COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

MONEY NEVER SLEEP

Autor: Rafael García Domínguez
Director: Pablo Zulaica Pérez

Madrid
Agosto de 2020

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título
MONEY NEVER SLEEPS
en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el
curso académico 2020/2021 es de mi autoría, original e inédito y
no ha sido presentado con anterioridad a otros efectos.
El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido
tomada de otros documentos está debidamente referenciada.

[illegible]

Fdo.: Rafael García Domínguez Fecha: 15/ 08/ 2020

Autorizada la entrega del proyecto
EL DIRECTOR DEL PROYECTO

[illegible]

Fdo.: Pablo Zulaica Pérez Fecha: 15/ 08/ 2020

6º. Fines y funcionamiento del Repositorio Institucional.
La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 15 de Agosto de 2020

ACEPTA

[illegible]

Fdo: Rafael García Domínguez

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

MONEY NEVER SLEEP

Autor: Rafael García Domínguez
Director: Pablo Zulaica Pérez

Madrid
Agosto de 2020

Agradecimientos

En primer lugar, dar las gracias a Pablo, mi director de proyecto por su dedicación, su tiempo y su ayuda.
Además, quería agradecer tanto a mi familia y amigos por el apoyo en estos años de carrera y los dos años de Máster.
Por último, a toda la comunidad de Comillas y en particular a el profesorado y mis compañeros de la escuela ICAI que han contribuido a mi formación tanto profesional como personal.

Gracias a todos.
Rafael.

MONEY NEVER SLEEPS

Autor: García Domínguez, Rafael
Director: Zulaica Pérez, Pablo
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Desarrollo de una solución tecnológica, un Bot de Telegram, que ayude a los usuarios a realizar los retos que quieren alcanzar. Creación de un modelo de negocio, startup, basado en el Business Model Canvas alrededor de este producto y validación de esta idea de negocio con el mercado.

Palabras clave: Bot, BMC (Business Model Canvas) , MPV (Mínimo Producto Viable)

1. Introducción

Tras observar que una parte de la población tiene el problema de que no es capaz de cumplir los retos que se proponen en este Trabajo de Fin de Máster se pretende solucionar dicho problema proponiendo la creación de un startup alrededor de un producto que se encargue de ayudar a estas personas a que lleguen a alcanzar esos objetivos.

2. Definición del proyecto

En este Trabajo de Fin de Máster se propone la creación de startup alrededor de un producto tecnológico muy escalable, un Bot de Telegram, cuya función es la de promover y ayudar a los usuarios a conseguir sus retos personales que por uno u otro motivo no son capaces de realizar. Para ello el producto se sustenta en la idea de que el mayor motivador de la sociedad actual es el dinero y que por ello asociando una cantidad de dinero, a la que llamaremos apuesta, a cada reto que se proponga el usuario tendrá una motivación adicional para cumplirlo, ya que de no hacerlo perderá el dinero apostado.

3. Problema

El problema en torno al cuál se estructura el proyecto es que la sociedad hoy en día, y en concreto algunos segmentos de la población (estudiantes y trabajadores 25-40 años), les cuesta cumplir los retos que se proponen o que han pensado en ocasiones conseguir y que finalmente ni se los llegan a proponer seriamente. Esto está sucediendo debido a varios hábitos que tienen estas personas a las cuáles se dirige el producto. En primer lugar, existe una falta de motivación a proponerse nuevas metas. En segundo lugar, muchos de estas personas no consiguen tiempo para ello y esto nace de una falta de organización que les hace no aprovechar el tiempo o muchas veces dedicarles este tiempo a otras cosas que no deberían de ser prioritarias. Todo esto unido a la situación actual en la que tanto a nivel académico como laboral las personas están más exigidas y tienen menos tiempo libre y mucha parte del que tienen se dedica al descanso por el estrés y el cansancio acumulado durante la semana.

4. Solución

Expuesto y analizado el problema se diseña un producto para dar solución a este. Dicho producto se sostiene bajo la asunción de que el principal motivador e incluso podríamos decir motor de la sociedad actual es la economía. Bajo este principio se estructura el producto, que introduce el dinero como un motivo adicional que consiga eliminar estas actitudes de pereza, falta de motivación y desorganización.
El producto por tanto se basa en que las personas que quieran comprometerse a cumplir sus retos asociarán una cantidad de dinero a cada reto que quieran hacer que solamente se les será devuelto en caso de cumplir

el reto, en caso contrario el usuario perderá el dinero. Por último, y para incentivar más el uso del producto se decidió introducir una rama solidaria y es que parte de ese dinero que se pierde se destinará a una causa solidaria.

5. Telegram Bots

El producto que se utilizará como solución es un Telegram Bot. Una solución tecnológica que se en marca dentro de la aplicación de mensajería Telegram. Estos bots son soluciones automatizadas que son capaces de mantener una conversación con usuarios reales de la aplicación y cuya función es la de servir como herramienta de bajo coste y muy escalable a las empresas. Un Bot funciona de la siguiente manera. En primer lugar, tiene que ser programado con una serie de estados que sean capaces de dar respuesta a cualquier mensaje (input) recibido por parte del usuario. El Bot no es capaz de enviar mensajes a usuarios sin que ellos hayan empezado la conversación, sin embargo, una vez una conversación acaba el bot puede quedar en estado de reposo y utilizar el último mensaje del usuario para iniciar una nueva conversación lo cual soluciona fácilmente esta restricción y lo hace más útil.

En la actualidad cada vez más el uso de estas herramientas automatizadas es más común y es que como se ha mencionado el bajo coste, la funcionalidad que tienen incluso pudiendo procesar pagos y la capacidad de poder funcionar conjuntamente con otras aplicaciones, bases de datos..etc, además de su escalabilidad hacen que su presencia en las empresas cada vez sea mayor para entender mejor que busca un cliente al hacer una consulta o directamente darle el servicio al cliente sin necesidad de un empleado.

6. Business Model Canvas (BMC)

Una vez explicada la idea y el producto que va a dar solución al problema, se diseña la estrategia de la empresa que se pretende crear alrededor de esta idea. Para ello, se utilizada una herramienta muy conocida en el mundo startup llamada Business Model Canvas. Esta herramienta define de forma clara y concisa cuales van a ser los agentes sobre los que se va a fundamentar el negocio, las funciones y relaciones con cada uno de ellos. Para ello, se definen nueve bloques entre los que existen relaciones claras. Los bloques principales son el de segmento de clientes, propuesta de valor, estructura de costes y flujo de ingresos. Sin embargo, estos de complementan con los demás bloques que tratan sobre los recursos que necesita la empresa, las actividades clave, las alianzas con otras empresas, la relación con los clientes y los canales por los que llega a usuarios y clientes.

En el apartado del BMC, se define con detalle cada bloque y la relación que existe entre los bloques de la empresa quedando una imagen clara del modelo de negocio de la empresa de forma muy visual, muy fácil de entender para alguien que no sepa nada sobre la empresa y que además permite reducir tiempo de preparación y recursos empleados en su desarrollo.

7. Cómo funciona el producto

Entendido el modelo de negocio, se pasa a explicar con detalle como funciona el producto. Como se ha indicado, el producto es un Bot hecho en Telegram, en él los usuarios podrán realizar cuatro funciones:

- Crear un nuevo reto: Se le pedirá una pequeña descripción, los datos de contacto de una persona que se encargará de juzgar si se consigue este o no, una cantidad de dinero asociada al reto que se perderá en caso de no hacerlo y una fecha máxima para realizarlo.

- Juzgar un reto: En caso de que alguien te nombre como juez de su reto, el Bot le pedirá el código del reto a juzgar y acto seguido le preguntará si el reto ha sido completado o no. En función de la respuesta de este se procederá a la devolución del importe apostado o no.

- Ver tus retos: En esta opción el usuario (si es que es un usuario que ya tiene algún reto creado), podrá ver los retos que ha ido haciendo usando el producto, ver el estado de cada un de ellos y sus características.

- Retar a otra persona: Con esta opción se permite a un usuario a diseñar un reto y planteárselo a otra persona la cual para que este se haga efectivo deberá de aceptarlo y entonces se le asignará.

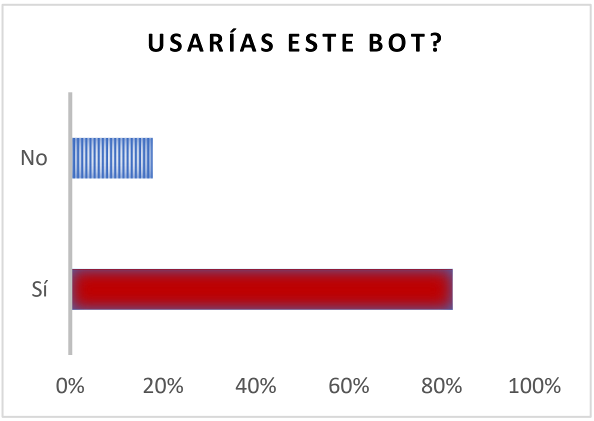
Estas son las funciones principales del producto actualmente, última versión validada con el mercado. Cabe destacar que el producto está abierto a futuras mejoras y actualizaciones en función de lo que el mercado objetivo vaya demandando. Este es uno de los principios básicos del modelo LearnStartup bajo el que se crea este producto, pivotar para ir adaptándose en cada momento a las necesidades de los clientes.

8. Validación de la idea

Una vez diseñado el producto se empieza han ha desarrollar bocetos de producto para testear con el mercado, los conocidos como Mínimo Producto Viable (MPV). Para testear con el mercado se crea un video promocional explicando el producto y que solución pretende dar y una encuesta para sacar conclusiones. De cada validación se obtienen feedback y se actualiza el MPV para mejorar la prestación que el producto de al cliente.

9. Resultados

Los resultados obtenidos tras la validación con el mercado y los estudios económicos muestran que la idea es atractiva para el mercado con un **82.65%** que usaría el Bot y también **rentable en el largo plazo** debido principalmente a la escalabilidad.



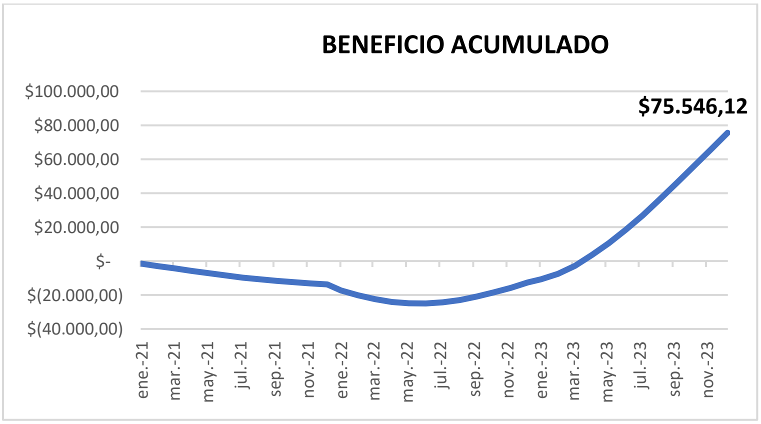


Figure: Resultado Encuesta y beneficio acumulado esperado

MONEY NEVER SLEEPS

Author: García Domínguez, Rafael
Supervisor: Zulaica Pérez, Pablo
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

Development of a technological solution, a Telegram Bot, that helps users to achieve the challenges they want to achieve. Creation of a business model, start-up, based on the Business Model Canvas around this product and validation of this business idea with the market.

Keywords: MVP (Minimum Viable Product), BMC (Business Model Canvas), Bot

1. Introduction

After observing that a part of the population has the problem of not being able to meet the challenges proposed in this Master Thesis, we intend to solve this problem by proposing the creation of a startup around a product that will help these people to achieve these objectives.

2. Definition of the project

This Master Thesis proposes the creation of a start-up around a very scalable technological product, a Telegram Bot, whose function is to promote and help users to achieve their personal challenges that for one reason or another they are not able to do. The product is based on the idea that the greatest motivator in today's society is money, and therefore, by associating an amount of money, which we will call a bet, to each challenge that is proposed, the user will have an additional motivation to meet it, since if he does not do so, he will lose the money bet.

3. Problem

The problem around which the project is structured is that society today, and in particular some segments of the population (students and workers 25-40 years), have difficulty meeting the challenges that are proposed or have thought at times and finally not even proposed seriously.

This is happening because of several habits that these people have to which the product is addressed. In the first place, there is a lack of motivation to set new goals. Secondly, many of these people do not get time to do so and this is born of a lack of organization that makes them not take advantage of the time or often devote this time to other things that should not be a priority. All of this, coupled with the current situation in which both academically and in the workplace people are more demanding and have less free time and much of the time they have is dedicated to rest due to the stress and tiredness accumulated during the week.

4. Solution

Once the problem has been exposed and analysed, a product is designed to provide a solution. Such a product is sustained under the assumption that the main motivator and even we could say engine of today's society is the economy. Under this principle the product is structured, which introduces money as an additional motive that manages to eliminate these attitudes of laziness, lack of motivation and disorganization.

The product therefore is based on people who want to commit to meet their challenges will associate an amount of money to each challenge they want to do that will only be returned if they meet the challenge, otherwise the user will lose the money. Finally, and to further encourage the use of the product it was decided to introduce a solidarity branch and that is that part of that money that is lost will go to a solidarity cause.

5. Telegram Bots

The product to be used as a solution is a Telegram Bot. A technological solution that is branded within the Telegram messaging application. These bots are automated solutions that are capable of maintaining a conversation with real users of the application and whose function is to serve as a low-cost and highly scalable tool for companies. A Bot works as follows. First, it has to be programmed with a series of states that are capable of responding to any message (input) received from the user. The Bot is not able to send messages to users without they have started the conversation, however, once a conversation is over the Bot can remain in the idle state and use the last message from the user to start a new conversation which easily solves this restriction and makes it more useful.

Nowadays the use of these automated tools is more and more common and as it has been mentioned the low cost, the functionality that they have even being able to process payments and the capacity of being able to work together with other applications, databases...etc, in addition to its scalability makes its presence in the companies more and more important to understand better what a customer is looking for when making a consultation or directly giving the customer service without the need of an employee.

6. Business Model Canvas (BMC)

Once the idea and the product that will provide a solution to the problem have been explained, the company's strategy is designed around this idea. To do this, we use a well-known tool in the startup world called Business Model Canvas. This tool defines in a clear and concise way the agents on which the business is going to be based, the functions and relationships with each one of them. For this purpose, nine blocks are defined between which there are clear relationships. The main blocks are customer segment, value proposition, cost structure and revenue flow. However, these are complemented by the other blocks that deal with the resources the company needs, key activities, alliances with other companies, the relationship with customers and the channels through which it reaches users and customers.

In the section of the BMC, each block is defined in detail and the relationship that exists between the blocks of the company leaving a clear image of the business model of the company in a very visual way, very easy to understand for someone who does not know anything about the company and also allows to reduce preparation time and resources used in its development.

7. How the producto works

Once the business model is understood, the product is explained in detail. As indicated, the product is a Bot made in Telegram, in which users can perform four functions:

- Create a new challenge: You will be asked for a short description, the contact information of a person who will be in charge of judging whether or not this challenge is achieved, an amount of money associated with the challenge that will be lost in case it is not achieved, and a maximum date for completion.

- Judging a challenge: In case someone names you as the judge of his challenge, the Bot will ask you for the code of the challenge to be judged and then will ask you if the challenge has been completed or not. Depending on the answer of this one will proceed to the refund of the amount bet or not.

- View your challenges: In this option the user (if he is a user who already has a challenge created), will be able to see the challenges he has been doing using the product, see the status of each one of them and their characteristics.

- Challenge someone else: This option allows a user to design a challenge and propose it to another person who must accept it and then it will be assigned.

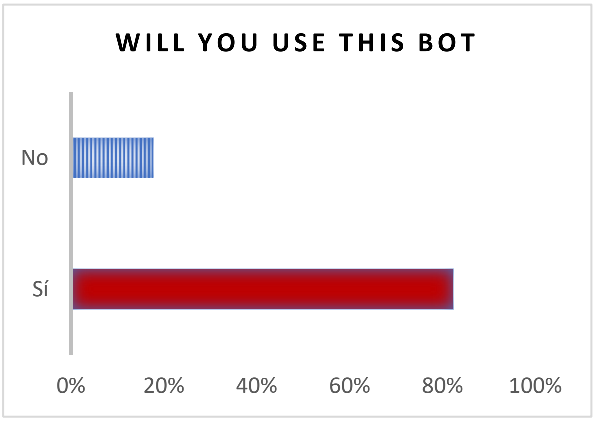
These are the main functions of the product currently, last version validated with the market. It is worth mentioning that the product is open to future improvements and updates depending on what the target market demands. This is one of the basic principles of the LearnStartup model under which this product is created, pivoting to be adapted at all times to customer needs.

8. Validation of the idea

Once the product is designed, we start developing product sketches to test with the market, known as Minimum Viable Product (MVP). To test with the market, a promotional video is created explaining the product and what solution is intended to give and a survey to draw conclusions. From each validation, feedback is obtained, and the MVP is updated to improve the performance of the product for the customer.

9. Results

The results obtained after the validation with the market and the economic studies show that the idea is attractive for the market with 82.65% that would use the Bot and also profitable in the long term mainly due to the scalability.



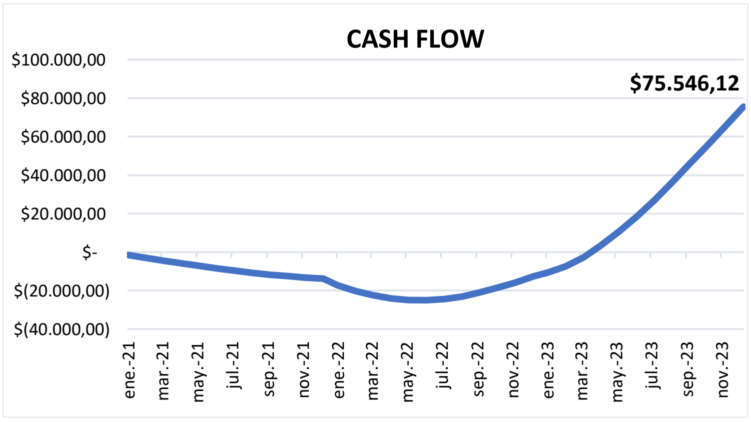


Figure: Results survey and expected cash flow

INDEX

FIGURES INDEX

TABLES INDEX

1. CHAPTER 1: INTRODUCTION

1.1 THE PROJECT

The project consist in create a startup around the idea of developing a Telegram Bot whose main feature is that is going to help the users to perform challenges that they have in theirs daily routine and they do not manage their time, motivation or other factors for doing it. Firstly, it is going to explain the problem that it is trying to be solve.

1.1.1 PROBLEM

Nowadays, people spend most of their time working instead of being with family, friends or doing whatever that they would like to do. Moreover, after working some people is less proactive to do other things because they are so tired or also because they prefer to rest for tomorrow.

How many times we heard one friend telling us that he/she is so tired to hang out, or to practice sport, etc..? They believe that they do not have enough time to combine work with friends, sport or to pursue their dreams. However, this is not true, the problem that most of the society is challenging is the lack of willpower, motivation, or organization to use prioritize their free time in order to comply with their goals.

The problem that some people face is that they are at a standstill and adjusted to their routine and they have stop challenging themselves.

After stating the main problem that the company is trying to solve by proving a service to the clients, now it is going to be explained the solution that the company provides to this problem, which is the most important part of the project.

1.1.2 SOLUTION

In this context it is suited Getyourchallenge. The company main objective is to meet the requirements of that part of the society who have this problem.

As it is stated in the previous part the problem of this community is their lack of

> motivation to take part in other activities. In other words, they have goals and plans to do over the year but at least they feel lazy, do not find time to do it or they finally lose their willingness to them.
>
> It is appropriate to say that money is the best motivator in the current society. People do their best when they are encouraged about earning money or instead not losing it. This statement could appear as hopeless, but why if we use this tool in order to improve our actual world? Actually, most of the people work because of money, so why do not apply this concept in order to motivate people to make the best of their free time?
>
> In this environment is where the solution outlined by the company lies.
>
> The solution exposed is to give a tool for this community in which they can create their challenges and bet an amount of money that they will lose in case that they do not achieve the goal settled.
>
> This tool is a Telegram Bot that interact with the users asking about the name, amount, due date and the most important part who is going to be the person who is going to say if they achieve the goal or not, the judge. Moreover, the Bot can be beyond this use. The Bot will storage in a database all the challenges created by the user and he/she can review then whenever they want.
>
> Finally, in order to continue challenging the users the Bot have a space in which the users can see other challenges that have been done by other users. In this way, they will have always new challenges and areas to continue improving to become the best of themselves.

The concept behind this bot is that for everyone, money is a principal motivator and that for everything that you do there is a price for which you will do everything you could in order to do not lose this money. The bot, with the help of a person related to the user, will determine if the user has performed the challenge and therefore, return the money that the user bet to the user or in the other hand, send this amount to the social project selected by the user.

In the project, after explaining the product, it will be explained the business strategy of the company around this product and for doing so it will be used the business model canvas (BMC) tool.

1.2 REASON

The main reason of the project is to create a sustainable startup around the product that it has been developed. By doing it is expected to put in place all the knowledge that have been acquired during the two years master's degree combining the engineering aspect, related to the programming part of the project with an entrepreneur and business aspects which focus on the business model canvas and company strategy around the product.

Moreover, with this project also there is a social purpose to achieve, provide people a new tool that they can use to satisfy people needs. In particular, the startup expects to solve the client's needs related to a lack of motivation, organization, and discipline to reach the objectives that they have and are not capable of doing in their own.

Using money as an incremental motivator to achieve this objective is the underlying concept that it is used with this product. As a last resort the company have also a charity or social contribution objective because the majority amount of the money lost by the users (by not achieving their objectives) is going to be transfer to charity organization or social institutions which have a goal to improve the social environment, helping in one way or another to develop a better world. This idea of transfer the money to a non-profit organization came because most of the time that someone collaborate with a cause, he does it without receiving anything else to himself that the feeling to help other. Sometimes, people think about that and they decide not to collaborate in important matters to our society. With this tool, we expect that the society is going to be more prompted to join into campaigns like these.

1.3 HOW IS GOING TO BE DEVELOPED

The startup main asset is the bot, which is the tool use by the users to provide them a service. However, the startup needs more elements or tools in order to complement the bot and create a complete company such as the database and a web.

Firstly, it should be designed the interface user-company, the bot. To design it, it must be stated the principal features that the users is expecting from the bot and also what the bot need to judge if the challenge is done or not. By doing so, the conversation between user and bot will be designed.

The next step is to implement this by creating the tool, which is the most difficult and time-consuming part of the project. For doing it, it will be used the Telegram app, using the BotFather, which is a bot created by telegram to whom you can talk in order to create new bots in telegram or manage the bots that you have created so far. Moreover, it will be used a programming interface, in this case Pycharm, in order to program the Telegram bot to create the conversation that is has been designed. The programming language that has been used is Python.

1.3.1 TELEGRAM BOTS

The value proposition of the company is a Telegram bot that help the users to reach their objectives. Therefore, the first thing that is going to be explained in detail is what is a telegram bot and also for what purpose are they being used nowadays.

Firstly, it is going to be explained what a telegram bot is about. Telegram bots are applications that are programmed to perform several functions, interacting with users automatically after receiving a user's message. The particular thing about telegram bots if that they work in the telegram environment and thus, they do not need and additional installation procedure to run. A more sophisticated definition provided by Telegram about the bots are:

"Telegram bots are ***AI-inspired apps*** *that can serve many functions: send relevant information about the weather or useful news articles, schedule reminders, play tunes, create to-do lists, and so much more. Such bots work in Telegram, a popular* ***instant messaging application*** *used by millions of people worldwide."*[1]

"Bots are a software application capable of running automated tasks and often aim to act as a real person" [2]

The main objectives of the bots are to reduce the human resources used in areas and also increase the velocity of the service provided to the client. Automatizing some

[1] (Laura 2020)
[2] (Treuberg 2020)

> areas will reduce the cost structure of the companies and make the enterprise more efficient and with less exposure to human capital.
>
> The main objectives of the bots are to reduce the human resources used in areas and also increase the velocity of the service provided to the client. Automatizing some areas will reduce the cost structure of the companies and make the enterprise more efficient and with less exposure to human capital.
>
> The most common bot is the chatbot, which is the type of bot used in this project. A chatbot recognize text or speech from the user, after that analyze the information and after that it response automatically. The ultimate goal of bots is to be prepared to act as a real person.
>
> Bots can be created to perform several functions and there are lots of them which codes are public and can be used by everyone. To create a bot by your own you only need to download the telegram app and start a conversation with the BotFather, another telegram bot, and as its name stated is the first and more powerful bot in Telegram.

After that, the bot needs to storage the information provided by the user and also by Telegram in the conversation. This is because the bot will interact with the user differently if they are new users, they have already challenge with the company and also if the user has already used some judges before for his challenges. All this info is storage in a database and use by the bot to interact with the user in a personalize way. This database is created using the online tool called, Airtable. Airtable main feature is that it is an online spreadsheet easy to use, that is powerful enough to act as a database that can be use for customer-relationship management (CRM). This function as a CRM is important because this database will be automatized to send emails directly to users and judges who are involve in a created challenge. The database consists in six tables: All, Users, Challenges, Judges, Variables and Defiance, that will be explained in more detail in the following chapters. Until this stage, this will be the minimum viable product.

After creating the Minimum Viable Product (MVP), the product will be tested with the market in order to analyze the product market fit. To do it, a promotional video explaining the product will be created and also a web to explain the motivation of the company and the services that it offers.

Finally, after testing the product with the market it will be created a monetization and cost structure forecast for the company as well as a marketing strategy to launch the product into the market and next steps.

1.4 OBJETIVES

The objective of this Master Thesis is to create a startup from the idea of a Telegram Bot which purpose is to help its users to achieve their challenges. Therefore, the crux of the Project is to help the users to improve and become the best of themselves.

Specifically, the project objectives have been split in several milestones that have been reached in order to achieve the final goal.

- Create a functional bot, MVP (Minimum Viable Product), which consist in a prototype of the product that can be tested with users in order to take insights of the product market fit and get information to turn the strategy and guide the product in a way to solve the clients needs.
  In this MVP, the user can:
  - Create a challenge (description, amount, judge)
  - Judge an existing challenge, giving the code to the Bot

- Create a database in which it would be recorded all the information about the challenges created by the users and that as last resort will be use to measure ratios and KPI's that will be useful to analyze.
  This database will be create using Airtable, an online database service that can be related with the Bot using its API key.
  The database will consist in 6 main tables:
  - **All table**: collect all the info of every interaction that a user have with the bot.
  - **Users table**: collect all the users info (name, ID, email and language)
  - **Challenge table**: collect the info related to the challenged that have been successfully created by the users. (description, amount, due date, judge name, judge email and status of the challenge)

    - **Judge table**: it is recorded the info of the judges selected by the users to judge their challenges (name, ID, email)
    - **Variables table**: records the global variables that are needed for the conversation between bot and user.
    - **Defy table**: This table is used to record all the challenges that have been created to challenge someone else and are waiting to be accepted or denied by this person (the future user).

- Develop a Business Model Canvas around the product created. This is the business part of the project. It is stated the strategy of the company in a startup environment. It is defined the following aspect of the company strategy:
    - Clients
    - Cash inflows (how the company generate money)
    - Value proposition, what is the company offering to the potential clients
    - Customer relationship
    - Channels
    - Key activities of the company
    - Key resources used by the company
    - Cost structure
    - Key partners, other business/enterprise related to the company operation

- Test the MVP, with potential clients and analyze their interactions to take insights and redirect the product options. To do so it will be used a survey that the users should answer after trying the product.

- Create a marketing campaign to introduce the idea into the market. It will be created by:

    - Create a video presenting the idea and how the product works
    - Develop a website
    - Ask for potential users emails to create a network

1.5 STRUCTURE

The document will be structured in order to firstly, in this Chapter one, introduce the startup that has been created and the product around it, as well as the main objectives that the project has from the start and the reason why it has been decided to perform this project.

After that in the Chapter two, it will be explained the business model around the startup and for doing it, the business model canvas tool will be used. It will be stated the BMC tool, the concept of each item defined in the model and explained the strategy of the company using this tool.

The Chapter three will cover the development of the product and the features and how has been developed the minimum viable product. Firstly, explain the product, after that how it has been created and finally show how does it works and how it interacts with the users.

In the fourth chapter the product will be tested with the market in order to find the product-market fit which is the most important goal during the development of a startup. To do it, a landing page and a survey for the early adopters of the startup will be created and tested. This chapter finish stating the next steps that the company should follow.

In Chapter five, it will be explained the planned launch to the market of the company and the steps that fill be followed to do it.

Finally, in the last chapter the main conclusions of the project will be explained. These conclusions will be split in two ways: the first one conclusions related to the company and the product and the second ones related to the personal knowledge and skills that have been created and developed by doing this project.

2. CAPÍTULO 2: BUSINESS MODEL

2.1 INTRODUCTION

In this chapter it is going to be explained the business model of the startup. For doing it as it is stated before the business model canvas tool will be used. Firstly, the BMC tool will be explained and contextualized, after that the busines model canvas of the startup created will be explained in detail.

2.2 BUSINESS MODEL CANVAS (BMC)

The business model canvas is a chart representation of the main variables that sustain the business in an organization. In particular, the business model canvas is a strategy tool use to develop the organization of the company and identify in a matter of minutes how the business work and who are the main agents that involves in the activities of the company.

The origins of the business model canvas start in 2004 by Alexander Osterwalder who develop this tool as his doctoral thesis. This tool was created to design and validate business model in a visual and easy way instead of the classical business plan which consist in several pages about future expectations that are only hypothesis.

Before this tool, the companies were structured around the historical business plan which requires more detail and expectations about the future of the company and whose format was strict and very tedious most of all for new business ideas which only need to design the structure and try to validate the idea in the market before any investment.

The main disadvantage of the old business plans is that it require so many time to do and the main part of it is expectations about a future where nobody knows how the market is going to respond to the proposition of the company.

Therefore, the business model canvas is a tool that solve this problem and very useful for startups because it is a way to structure the main variables of the business. Moreover, it allows the company to change the value proposition easily in order to redirect it to the necessity of the market.

The model consists in 4 areas and 9 blocks. The areas are:

- What is the company offering?
- How the company is going to earn money from the business?
- Who is going to be directed the value proposition of the company to?
- How is going to perform the business the value proposition

As it has been stated before, the business model canvas is a tool to design the business of the company. Therefore, the main objective is to answer to these four questions. To do so the model used nine building blocks in which it is explained in more detail the variables which are related to the company business.

At the same time that each block is being explained, the BMC of the company, Getyourchallenge, it is going to be explained. By this way, the understanding of the company key business factors will be complete. Therefore, a summary of the BMC of Getyourchallenge is provided below.

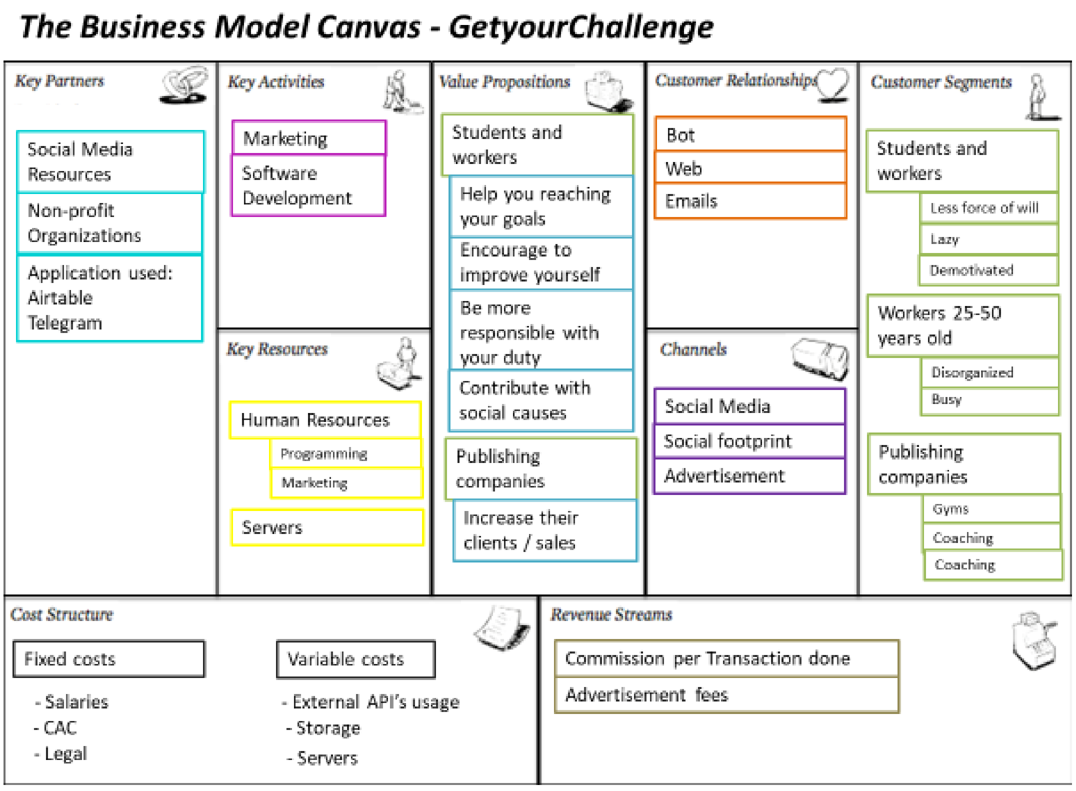


Figure 1. Business Model Canvas

To answer the first question, it is used the first and most important block inside the model, the value proposition.

2.2.1 VALUE PROPOSITION

In this block it is defined the core activities and services that are provide by the company. The products that the company offer. While developing this block the management has to think in answering by these activities or services the customers needs. That is to say that in the value proposition the company has to show how they are distinct from it potential competitors and by doing so stablish its core services or products that solve their clients needs.

Once the target market it is set, then the company have to explain how their product and services can solve the issues, pains of the clients. Moreover, they should also describe the main benefits that the customer will obtain in exchange of the services provided. The tool that startup uses to explain all this is the value proposition.

A good explanation of what is the value proposition and how it is used is the one provided by Peter J. Thomson:

"A value proposition is where your company's Product offer intersects with your customer's desires. It's the magic fit between what you make and why people buy it. Your value proposition is the crunch point between business strategy and brand strategy."

•Features: The features also provide the 'reasons to believe'. It is the description of how your product is going to work.

•Benefit: The benefits are the characteristic that your product/services is going to do for the customer and also how it is going to make their life easier.

•Experience: Describe the sentiment and feelings that the customer is going to experience due to the use of the product/service. Therefore, it can be defined as the sum of the benefits and features.

2.3.2.1 VALUE MAP

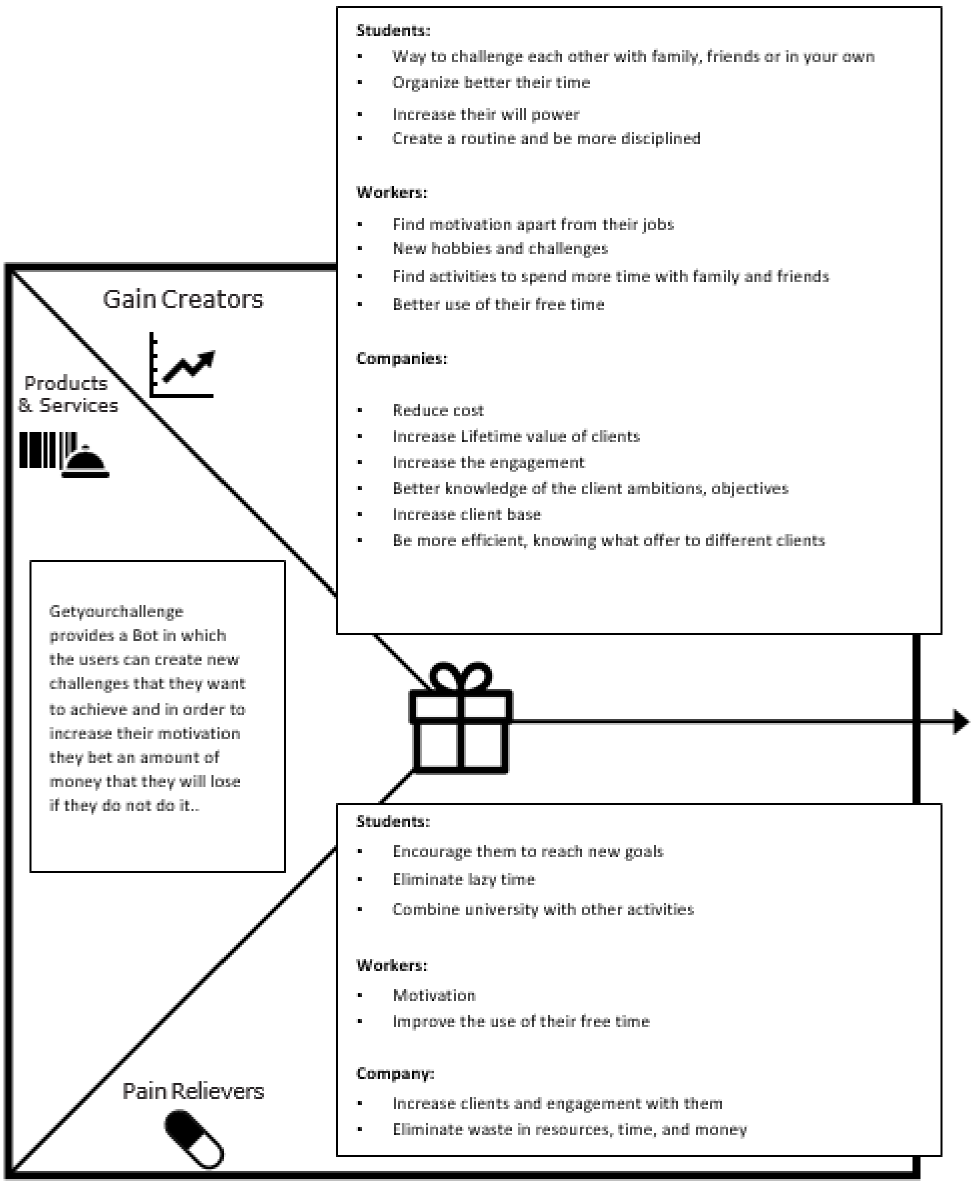
The value map is the tool used by startup in which they state the value proposition of the company and at the same time addressing the pains, desires and goals that have been analyzed previously in the customer profile.

Therefore, the Value map answer to three main features:

• Products and services: The company list the product and services in which it is sustain their value proposition, this is the most important part of the value map.

• Pains and killings: In this section the company express the features of their value proposition in which they mitigate or reduce the risk perceived by the customer and also the obstacles that they are facing.

• Gains creators: This section provides the expected utility that the product is going to give to the client. Increasing time, functional utility, profitability or generating positive emotions.



Students:
- Way to challenge each other with family, friends or in your own
- Organize better their time
- Increase their will power
- Create a routine and be more disciplined

Workers:
- Find motivation apart from their jobs
- New hobbies and challenges
- Find activities to spend more time with family and friends
- Better use of their free time

Companies:
- Reduce cost
- Increase Lifetime value of clients
- Increase the engagement
- Better knowledge of the client ambitions, objectives
- Increase client base
- Be more efficient, knowing what offer to different clients

Getyourchallenge provides a Bot in which the users can create new challenges that they want to achieve and in order to increase their motivation they bet an amount of money that they will lose if they do not do it..

Students:
- Encourage them to reach new goals
- Eliminate lazy time
- Combine university with other activities

Workers:
- Motivation
- Improve the use of their free time

Company:
- Increase clients and engagement with them
- Eliminate waste in resources, time, and money

Figure 2. Value Map

2.3.2.2 COMPETITORS AND SUBSTITUTIVES

After expressing the value proposition of the company is important from an investor perspective to know the competitors of the company, those companies which are offering a similar value proposition. In fact, the most important part for the investors is to compare the value proposition of the company, in this case Getyourchallenge with its competitors and substitutes companies.

- Competitors: Are those companies which are offering the same service or product as the company analyzed.
- Substitutes: Offer to the same clients a services that can be use instead of the one that the company analyzed is using

As an startup, at the beginning there are not many companies which offer a similar value proposition or other services that can be used instead of the one provided by the company. The startup ambition is to create a new market without this competition and by this way create a large community of users and barriers of entry for potential players that would like to enter in the market in the future.

In the case of Getyourchallenge, there is a service that was created some years ago which value proposition was close to their value proposition, which can be seen as a potential competitor.

COMPETITOR VALUE PROPOSITION

In this case the competitor is a website named "GO FUCKING DO IT". In this website the service offered is similar to the one that Getyourchallenge provides with the Telegram Bot.

On the website, the user can create a challenge inside the web, to do it the user has to complete the following items:

- Goal name
- Due date
- Amount to pay
- Your name and email
- Judge friend and email

Moreover, in the website there are some examples of goals that can be done in order to motivate the users to pursue new challenges.

The website makes money in two ways:

- Amount of losses challenges
- Fee from adds in the website

COMPETITOR LAUNCHING AND RESULTS

Due to the fact that the type of company can be seen as a startup like Getyourchallenge, it is important to analyze how was the launching process of the company and which were the results obtained from the company.

The launching process was simple. Firstly, the company idea was promotion the idea in a blog called "Hacker News". The idea of doing this was to create a strong viral effect because very influential people can read it. After that, the creator of the website talked with a journalist in order to show the business in an online newspaper article. Also, they created a Twitter account and start posting there to promotion the website.

Finally, after validating the idea they launch the product and the results of users, visitors to the website and challenge created were in the first month were:

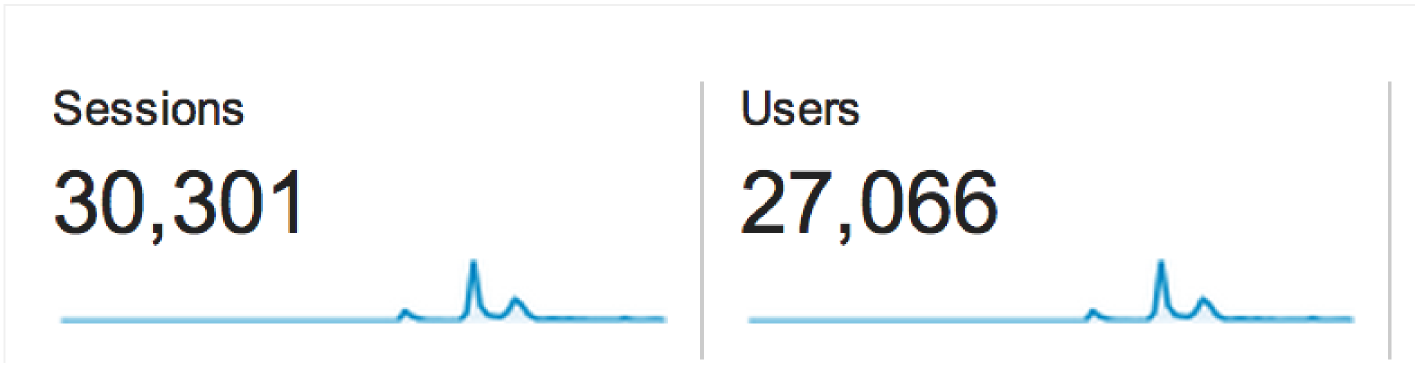


Figure 3. Go Fucking Do It, first users data

"In the end 1% of visitors set a goal and entered their credit card details. That's a pretty average conversion rate for most sites that require payment. The average price set was $78.54. Of the people that set a goal, 84% actually reached it (or their supervisor stated they did).

However, 55% of all goals were never set as reached or not to by the supervisor! That means the emails were probably not being read! So with that deducted, I was left with

1% actually being charged. That left only a few people failing and so just a few charges.

So for this model to make ~$2000 per month, it'd have to be scaled up to get 250,000 visits per month, with 2,500 people converting, and 25 people failing. That's only about 10x what it's at now. Still a challenge."[3]

COMPETITIVE ADVANTAGES OF GETYOURCHALLENGE

Once the value proposition of the competitor have been explained, the comparison between Getyourchallenge value proposition and the competitor gives the competitive advantages that Getyourchallenge has over the competitor.

Both companies have a similar clients segments and revenues streams. The main difference is in the value proposition.

In the website, each time that the user use the web, he/she has to fill all the items over and over again because the website cannot storage and recognize the user as one person who has used the website before. For the same reason, the website cannot offer a personalize treatment to the client because it cannot show to him/her the challenges that they are pursuing at the moment or send to him/her information of additional services from other companies that can be useful to achieve the goal.

On the other hand, as explained above in the value proposition of Getyourchallenge, the main focus of the company apart from the product service to the client is to create a real engagement with the user and try to do it by personalizing the treatment and having the information of the user with the bot storage to increase this engagement. Moreover, having partnership with companies which can offer services related to the challenges created in the Bot represent also a competitive advantage against the website because the revenues from adds of this companies have more value to the user and also for the companies (Getyourchallenge can increase the price for these adds).

[3] (2014)

SUBSTITUTES

On the other hand, there are some substitutes of the service provided by Getyourchallenge. Inside the potential substitutes, the most important one is focus in the sport sector. One of the main types of challenge that is created by the users is to pursue a sport challenge like, run a marathon, start practicing a new sport.

Some sport apps, mainly the running ones, which main service is to provide a tool to the user to measure their effort. However, the app are implementing a new service which is the one that can substitute the service of the Telegram Bot of Getyourchallenge. In this new section the app recommend the users new challenge related with sport activities and the history of the user with the app, having a personalize treatment as the one intended by Getyoutchallenge.

The following figures show this new service in the running app of a known brand as Adidas.

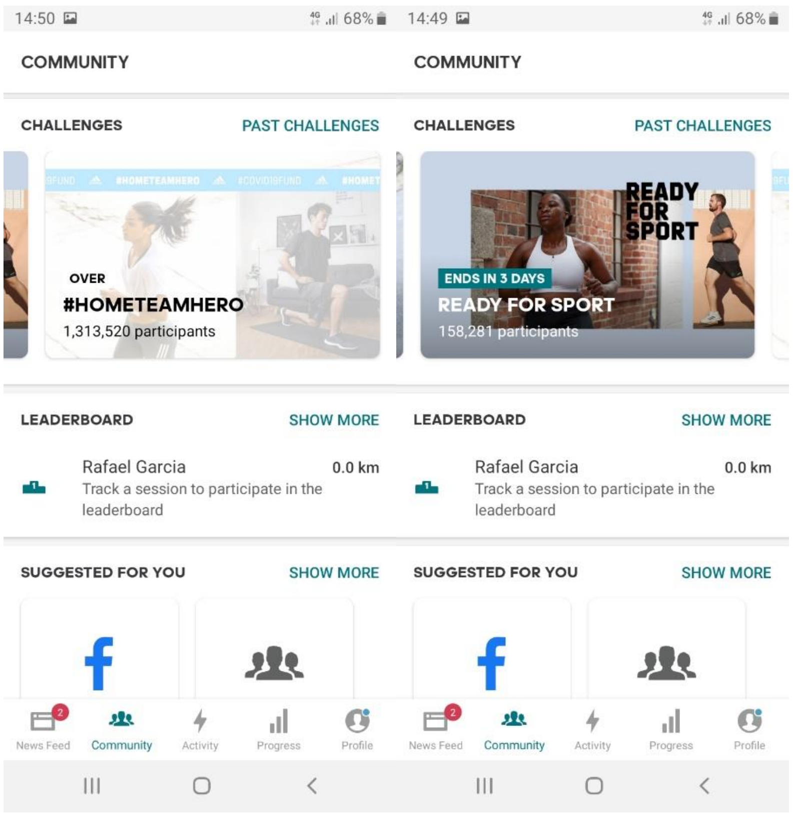


Figure 4. Challenge section Adidas app

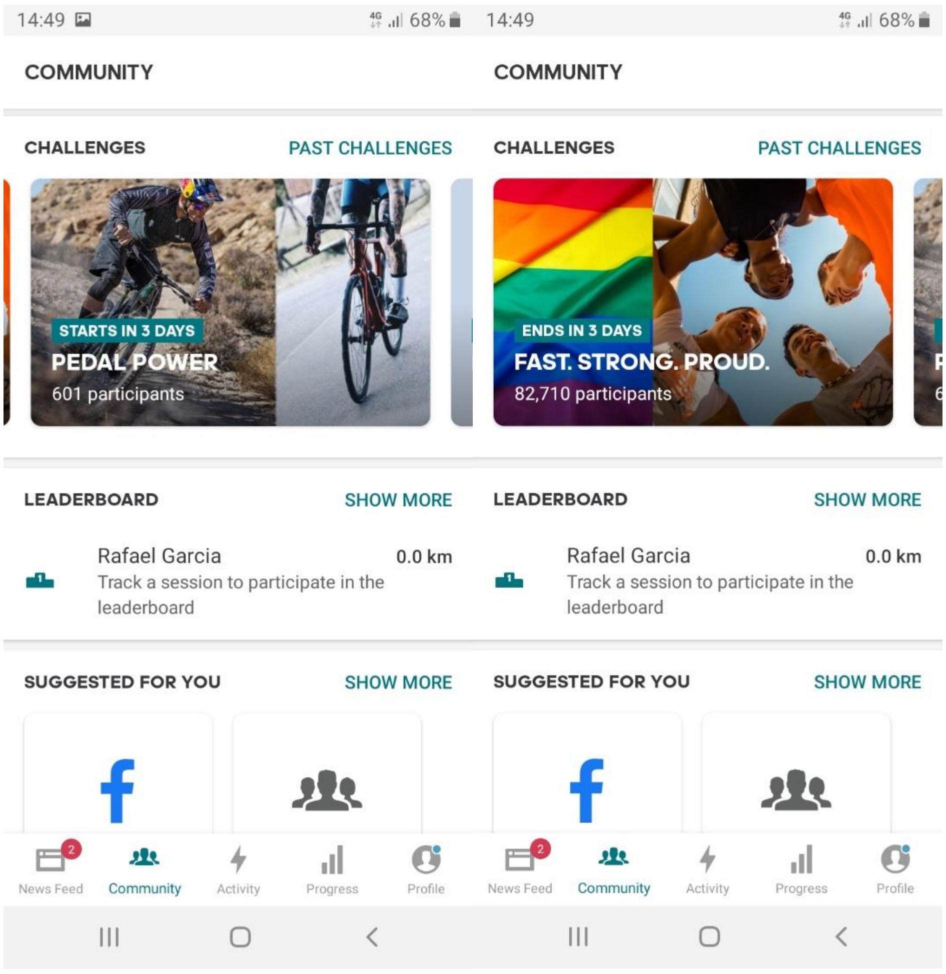


Figure 5. Challenge section Adidas app 1

However, the service is not as good as the value presented by Getyourchallenge. This makes sense because this is not the main idea of the running app and therefore the value provided is not the same.

Secondly, the model answer to whom is this value proposition directed. It is answer using three blocks: Clients segments, channels and client's relationship.

2.2.2 CLIENTS SEGMENTS

In this block the company state who are going to be the main types of clients, meaning, which types of persons are we focus in order to identify their needs and develop the value proposition around it. It is important to mention that the company should group different clients in the same category or type if they have the same needs from the company in spite of the fact that they can be very different. For example, the clients can be grouped by wealth, age, gender… depending on the value proposition the company have to segment the clients in different groups that will received or be reached by the company in a similar way.

By identifying the needs and requirements of each group, the value proposition can be modified to be more efficient oriented and lead to a greater customer satisfaction and therefore a competitive advantage against the competition.

The clients are those persons or enterprises which are going to benefit of the tool provided by a company. Therefore, the clients of Getyourchallenge are going to be all individuals and companies which will improve and take advantage due to the use of the Telegram Bot.

As we stated in previous chapter the main objective is to help people to achieve their goals, so it is obvious that the main clients are going to be individuals which have a lack of willpower.

However, there will be companies that also will benefit from the Bot. They can be split in two types of companies.

- Firstly, those which will encourage the use of the Bot inside their company in order to motivate their workers to complement their formation and challenge them. Specially, those companies whose business is to help customer to do new challenges.
- Secondly, enterprises which are related with the challenges that the users want to complete and can provide them a useful way to do it and therefore will be interested in advertising to the users.

These three types of clients are going to be defined with more detail with two tools: the customer persona and the customer profile.

2.3.1.1 CUSTOMER PERSONA

Find the ideal or target customer is one of the key issues that a startup face at the beginning when it is creating the business plan. If you do not define with detail the segment of the population to which you are going to focus, you will probably lose money, time and resources trying to capture people who are not really interested in your product. Because of that, defining the target market, specifically the type of person to whom the company is going to head of.

Customer persona is the first resources that startups use to face this issue. This tool is the best way to unify the vision all along the company of the characteristic of the ideal customer for the enterprise. A consumer persona is the idealized personification of that group. In other words, it's a fully fleshed out individual.

However, some companies have different types of customer which have different preferences, ambitions, necessities and objectives. Therefore, to capture and understand deeply all the profiles the company must outline one customer persona for it client segment who share the same characteristic.

The characteristic define in the customer persona can vary but mainly are:

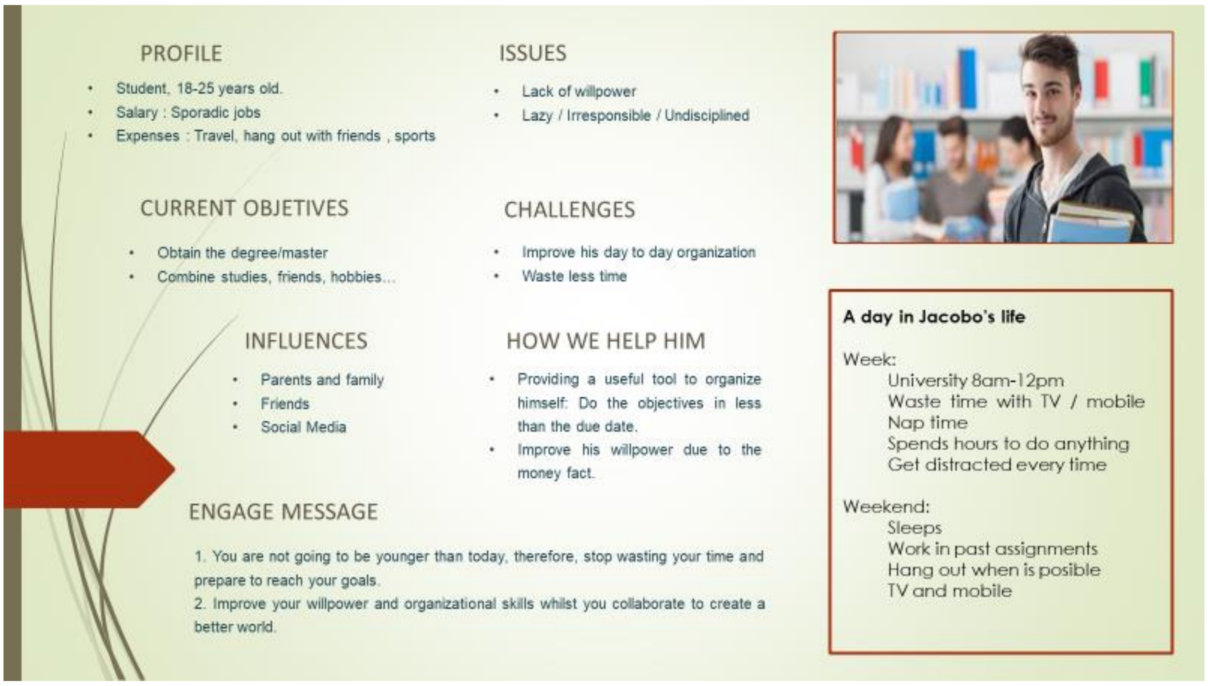
- Profile (occupation, age, expenses…)
- Issues
- Objectives
- Influences
- A day in his life

Getyourchallenge have two core clients:

- Individual users of the Bot.
- Companies related to the challenges that are pursuing the users and are interested in acquired them as customer, therefore want direct advertisement to them.

a) Customer Persona Individual Users

1) Students (18-25 years old)



PROFILE

- Student, 18-25 years old.
- Salary : Sporadic jobs
- Expenses : Travel, hang out with friends , sports

ISSUES

- Lack of willpower
- Lazy / Irresponsible / Undisciplined

CURRENT OBJETIVES

- Obtain the degree/master
- Combine studies, friends, hobbies…

CHALLENGES

- Improve his day to day organization
- Waste less time

INFLUENCES

- Parents and family
- Friends
- Social Media

HOW WE HELP HIM

- Providing a useful tool to organize himself: Do the objectives in less than the due date.
- Improve his willpower due to the money fact.

ENGAGE MESSAGE

1. You are not going to be younger than today, therefore, stop wasting your time and prepare to reach your goals.
2. Improve your willpower and organizational skills whilst you collaborate to create a better world.

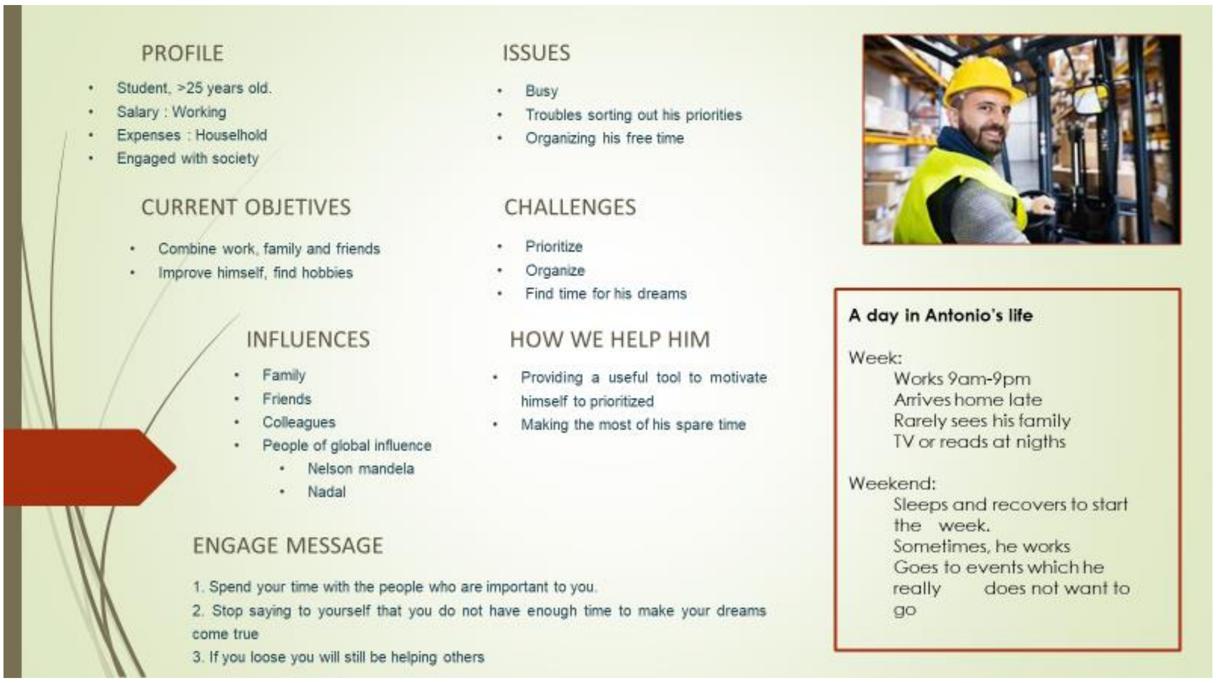
A day in Jacobo's life

Week:
University 8am-12pm
Waste time with TV / mobile
Nap time
Spends hours to do anything
Get distracted every time

Weekend:
Sleeps
Work in past assignments
Hang out when is posible
TV and mobile

Figure 6. Students Customer Persona

2) Workers (25-50 years old)



PROFILE

- Student, >25 years old.
- Salary : Working
- Expenses : Houselhold
- Engaged with society

ISSUES

- Busy
- Troubles sorting out his priorities
- Organizing his free time

CURRENT OBJETIVES

- Combine work, family and friends
- Improve himself, find hobbies

CHALLENGES

- Prioritize
- Organize
- Find time for his dreams

INFLUENCES

- Family
- Friends
- Colleagues
- People of global influence
    - Nelson mandela
    - Nadal

HOW WE HELP HIM

- Providing a useful tool to motivate himself to prioritized
- Making the most of his spare time

ENGAGE MESSAGE

1. Spend your time with the people who are important to you.
2. Stop saying to yourself that you do not have enough time to make your dreams come true
3. If you loose you will still be helping others

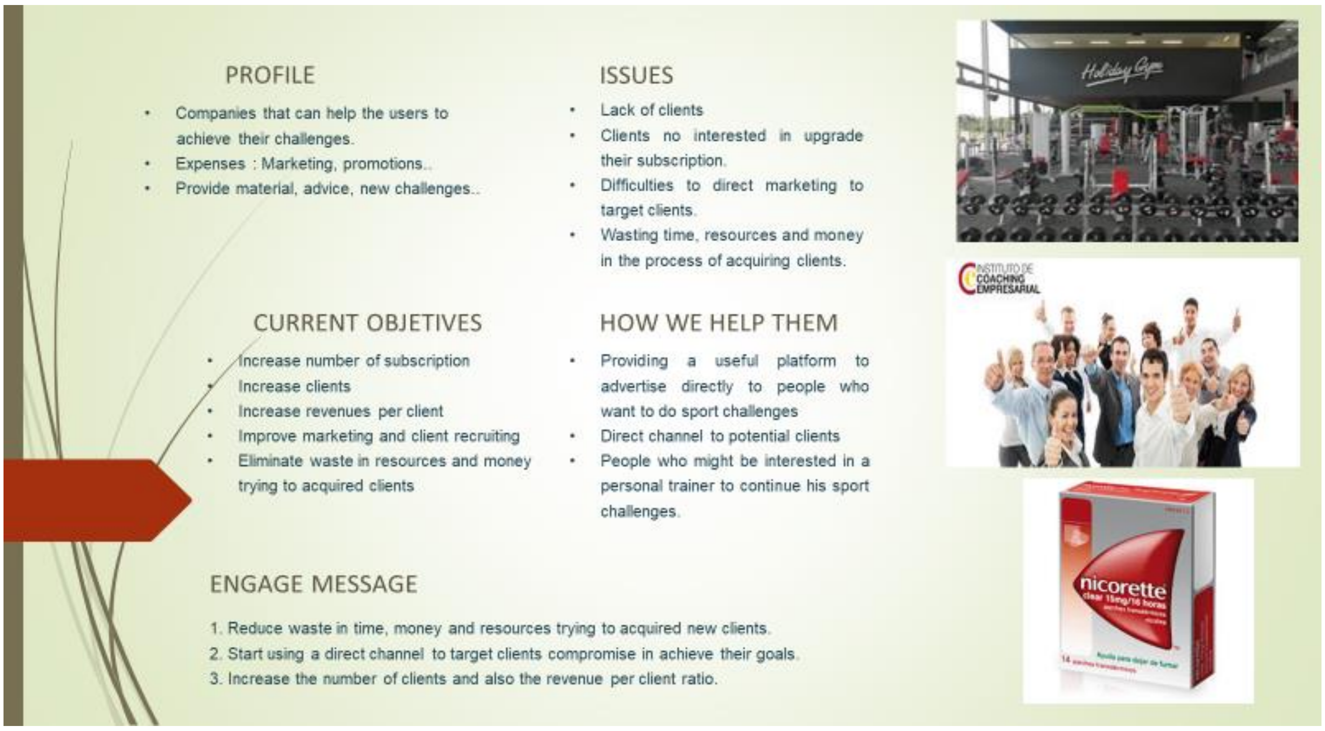
A day in Antonio's life

Week:
Works 9am-9pm
Arrives home late
Rarely sees his family
TV or reads at nigths

Weekend:
Sleeps and recovers to start the week.
Sometimes, he works
Goes to events which he really does not want to go

Figure 7.Workers Customer Persona

b) Customer Persona Publishing Company



PROFILE

- Companies that can help the users to achieve their challenges.
- Expenses : Marketing, promotions..
- Provide material, advice, new challenges..

ISSUES

- Lack of clients
- Clients no interested in upgrade their subscription.
- Difficulties to direct marketing to target clients.
- Wasting time, resources and money in the process of acquiring clients.

CURRENT OBJETIVES

- Increase number of subscription
- Increase clients
- Increase revenues per client
- Improve marketing and client recruiting
- Eliminate waste in resources and money trying to acquired clients

HOW WE HELP THEM

- Providing a useful platform to advertise directly to people who want to do sport challenges
- Direct channel to potential clients
- People who might be interested in a personal trainer to continue his sport challenges.

ENGAGE MESSAGE

1. Reduce waste in time, money and resources trying to acquired new clients.
2. Start using a direct channel to target clients compromise in achieve their goals.
3. Increase the number of clients and also the revenue per client ratio.

Figure 8. Publishing Company Customer Persona

2.3.1.2 CUSTOMER PROFILE

After personalizing the core clients of the company, the second step in relation with clients is to identify most specifically their pains, gains and what they are trying to do in their lives.

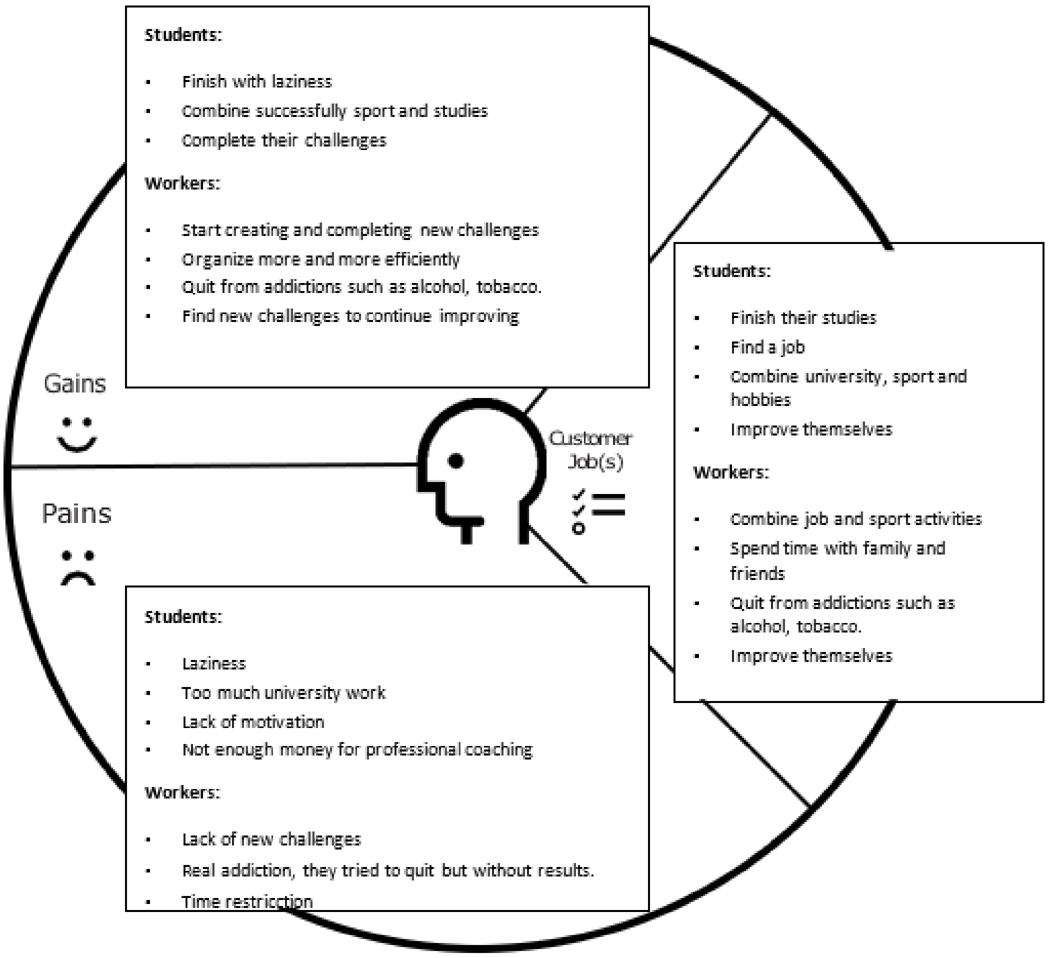
The customer profile is the ideal tool to do it. With this tool it is going to be defined three things for each client segment.

1) Customer Jobs: what customer are trying to get done in their work and in their lives. Jobs are the tasks they are trying to perform, the problems they are trying to solve or the needs they are trying to satisfy.

2) Customer Pain: In this section it is described the bad outcomes, risks and obstacles related to the customer jobs.

3) Customer Gain: Describe the more expected benefits the customers are seeking.

Customer Profile for Students and Workers



Students:

- Finish with laziness
- Combine successfully sport and studies
- Complete their challenges

Workers:

- Start creating and completing new challenges
- Organize more and more efficiently
- Quit from addictions such as alcohol, tobacco.
- Find new challenges to continue improving

Students:

- Finish their studies
- Find a job
- Combine university, sport and hobbies
- Improve themselves

Workers:

- Combine job and sport activities
- Spend time with family and friends
- Quit from addictions such as alcohol, tobacco.
- Improve themselves

Students:

- Laziness
- Too much university work
- Lack of motivation
- Not enough money for professional coaching

Workers:

- Lack of new challenges
- Real addiction, they tried to quit but without results.
- Time restricction

Figure 9. Customer Profile 1

Customer Profile for Advertisement Companies

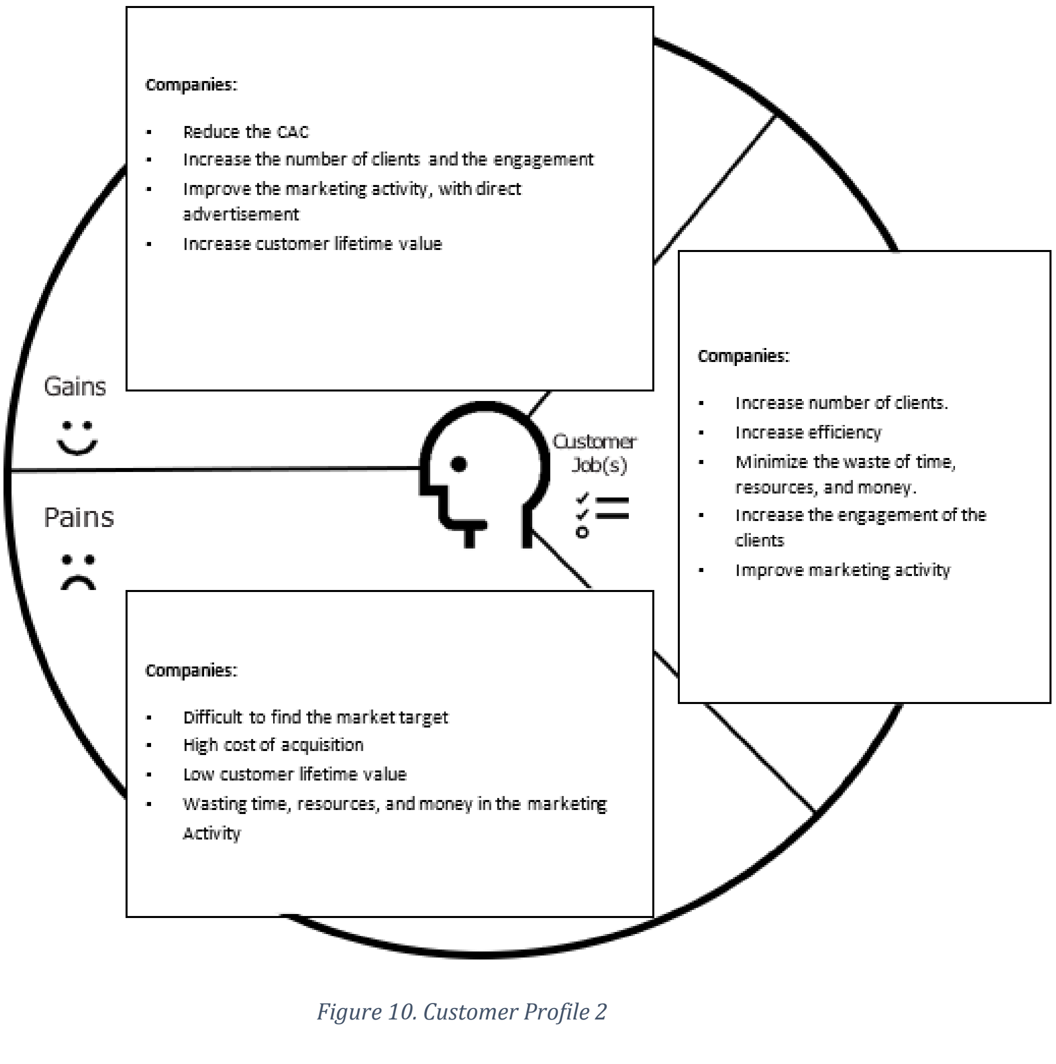

Companies:
• Reduce the CAC
• Increase the number of clients and the engagement
• Improve the marketing activity, with direct advertisement
• Increase customer lifetime value
Gains
Pains
Customer Job(s)
Companies:
• Increase number of clients.
• Increase efficiency
• Minimize the waste of time, resources, and money.
• Increase the engagement of the clients
• Improve marketing activity
Companies:
• Difficult to find the market target
• High cost of acquisition
• Low customer lifetime value
• Wasting time, resources, and money in the marketing Activity

Figure 10. Customer Profile 2

2.2.3 CHANNELS

It is not only about how the customer is reached and how the customer contact the company. In the channel section the company state where the services are provided, the location. To specify it the company should look to the stages of buying by the customer which are:

- Awareness of the product: which platform announces the product or make the customer get notice of the product/service offer by the company.
- Purchase: which platform location is used to purchase.
- Delivery: channel used to give the product service to the client.
- After sales: client attention and platform use to post-sales relationship with the client.

The channels where the service arrived to the potential and existing channels for Getyourchallenge are mainly three:

- **Social Media**: Mainly, the platform in which the bot is developed, Telegram, but also other social media apps in which the product can be advertise in order to reach clients.
- **Social footprint:** Due to the fact is thought to help people to reach their goals by helping them, the people will start to talk about the service to their friends, family and people related to them. This represent a source to reach potential clients and by the increase in the number of clients and therefore and increase in the value provided by the company
- **Advertisement**: The last channel to reach clients or clients to reach the company is to use the advertisement complemented with the video in which the product and company is explained and also the website.

2.2.4 CUSTOMER RELATIONSHIP

In this block the company design the strategy of how is going to be the relation customer-company for each segment. Each segment has different needs and therefore requires a different treatment and how the company contact the client and approach to them.

For Getyourchallenge the interaction with the user can take place in different ways. However, it is obvious that the most important one is the interaction using the service in this case the Telegram Bot. In particular, this relationship is so important because the service is different and personalized due to the capacity of take information from the database. In this way, the bot is able to recognize if the user is new or not, if he/she has already some judges and how and also they can see the status of their existing and past challenges that have been created before.

Moreover, the CRM created to automatize the send of emails to the user and judge when a new challenge is created is the second way of interaction between user and potential user with the product. Finally, the website to launch the product is the last resort used to interact with potential clients.

Thirdly, the company design how they expect to earn money from the business. To do so they have to structure the cash inflows sources and the cost structure, the financial planning, of the company for each product/service that compound the value proposition.

2.2.5 REVENUES & COST STRUCTURE. FINANCIAL PLANNING

What determines if a company is successful or not is the profits that it is able to capture by providing their services. The profit is a relation between the revenues streams of the company by providing services to the clients and the cost that the company incur in order to provide these services.

In this section, firstly, it is going to be explained the revenues streams of the company and which type of monetization of the services has been decided to use. After that it will explain the cost related. Finally, it will be put all together to develop a financial planning for the company in the foreseeable future.

2.2.5.1 REVENUES

This block it is used to design how the company is going to sell the product or service and how they are going to monetize it. There are several ways of earn money from a business, margin from the sell of products, commissions, advertisement, sell the time of your human capital if you offer services. The company has to think about what sources of cash inflows can and will increase the value of the company and how to earn money from each source, which collection method to use.

The revenues streams are the sources of revenues for which a company earn money from the users of their services in exchange of the services provided to them.

The startup has a similar value proposition as a mobile app, in spite of the fact that in this case the service is not an app but a telegram bot, therefore its monetization options are alike.

The monetization options that are available for this type of services are:

- **Pay to download:** The user has to pay to download the app before use it. However, once downloaded, the user has completely access for everything in the app.
- **Premium:** The users pay a monthly subscription in order to use the services provided by the app with a premium service which usually includes that it avoid adds in the app and special services more personalized.
- **Freemium:** The users have limited access to the services provided and usually in their interaction in the app they face advertisement of companies related to the interest of the users.
- **Commissions:** The company charge a % of the transactions that the users make in the app.

Once defined the multiple monetization options that are available for a company with similar services as Getyourchallenge.

As it is stated in previous sections, the service provided by the company is a Telegram bot. Although it can be thought that that the first two monetization options are impossible to follow, due to the fact that the bot resides in Telegram an external platform. There are ways to implement this monetization options that are created in Telegram in order to solve this initial problem. The third type of monetization, however, is not possible to implement because while the user is conversating with the bot, no external and indirect adds appear when the user is using the service.

In order to decide which monetization option use for the business the company developed one table where it is shown the advantages and disadvantages for each type, and a clients and revenues forecast with each option.

	ADVANTAGES	**DISADVANTAGES**
PAY TO DOWNLOAD	Higher revenues per user No need to worry for conversion Do not need too many users to be profitable	Higher barrier to entry for the users→ a smaller number of users "*Only 20% of this apps have more than 100 downloads*" The existing user do not pay for the updates in the app
PREMIUM	Constant revenue stream Better forecast of cash flows	You have to worry to capture clients and also to retain them Higher needs in user personal attention
COMMISIONS	Easy to increase number of users, low barriers to entry High engagement, increase conversion High number of users, quick spread Brand knowledge, market positioning	Low conversion "*For example, Dropbox 4%, the highest is Spotify with 27%*"

Table 1. Monetization Options

After analyzing both tools, the commissions monetization option is the most suitable for this type of business and product. However, the next step is to decide for which transaction apply a % of commissions. There are mainly two types of commissions charges options:

- **Commission for each challenge created and performed by the user:** In this option, the user bet an amount in order to pursue the challenge created. This amount will be transfer to a social project or ONG decided by him if he/she fails in doing it. But, if he/she perform the challenge the amount less a transaction fee (the income for the company) will be returned to him/her.
- **Commission for each challenge created and not performed by the user:** In this option, if the user performs the challenge the amount bet will be returned to him/her completely. However, if they fail to do it one % of the amount will be revenue for the company as a fee, and the rest will be sent to a social project or ONG.

Between these two options, the most suitable with the context of the company and the value proposition offer by the company is the second one. This is because, the service that the company offer is to help the users to do their challenges. Therefore, if the company charges a fee when the users perform the challenge this will be against the concept the company is created around.

Finally, some features of the other type of monetization are compatible with this monetization option. In particular, as the challenges will be related to other companies business a direct advertisement revenue stream can be implemented to increase the revenues of the company and as the same time improve the services provides to the users because the adds will be personalized for each user challenges, in order to improve information to the users and by this increase the probability of perform the challenge and increase the user commitment to it.

Thus, in summary the revenues stream for the startup can be split in two different ways:

1. Revenues as a % of the amount bet by the user to perform a challenge, only when the user is not able to perform the challenge.

2. Revenue related to the adds send to the users of the bot, this type of revenue can by priced in different parts:

a. Fixed fee for each adds sent

b. Variables fee for each user that clic in the add that have received

c. Variable or fixed fee for each user that because of the add has pay for a service of the advertisement company.

2.2.5.2 COST STRUCTURE

On the other hand, the company has to structure the cost sources related to each product or service and take this as a based to the pricing which is so related to the Revenues block. By analyzing the competition environment, the competitive advantages, the market elasticity and the cost structure, the company will develop a pricing model and stablish margins for each category that form the value proposition

Finally, the company has to answer how they are going to perform all this plan and what they are going to need for doing it. The last three blocks of the business model canvas explain it, they are: key activities, key resources and key partners.

The cost structure of the company as always can be split in two types of cost. Firstly, the fixed cost, those that you have to pay for them whatever the revenues are. On the other hand, the variable cost those that you have to paid related to the labor or resources use to sell an additional item.

The cost structure and drivers of cost are so related to the key resources and key activities that will be explained in further sections, however, to better understand the cost structure are going to be listed below:

Key activities: Software development and Marketing.

Key resources: Human resources, serves and other applications used in the development of the product.

Once the key resources and activities are listed, the cost structure can be explained in more detail.

As it is stated in the first chapter, the product consists on a telegram bot will be automatically maintain conversations with the user and create challenges that the users want to perform. Therefore, this type of business can be defined as labor intensive before the start of the business and no capital intensive because it is not necessary to invest huge amount of money in order to build an infrastructure or equipment.

The main costs of the company are the cost related to the development of the product (programming the product) and the marketing (launch the product and promotion it), both activities are labor intensive because both needs human resources to be done and therefore the most part of the cost structure of the company will be the salary of these employees.

Being a technological tool labor intensive in the beginning only gives the company the advantage of the scalability. This phenomenon is only possible if the engagement between users and the product is high enough to increase the number of transactions that are done because of the bot conversation. This engagement in the startup world is called the network effect. The network effect lies in the effect that when the number of participants using a good or services, the value of the good or services also increase. In this case, the higher number of challenges created talking with the bot and storage in the database makes easier to other people to find new challenges to pursue in their lives and also see that other people can do this challenges motivate the others users to increase their willingness to do them, which is the main objective of the service provided.

This network effect is only possible if the product gain knowledge around the community and that makes the marketing activity one of the most important part of the business. The main resources use to develop this activity are human resources and therefore the cost related to the activity are mostly the salaries of the employees who do this job in the company. This type of cost in a startup is better defined as Cost of Acquisition of a Client (CAC), this metric is so useful to know the amount of money that the company has to invest in marketing in order to gain new users. Finally the total cost of marketing will be CAC x (number of new users).

Moreover, as it is mentioned before, all the information provided by the users to the bot are storage in a database, as well as the bot which is also storage in a server in order to function at any time. These two storage services are also main cost related to the operation of the company.

Apart from that, the company will need also legal services and other administrative employees. Finally, the use of API's related to other apps like Airtable, Telegram..will be in the future a cost that the company has to assume in order to continue the operation using this platforms.

The startup has a similar value proposition as a mobile app, in spite of the fact that in this case the service is not an app but a telegram bot, therefore its monetization options are alike.

In short, the cost structure of the company will be:

Fixed Cost	Variable Cost
Salaries	Storage services
-Programming	-Airtable (database)
-Marketing	-Servers
- Client attention	Legal services
	Marketing campaigns
	External API usage

In the next section, Financial Forecast, all the stream revenues and cost structure expected for the future of the company will be detailed.

2.2.5.3 FINANCIAL PLANNING

After defining the cost structure and the revenue source of the company, the forecast for the following years is explained in this section.

In this section it is going to be exposed:

1. The three-year forecast of the company revenues, which it is based on the forecast of the expected users and how the conversion of them is going to be.
2. The three-year forecast of the expenses of the company which is split mainly in three

categories: operating expenses, salaries and marketing expenses.

3. The three-year forecast of the cash flows that the company is going to be accumulating month by month which reflect the profit of the company in the short-term and it can be used as a model to infer the potential of the company business.

REVENUES

In order to forecast the revenues of the company it has been used some parameter as guide to improve the forecast. As it is stated in the monetization section, the company revenues are a portion of the bets in which the challenge had not been performed successfully.

The parameters that are adjusted in order to forecast more precisely the revenues are:

- Users
- % Increase in number users per month
- % Conversion
- % Increase/Decrease in conversion
- Number of transactions per user and month
- % Increase in the number of transactions
- Average bet
- % Revenue for each bet: The percentage of the bet that is not directed to the social organization or project elected by the user.

The revenue is therefore calculated following this pattern:

1. Number of users = Number of users previous month * (1 + % increase)
2. % Conversion = %Conversion previous month * (1+ % increase/decrease)
3. # transactions/user = # transactions/user * (1 + % increase/decrease)

4. Transaction per month = Number of users * % Conversion * # transactions

Revenues= Transaction per month * Avg Bet * % Revenue per bet

Once it has been explained how the revenues is calculated, then how the assumptions of these items that are needed to calculate the revenues is stated.

Firstly, the most important driver of the revenues is the number of users. In order to determine this parameter accurate the company has created a potential market for each segments: Spain and EEUU.

MARKET POTENTIAL

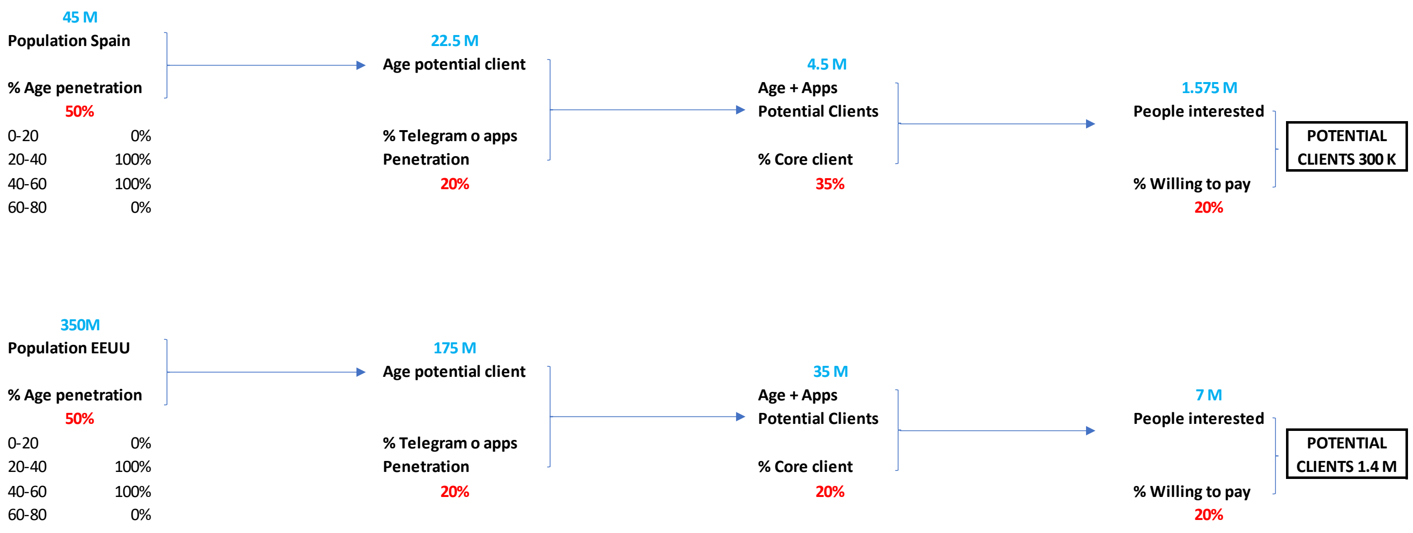


Figure 11. Market Potential Analysis

This chart serves a base in order to compare is the number of users that the company is assuming in the financial planning has sense or not, comparing it with the potential market presented above.

The expected number of users calculated as a driver of the forecast revenue model is shown in the following chart.

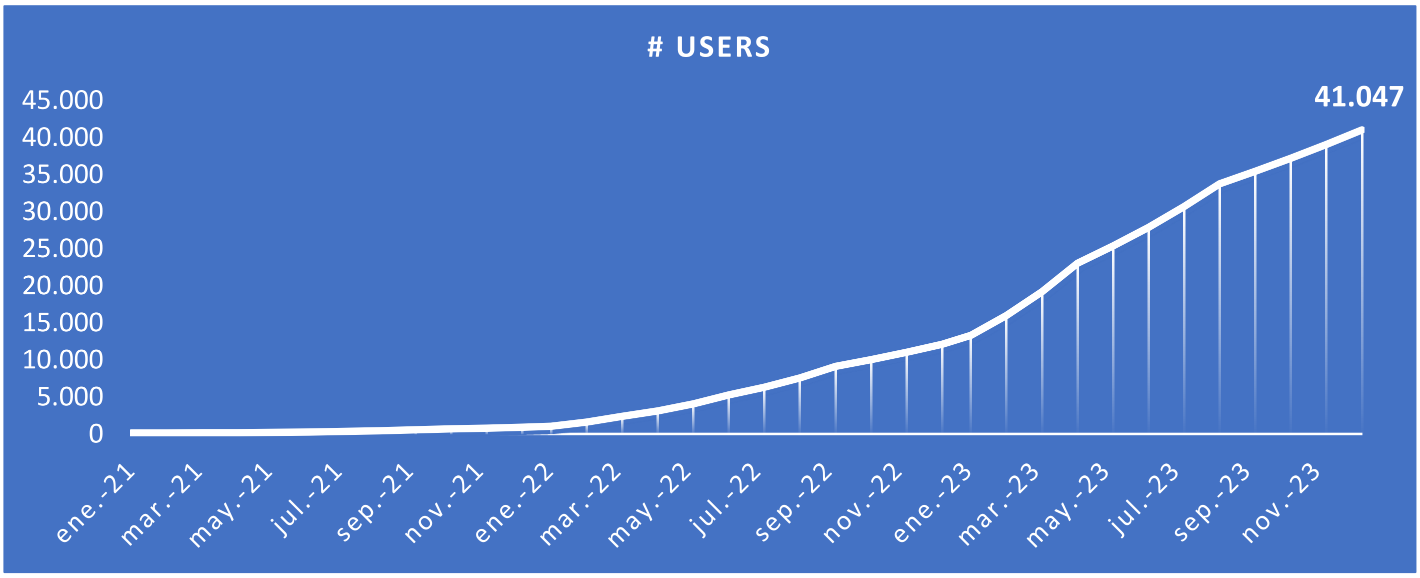


Figure 12. Expected number of users

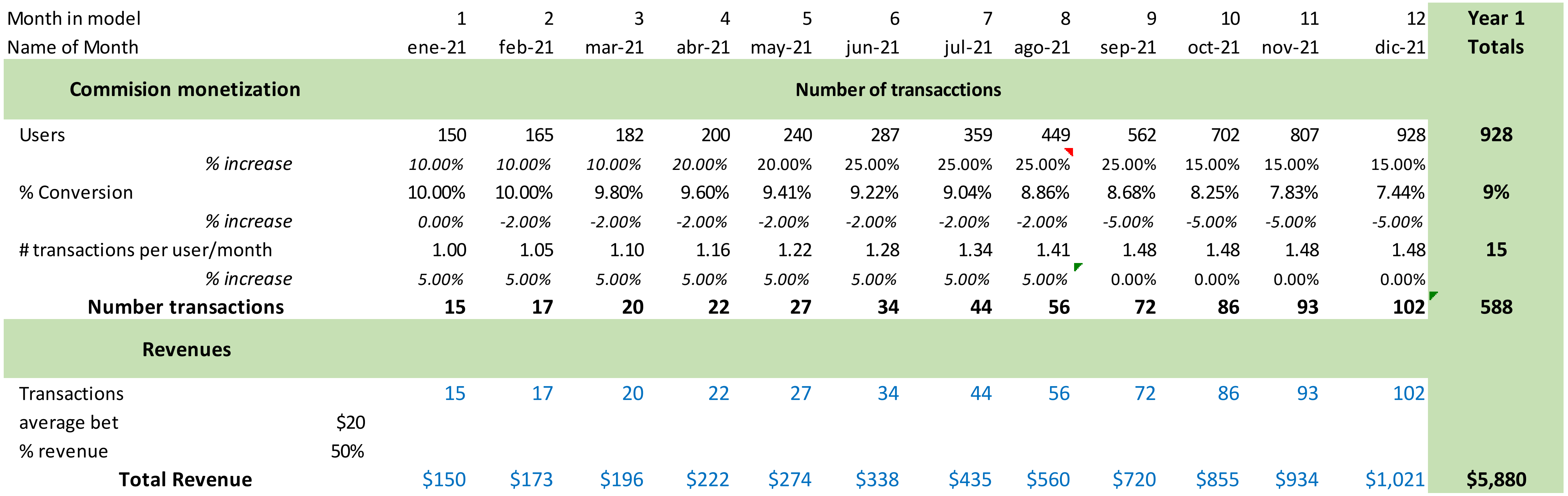
MODEL PARAMETERS

Continuing with the estimation of the parameters, the period forecasted has been split in three parts. In each part the parameters assumptions changes, however some of them follow a trend. The most important one is the relation with the number of users and the rate of conversion. The more user that product has, the less conversion rate because it is expected that more challenges created will not be judge and therefore the conversion rate will be lower. The engagement also explain this phenomenon because when you have less clients, this clients are more loyalty to you and therefore the conversion ratios are higher and are decreasing when the number of users increase because they are expected to have less engagement. Finally, the average bet and the % of the bet that is considered revenue are fixed and tested in three different scenarios that are show below. The conservative scenario, which is the most suitable for the forecast is the one use to present the forecast in the section.

The first one, during the first year in which the product is launched in Spain, is characterized by a medium rate increase in the number of users (10-25% per month) it is sustained because at the beginning there will be few users and therefore by a little bit on marketing and creating engagement with the users that start using the app, it will not be difficult to reach each month at least one additional users from five existing users of the product (20% increase).But at the beginning the rate is set at 10% to be more conservative and because the product will not be as popular as it should be later.

Moreover, it is assumed there will be around 150 users by the time the product is going to be launch (January-2021), this is because during the validation phase the company has captured 52 users in about a month and they are assume to remain and still increasing in next 4 months. Additionally, the conversion rate will decrease but as the number of users remain low, it only will decrease by 2% per month which is not so much. On the other hand, the number of transactions per user and month in this period it is expected to be increasing because for most of the users it represent a new tool and the client try to use it in each activity in which it can be useful.

A summary of this phase is show below:



Month in model		1	2	3	4	5	6	7	8	9	10	11	12	**Year 1**
Name of Month		ene-21	feb-21	mar-21	abr-21	may-21	jun-21	jul-21	ago-21	sep-21	oct-21	nov-21	dic-21	**Totals**
Commision monetization							**Number of transacctions**							
Users		150	165	182	200	240	287	359	449	562	702	807	928	**928**
% increase		*10.00%*	*10.00%*	*10.00%*	*20.00%*	20.00%	25.00%	25.00%	25.00%	25.00%	15.00%	15.00%	15.00%	
% Conversion		10.00%	10.00%	9.80%	9.60%	9.41%	9.22%	9.04%	8.86%	8.68%	8.25%	7.83%	7.44%	**9%**
% increase		*0.00%*	*-2.00%*	*-2.00%*	*-2.00%*	*-2.00%*	*-2.00%*	*-2.00%*	*-2.00%*	*-5.00%*	*-5.00%*	*-5.00%*	*-5.00%*	
# transactions per user/month		1.00	1.05	1.10	1.16	1.22	1.28	1.34	1.41	1.48	1.48	1.48	1.48	**15**
% increase		*5.00%*	*5.00%*	*5.00%*	*5.00%*	*5.00%*	*5.00%*	*5.00%*	*5.00%*	0.00%	0.00%	0.00%	0.00%	
Number transactions		**15**	**17**	**20**	**22**	**27**	**34**	**44**	**56**	**72**	**86**	**93**	**102**	**588**
Revenues														
Transactions		15	17	20	22	27	34	44	56	72	86	93	102	
average bet	$20													
% revenue	50%													
Total Revenue		$150	$173	$196	$222	$274	$338	$435	$560	$720	$855	$934	$1,021	**$5,880**

Figure 13. Revenue forecast year 1

During the second year and expecting to have around 1,000 users, the company will start an aggressive marketing campaign in order to escalate the product use. For this purpose, the company will develop this marketing campaign in Spain and also, they will enter into the American market in order to impulse the number of users in a bigger market. In this phase it is expected in the beginning a higher increase in the number of users with the rate around 50% increase each month, for the first two month because the product will be recognize in a bigger circle, the base client is not too big, and the marketing strategy will be more powerful when the product is not popular yet. However, as the time pass the rate will stabilize at a rate similar to the first phase (15-20%), this is because the marketing campaign will show an increase at first but when the company reach a higher number of clients it is more difficult to capture the same rate of new clients per month and it will stabilize the rate for the rest of the year, maintaining the marketing campaign active. At the same time, the conversion rate as well as in the first phase is going to be decreasing at a rate of -5%, which is higher than in the previous phase. This has sense because the increase in the number of users is higher too. Moreover, the number os transaction per user and month is stabilize because the existing users will decrease the use of the app after the first months and a huge increase in the number of users makes difficult to maintain a high level of engagement with all of them. Probably, some of the new users will use a lot the app at the beginning but this will be balance with a group of them that will only use it for a few challenges.

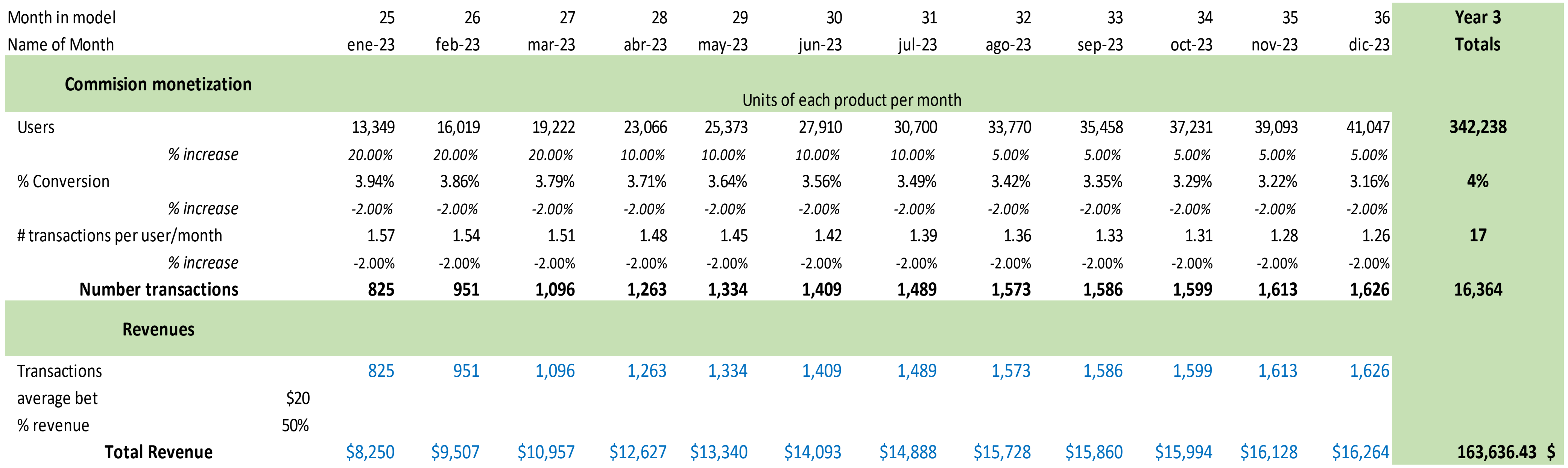
The results of the forecast are presented in the chart below.

Month in model		13	14	15	16	17	18	19	20	21	22	23	24	**Year 2**
Name of Month		ene-22	feb-22	mar-22	abr-22	may-22	jun-22	jul-22	ago-22	sep-22	oct-22	nov-22	dic-22	**Totals**
Commision monetization		Units of each product per month												
Users		1,067	1,601	2,402	3,122	4,059	5,277	6,332	7,598	9,118	10,030	11,033	12,136	**73,776**
% increase		*50.00%*	*50.00%*	*30.00%*	*30.00%*	30.00%	20.00%	20.00%	20.00%	10.00%	10.00%	10.00%	10.00%	
% Conversion		7.07%	6.72%	6.38%	6.06%	5.76%	5.47%	5.20%	4.94%	4.69%	4.46%	4.23%	4.02%	**5%**
% increase		*-5.00%*	*-5.00%*	*-5.00%*	*-5.00%*	*-5.00%*	*-5.00%*	*-5.00%*	*-5.00%*	*-5.00%*	*-5.00%*	*-5.00%*	*-5.00%*	
# transactions per user/month		1.48	1.51	1.54	1.57	1.57	1.57	1.57	1.57	1.57	1.57	1.57	1.57	**19**
% increase		2.00%	2.00%	2.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
Number transactions		**112**	**162**	**236**	**297**	**367**	**453**	**516**	**588**	**671**	**701**	**732**	**765**	**5,599**
Revenues														
Transactions		112	162	236	297	367	453	516	588	671	701	732	765	
average bet	$20													
% revenue	50%													
Total Revenue		$1,115	$1,621	$2,356	$2,968	$3,665	$4,527	$5,160	$5,883	$6,706	$7,008	$7,323	$7,653	**55,985.81 $**

Figure 14. Revenue Forecast Year 2

Finally, in the last year of the forecast starting with a base of around 13,000 users by January 2023, the increase in the number of users will decrease and start to stabilize. After launching the product two years ago and having a huge marketing campaign during the last year in this last year of the forecast is when the company will try to start collecting the profits. In this year the expenses in marketing will decrease and the company will focus in a marketing strategy of retargeting in order to capture people that already know the product and maybe have been an user. On the other hand, and having a base of 13,000 users at the moment, the company will try to create engagement with them and still increasing the number of users by the promotion of these users to their family and friends. Because the increase in the number of users is lower than the previous year, the rate of conversion will still be decreasing but with a lower rate (-2%). The number of transactions will also decrease a this rate for the last year.

The result is represented in the chart below.



Month in model		25	26	27	28	29	30	31	32	33	34	35	36	**Year 3**
Name of Month		ene-23	feb-23	mar-23	abr-23	may-23	jun-23	jul-23	ago-23	sep-23	oct-23	nov-23	dic-23	**Totals**
Commision monetization							Units of each product per month							
Users		13,349	16,019	19,222	23,066	25,373	27,910	30,700	33,770	35,458	37,231	39,093	41,047	**342,238**
% increase		*20.00%*	*20.00%*	*20.00%*	*10.00%*	*10.00%*	*10.00%*	*10.00%*	*5.00%*	*5.00%*	*5.00%*	*5.00%*	*5.00%*	
% Conversion		3.94%	3.86%	3.79%	3.71%	3.64%	3.56%	3.49%	3.42%	3.35%	3.29%	3.22%	3.16%	**4%**
% increase		*-2.00%*	*-2.00%*	*-2.00%*	*-2.00%*	*-2.00%*	*-2.00%*	*-2.00%*	*-2.00%*	*-2.00%*	*-2.00%*	*-2.00%*	*-2.00%*	
# transactions per user/month		1.57	1.54	1.51	1.48	1.45	1.42	1.39	1.36	1.33	1.31	1.28	1.26	**17**
% increase		-2.00%	-2.00%	-2.00%	-2.00%	-2.00%	-2.00%	-2.00%	-2.00%	-2.00%	-2.00%	-2.00%	-2.00%	
Number transactions		**825**	**951**	**1,096**	**1,263**	**1,334**	**1,409**	**1,489**	**1,573**	**1,586**	**1,599**	**1,613**	**1,626**	**16,364**
Revenues														
Transactions		825	951	1,096	1,263	1,334	1,409	1,489	1,573	1,586	1,599	1,613	1,626	
average bet	$20													
% revenue	50%													
Total Revenue		$8,250	$9,507	$10,957	$12,627	$13,340	$14,093	$14,888	$15,728	$15,860	$15,994	$16,128	$16,264	**163,636.43 $**

Figure 15. Revenue Forecast Year 3

SCENARIOS ANALYSIS

In order to test the impact of the two most determinant variables in the model, the %increase in the number of user and the %conversion it has been developed a scenario analysis in which these two inputs has been changed.

In this scenario analysis two new scenarios have been added to the one exposed above which represent the expected scenario. These two scenarios are the optimistic one and on the other hand the pessimistic.

The scenarios tested are the following.

Scenario 1: Expected

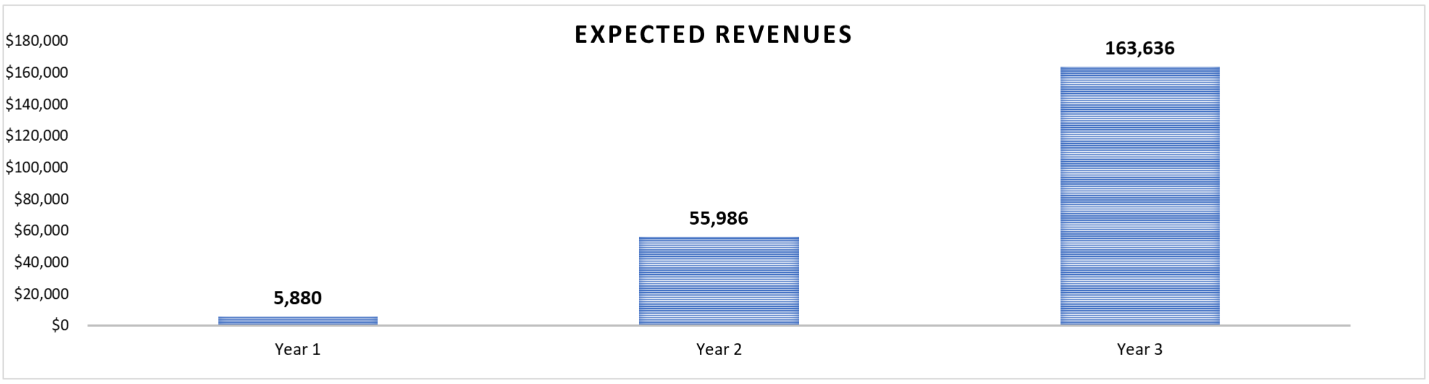


Figure 16. Revenues expected scenario

A summary of the inputs used to create the scenario are presented below:

- **Initial users: 150**
- **% Conversion initial: 10%**

	Year 1	Year 2	Year 3
Final Users	928	12,136	41,047
Avg %user increase (monthly)	17.92%	24.17%	10.42%
Avg %Conversion (monthly)	9.01%	5.42%	3.54%

Table 2. Inputs expected scenario

Moreover, in the model there is a crucial input in the second year due to the aggressive marketing campaign that it is expected to be made. For that reason, for some months the expected % monthly user increase is set to 50% which could be seen to aggressive. Therefore, a small sensitivity analysis, downgrading the parameter -5% and -10%, has been performed to see the impact of the change in this assumption on the final profitability.

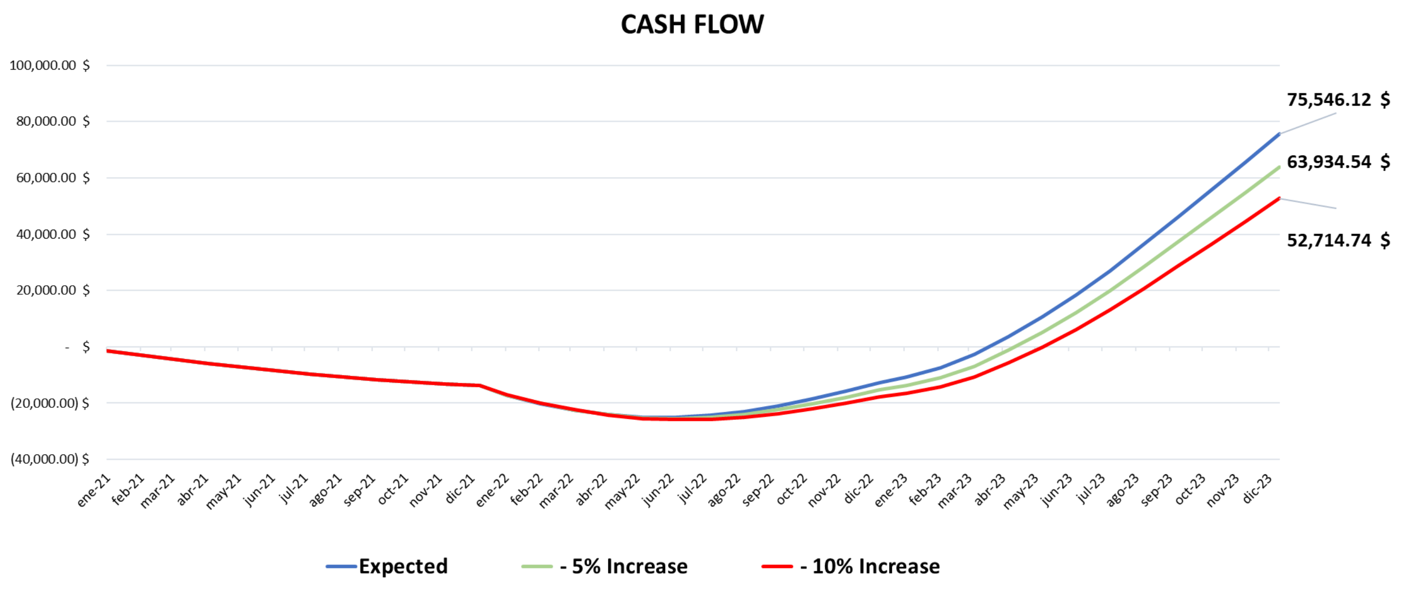


Figure 17. Profitabiity vs %users increase

From the chart can be obtained that from each 1% decrease in the input analyzed, the overall profitability will fell by 3.07%. This also show the potential impact of the changes in the % conversion in the final profitability.

Scenario 2: Optimistic

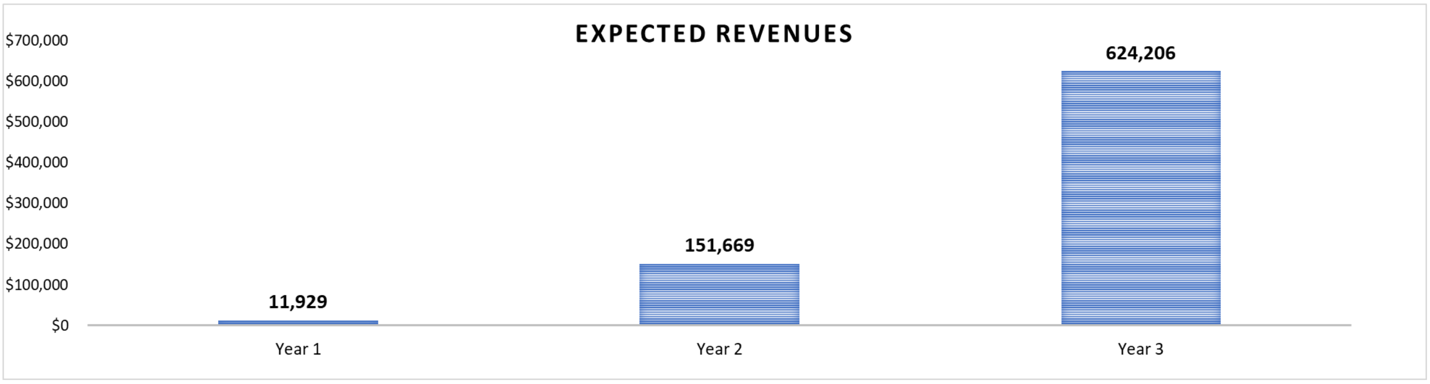


Figure 18. Revenues optimistic scenario

A summary of the inputs used to create the scenario are presented below. The % user increase is approximately 2.5% up in relation to the expected model as well as the %conversion each month.

- **Initial users: 200**
- **% Conversion initial: 12.5 %**

	Year 1	**Year 2**	**Year 3**
Final Users	1,559	25,938	114,734
Avg %user increase (monthly)	20.42%	26.67%	12.92%
Avg %Conversion (monthly)	11.52%	7.35%	5.31%

Table 3. Inputs optimistic scenario

Scenario 3: Pessimistic

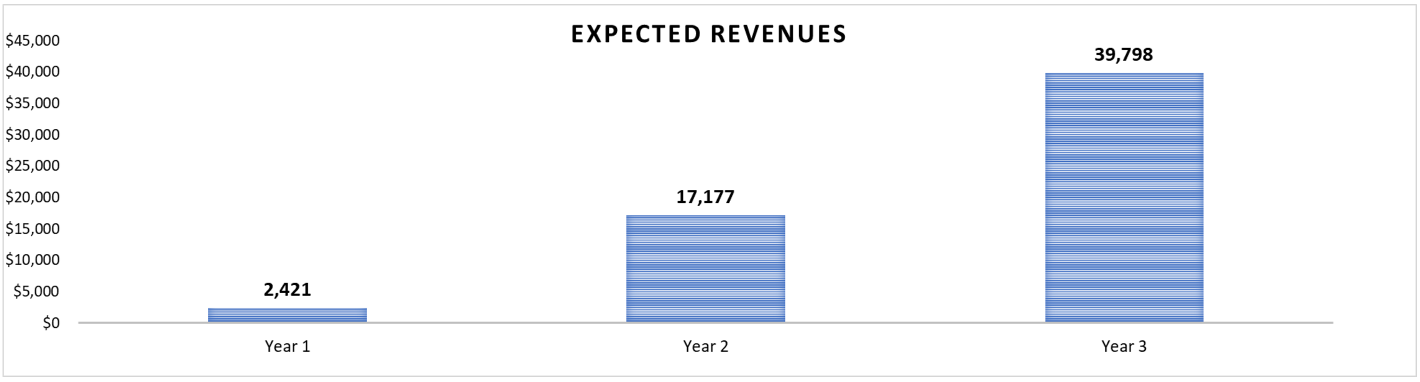


Figure 19. Revenues pessimistic scenario

A summary of the inputs used to create the scenario are presented below. The % user increase is approximately 2.5% down in relation to the expected model as well as the %conversion each month.

- **Initial users: 100**
- **% Conversion initial: 7.5%**

	Year 1	Year 2	Year 3
Final Users	489	4,997	12,845
Avg %user increase (monthly)	15.42%	21.67%	7.92%
Avg %Conversion (monthly)	6.61%	3.74%	2.43%

Table 4. Inputs pessimistic scenario

To validate the accuracy of the forecast it has been compare to similar projects that have took place before. It has been compared to the potential competitor which has been explained some section above, the website called 'Go fucking do it'. This website with a similar product to the one created by Getyourchallenge, reach 30,000 users in a month with a conversion rate of 1% (people who visit and create a challenge), and a 1% conversion of the people that create a challenge. Moreover, the average amount of the bets was 78.54$, and all the amount of the bet losses were considered revenue for the company. Therefore, in a month the revenues were:

Revenue= 30k users * 0.01 * 0.01 * 78.54 = 235.62 $ in the first month.

Comparing it with the 150$ forecasted for the first month it makes sense. However, this is not the key in the comparison.

As it can be seen, in month the website had 30,000 visitors which is the forecasted number of users for Getyouchallenge at the beginning of the first year. From this fact the company can infer two aspect:

- First: as a website the potential clients does not have to download the app before to try it which represent a lower barrier to entry. But for this fact, the conversion rate (visitor-user) is so low around 1%, that it is expected to be a lot higher in the business model of Getyourchallenge.
- Secondly, this high number of visitor in the first month validate the forecast of users for Getyouchallenge because it seems that with a powerful marketing campaign and entering in a big market as the American (is the market where the competitor which is being compared launched the website) makes possible to capture around 90,000 users in the first three years.

COSTS

The costs forecast is so related to the revenues model because the three cost are dependent of the number of users of the product at the beginning, but when some scale is reached, the expenses start to be fixed while the revenues continue increasing. This fact represents the **scalability of the business** which is the main advantage of the company.

The cost forecast for the first three years is presented below and after that it is explained the rest of the cost's sources.

KEY HUMAN RESOURCES FORECAST						
	2021		2022		2023	
	Full time	Part time	Full time	Part time	Full time	Part time
CEO	0	1	1	0	1	0
Software	0	0	0	0	0	1
Marketing Technician	0	0	0	1	0	1
Administrative		0	0	0	0	0
Total	**0.5**		**1.5**		**2**	
Expense	**15,000.00 $**		**37,500.00 $**		**45,000.00 $**	
% Revenues	**255%**		**67%**		**27%**	
OPERATING EXPENSES						
	2021		2022		2023	
External API usage (Airtable)	240.00 $		240.00 $		600.00 $	
Servers	3,000.00 $		3,000.00 $		3,000.00 $	
CRM	523.92 $		523.92 $		523.92 $	
Website	102.00 $		102.00 $		294.00 $	
Expense	**3,865.92 $**		**3,865.92 $**		**4,417.92 $**	
% Revenues	**66%**		**7%**		**3%**	
MARKETING EXPENSES						
	2021		2022		2023	
Instagram Advertising	835.51 $		13,449.48 $		26,021.06 $	
NEW USERS	928.26		11,207.48		21,683	
CAC	0.90 $		1.20 $		1.20 $	
Expense	**835.51 $**		**13,449.48 $**		**26,021.06 $**	
% Revenues	**14%**		**24%**		**16%**	
TOTAL EXPENSES	**19,701.43 $**		**54,815.40 $**		**75,438.98 $**	

Table 5. Cost forecast

In the chart it is presented the cost split by the nature of each one.

Firstly, the human capital which is necessary to carry through the business is not too much and will not change a lot in the life of the company.

In the first two years, with the product designed and working only by the time of the CEO will be sufficient to carry out the activities of the company. However, as it was mentioned in the revenue section, during the second year the company will launch an aggressive marketing campaign and for that purpose a marketing technician will be hire part time for the second and also in the third year.

Finally in the last year of the forecast, a software technician will be hired in order to update the product and develop new function to maintain the community of users motivate to use the product and also try to increase the average revenue from user extending the value proposition.

The salaries used to perform the forecast are shown in the next graph.

SALARIES	
STAFF TYPE	ANNUAL SALARY
CEO	$30,000
Software	$15,000
Marketing Technician	$15,000
Administrative	$20,000

Table 6. . Human capital salaries

On the other hand, the operating expenses increase with the number of transactions until its reach more than 2,000 challenges created per month. Once a certain level of transactions is reached the operating expenses are fixed and therefore the business will start to benefit from the economies of scale that the type of business has.

Finally, the cost of marketing which is related with the increase in the number of users. In order to estimate the marketing cost with accuracy the company has estimate firstly a metric that is so useful in startups the CAC (Cost of acquisition of new clients). The source to capture users have been stablish to be Instagram because is the source with more information to evaluate the CAC, however others social media platform such as Youtube or Facebook have a similar CAC and will also be used in order to capture new clients. The CAC have been split in two parts because the cost of advertising and the rate of capture is different by region based in the historic data. As it was mentioned in the revenue section, the marketing campaign will be directed firstly to the Spanish market and after that to the American market.

Thus, the CAC have been calculated independently for these two regions:

INSTAGRAM CAC	SPAIN	USA
# visitors	15,000,000.00	132,000,000.00
clic conversion ratio	3%	3%
# users	149,985.00	792,000.00
CPC	0.30 $	0.30 $
TOTAL COSTS	135,000.00 $	1,188,000.00 $
CAC	**0.90 $**	**1.5 $**

Table 7. Calculation of the CAC per region

Starting with the number of users in each country, the clic conversion ratio of 3% and if a third of these visitors who clic in the add are converted to users. The cost per clic multiply by the number of visitors and the clic conversion ratio results in the total costs. This total cost divided by the number of clients that can be converted makes the CAC.

To understand better the cost structure the percentage of the revenues that it types of cost represent per year, which is show in the following chart with more detail.

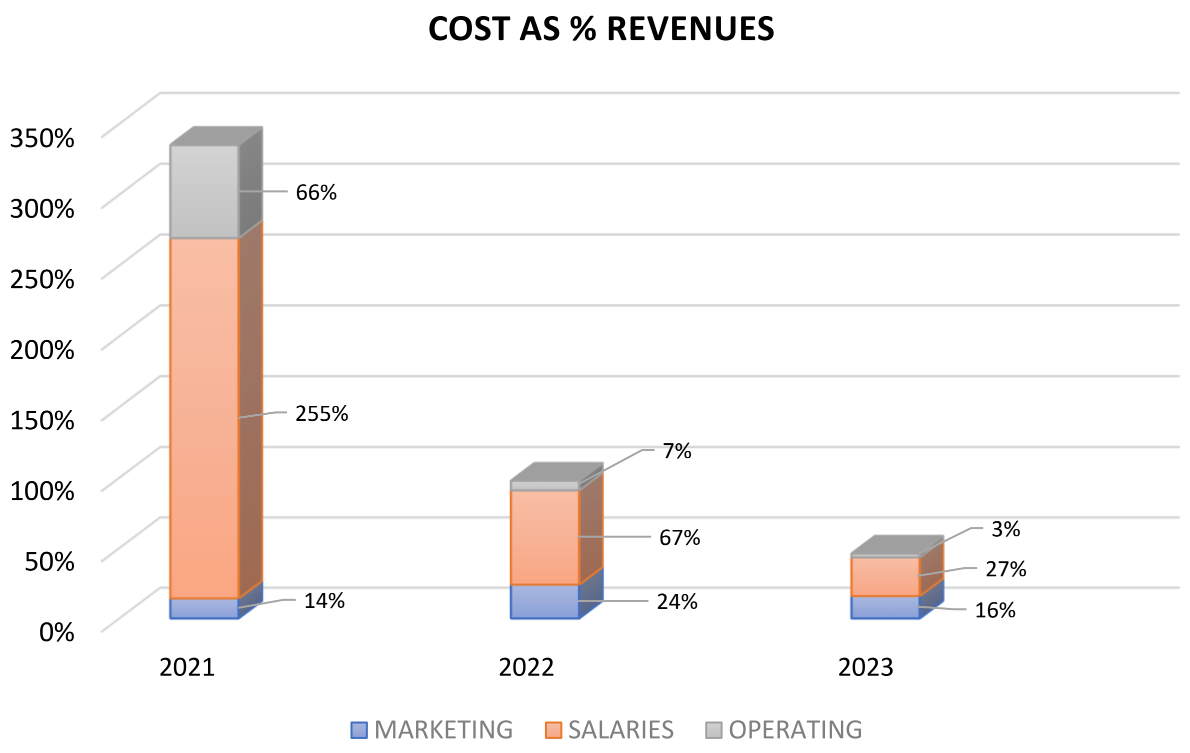


Figure 20. Cost as % Revenue per year

The chart shows that during the first year and due to the low expected revenues, the percentage of cost is higher than the 300% of the revenues. This trend is reverted in the subsequent years.

In spite of the fact that the cost of the salaries is the biggest one this percentage is decreasing each year due to the scalability of the business as well as the operating cost. However, the marketing cost increase when the company do a marketing campaign in order to increase the number of users or also to increase the monthly transaction per user.

PROFIT

Putting the revenue and cost forecast for the next three year the last step is to compute the expected profit and cash flow of the company for the next three years.

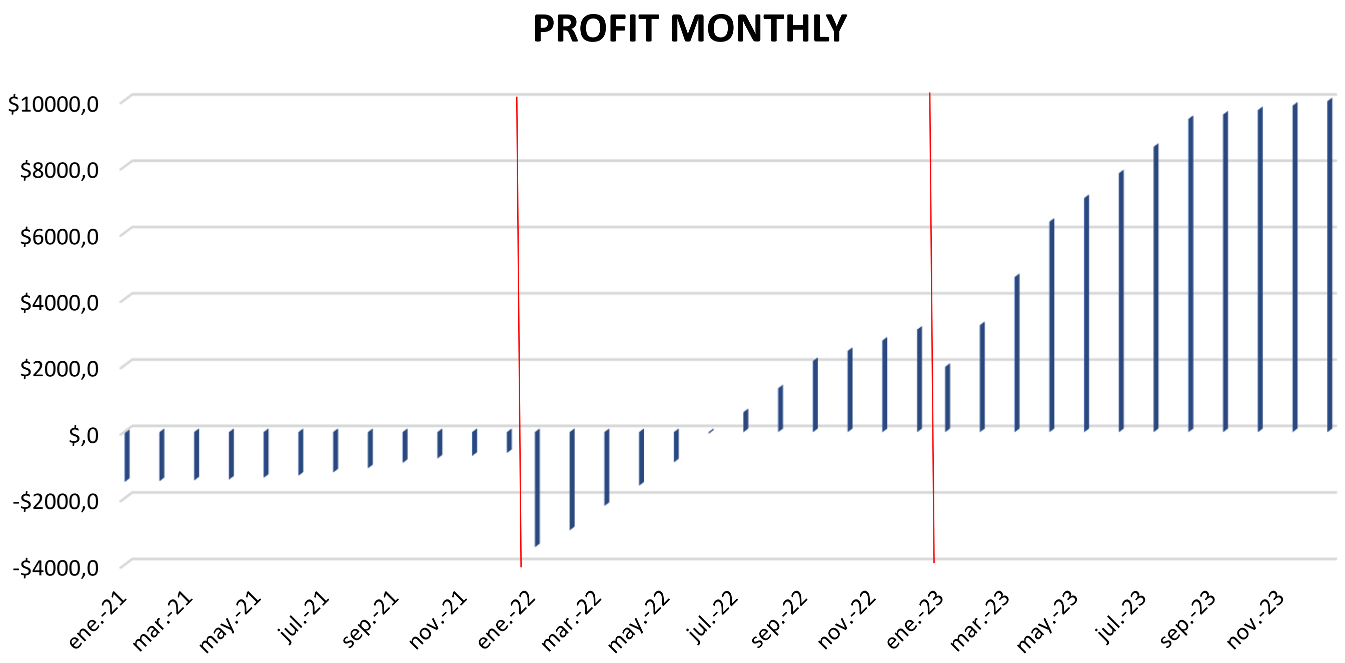


Figure 21. Monthly profits forecast

The chart shows three different zones which match with the three years forecasted in the model.

During the first stage or year, the company is losing money because the revenues are not high enough to cover the fixed expenses as the salaries.

In the second stage, the company start earning money in July and is increasing the profit each month after that. This is because when the fixed cost is cover the scalability of the company take part and show this increase.

Finally, in the last stage, with a high base of clients and spending less money in marketing than in the previous year the profits are enhanced increasing a higher rate at the beginning of the year and stabilizing the profit as the year is going on.

To have an overall picture of the profitability of the company, a cumulative profit chart is presented.

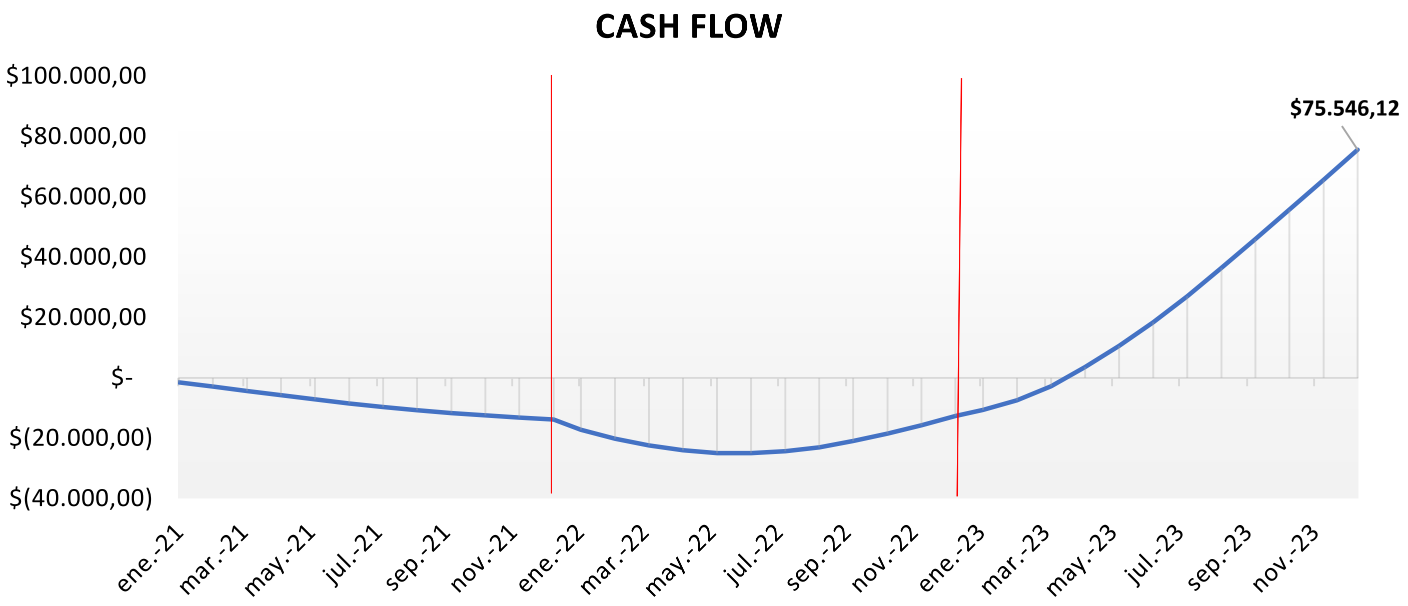


Figure 22. Expected profits

As it was expected seeing the previous chart, this one also presents three stages that match with the three years forecasted.

The main insight that can be taken from the chart is that the company is not generating returns until April '23, however, from that point on the returns curve is solid and expected to increase for the a few years before stabilizing. This will be leaded by the scalability of the business that was explained before.

The cash flow chart shows a trend that once the breakeven point of number of users is reached and the company start to make profits, the company have a high potential of increasing profits.

However, for this increase to be materialized the company needs to create the engagement necessary to retain the users in the long term and obtain a higher Lifetime value of the customer related to the cost of acquisition of each user.

SENSITIVITY ANALYSIS

In the three phases there are some assumptions that are fixed, the average bet and also the percentage of revenue from the amount loss in the bets. To determine these two parameters the company has create one sensitivity analysis in which changing these two parameters the company has different outcomes from the revenues. This two parameter are different in nature because the average bet is determine by the users of the product, whereas the % of revenue is determine by the company, which has two balance the increase in revenue with the value provided to the clients that want to contribute to social causes with the money they lose.

The output is the profit after the first three years of the business. For this analysis, the rest of parameter described at the beginning has hold constant.

From this analysis the company were trying to understand in which direction focus the time and resources, and at the same time see in a matter of seconds which levels of the inputs are necessary to be profitable.

SCENARIOS SENSITIVITY ANALYSIS					
PROFITS AFTER 3 YEARS					
% Revenue	**AVERAGE BET**				
	5.00 $	10.00 $	20.00 $	30.00 $	50.00 $
25%	(121,768.08 $)	(93,580.33 $)	(37,204.85 $)	19,170.63 $	131,921.60 $
50%	(93,580.33 $)	(37,204.85 $)	75,546.12 $	188,297.09 $	413,799.02 $
75%	(65,392.59 $)	19,170.63 $	188,297.09 $	357,423.54 $	695,676.44 $
90%	(48,479.95 $)	52,995.92 $	255,947.67 $	458,899.41 $	864,802.89 $

Table 8. Sensitivity analysis: Avg Bet & % Revenue

In the table there are shown the yellow cell represent the output under the expected scenario. The average bet and % of revenues used to pursue this analysis are set at the values from the expected scenarios.

From the analysis the following conclusions can be taken as insights:

- With an average bet **below 7$** the business is not profitable under neither the % of revenue studied, which means that at least the company has to focus in having an average bet higher than it.
- Having an average bet **above 27$** makes the business profitable for all levels of % of revenue studied, which means that obtaining this average bet the other parameter can be used to increased the profits whenever the number of users and transaction do not decrease due to that fact.
- Assuming an expecting average bet of **10$**, is necessary a % of revenues from the bet lost of at least **66%** to be profitable.
- With an expecting average bet of **20$**, is necessary a % of revenues from the bet lost of **33%** to be profitable.

In short, the company will need to focus on increase the average bet more than increasing the % of revenues of the lost bet. It will be easier to increase the average bet, encouraging people who want to perform important challenges and want to take them seriously. The % of revenue from lost bet is difficult to increase since is limit to 100% whereas the average bet not.

The second sensitivity analysis uses as input variables the initial expected conversion rate and the initial average transaction per user and month. For both parameter the monthly increase/decrease is set in the model using this initial value as input, therefore only by changing the initial value all the model change around these two parameters.

From this analysis the company were trying to understand the impact of the % conversion and the number of transactions per user monthly in the profit after the first three year of the business. By doing this the company will decide with more information if it is more important to promote an increase in the %conversion or, on the other hand, enhance the monthly number of transactions per user.

SENSITIVITY ANALYSIS					
PROFIT AFTER THREE YEARS					
Monthly Transaction per Client	**Initial %Conversion**				
	2.50%	5.00%	10.00%	15.00%	20.00%
0.50	(121,768.08 $)	(93,580.33 $)	(37,204.85 $)	19,170.63 $	75,546.12 $
1.00	(93,580.33 $)	(37,204.85 $)	75,546.12 $	188,297.09 $	301,048.05 $
1.25	(79,486.46 $)	(9,017.11 $)	131,921.60 $	272,860.31 $	413,799.02 $
1.50	(65,392.59 $)	19,170.63 $	188,297.09 $	357,423.54 $	526,549.99 $

Table 9. Sensitivity analysis: Initial %Conversion & Monthly transactions

In the table there are shown the yellow cell represent the output under the expected scenario. The average bet and % of revenues used to pursue this analysis are set at the values from the expected scenarios.

From the analysis the following conclusions can be taken as insights:

- Is needed **at least a 4.43% initial conversion rate** to have profits with the number of monthly transactions per user used in the analysis.
- Having an **initial conversion rate higher than 13.3%** the business is profitable for each of the number of monthly transactions per user used in the analysis.
- In the expected scenario with an initial conversion rate of 10%, the business has profit if **the number of monthly transactions per user** is higher than **0.66**.
- For each **1% increase in the initial conversion rate** the company **increase its profits by 22,550.19$** at year three under the expected scenario, **5,637.55$** under the pessimistic scenario and **84,563.23$** in the optimistic. **Thus, the better the scenario the most important the increase in the conversion rate to maximize the profits.**

On conclusion, the number of monthly transactions per user is a more difficult parameter to increase because some challenges cannot be pursued at the same time and take more than a month to complete most of the challenges. Therefore, as both parameters have the same weight in the calculation of the profits the company will have to focus in increasing the conversion rate. Knowing that achieving a conversion rate higher than 13.3% the business is profitable regardless for several monthly transactions higher than 0.25 per user, the company can maximize profits for all the values higher than this one.

2.2.6 KEY ACTIVITIES

Once defined the value proposition the company has to design how they are going to do this product or service. Meaning that they have to define the core activities of the company that will lead to the final product. It is not only focus in the production, it is related to all the activities that are needed to complete efficiently the value proposition stated for each client segment. For instance, activities related to client approach, manage teams inside the company apart from the production or execution of services

The key activities of the company are the principal work that has to be done in order to give to the clients the services that they are wondering.

In the case of Getyourchallenge the activities can be split in two types: marketing activities and operational activities (programming or technological development activities).

The second one, are those which function is to satisfy the clients needs when it is using the service and can be classify by it need in the following table.

CLIENTS NEEDS	KEY ACTIVITIES
Easy understanding of the product	Promotional video
Easy interaction with the bot	Bot design
Storage info of the challenges, users and judges	Database development
Reminders to the users	Automatization (CRM)
New challenges	Web development

Table 10. Key activities

On the other hand, the marketing activities are those which focus is to increase the number of users and also increase the interaction of the users that the company already have. This marketing activities can be divided in continuous activities and campaigns. The different between the two is that the first one are activities of marketing and brand positioning that should be done in a continuous basis whereas the campaigns are specific marketing activities that will focus in increase a special type of users, or related to a specific date during the year. The first one are considered fixed cost and the second one will be considered variable cost. Both of them have to be measure to identify the metrics of each campaign and how much money is the company investing in capture new clients and the percentage of increase during the life of the company and for each implementation of the value proposition.

2.2.7 KEY RESOURCES

What the company needs to perform the activities stated before. The resources can be categorized as human resources (employees, staff aspect), intellectual (know-how, brand or patents), physical (assets as PP&E or equipment), technological or financial resources (funds sources and flows of income, needed for the day to day activity).

In order to perform these activities, the company requires the use of different resources. The resources that Getyourchallenge needs are human resources and technological resources.

The following table shows the relation between the activities perform by the company and the resources needed to do it.

KEY ACTIVITIES	KEY RESOURCES
Marketing	(Human resources)
Promotional video	Web designer (Human resource)
Bot design	Programmers (Human resources)
	Servers and external platforms (Technological tools)
Database development	Programmers (Human resources)
	External platforms (Technological tools)
Automatization (CRM)	Programmers (Human resources)
	External platforms (Technological tools)
Web development	Web designer (Human resources)

Table 11. Key resources

Related to the use of external platform is the next block, the key partners.

2.2.8 KEY PARTNERS

The company will have relation with other companies from the same sector or related such as suppliers, competitors, government or regulators. The partners consist in other companies to which you can obtain benefits from in exchange of benefits that you will offer then searching for double gain for both parts.

This block shows the relation with other companies that is sustained because both part of the agreement benefit from it. In the case of Getyourchallenge the partners are companies or services that are related to the service provided by the company.

Firstly, all the online application that are used to create the minimum viable product, the Telegram Bot. Therefore, the first key partner is Telegram and after that, the database platform used to storage all the info collected by the bot, Airtable.

Secondly, as a percentage of the money bet and lose by the users by not achieving the challenge proposed is sent to ONG's or other non-profit organizations, all the entities that can be benefit from this agreement will become key partners of the company.

Finally, the social media resources use to launch the product and capture users and clients are also key partners.

These nine blocks, expressed in eight points, are the main structure of a company business and at it have been explained in each block they are interrelated between them. Therefore, to create a good business model canvas it should be created and design as a whole and not splitting the work by defining each block in isolation. The following charts show the representation of a business model canvas and their relationships.

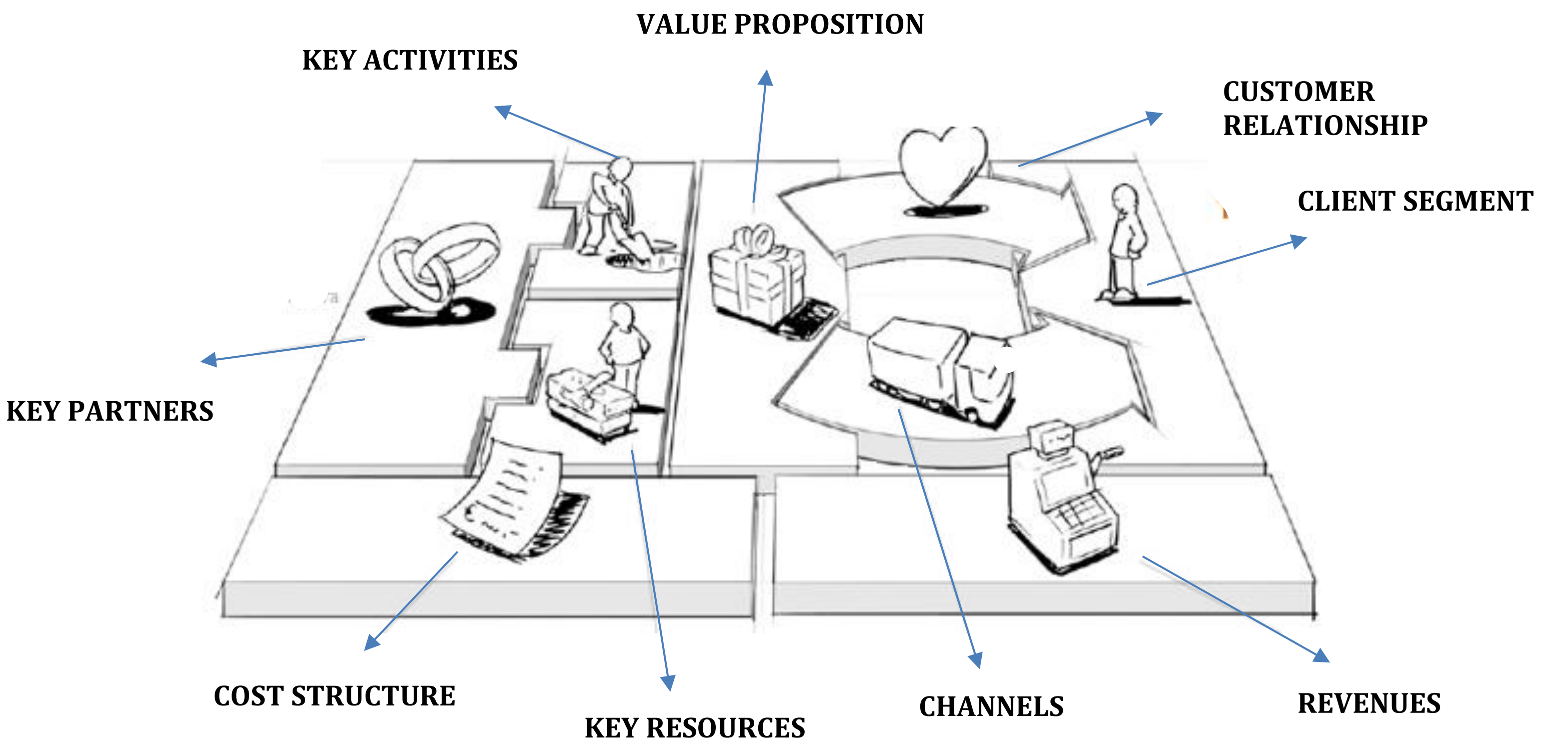


Figure 23. BMC blocks

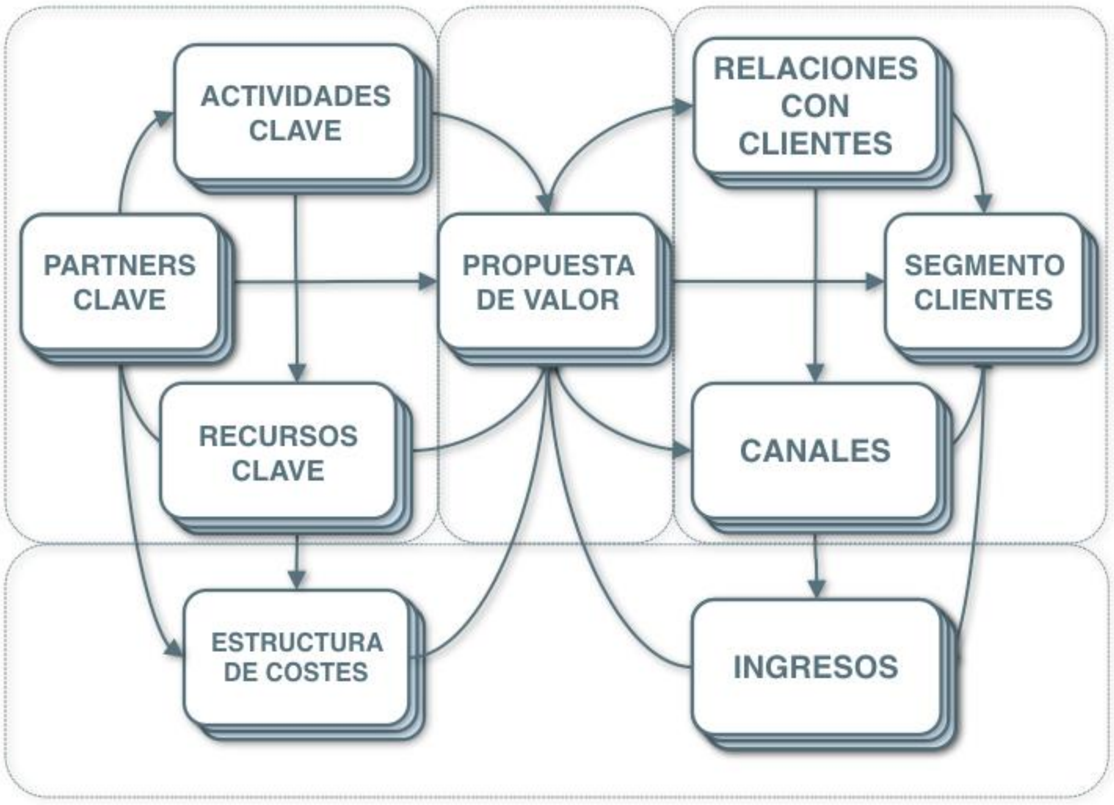


Figure 24. BMC blocks relationships

2.3 CONCLUSIONS

This section has exposed the business model canvas tool and after that the business model proposed by the startup created, Getyoutchallenge. By using the BMC the company business and strategy is shown and that tool is useful to always think in which way and how to proceeds in the several activities and factors that involve a company. After defining the business of the company, the next step which will be explained in Chapter three is the product/service in detail and how it has been developed.

3 CAPÍTULO 3: PRODUCT DEVELOPMENT

3.1 INTRODUCTION

In this chapter, the product in which the startup is sustained, the Telegram Bot and the database associated are going to be explained. Firstly, it is going to be expressed what the product does and how it works combine with the database. After that, how it has been developed, explaining the process and also which tools have been used. Finally, in the last part of the chapter it is going to be shown how it interact with the users and provide some examples of conversation with the bot an already users.

3.2 WHAT THE PRODUCT DOES

The product that offer the services provided by the startup Getyourchallenge is a Telegram Bot which main function is to interact with the clients as an automatic gadget that it is used by the clients to:

1. Create challenge that they want to perform.
2. Judge challenge of their relatives or friends, that have created a challenge.
3. See the status of challenges that have to be perform by the client or that have been already done (successfully or not).
4. Defy another person to pursue a challenge that you have created.

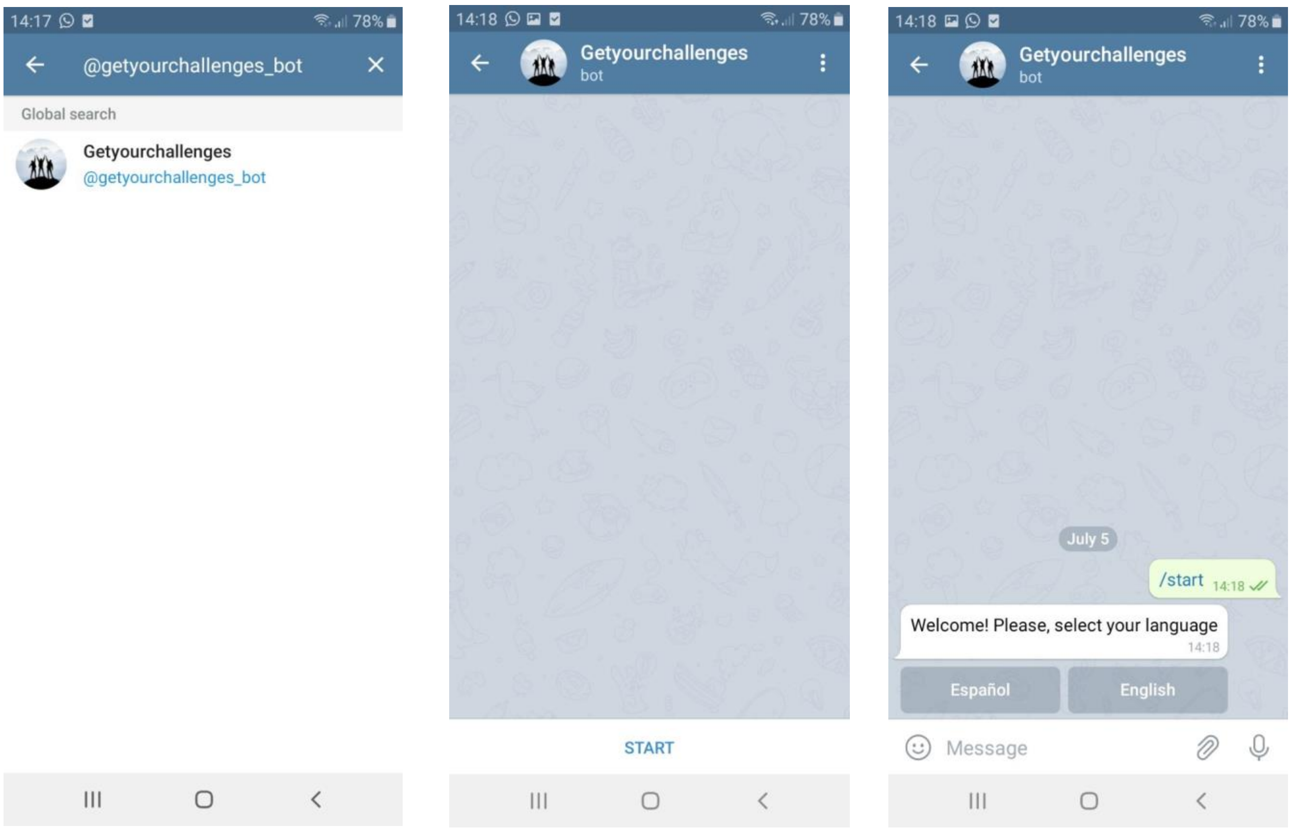
The functions that are stated above are the function that the user can do interacting with the Bot every time that he/she wants. In this section is exposed how the product works in reality, in particular the Minimum Viable Product that has been created and it is expected to be launched.

3.2.1 STARTING

From the beginning, the user has to download the app in which the bot resides, Telegram. Once the user has the app, then he/she can start using the service but first the user has to find the bot, searching for it:

Then, if the user is new and does not have any conversation with the bot, the user has to press the start button and the conversation will start.

However, if the user already has a conversation with the bot, she/he will start a new interaction with it using the last Menu provided by the bot in the last message.

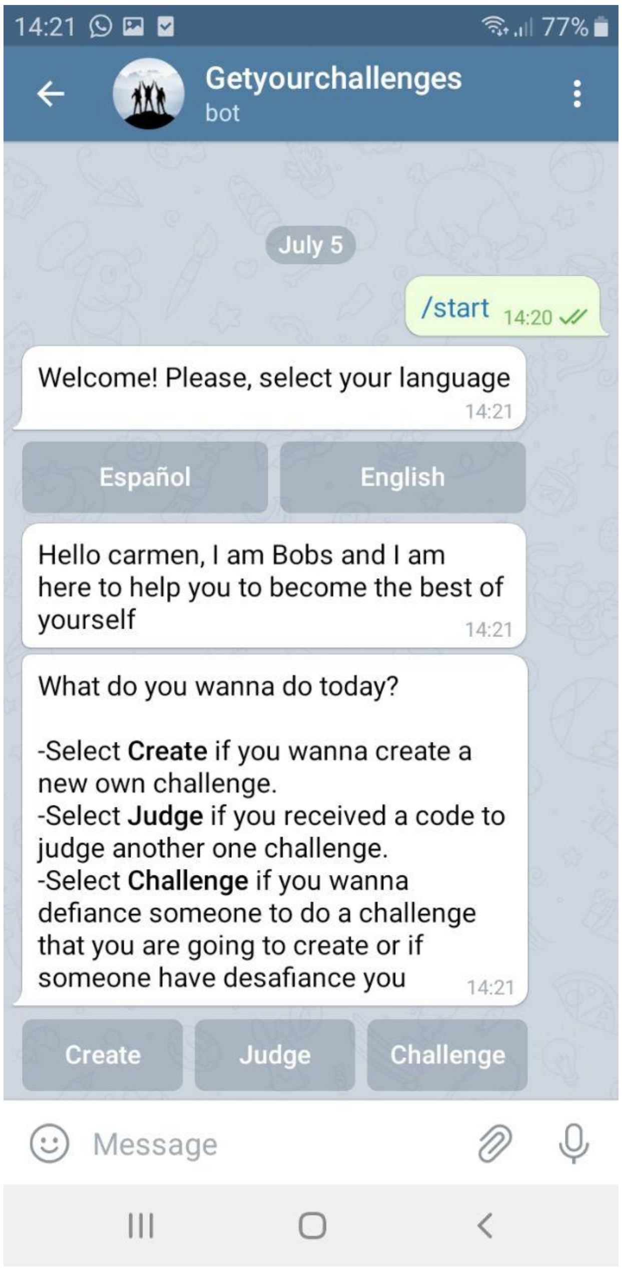


/start 14:18

Welcome! Please, select your language 14:18

Figure 25. Starting the conversation

As the previous section explain there are four function that a user can choose to do with the bot in each interaction. The function of each one are going to be explained below.


14:21 77%

Getyourchallenges
bot

July 5

/start 14:20

Welcome! Please, select your language 14:21

Hello carmen, I am Bobs and I am here to help you to become the best of yourself 14:21

What do you wanna do today?

-Select **Create** if you wanna create a new own challenge.
-Select **Judge** if you received a code to judge another one challenge.
-Select **Challenge** if you wanna defiance someone to do a challenge that you are going to create or if someone have desafiance you 14:21

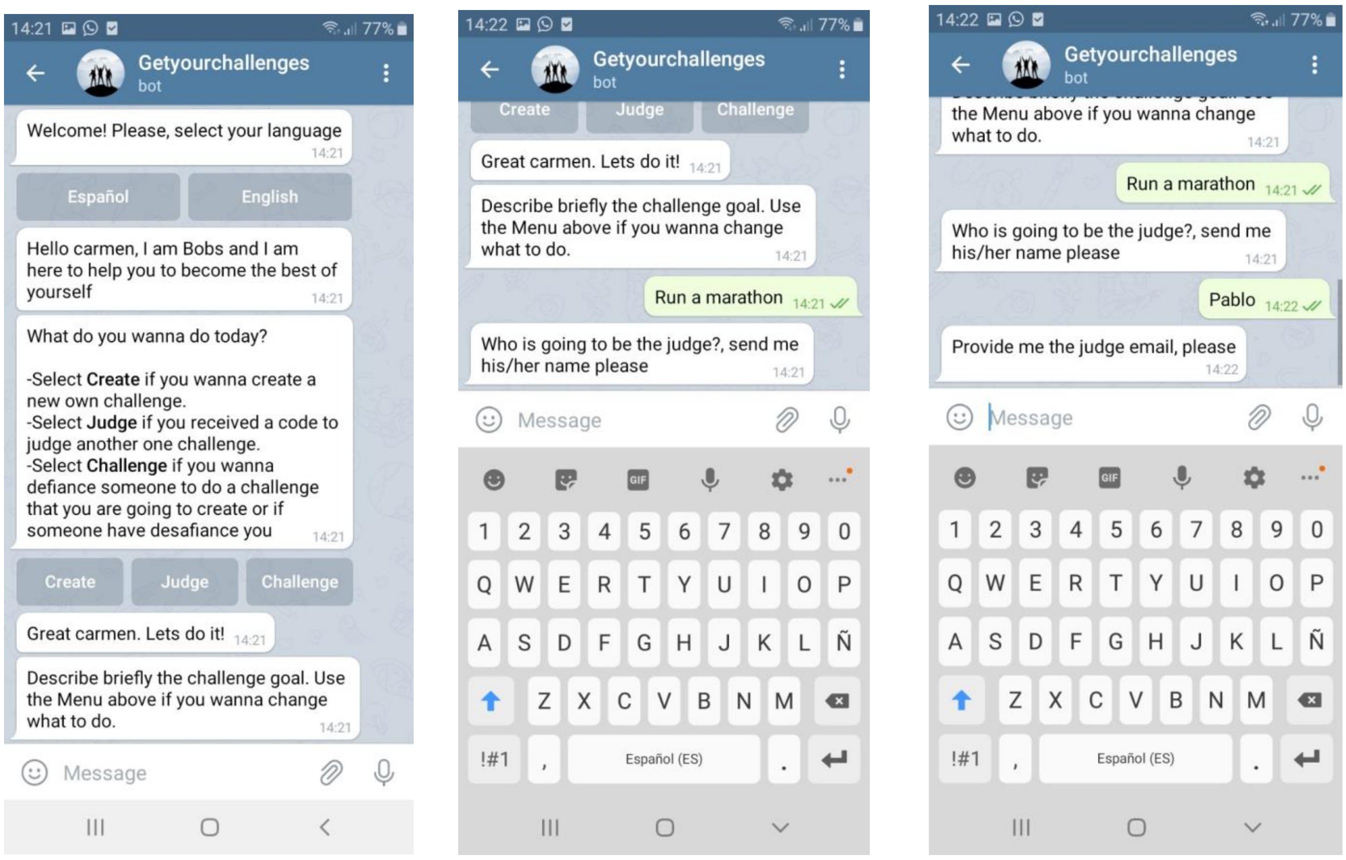
Figure 26. Bot initial options

3.2.2 CREATE

After deciding to create a new personal challenge the bot will ask the user some features of the challenge.

First, the bot is going to ask about a brief description of the challenge that the user want to perform. By storing it, the company increase the value that it gives to the clients because it has a higher number of challenges to propose to other users.

The second part is to decide who is going to evaluate the challenge, the judge. The bot is going to ask for a name and a email to contact him/her. However, if the user has used previous judges before the bot will send to him a list of them because probably the user will use one of them again, but also with the possibility to add a new one.



Welcome! Please, select your language 14:21

Hello carmen, I am Bobs and I am here to help you to become the best of yourself 14:21

What do you wanna do today?

-Select **Create** if you wanna create a new own challenge.
-Select **Judge** if you received a code to judge another one challenge.
-Select **Challenge** if you wanna defiance someone to do a challenge that you are going to create or if someone have desafiance you 14:21

Great carmen. Lets do it! 14:21

Describe briefly the challenge goal. Use the Menu above if you wanna change what to do. 14:21

Run a marathon 14:21

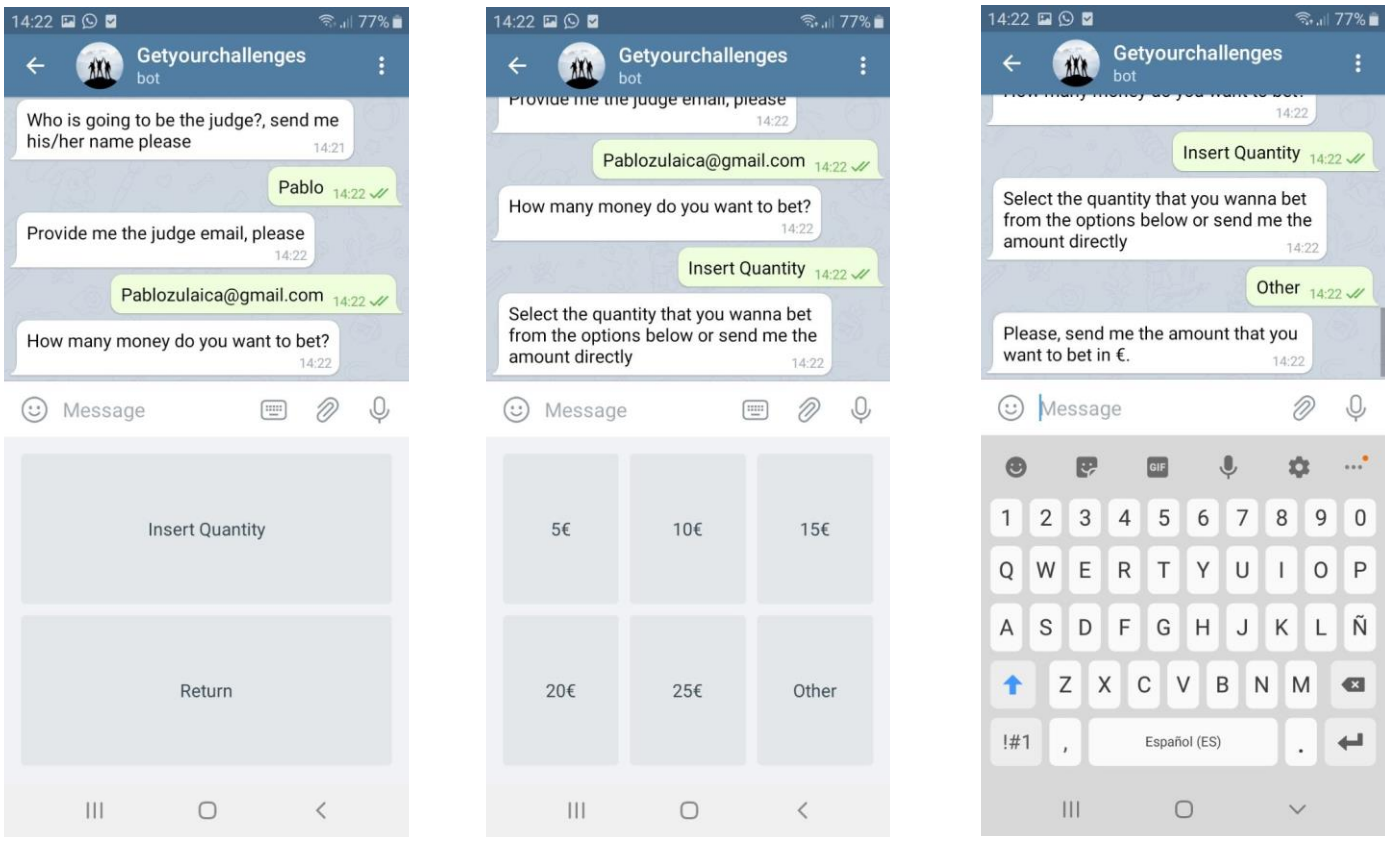
Who is going to be the judge?, send me his/her name please 14:21

Pablo 14:22

Provide me the judge email, please 14:22

Figure 27. Challenge description and judge details

After receiving the judge contact, the next step is to select the amount that the user want to lose if he/she is not able to perform the challenge, there are two options. Select one of the options of an external keyboard, or select in this keyboard the other option if the user want to bet another quantity, the user will be asked to send the amount in numbers with or without the € symbol.


14:22 77%
Getyourchallenges
bot

Who is going to be the judge?, send me his/her name please 14:21

Pablo 14:22

Provide me the judge email, please 14:22

Pablozulaica@gmail.com 14:22

How many money do you want to bet? 14:22

Insert Quantity 14:22

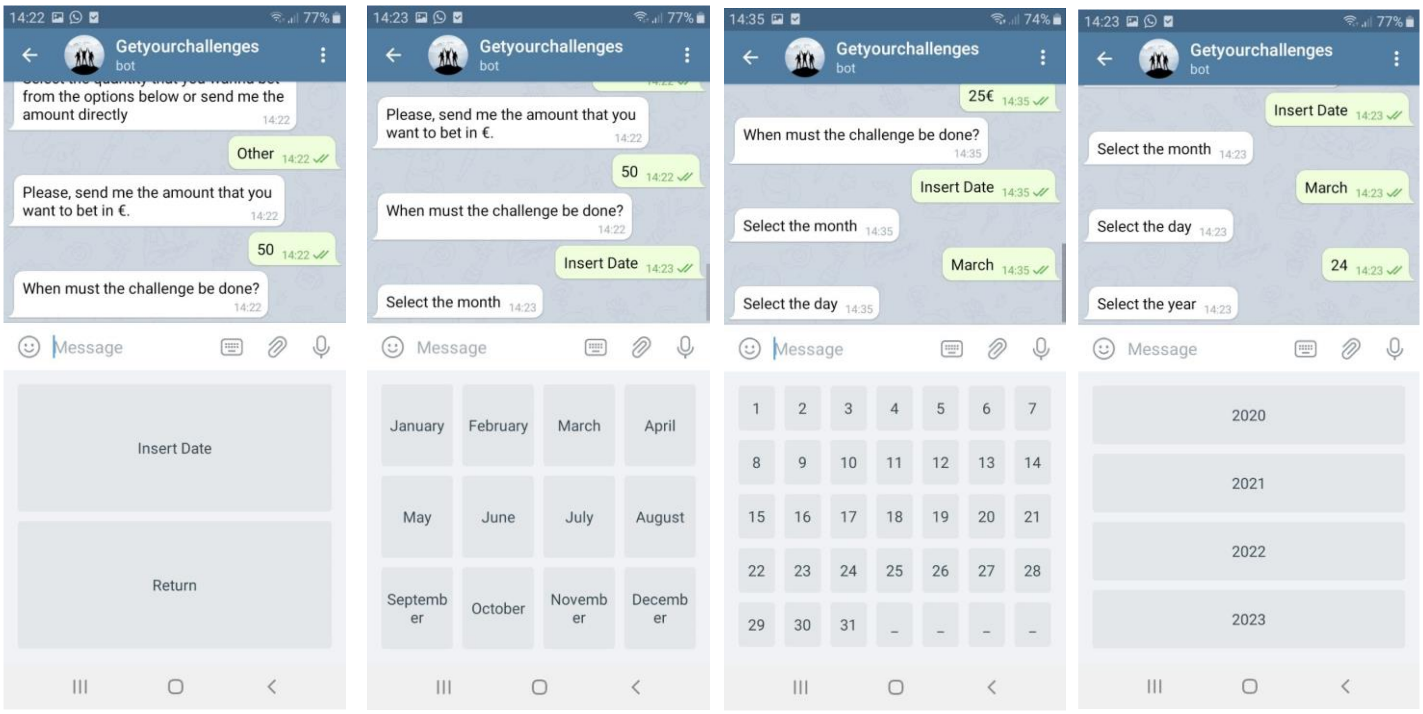
Select the quantity that you wanna bet from the options below or send me the amount directly 14:22

Other 14:22

Please, send me the amount that you want to bet in €. 14:22

Figure 28. Inserting the amount to bet

The next step is to select the due date, for this stage the user has to select from an external keyboard, the month, the day and at least the year of the due date. By this method the due date is storage in the database in a standardized way.


from the options below or send me the amount directly 14:22

Other 14:22

Please, send me the amount that you want to bet in €. 14:22

50 14:22

When must the challenge be done? 14:22

Please, send me the amount that you want to bet in €. 14:22

50 14:22

When must the challenge be done? 14:22

Insert Date 14:23

Select the month 14:23

25€ 14:35

When must the challenge be done? 14:35

Insert Date 14:35

Select the month 14:35

March 14:35

Select the day 14:35

Insert Date 14:23

Select the month 14:23

March 14:23

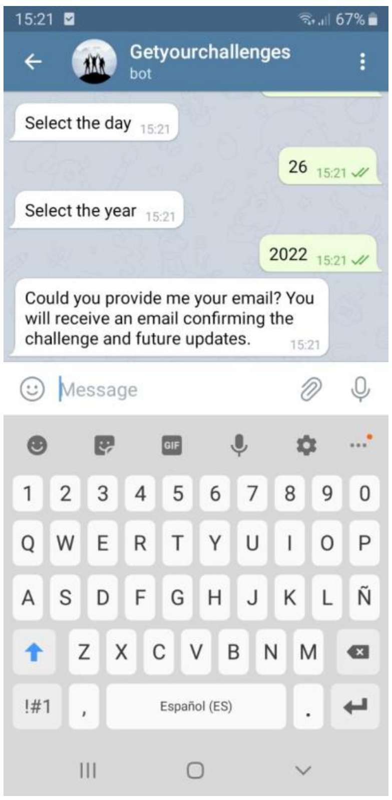
Select the day 14:23

24 14:23

Select the year 14:23

Figure 29. Inserting the due date

The last question is the email of contact of the user, this step is not necessary if the user information is already in the database.


Select the day 15:21

26 15:21

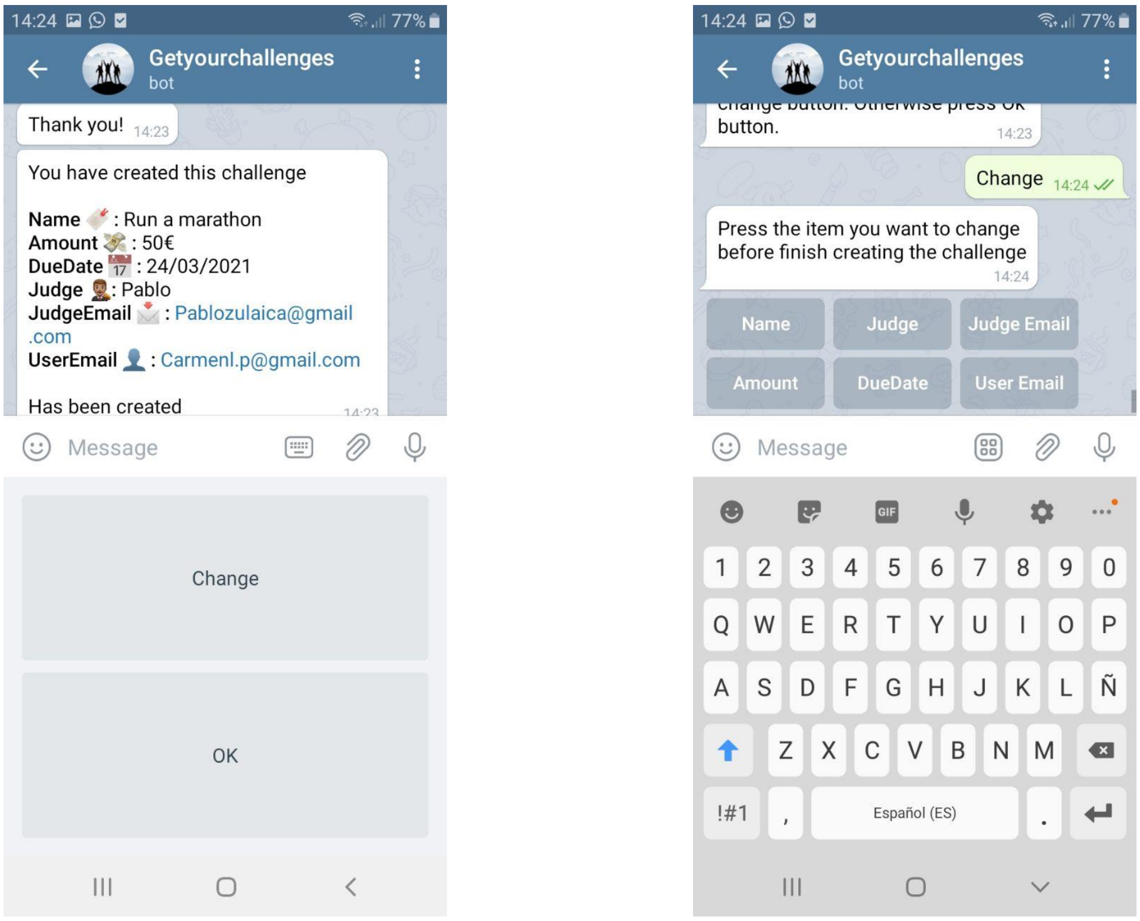
Select the year 15:21

2022 15:21

Could you provide me your email? You will receive an email confirming the challenge and future updates. 15:21

Figure 30. Asking for user email

Before completing the creation of the challenge, the bot sends a summary of the challenge that have been created to the user. Then the user can select the change button to change any field of the challenge in which there is a mistake or instead press the Ok button to accept and create the challenge completely.


Getyourchallenges
bot

Thank you! 14:23

You have created this challenge

Name : Run a marathon
Amount : 50€
DueDate : 24/03/2021
Judge : Pablo
JudgeEmail : Pablozulaica@gmail.com
UserEmail : Carmenl.p@gmail.com

Has been created

button. 14:23

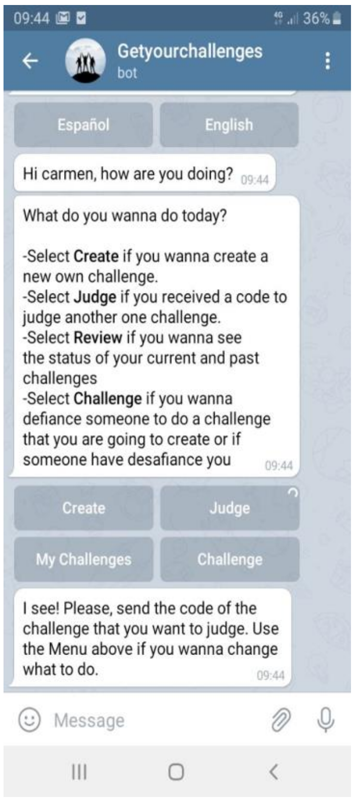
Change 14:24

Press the item you want to change before finish creating the challenge 14:24

Figure 31. Final step to create the challenge

3.2.3 JUDGE

If the user wants to judge an existing challenge the process is fast and easy. The bot will ask the user about the challenge code that he/she wants to evaluate.


09:44 36%

Getyourchallenges
bot

Hi carmen, how are you doing? 09:44

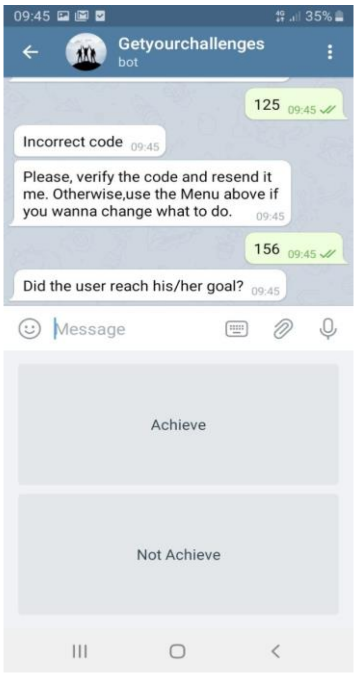
What do you wanna do today?

-Select **Create** if you wanna create a new own challenge.
-Select **Judge** if you received a code to judge another one challenge.
-Select **Review** if you wanna see the status of your current and past challenges
-Select **Challenge** if you wanna defiance someone to do a challenge that you are going to create or if someone have desafiance you 09:44

I see! Please, send the code of the challenge that you want to judge. Use the Menu above if you wanna change what to do. 09:44

Figure 32. Bot asking the challenge code

Then after receiving the code, the bot sends an external keyboard to the user in which he/she select the option of Achieved if the challenged was performed or Not Achieved otherwise.


09:45 35%

Getyourchallenges
bot

125 09:45

Incorrect code 09:45

Please, verify the code and resend it me. Otherwise,use the Menu above if you wanna change what to do. 09:45

156 09:45

Did the user reach his/her goal? 09:45

Figure 33. Judging options

Finally, a summary clarification the judge of the challenge is sent to the user as a prove, and an email is automatically sent to the user that created the challenge.

3.2.4 MY CHALLENGES

This function is only available for the existing users because they are the only ones who already have challenges in the database.

After selecting this function, the bot will search in the database for all the challenges that the user has with the company, using the User ID. After that the bot sends a list of challenges to the user in an inline keyboard, each challenge by its brief description.

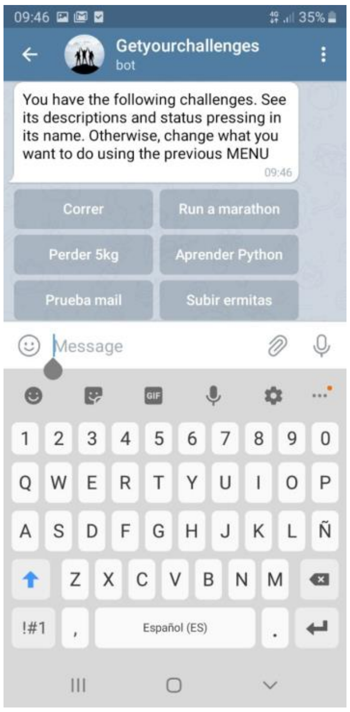
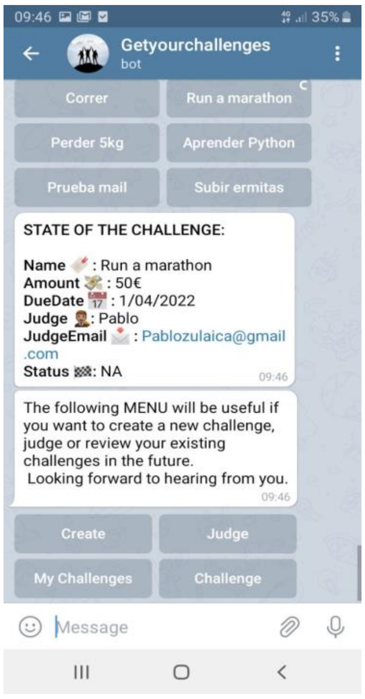

You have the following challenges. See its descriptions and status pressing in its name. Otherwise, change what you want to do using the previous MENU

Figure 34, User challenges with the Bot

The user selects the one he/she wants to see and a summary of it and its status is send by the bot to the user.


STATE OF THE CHALLENGE:

Name : Run a marathon
Amount : 50€
DueDate : 1/04/2022
Judge : Pablo
JudgeEmail : Pablozulaica@gmail.com
Status : NA

The following MENU will be useful if you want to create a new challenge, judge or review your existing challenges in the future.
Looking forward to hearing from you.

Figure 35. Challenge status

3.2.5 CHALLENGE

This function is used when the user wants to create a challenge to defy someone else or when the user has been challenged.

The first step is therefore, decide between these two options.

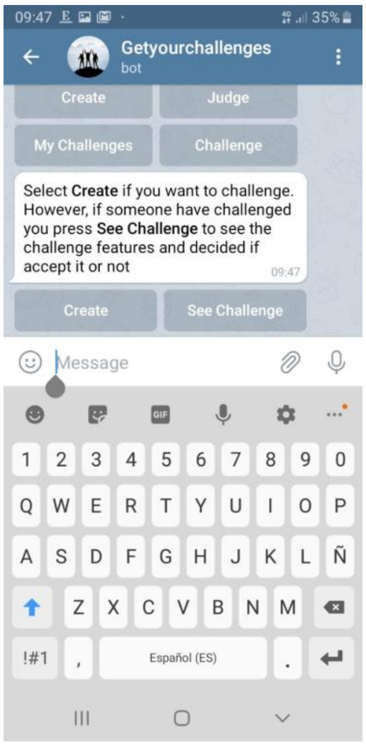
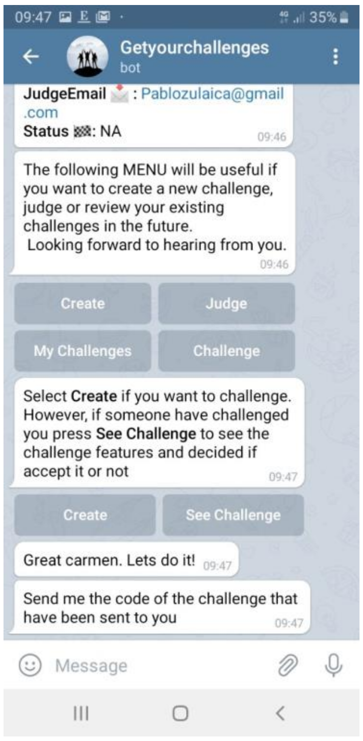

Select **Create** if you want to challenge. However, if someone have challenged you press **See Challenge** to see the challenge features and decided if accept it or not

Figure 36. Defy someone options

The first option create a challenge to defy someone has the same shape as the create function. However, the other option has an unique way. When selecting it, the bot will ask you for the code of the challenge that you have been challenged. After sending the code to the bot, it will answer you with the summary of the challenge and two options, Take the challenge if the user want to accept it or on the other hand to Leave it if he/she does not want to do it.


09:47

Getyourchallenges
bot

JudgeEmail 📩 : Pablozulaica@gmail.com
Status 🏁: NA 09:46

The following MENU will be useful if you want to create a new challenge, judge or review your existing challenges in the future.
Looking forward to hearing from you. 09:46

Select **Create** if you want to challenge. However, if someone have challenged you press **See Challenge** to see the challenge features and decided if accept it or not 09:47

Great carmen. Lets do it! 09:47

Send me the code of the challenge that have been sent to you 09:47

Figure 37. See the challenge that you have been defy

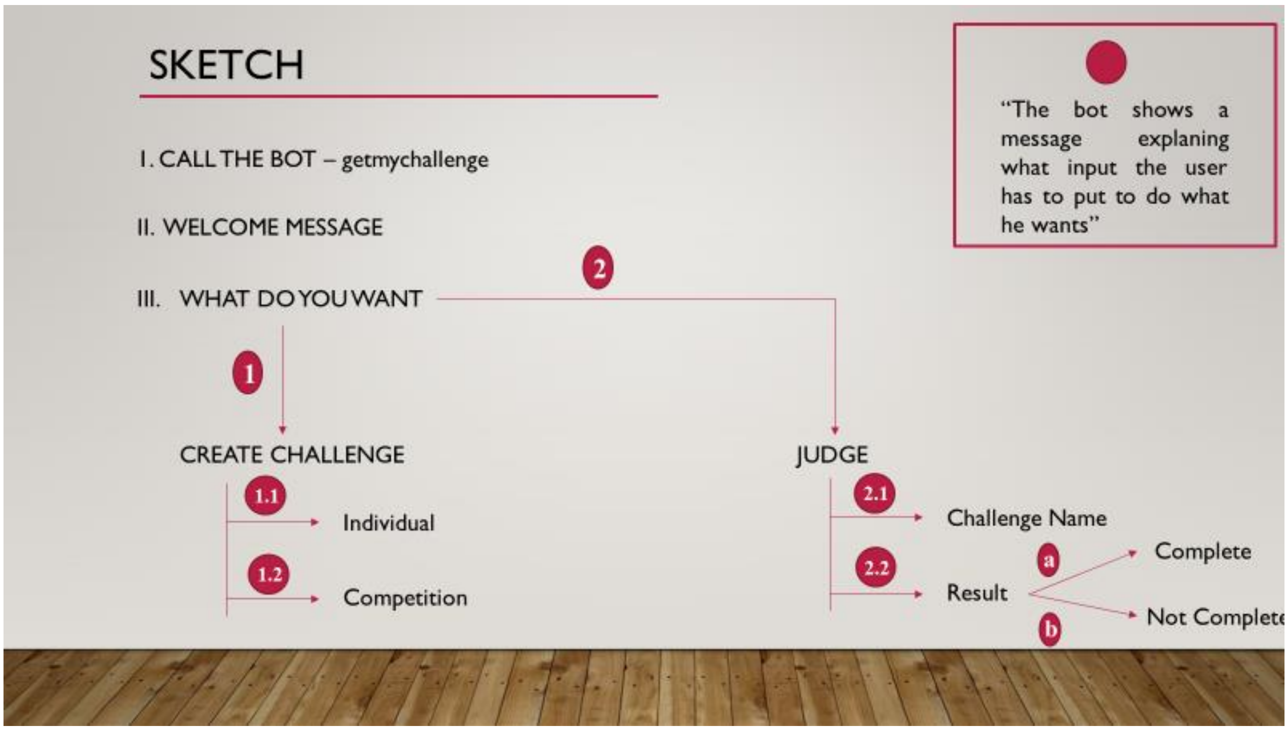
The functions that are stated above are the function that the user can do interacting with the Bot every time that he/she wants.

3.3 HOW THE PRODUCT HAVE BEEN DEVELOPED

In this section it is explained how the product have been developed. Starting from the design and first sketch to the final stage, the minimum viable product.

The first part of the product development was the design of the product, which main focus was to define what the product have to do in order to provide the service that it is exposed in the value proposition of the company BMC.

In this stage, the first sketch of the product was made and is the following:



SKETCH

I. CALL THE BOT – getmychallenge

II. WELCOME MESSAGE

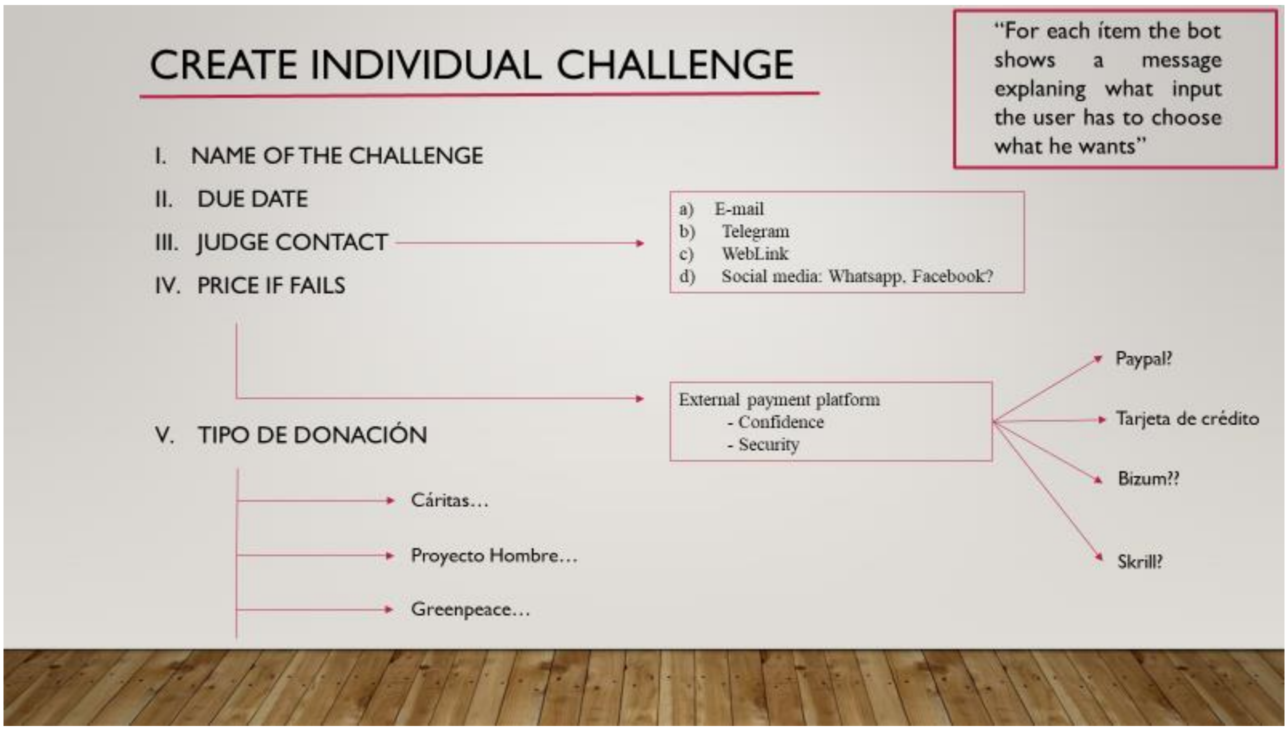
III. WHAT DO YOU WANT

1 CREATE CHALLENGE
- 1.1 Individual
- 1.2 Competition

2 JUDGE
- 2.1 Challenge Name
- 2.2 Result
  - a Complete
  - b Not Complete

● "The bot shows a message explaning what input the user has to put to do what he wants"

Figure 38. First sketch of the product



CREATE INDIVIDUAL CHALLENGE

I. NAME OF THE CHALLENGE

II. DUE DATE

III. JUDGE CONTACT
- a) E-mail
- b) Telegram
- c) WebLink
- d) Social media: Whatsapp, Facebook?

IV. PRICE IF FAILS

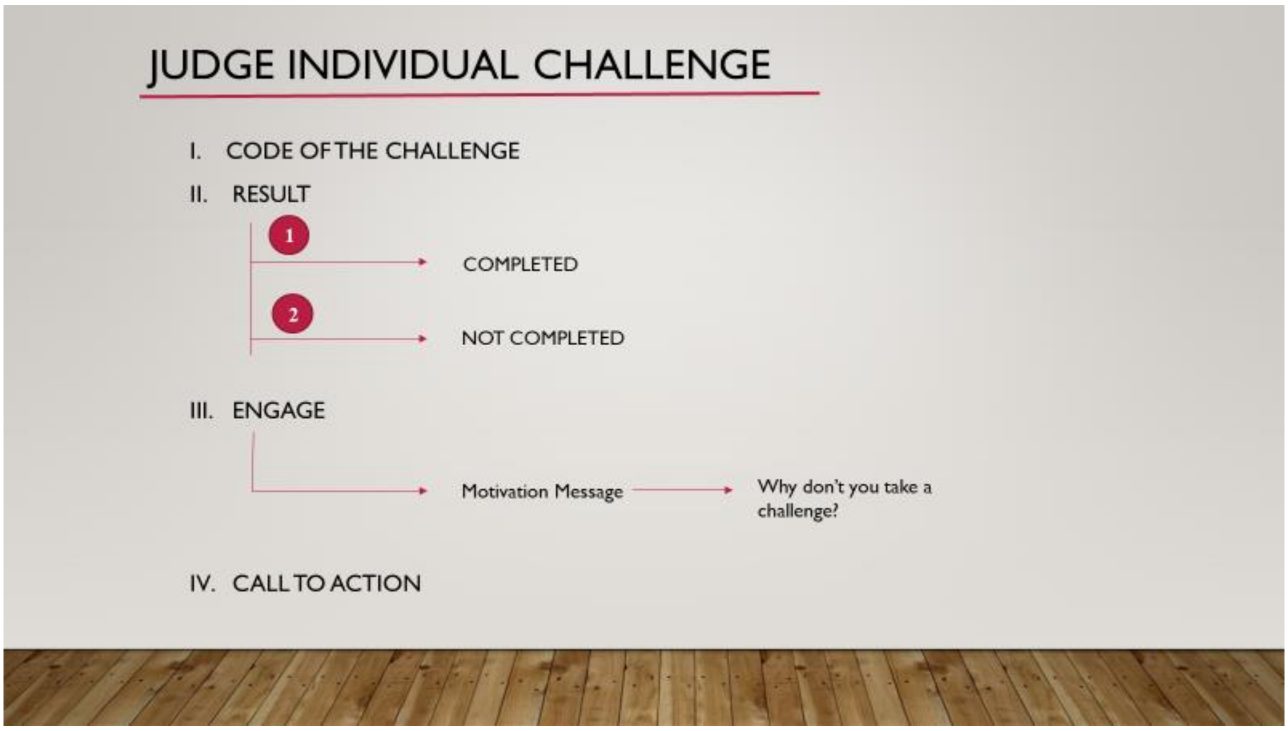
External payment platform
- Confidence
- Security

Paypal?
Tarjeta de crédito
Bizum??
Skrill?

V. TIPO DE DONACIÓN
- Cáritas…
- Proyecto Hombre…
- Greenpeace…

"For each ítem the bot shows a message explaning what input the user has to choose what he wants"

Figure 39. First product design: create function


JUDGE INDIVIDUAL CHALLENGE

I. CODE OF THE CHALLENGE

II. RESULT

III. ENGAGE

IV. CALL TO ACTION

Figure 40. First product design: judge function

The figures above, shows the first design of what the product had to do in order to provide the clients the service proposed by the company. In the first design as it is exposed, the product only proposed two function, one to create a challenge and a second one two judge the challenges created.

With this design in mind the first sketch of the product started to be developed. To do it the first step was to create a new bot inside Telegram, and to do it the @BotFather had to be used. Interacting with BotFather it is possible to create new bots inside Telegram, you only have to decide the name of the bot and botfather gives to you the TOKEN, which is the unique identification of the bot that has been created.

Once the bot TOKEN has been provided, the next step is to start creating a Bot programming it and in the case of this project the Bot is programmed with Python, in a platform called Pycharm.

Using the Github repository, which has several example of different bots created by others users, the first sketch was designed. The code of this first bot can be found in the Anexo of this document.

As it is shown in the figures above, the first sketch was functional but so simple. It only used an external keyboard to select between the functions of the bot at the beginning and then the rest is a simple conversation without the possibility to go back if you make a mistake in the email or in the amount. Moreover, the data was not standardized in the database because there were not options to choose in a keyboard to storage a standardized form.

After taking contact with the telegram bots by creating the first sketch, the next step was to decide which features of the bot could be improve and to do it the product was tested with some relatives and friends in order to find insights to increase the value added to the client in the interaction with the bot.

The first problem was the lack of standardized information that was received from the user. To solve this problem, the best option was to create new keyboards inline and external ones that make easier for the user to choose the amount to bet or the due date.

Moreover, another technical improvement was to change some external keyboards to inline keyboards which are options that are send to the user inside the conversation and can be press anytime during the conversation to select an option. Being inside the conversation made possible to the user to continue writing to the bot if he/she does not want to select any option of the inline keyboard, a feature that it is less clear to the user using the external keyboard. This inline keyboard was implemented in several part of the conversation starting from the first Menu in order to select between the functions that the user can do with the bot.

Apart from the technical improvements that have been stated before. The most important update was in the design of the bot. From the opinions of the users that take contact with the first sketch several insights were extracted, and the design of the bot pivoted to other directions.

The most important strategic insight that were extracted from the users of the first sketch was the option of challenge someone else using the bot. Instead of only have the opportunity of creating a personal bot, why the user cannot create a challenge to defy someone else to do it?

This function will increase the value offer to the users because one of the main objective is to increase the users motivation to do challenges and one way of doing it is having the opportunity of share this challenge with a relative or a friend and also is a great way of find new challenges for the users.

In relation with this strategic improvement was the creation also of a function in which the existing users has the opportunity of see the challenges that they have created with the bot or the challenges that they have accepted when someone had defied them. The new sketch of the bot was modified to the following one:

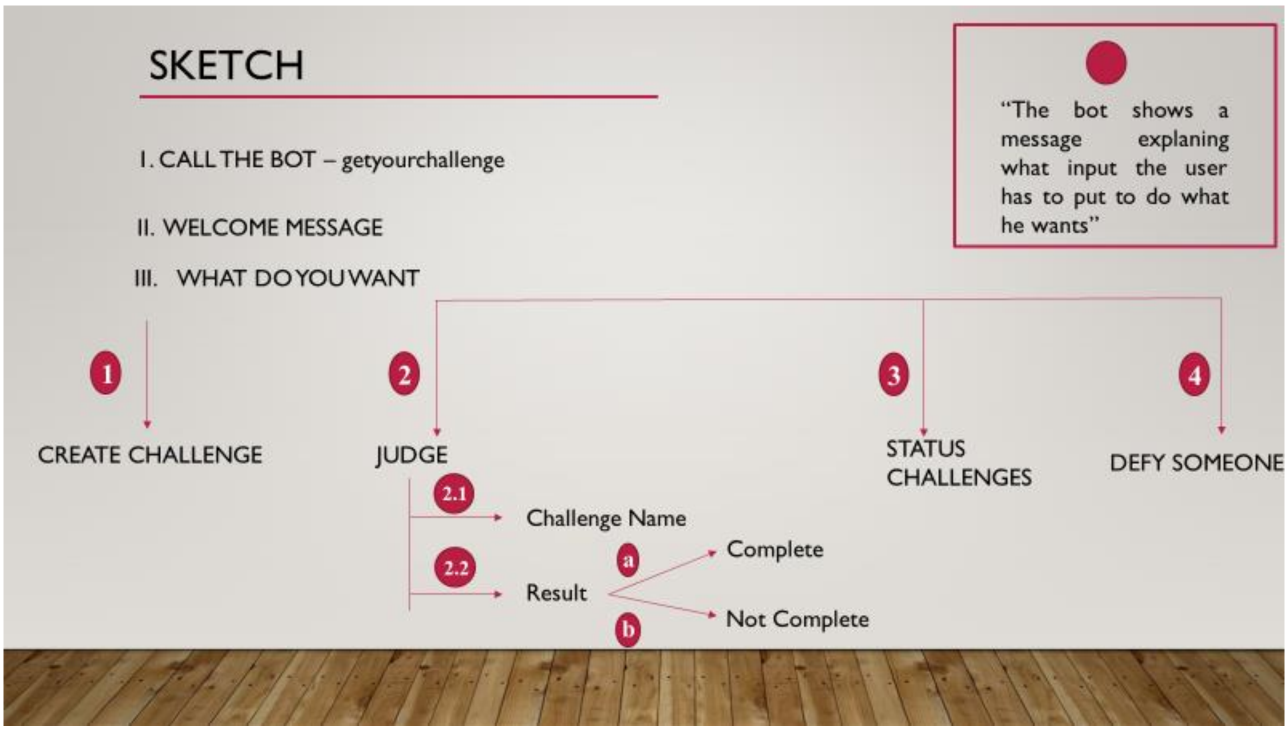


Figure 41. Sketch Minimum Viable Product

Therefore, with these two updates, the main menu in which the user decides what to do with the service provided by Getyourchallenge improve to the following one.

Firstly, there is a Menu for new users, as they do not have any challenge yet there is no option to see the challenges. However, for the existing users, the four options are available. Both Menus use an inline keyboard, the technical improvement that was explained above.

Once the bot development has been explained, at the same time that the bot was being developed the other essential part of the product, the database, was developed too.

At once only one table was created, all table. However, as the new strategic improvements arrived the database had to change to.

In order to classify the users, two new tables were developed, the users table and the judge table. This two table were useful because:

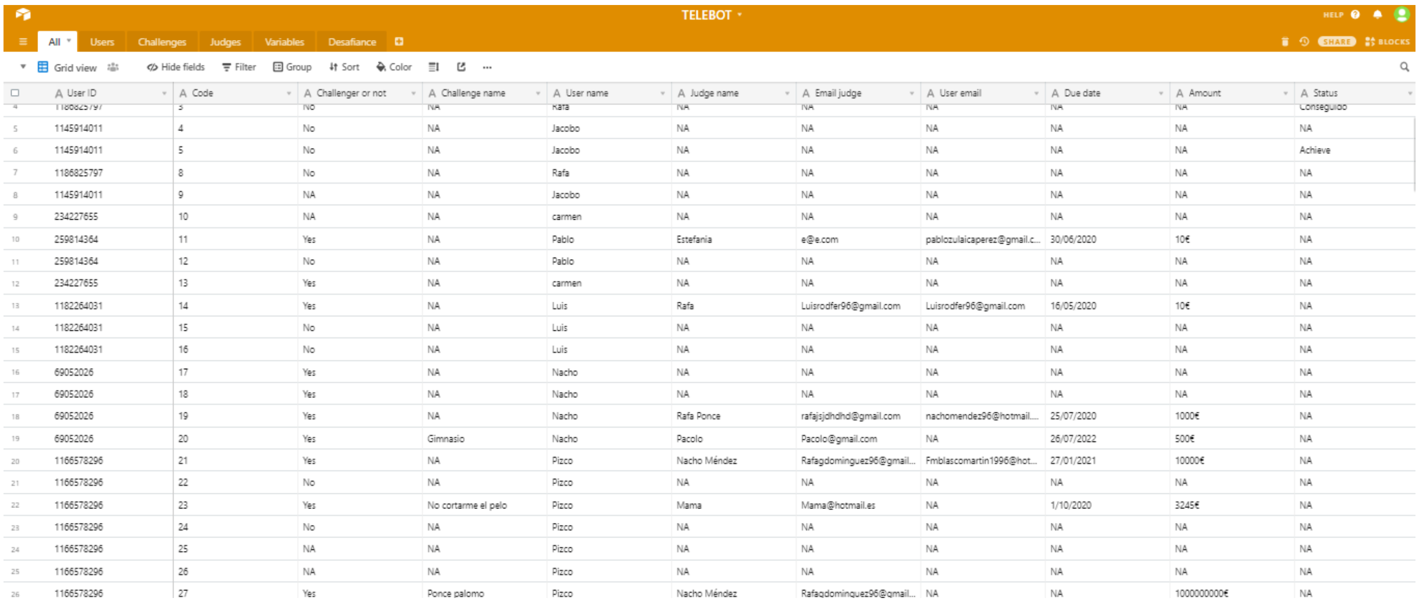
- In the users table were collected all the information provided by the users, therefore, the existing users do not have to give again to the bot this information and the challenge information was complete with the information of the users table.
- The judge table followed the same logic but instead of the users (creators of the challenges), with the persons in charge of judge the challenges, the judges.

Moreover, to make easier the interaction of the user and to take advantage of the database, a new table was created in which only the challenges that have been completely created (judged or not) appears. This table is so important because it allow several functions.

- The first one to present a list of existing judges that the user has used before in other challenges.
- The second one to allow the user to see the all the challenges that he/she has with the company.
- Provides a better information which can be used to measure the percentage of challenge in process, the challenges that have been achieve or not achieve and compare the number of interactions with the number of completely created challenges.

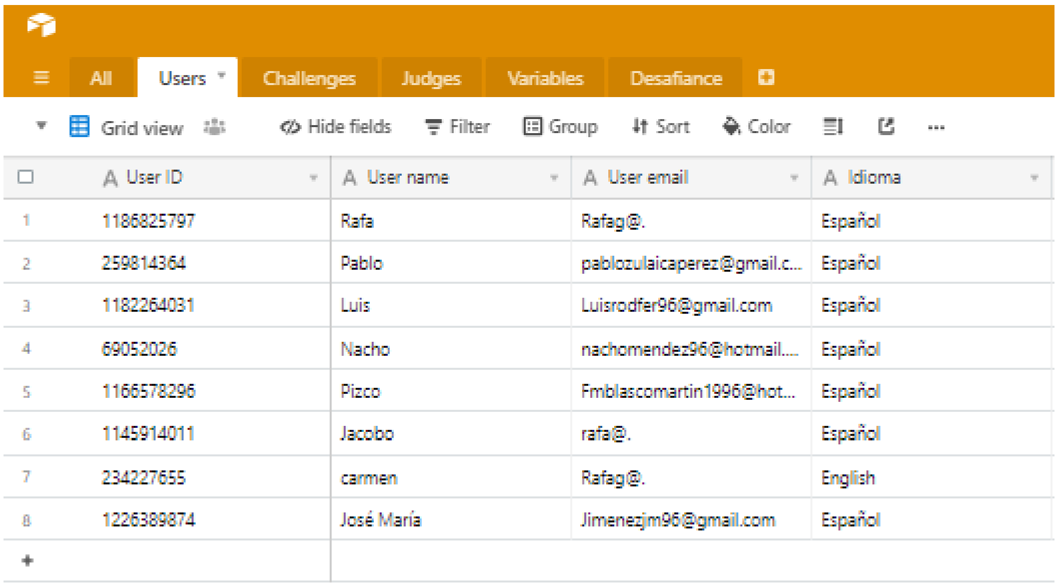
The last two improvements of the database were the Variables table and the Defy table. The first one was created to allow the interaction of the bot with several users at the same time, which was impossible in the first sketch made. The second one was created to implement the new function of challenging another user to pursue a challenge created by yourself.

The following figures show the state of the database before launching the product.



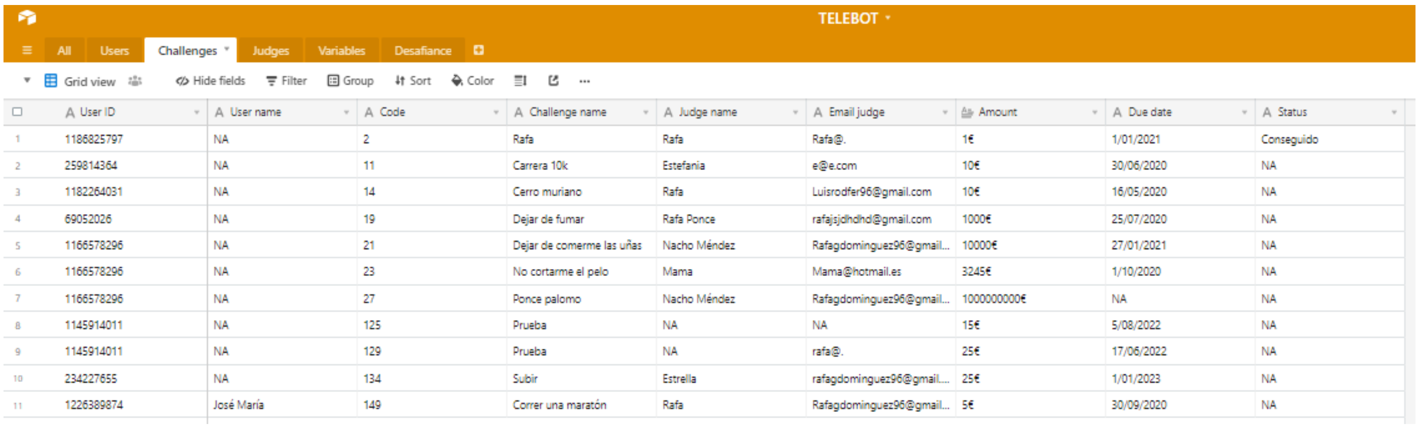
	User ID	Code	Challenger or not	Challenge name	User name	Judge name	Email judge	User email	Due date	Amount	Status
4	1186825797	3	No	NA	Rafa	NA	NA	NA	NA	NA	Conseguido
5	1145914011	4	No	NA	Jacobo	NA	NA	NA	NA	NA	NA
6	1145914011	5	No	NA	Jacobo	NA	NA	NA	NA	NA	Achieve
7	1186825797	8	No	NA	Rafa	NA	NA	NA	NA	NA	NA
8	1145914011	9	NA	NA	Jacobo	NA	NA	NA	NA	NA	NA
9	234227655	10	NA	NA	carmen	NA	NA	NA	NA	NA	NA
10	259814364	11	Yes	NA	Pablo	Estefania	e@e.com	pablozulaicaperez@gmail.c...	30/06/2020	10€	NA
11	259814364	12	No	NA	Pablo	NA	NA	NA	NA	NA	NA
12	234227655	13	Yes	NA	carmen	NA	NA	NA	NA	NA	NA
13	1182264031	14	Yes	NA	Luis	Rafa	Luisrodfer96@gmail.com	Luisrodfer96@gmail.com	16/05/2020	10€	NA
14	1182264031	15	No	NA	Luis	NA	NA	NA	NA	NA	NA
15	1182264031	16	No	NA	Luis	NA	NA	NA	NA	NA	NA
16	69052026	17	Yes	NA	Nacho	NA	NA	NA	NA	NA	NA
17	69052026	18	Yes	NA	Nacho	NA	NA	NA	NA	NA	NA
18	69052026	19	Yes	NA	Nacho	Rafa Ponce	rafajsjdhdhd@gmail.com	nachomendez96@hotmail....	25/07/2020	1000€	NA
19	69052026	20	Yes	Gimnasio	Nacho	Pacolo	Pacolo@gmail.com	NA	26/07/2022	500€	NA
20	1166578296	21	Yes	NA	Pizco	Nacho Méndez	Rafagdominguez96@gmail...	Fmblascomartin1996@hot...	27/01/2021	10000€	NA
21	1166578296	22	No	NA	Pizco	NA	NA	NA	NA	NA	NA
22	1166578296	23	Yes	No cortarme el pelo	Pizco	Mama	Mama@hotmail.es	NA	1/10/2020	3245€	NA
23	1166578296	24	No	NA	Pizco	NA	NA	NA	NA	NA	NA
24	1166578296	25	NA	NA	Pizco	NA	NA	NA	NA	NA	NA
25	1166578296	26	NA	NA	Pizco	NA	NA	NA	NA	NA	NA
26	1166578296	27	Yes	Ponce palomo	Pizco	Nacho Méndez	Rafagdominguez96@gmail...	NA	NA	1000000000€	NA

Figure 42. All Table



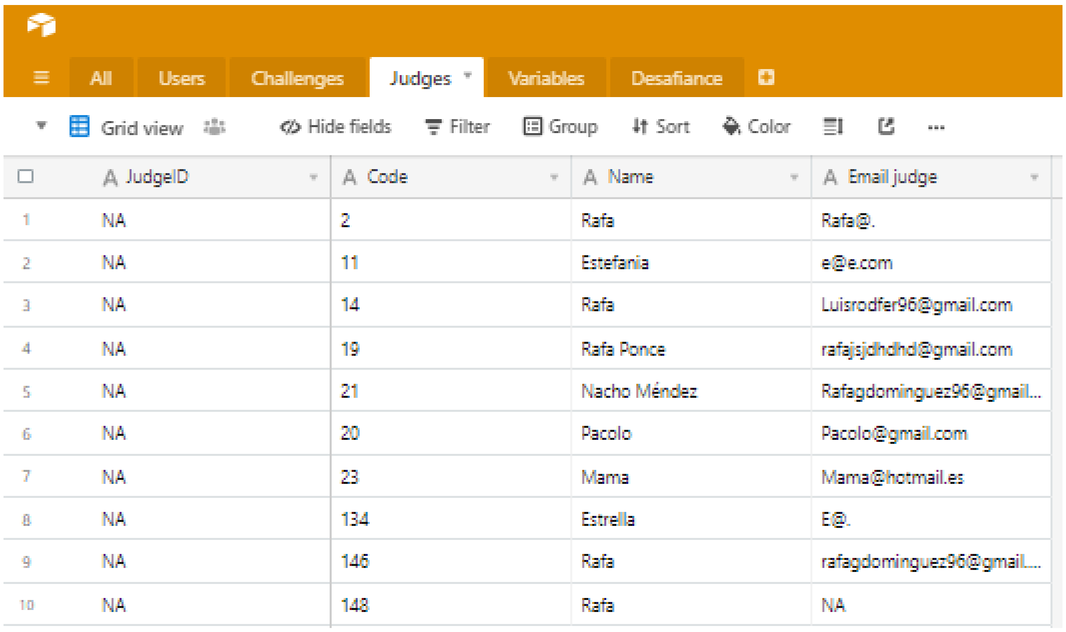
	User ID	User name	User email	Idioma
1	1186825797	Rafa	Rafag@.	Español
2	259814364	Pablo	pablozulaicaperez@gmail.c...	Español
3	1182264031	Luis	Luisrodfer96@gmail.com	Español
4	69052026	Nacho	nachomendez96@hotmail....	Español
5	1166578296	Pizco	Fmblascomartin1996@hot...	Español
6	1145914011	Jacobo	rafa@.	Español
7	234227655	carmen	Rafag@.	English
8	1226389874	José María	Jimenezjm96@gmail.com	Español

Figure 43. Users Table



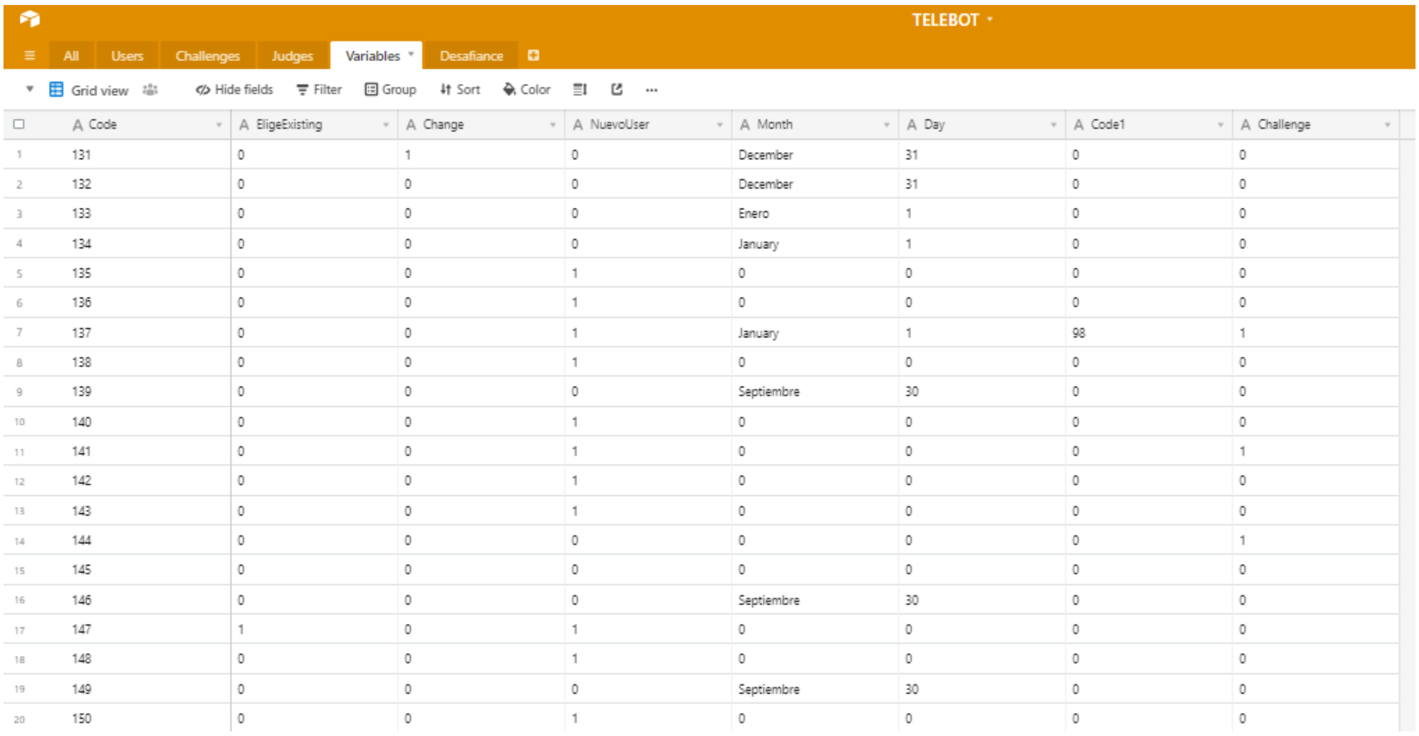
	User ID	User name	Code	Challenge name	Judge name	Email judge	Amount	Due date	Status
1	1186825797	NA	2	Rafa	Rafa	Rafa@.	1€	1/01/2021	Conseguido
2	259814364	NA	11	Carrera 10k	Estefania	e@e.com	10€	30/06/2020	NA
3	1182264031	NA	14	Cerro muriano	Rafa	Luisrodfer96@gmail.com	10€	16/05/2020	NA
4	69052026	NA	19	Dejar de fumar	Rafa Ponce	rafajsjdhdhd@gmail.com	1000€	25/07/2020	NA
5	1166578296	NA	21	Dejar de comerme las uñas	Nacho Méndez	Rafagdominguez96@gmail...	10000€	27/01/2021	NA
6	1166578296	NA	23	No cortarme el pelo	Mama	Mama@hotmail.es	3245€	1/10/2020	NA
7	1166578296	NA	27	Ponce palomo	Nacho Méndez	Rafagdominguez96@gmail...	1000000000€	NA	NA
8	1145914011	NA	125	Prueba	NA	NA	15€	5/08/2022	NA
9	1145914011	NA	129	Prueba	NA	rafa@.	25€	17/06/2022	NA
10	234227655	NA	134	Subir	Estrella	rafagdominguez96@gmail....	25€	1/01/2023	NA
11	1226389874	José María	149	Correr una maratón	Rafa	Rafagdominguez96@gmail...	5€	30/09/2020	NA

Figure 44. Challenges Table



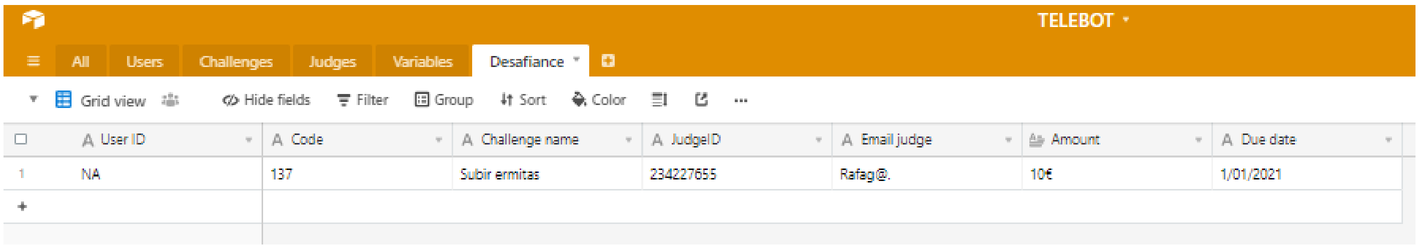
	JudgeID	Code	Name	Email judge
1	NA	2	Rafa	Rafa@.
2	NA	11	Estefania	e@e.com
3	NA	14	Rafa	Luisrodfer96@gmail.com
4	NA	19	Rafa Ponce	rafajsjdhdhd@gmail.com
5	NA	21	Nacho Méndez	Rafagdominguez96@gmail...
6	NA	20	Pacolo	Pacolo@gmail.com
7	NA	23	Mama	Mama@hotmail.es
8	NA	134	Estrella	E@.
9	NA	146	Rafa	rafagdominguez96@gmail....
10	NA	148	Rafa	NA

Figure 45. Judges Table



	Code	EligeExisting	Change	NuevoUser	Month	Day	Code1	Challenge
1	131	0	1	0	December	31	0	0
2	132	0	0	0	December	31	0	0
3	133	0	0	0	Enero	1	0	0
4	134	0	0	0	January	1	0	0
5	135	0	0	1	0	0	0	0
6	136	0	0	1	0	0	0	0
7	137	0	0	1	January	1	98	1
8	138	0	0	1	0	0	0	0
9	139	0	0	0	Septiembre	30	0	0
10	140	0	0	1	0	0	0	0
11	141	0	0	1	0	0	0	1
12	142	0	0	1	0	0	0	0
13	143	0	0	1	0	0	0	0
14	144	0	0	0	0	0	0	1
15	145	0	0	0	0	0	0	0
16	146	0	0	0	Septiembre	30	0	0
17	147	1	0	1	0	0	0	0
18	148	0	0	1	0	0	0	0
19	149	0	0	0	Septiembre	30	0	0
20	150	0	0	1	0	0	0	0

Figure 46. Variables Table



	User ID	Code	Challenge name	JudgeID	Email judge	Amount	Due date
1	NA	137	Subir ermitas	234227655	Rafag@.	10€	1/01/2021

Figure 47. Defy Table

The last improvement made to finally create the Minimum Viable Product was the CRM automatization using the database. For that purpose, an online tool called Zappier was used. With this tool the database was connected to a company email in order to automatically send emails to the users and judge of the challenge that where created. The automatization follow start when the user selects the Ok button which means that all the features of the challenges are correct. Then, the company email sends a message to the usermail which is storage in the users table and in this email, it is reflected the features of the challenge that the user has created. At the same time another email is send to the judge selected by the user, using the judge email provided by the user too. In this email, as well as in the email sent to the user, is exposed the features of the challenge and also instruction for the judge to what to do in order to evaluate the challenge.

3.4 HOW DOES THE PRODUCT WORKS?

3.4.1. DATABASE, AIRTABLE

Once the user select the function for which he/she wants to use the Bot, the product needs to storage some information in order to provide the service that the client is expecting. For that purpose the product combines the Bot with an external database which is made in an online platform called Airtable.

The database consist in six different tables in which the bot allocate the information provided by the users. The six tables which conform the database and what they stored are:

- All Table: In this table is collected every interaction between an user and the bot, does not matter if at least the user does not completely create a challenge or judge an existing challenge. This allow the company to compare between the functions of the bot which is the most demand (create own challenges, defy someone else, see the existing challenges) and also measure the percentage of challenges that have been judge in time or not for example.
- User Table: This table collects all the personal information that characterize an user. In particular it storage the User ID (Telegram personal code for each user), the user name (which is also obtained from the telegram user name that each user decide to use) and the user mail which is the most important contact way for the company to reach the user. This information is very useful, mainly, the email in the starting point in order to make contact with clients or potential clients that have already use the bot.
- Challenge Table: In this table it is storage the main characteristics of challenges that have been completely created. The features that define a challenge are: the challenge code which is used to identify the challenge, the challenge description, the amount bet in the challenge, the due date before what the challenge have to be perform and also the judge name and email associated with the challenge.
- Judge Table: This table is similar to the user table, however, this one is thought to storage the information of the judges that have been use in challenges created. The main function of the table is to provide the users a list of existing judges that

they have used before in previous challenges to simplify the process of creating a challenge and reducing the time.

- Variables Table: This table is internal and it is used by the bot in order to storage variables that change in each interaction between the bot and the user and are unique for each interaction. This allows the Bot to make several interaction at the same time with different users without having to wait for the first user to finish to start a new conversation.
- Defy Table: In this table are storage the challenges created by a person in order to defy someone else for the time that the person who have been challenged has not accepted the challenged yet. Once the user has accepted the challenge all the information of the challenge change to the challenge table and are deleted from this table.

Because it is an automatic product, for each function the interaction differs. Therefore, what the product does in each function is going to be explained next.

3.4.2. BOT FUNCTIONS

3.2.2.1 CREATE

In this function the bot asks specific questions to the user about the challenge that he/she wants to perform by themselves. Because it is going to be a personal challenge, the bot needs a third party in order to verify if the user has been able to achieve the goal or not. Therefore, the bot asks for a judge and an email contact in order to send him the description of the challenge and before when it has to be done. When the bot has storage all the information needed to create the challenge the user has the opportunity to change any feature if there is an error in the information storage. To do it, the user only has to select with feature is wrong and the bot will ask again for it. The bot will not create the challenge until all the information is correct, this is when the user presses the OK button to verify that all the information of the challenge is correct.

3.2.2.2 JUDGE

In this function the judges, the third party which function is to verify the performance of the challenge, select if the goal of the challenge has been achieve or not. To do it, firstly the bot ask the judge about the challenge code which is an unique identification for each challenge in the database. Once the bot finds the challenge that it wanted to be judge it send to the judge two options (Achieve or Not Achieve), by pressing any of those two options the challenge is judged and a message is sent to the user who created the challenge.

3.2.2.3. MY CHALLENGES

In this function the user ask the bot about the challenges that they have done in the past and those that they already have currently to perform. The bot search in the Challenge table for all the challenges that have as User ID the same one as the user who is asking for it. After that a complete list of the challenge is send to the user and he/she can decide which one want to see in detail by pressing in it description. After pressing in one of them, the complete description and status of the challenge is sent to the user.

3.2.2.4. CHALLENGE

In this function the user wants to defy someone else to perform a challenge that he/she is going to define or on the other hand, accept or deny a challenge that have been proposed to him. In the first option, the user define a challenge and who is going to be defy, the bot with the database and CRM automatization send an email to this person. Once the person that have been challenged receive the email and choose the challenge function, he/she select the second option which is though to see the challenge that have been proposed to them and decide to take it or leave it. If they take it, the challenge is created and storage in the challenge table if not, the challenge is deleted from the temporal Defy table.

3.5 CONCLUSION

This chapter has explained in detail the product in which the company is based, the Telegram bot and the database related to it. These two products combine represent the most important part of the value proposition that the company offer the clients and potential clients and for that reason is one of the most important part of the project. Moreover, both are the most time-consuming part of the project as well as labor intensive as well as the design of the BMC which was exposed in the previous chapter.

4. CHAPTER 4: PRODUCT-MARKET FIT

4.1 INTRODUCTION

In this chapter it is going to be exposed the test of the product explained in the chapter before with the market what it is called, the product market fit. It is the degree to which a product provided by a company, in this case the startup Getyourchallenge, satisfies the market demand.

The product market-fit is the main focus along the first step of a startup because it is necessary to validate the idea and verify that there is a real market demand that is going to find useful the tool provided by the company.

4.2 TESTING THE PRODUCT

For testing the product with the market, the company has to follow a strategy. In the case of Getyourchallenge as it was explained in the product development chapter the product sketches has been tested in a continuous bases with a circle of confidence and the product have been pivoting with the insights that had been identified.

However, neither of this test have been meaningful in relation to the product market-fit because there were not enough users and interaction to test the necessity and value that the potential users can perceive from the product. For this purpose a launching strategy has to be implemented.

The main focus of the test is to reach as many potential users as the company can in the less time possible because what the company needs is to create the network effect that was mention above. This network effect depends on the number of users that start using the product and by having a good interaction with it, start spreading the value of the product to their friend, work and family circle.

In order to achieve the objective, the test at the beginning, with no monetary resources, is based on three tools: a promotional video, a landing page and a survey for the users after using the product.

4.2.1 PROMOTIONAL VIDEO

Firstly, the users need to know the value that the product provides to them. For this purpose, a promotional video explaining what the user can do and how to do it is essential.

The video should attract users to interact with the bot and because of that, the video has to explain clearly and without any doubt how the user can easily start a conversation with the bot. The second feature that the video needs is to make the potential user feel motivated, which is one of the main characteristics of the core clients of the company. By seeing the video, the potential users have to start thinking in all these challenge that hey had or they already have and see themselves achieving it at least.

Finally, the video has to be sent to the potential clients to do that, the channels explained in the business model canvas will do this task.

4.2.2 SURVEY

The last tool used to evaluate the product market fit and learn from the interaction of the early users of the product is the survey.

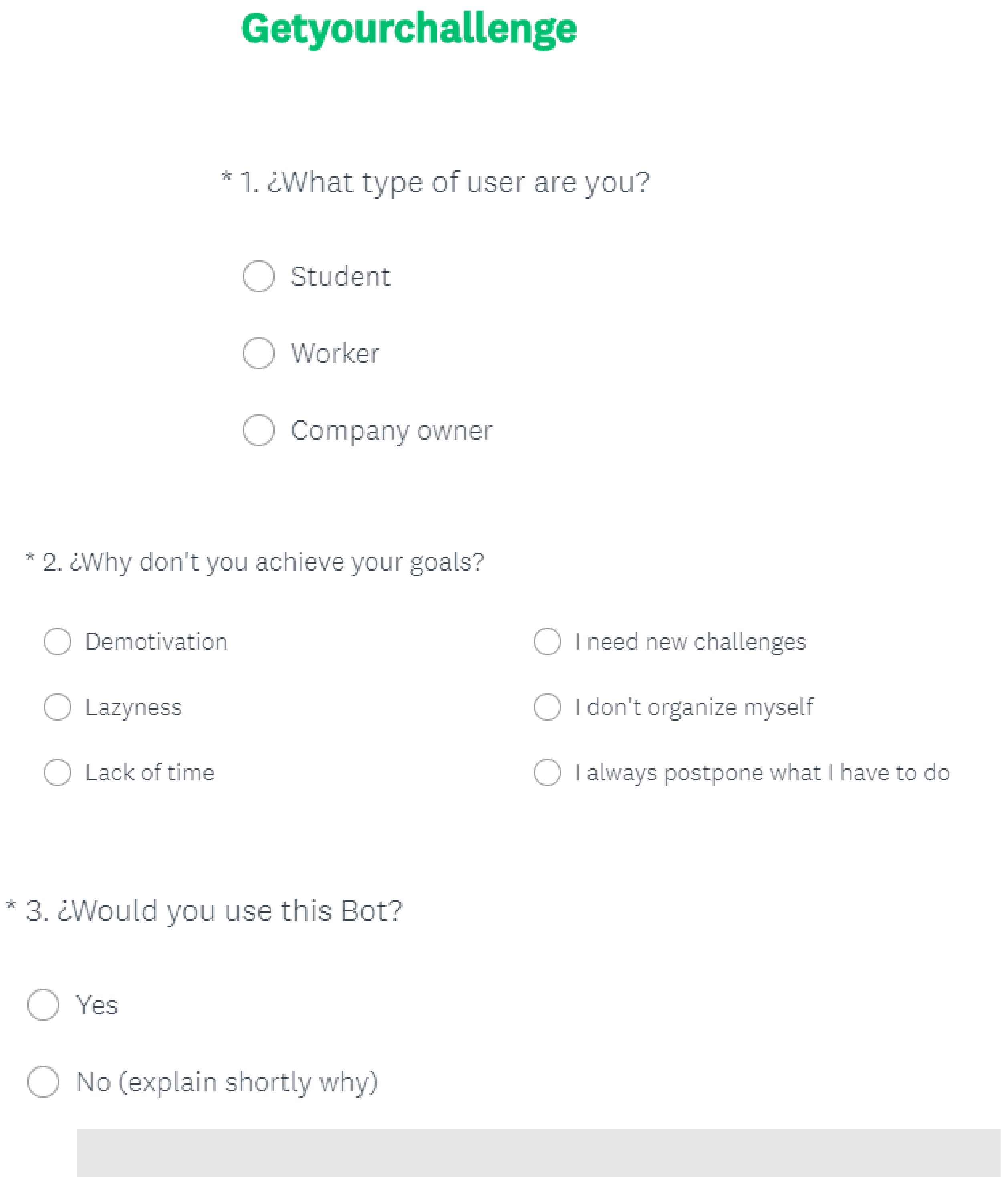
The function of the survey is to gain insights of which way to pivot the product and what to improve or develop in the product to completely satisfy what the clients demands from the service.

For that purpose, the following survey was created and sent to all the early users with whom the service was tested before the launching process.

For that purpose, Getyourchallenge decided to test the product using a minimum viable product (MVP) with the clients and also ask them to answer a quick survey.

The MVP used had all the features of the final product except from the payment part because it was not necessary to test how was the interaction between user and the Bot.

After the interaction they were asked to fill the following survey:



Getyourchallenge

* 1. ¿What type of user are you?

○ Student

○ Worker

○ Company owner

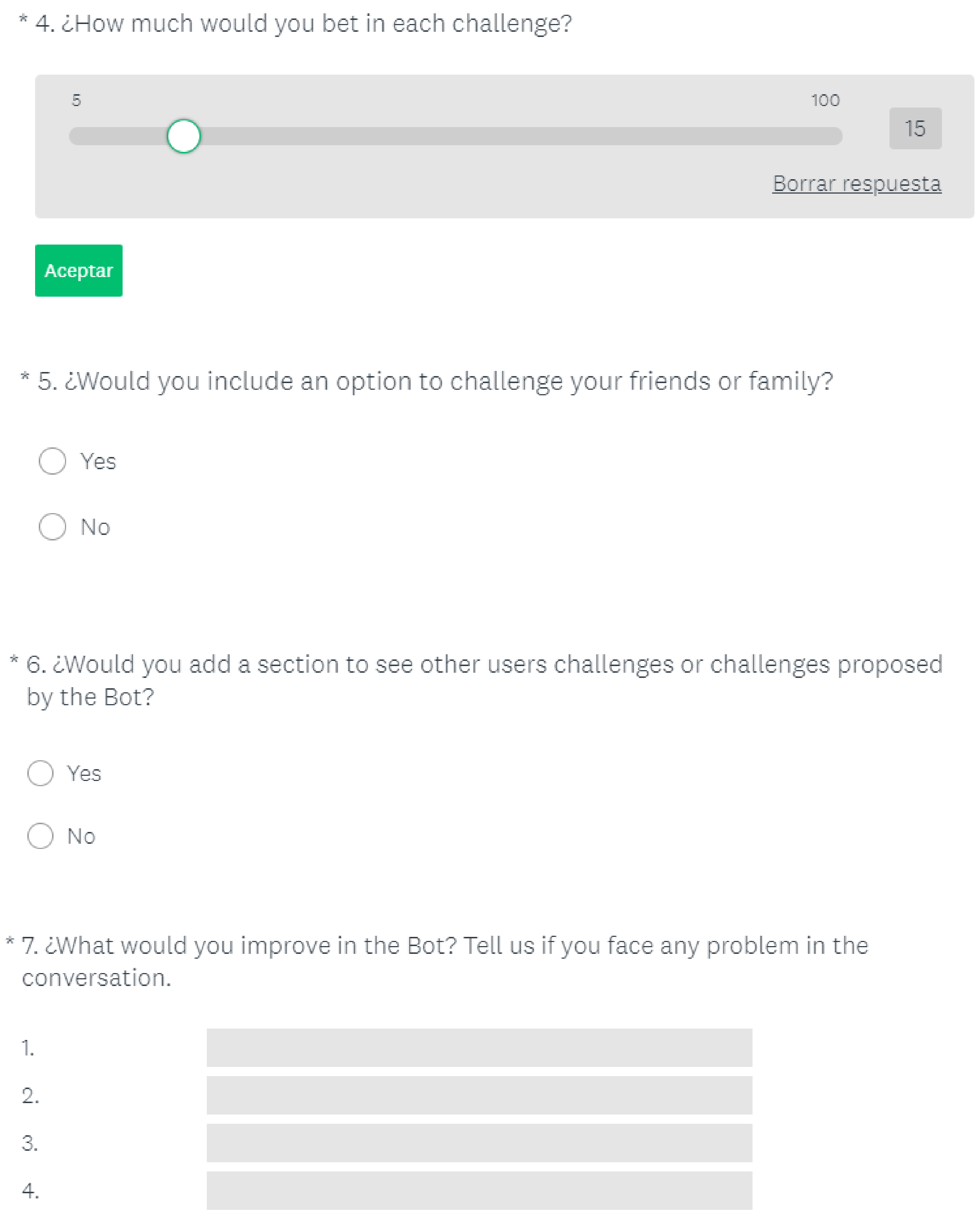
* 2. ¿Why don't you achieve your goals?

○ Demotivation	○ I need new challenges
○ Lazyness	○ I don't organize myself
○ Lack of time	○ I always postpone what I have to do

* 3. ¿Would you use this Bot?

○ Yes

○ No (explain shortly why)



* 4. ¿How much would you bet in each challenge?

5 — 100 | 15

Borrar respuesta

Aceptar

* 5. ¿Would you include an option to challenge your friends or family?

○ Yes

○ No

* 6. ¿Would you add a section to see other users challenges or challenges proposed by the Bot?

○ Yes

○ No

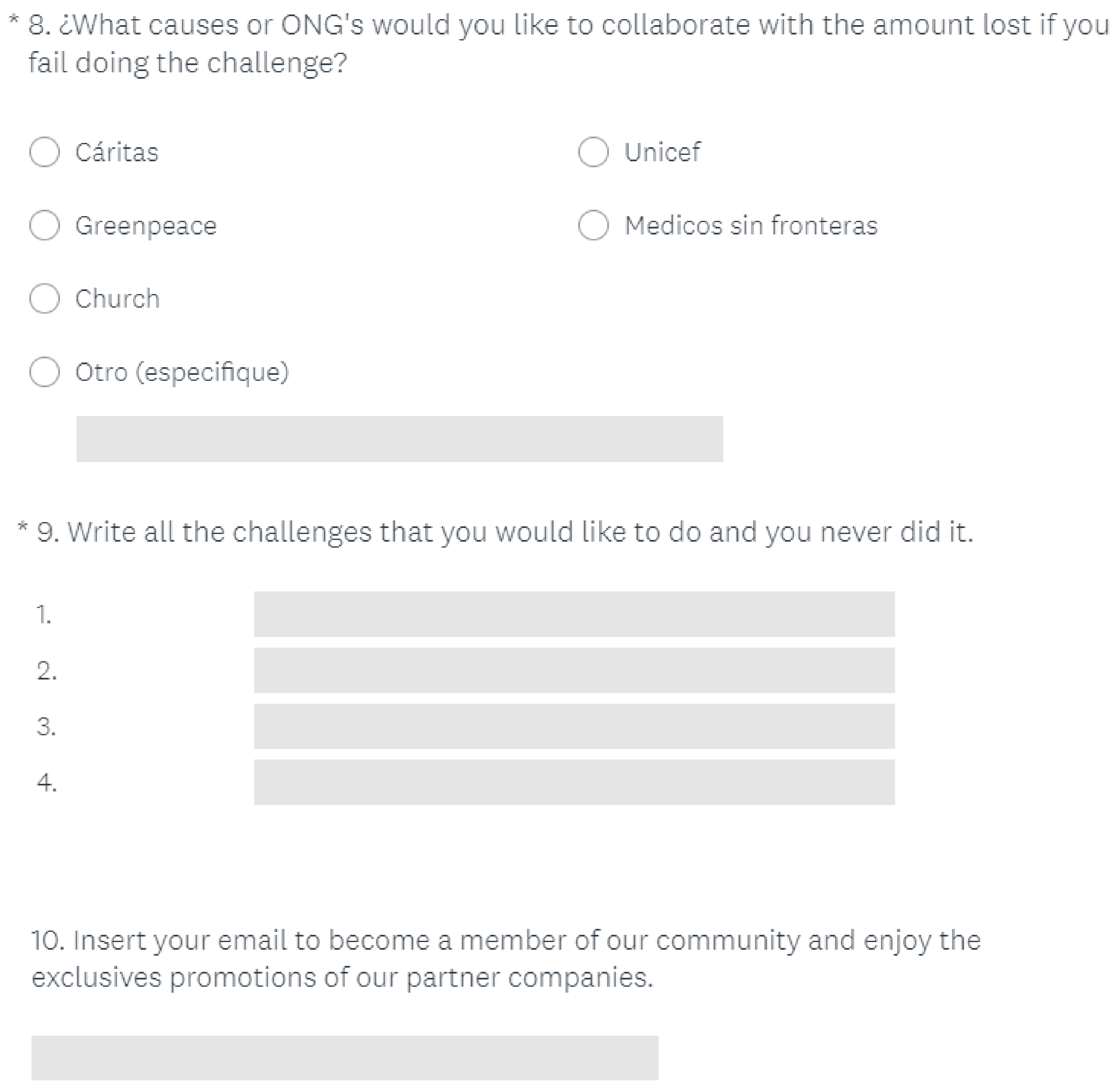
* 7. ¿What would you improve in the Bot? Tell us if you face any problem in the conversation.

1.

2.

3.

4.


* 8. ¿What causes or ONG's would you like to collaborate with the amount lost if you fail doing the challenge?

○ Cáritas	○ Unicef
○ Greenpeace	○ Medicos sin fronteras
○ Church	
○ Otro (especifique)	

* 9. Write all the challenges that you would like to do and you never did it.

1.
2.
3.
4.

10. Insert your email to become a member of our community and enjoy the exclusives promotions of our partner companies.

Figure 48. Survey

4.3 RESULTS

This section provides the results once the product has been tested with the potential users, this was done using the promotional video, the bot and the survey with friends and family and also expanding the circle to family and friends of these first users.

Firstly, the result and insights taken from the survey are going to be explained and after that the number of users and challenges created by these first users collected in the database.

Finally, is important to mention that as the first target is the national Spanish market the survey results are shown in Spanish, because the company wanted to reach the higher number of answer and sending an English survey will probably decrease this number because some people will need more time to complete it.

4.3.1 SURVEY RESULTS

After fifteen days of sharing the promotional video to potential clients and the survey, the results are the following.

The survey was completed by 174 potential clients. Each question insights and results are provided below:

Q1. WHAT TYPE OF USER ARE YOU?

Answered: 174 Skypped:0

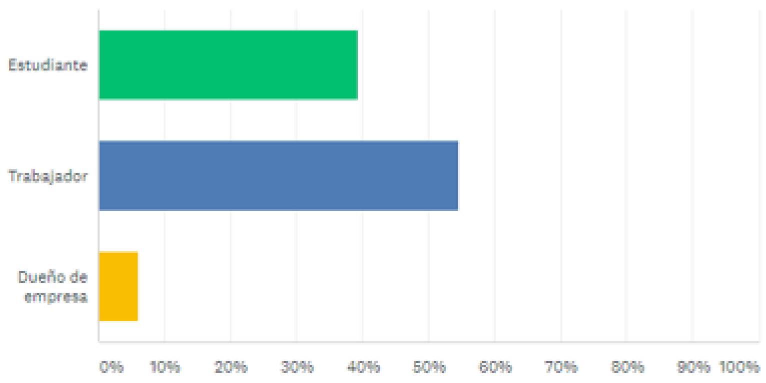


Figure 49. Survey results question 1

More than 90% of the people who took the survey was either workers or students at this moment. Most of them are shown to be workers, however, most of them are people who have finished their studied this year and are starting their professional carrers.

From this question the company see that they will need different channels to reach owner of companies who would like to make a partnership or a contract to promotion their companies to the users of the bot. However, it makes sense that most of the users are not owners of companies.

Q2. WHY DO NOT ARCHIEVE YOUR GOALS ?

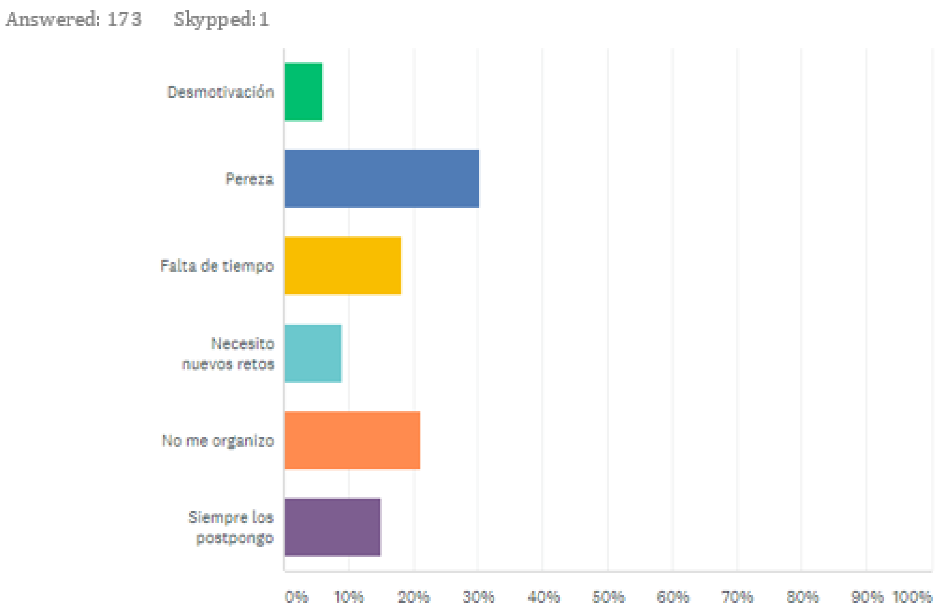


Figure 50. Survey results question 2

This question is so important to know the problems of the potential users and to improve the bot in order to relieve these problems and potential the use of the bot of these clients.

From the results the company infer that the most important driver that lead the people to not reach their goals is laziness, followed by lack of organization and lack of time. These three represent almost 70% of the motives why people do not reach their goals.

Moreover, as all the items exposed were selected, we can validate that the problems that was infer for the potential users were correct.

Q3. WOULD YOU USE THIS BOT ?

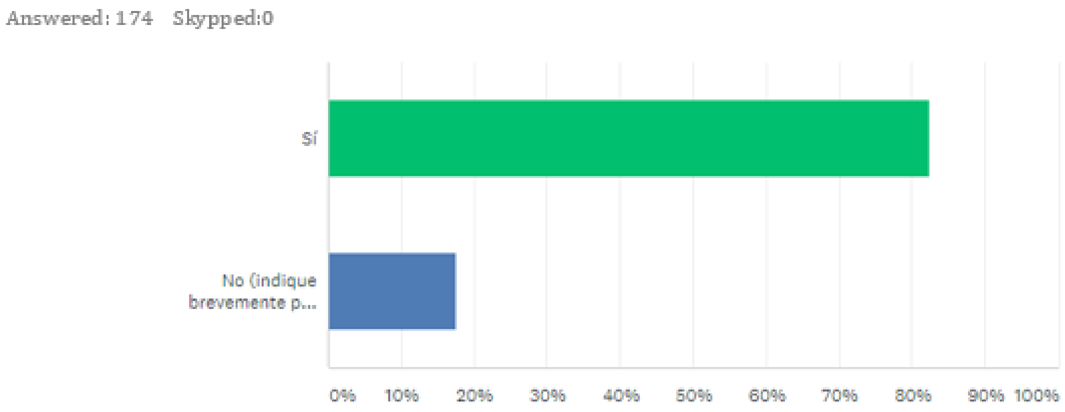


Figure 51. Survey results question 3

More than 80% of the 174 people who answered the survey said that they will use the bot. This is a pretty good ratio and the company can used it to show the potential investor the engagement and validation of the idea.

Apart from that the most important insights are taken from the negative's responses. The people answered why they will not use the bot and the most repeated responses were:

- The bet will not be enough to make me reach my goal
- I can cheat the bot easily and therefore I will not improve my laziness
- I will not bet money against myself

The first answer is important because we can infer that for some challenges and people, the bet should be so high in order to commit to it. The higher the bet the higher the revenues for the company if the bet is not performed.

On the other hand, the other two responses are perceived as a type of potential user that is not going to be interested in the bot. Because one of the main principles of the idea is the willingness of the user to do the challenge and therefore it is sustained in the credibility of both the user and the judge.

Q4. HOW MUCH WOULD YOU BET IN EACH CHALLENGE ?

Answered: 174 Skypped:0

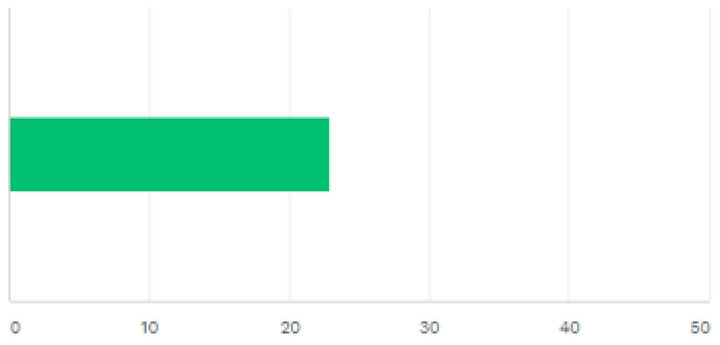


Figure 52. Survey results question 4

This question is related with one of the most important drivers of the revenue of the company, the average bet per challenge. The answer validates the normal scenario analyzed in the financial planning section, which was the one used to forecast the revenues and profits of the company in the near future.

Q5. WOULD YOU LIKE TO INCLUDE AN OPTION TO CHALLENGE YOUR FAMILY AND FRIENDS ?

Answered: 173 Skypped:1

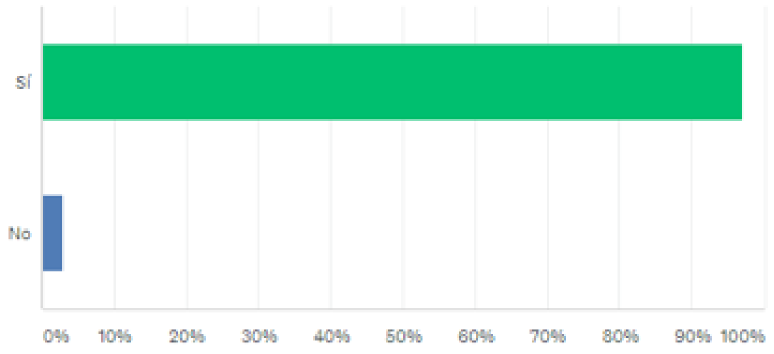


Figure 53. Survey results question 5

Due to this results, the first MVP launched by the company to the first users was improved adding the option of defying someone else to perform a challenge that you create for him/her and with the amount of the bet that you decide.

The MVP which is explained in the section 3 already has this section added due to the response obtained in the survey.

This is a really powerful tool for the company because by letting the users challenge each other the engagement will increase and also the number of users because if someone perform a challenge and after that want to do challenges with another person this users will spread the bot all over his/her contacts increasing the pool of users of the company and the number of challenges created.

Q6. WOULD YOU LIKE TO ADD A SECTION TO SEE OTHERS CHALLENGES OR CHALLENGES PROPOSED BY THE BOT ?

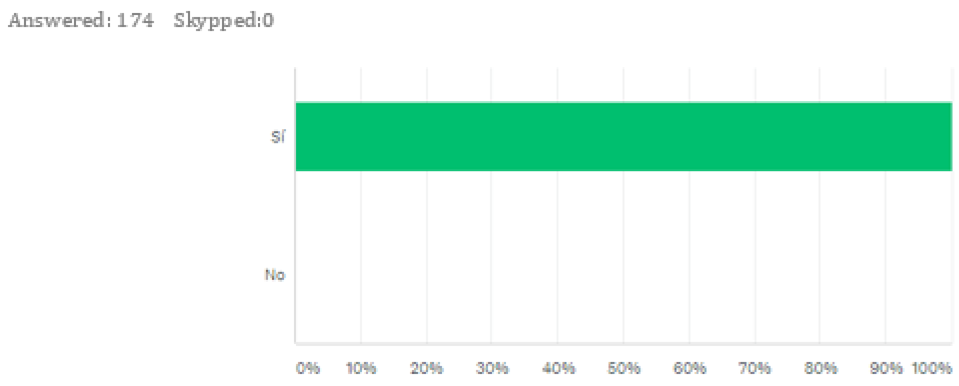


Figure 54. Survey results question 6.

The 174 ones polled would like to add this type of section. From this answer the landing page was modified. A new section explained above, Challenges, was created and its objective is to present the most common goals that the users are performing with the help of the bot in order to give ideas and new challenges to other users.

Moreover, the company will try to implement a new option in the bot in order to help the users to find new challenges without having to look into the website. **This is another example of pivoting after the interaction with clients.**

Q7. WHAT WOULD YOU IMPROVE FROM THE BOT ? TELL US IF YOU FACE ANY PROBLEM DURING THE CONVERSATION.

This question was only answered by around a half of the people (48%) who took the survey. It has sense because people who only have seen the promotional video or people who has not seen any problem or thing to improve from the bot have not answered it.

However, some improvements were suggested by the ones polled, the most meaningful for the company are listed below:

- Create an app instead of having inside Telegram.
- Group challenges per groups of people with similar interests
- Include reference of people who have performed similar challenges using the bot.
- Send motivation messages to the user while they are supposed to be performing the challenge.
- Possibility of creating group challenges, with more than one user and more than one bet.
- Improve the interface, created one more attractive.
- Include incentives if you perform your challenges.
- Increase the velocity of response from the bot.
- Possibility of implementing on WhatsApp.
- Send alerts or reminders to the users while they are supposed to be performing the challenge.
- Do not send money to ONG's

With all this knowledge from the user's perspective of the bot the company can pivot the product and develop a improve product that will satisfy the necessities that the users are providing. Moreover, some the improvements presented by the ones polled can be used to increase one of the main principles of the bot, the engagement, and the increase in revenues and profits too.

Q8. WHAT CAUSES OR ONG'S WOULD YOU LIKE TO CONTRIBUTE WITH THE AMOUNT LOST IF YOU FAIL TO DO THE CHALLENGE ?

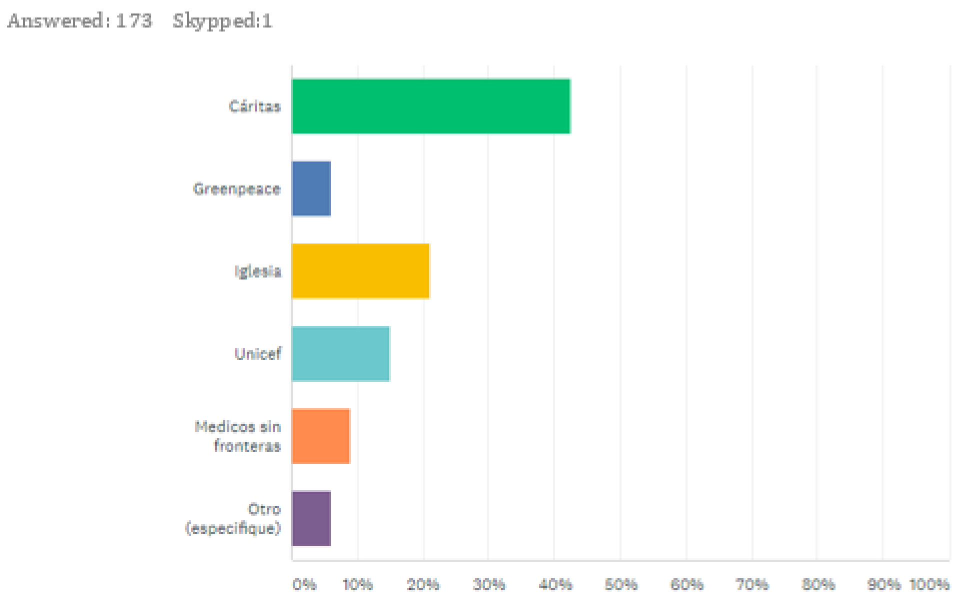


Figure 55. Survey results question 8

After doing this question it is seems that there are several options in which the users will want to contribute. However, the most important segment for the company is the Other (6.06%) which will make infer that some people are not using the bot because of this tool.

However, for futures surveys the question will be modify in order to take from this question the % of people who have real interest in using the bot due to the option of collaborate and which % of people does not take care of this. From this answer the company will decide to change another important parameter from the financial planning, the % of the bet that are revenue. If the company can increase this parameter, the revenues and profits will be multiplied.

Q9. WRITE ALL THE CHALLENGES THAT YOU WOULD LIKE TO DO AND YOU HAVE NEVER DONE ?

The result of this questions is used to implement the new section of the landing page in which the company present and suggest new challenges for the users and potential users.

The most common challenges that the people who answered the survey want to perform but they never for whatever reason done are:

- Improve the sport condition and increase sport training, go to the gym, improve in some sports (golf, athletics,yoga, swimming..)
- Read more
- Do a social labor: voluntary services
- Take a healthy diet, loose weight
- Learn languages, cooking, programming languages, musical instruments, new sports
- Improve my working activities
- Do a startup, create a business
- Improve in my studies
- Write a book
- Travel
- Stop smoking
- Take some certifications, CFA, programming certificates…
- Save money.

Q10. SEND US YOUR EMAIL TO ENTER IN OUR COMMUNITY AND REVEIVE SPECIAL PROMOTIONS.

This question was posted to create a community of potential users and also having their emails the company will use them to direct a special retargeting marketing campaign to increase the number of users or number of challenges per user.

Moreover, having a direct contact with the users is the way to continue knowing their pains and interest in order to update the challenges presented by the company and also the function of the bot.

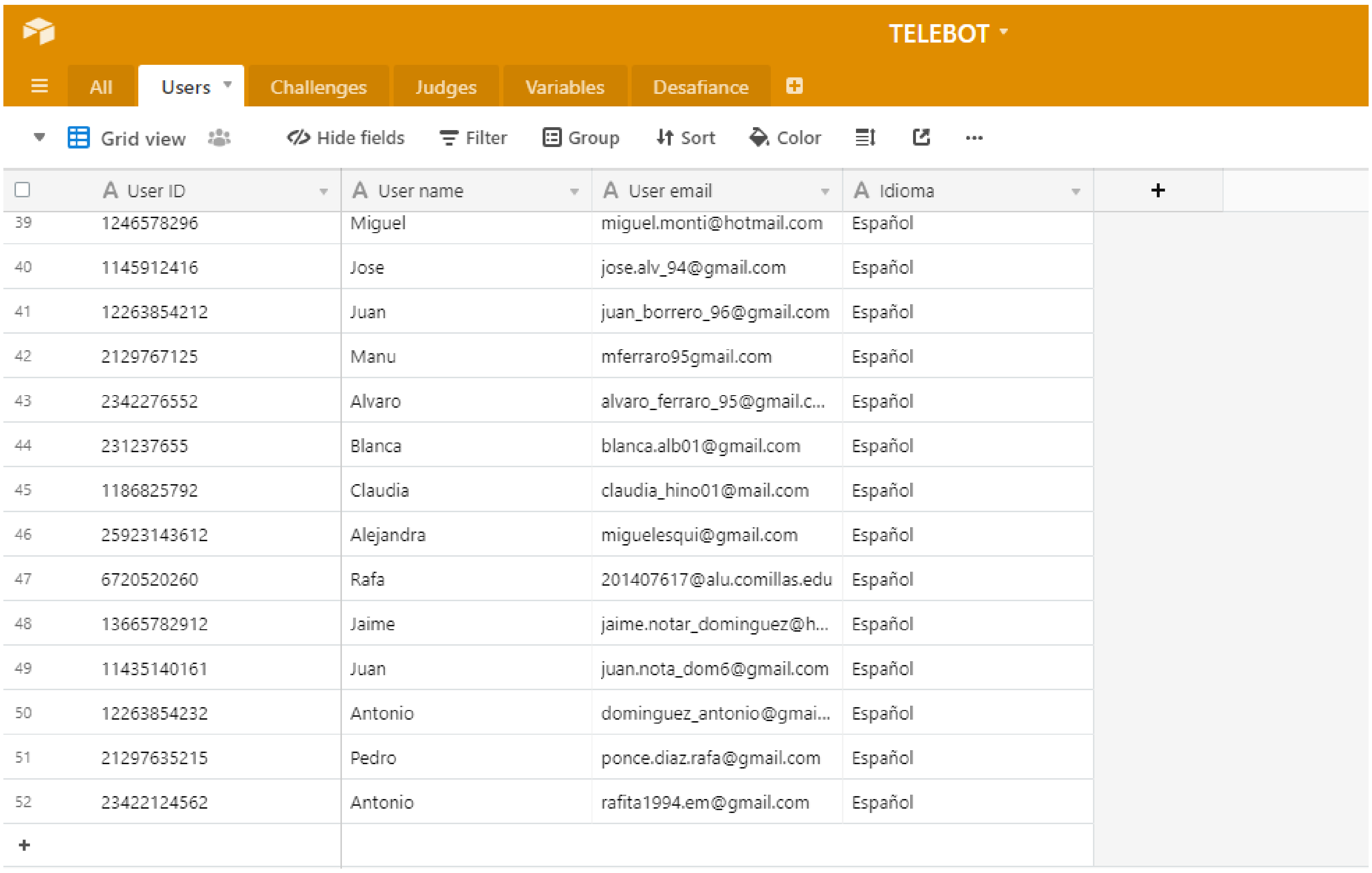
The result of the question was not as successful as the others because some people is more reticent to give their personal emails because they do not want spam and emails from sources that do not matter to them. Only 43% of the ones polled, 75 users, gives us their personal email.

4.3.2 DATABASE RESULTS

The results obtained in the database show the number of users, number of challenges created, and number of judges used in the probation period of the MVP.

It is important to mention that this results represent only information data because the product has not been launched yet and the information it has been used to infer the starting number of users from the beginning of January 2021, when it is expected to be launched the product.

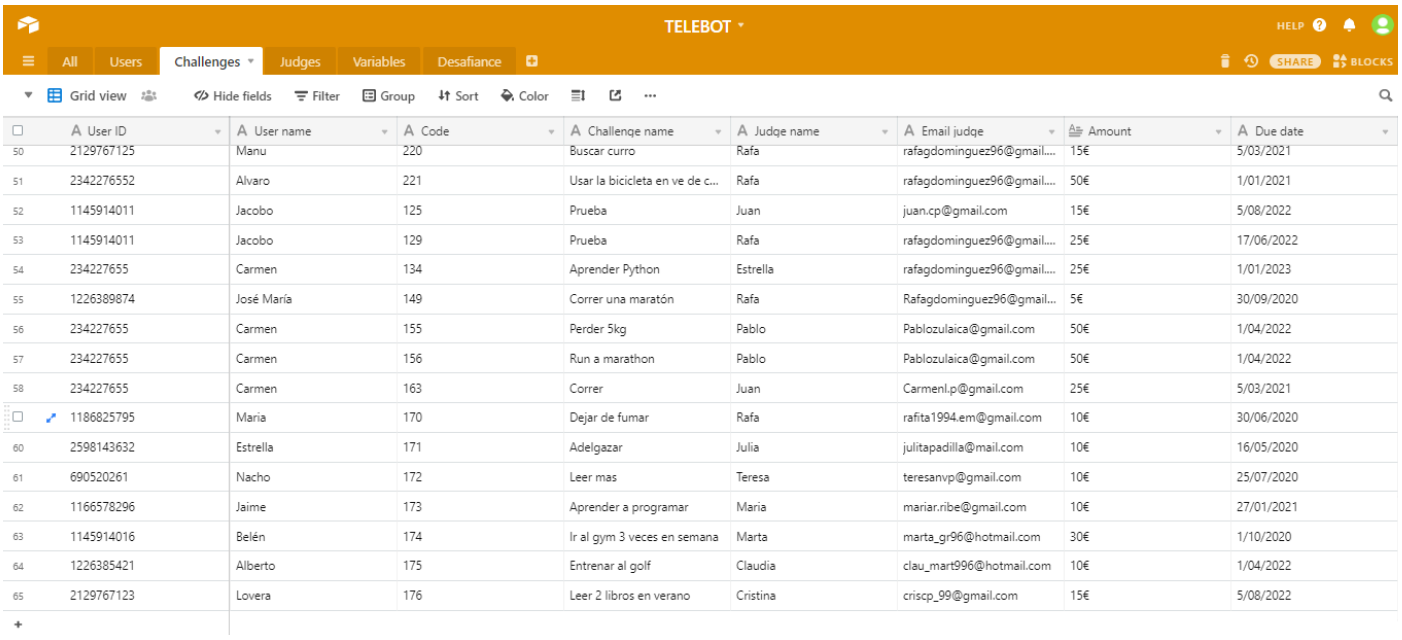
In August 2020, the number of users is 52.



	User ID	User name	User email	Idioma
39	1246578296	Miguel	miguel.monti@hotmail.com	Español
40	1145912416	Jose	jose.alv_94@gmail.com	Español
41	12263854212	Juan	juan_borrero_96@gmail.com	Español
42	2129767125	Manu	mferraro95gmail.com	Español
43	2342276552	Alvaro	alvaro_ferraro_95@gmail.c...	Español
44	231237655	Blanca	blanca.alb01@gmail.com	Español
45	1186825792	Claudia	claudia_hino01@mail.com	Español
46	25923143612	Alejandra	miguelesqui@gmail.com	Español
47	6720520260	Rafa	201407617@alu.comillas.edu	Español
48	13665782912	Jaime	jaime.notar_dominguez@h...	Español
49	11435140161	Juan	juan.nota_dom6@gmail.com	Español
50	12263854232	Antonio	dominguez_antonio@gmai...	Español
51	21297635215	Pedro	ponce.diaz.rafa@gmail.com	Español
52	23422124562	Antonio	rafita1994.em@gmail.com	Español

Figure 56. Number of users August 2020

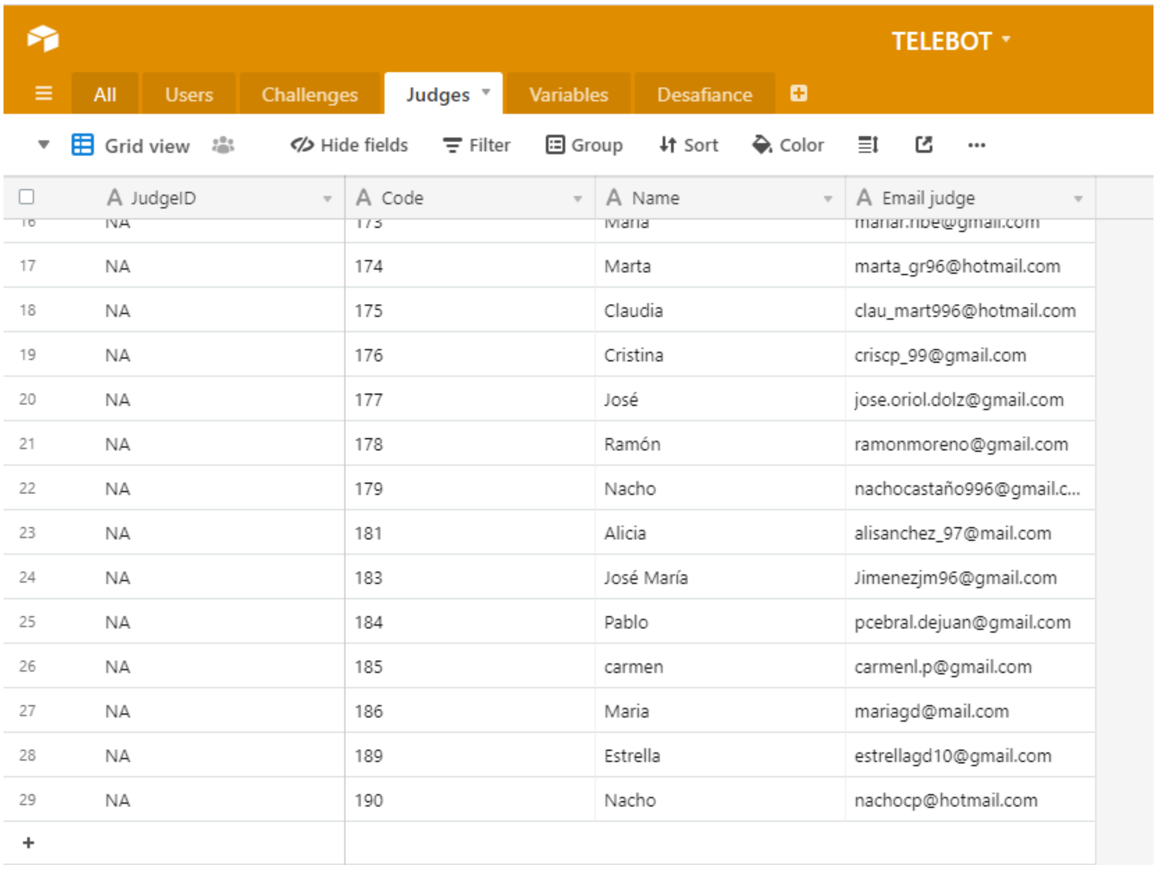
These users have created 65 challenges which create a rate of 1.25 challenges per user in a month, because the validation period started in July.



	User ID	User name	Code	Challenge name	Judge name	Email judge	Amount	Due date
50	2129767125	Manu	220	Buscar curro	Rafa	rafagdominguez96@gmail....	15€	5/03/2021
51	2342276552	Alvaro	221	Usar la bicicleta en ve de c...	Rafa	rafagdominguez96@gmail....	50€	1/01/2021
52	1145914011	Jacobo	125	Prueba	Juan	juan.cp@gmail.com	15€	5/08/2022
53	1145914011	Jacobo	129	Prueba	Rafa	rafagdominguez96@gmail....	25€	17/06/2022
54	234227655	Carmen	134	Aprender Python	Estrella	rafagdominguez96@gmail....	25€	1/01/2023
55	1226389874	José María	149	Correr una maratón	Rafa	Rafagdominguez96@gmail...	5€	30/09/2020
56	234227655	Carmen	155	Perder 5kg	Pablo	Pablozulaica@gmail.com	50€	1/04/2022
57	234227655	Carmen	156	Run a marathon	Pablo	Pablozulaica@gmail.com	50€	1/04/2022
58	234227655	Carmen	163	Correr	Juan	Carmenl.p@gmail.com	25€	5/03/2021
	1186825795	Maria	170	Dejar de fumar	Rafa	rafita1994.em@gmail.com	10€	30/06/2020
60	2598143632	Estrella	171	Adelgazar	Julia	julitapadilla@mail.com	10€	16/05/2020
61	690520261	Nacho	172	Leer mas	Teresa	teresanvp@gmail.com	10€	25/07/2020
62	1166578296	Jaime	173	Aprender a programar	Maria	mariar.ribe@gmail.com	10€	27/01/2021
63	1145914016	Belén	174	Ir al gym 3 veces en semana	Marta	marta_gr96@hotmail.com	30€	1/10/2020
64	1226385421	Alberto	175	Entrenar al golf	Claudia	clau_mart996@hotmail.com	10€	1/04/2022
65	2129767123	Lovera	176	Leer 2 libros en verano	Cristina	criscp_99@gmail.com	15€	5/08/2022

Figure 57. Number of challenges created by August 2020

Finally, the users during this time have used 29 different judges, which are collected in the judge database.



	JudgeID	Code	Name	Email judge
16	NA	173	Maria	mariar.ribe@gmail.com
17	NA	174	Marta	marta_gr96@hotmail.com
18	NA	175	Claudia	clau_mart996@hotmail.com
19	NA	176	Cristina	criscp_99@gmail.com
20	NA	177	José	jose.oriol.dolz@gmail.com
21	NA	178	Ramón	ramonmoreno@gmail.com
22	NA	179	Nacho	nachocastaño996@gmail.c...
23	NA	181	Alicia	alisanchez_97@mail.com
24	NA	183	José María	Jimenezjm96@gmail.com
25	NA	184	Pablo	pcebral.dejuan@gmail.com
26	NA	185	carmen	carmenl.p@gmail.com
27	NA	186	Maria	mariagd@mail.com
28	NA	189	Estrella	estrellagd10@gmail.com
29	NA	190	Nacho	nachocp@hotmail.com

Figure 58. Number of judges by August 2020

4.4 CONCLUSIONS

Analyzing the results obtained in the validation phase of the product, the company validate the idea and the product represented by the MVP.

This conclusion is sustained in the data presented above. After spreading the promotional video, creating the landing page and the survey.

The product satisfies some of the problems that the users have and therefore it seems to be a good tool to be used by them. Moreover, using the results of the survey the MVP has been improved, adding a new option in which the users can defy someone else to perform a challenge that the person who is defying is going to create.

On the other hand, the result show that right now there is no real engagement with the product because the average number of challenges created per user us 1.25, a low ratio in relation with the low mass of users that the company has at this moment. What is more, none of the challenged created have been judge.

This two aspects has been studied and the conclusions are that while the product it is not in a server and the payment platform is not ready to use the users do not really take into account the product and see it as a future idea that can be useful for them.

Finally, once the idea has been validated the company has to focus in two new aspects, which are explained briefly in the following section.

- How to launch the product
- Maintain and increase engagement with the users.

4.5 NEXT STEPS

With the idea validated the main next step are to put the product available to everyone and with all its function.

To do it the company has to split the launching in two different parts:

- The first one: Implement an external payment platform or on the other hand try to implement into the code of the bot the telegram payment option. After that, upload the final product into a server in order to be available for everyone who has Internet access.

- The second one the company has to design a launching strategy.

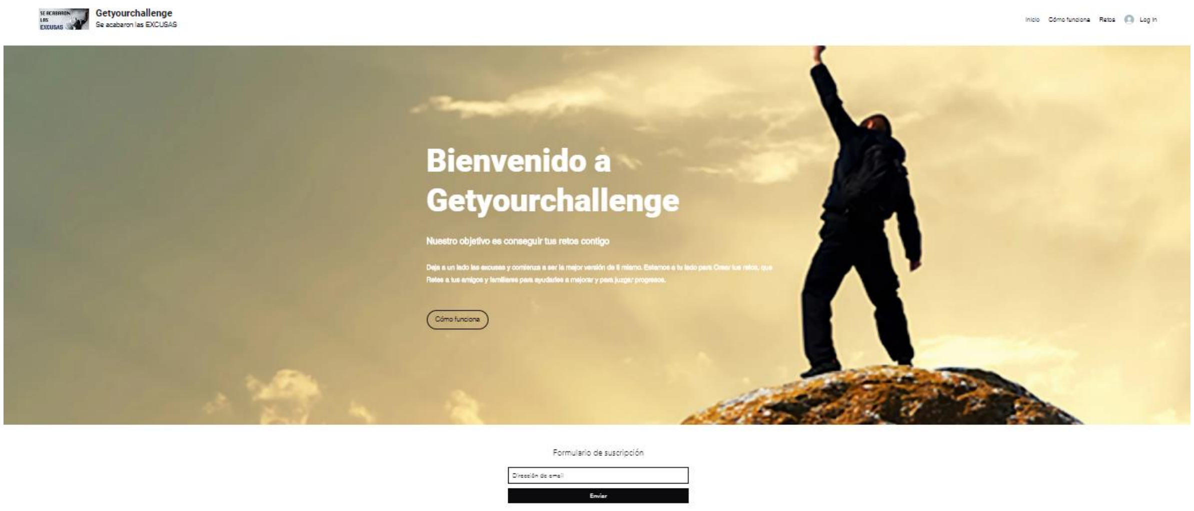
For the launching strategy the company will follow the same strategy used in the validation phase but in a big scale. An improve promotional video and a landing page will be spread to potential users.

4.5.1 LANDING PAGE

The landing page represent the image of the company. The first function of the website is to gain credibility for the users and possible investors. Besides, nowadays, whenever you are going to buy a product or use a service from a company the first thing you do is to search the company in Internet to understand what they do.

Secondly, the service offer by Getyourchallenge for so many people, potential users, is disruptive in a way because for some of them Telegram is a new app that they have never used and therefore they do not know how to interact with the bots and they will need some guide in doing it.

The website is segmented in three pages. The first one is the home page, with a motivating image and text, which is related to what the user needs. Also, there is stated what is the company offering and what the company value proposition to the user is.


Getyourchallenge
Se acabaron las EXCUSAS

Bienvenido a Getyourchallenge

Nuestro objetivo es conseguir tus retos contigo

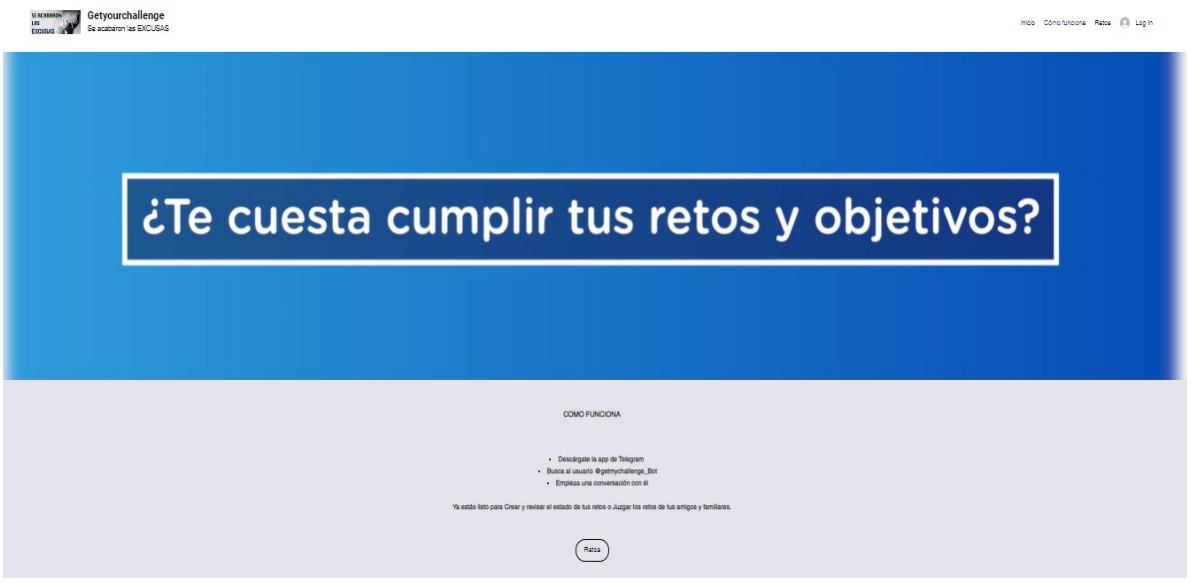
Deja a un lado las excusas y comienza a ser la mejor versión de ti mismo. Estamos a tu lado para Crear tus retos, que Retes a tus amigos y familiares para ayudarles a mejorar y para juzgar progresos.

Formulario de suscripción

Dirección de email

Figure 59. Website home page

Secondly, the next page is the one in which the product is explained in detail. For that purpose, the promotional video is posted there with the instruction written to help the user.



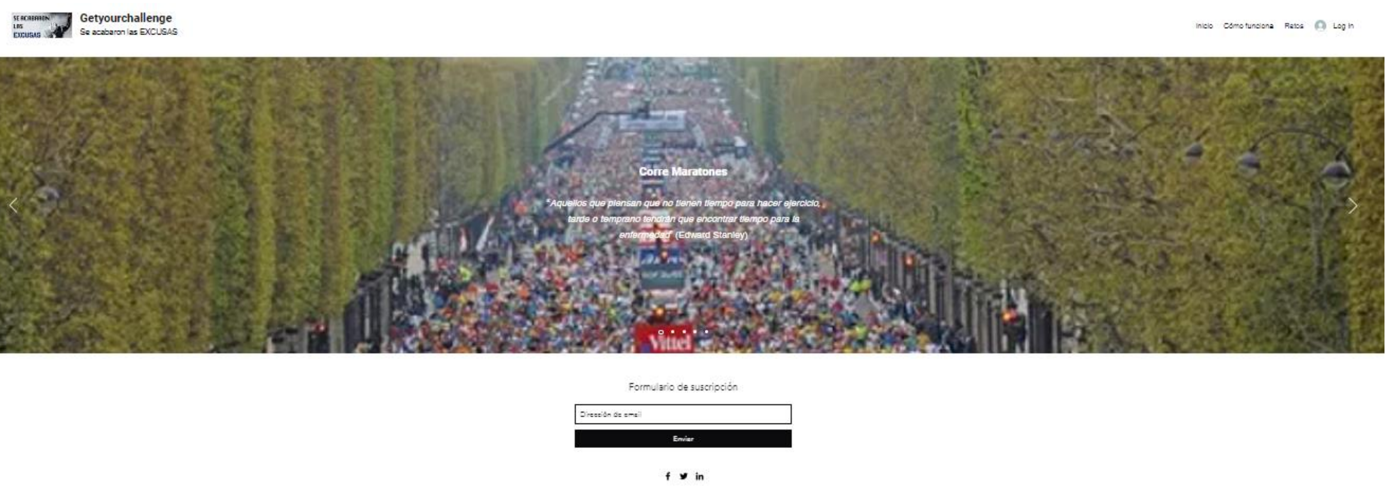
COMO FUNCIONA

- Descárgate la app de Telegram
- Busca al usuario @getmychallenge_Bot
- Empieza una conversación con él

Ya estás listo para Crear y revisar el estado de tus retos o Juzgar los retos de tus amigos y familiares.

Figure 60. How the product works page

Finally, the last page of the website consists in a list of different types of challenge that the users of the Bot can pursue. The function of this page is to increase the engagement between the company and the clients. As soon as new challenges types are created in the Bot for the users, this information will be entered in this page and the value will increase for both the users and the company. The last objective will be to create a community of users of all these people that using the Bot have increased their willingness to pursue new challenges and decrease the procrastination.



"Aquellos que piensan que no tienen tiempo para hacer ejercicio, tarde o temprano tendrán que encontrar tiempo para la enfermedad" (Edward Stanley)

Figure 61. Challenges page

Furthermore, the potential clients that enter in the website and are interested in the idea presented by the company can enter their mails in the box which is placed at the bottom of the website in order to develop a direct advertising campaign to launch the product.

To find sources to spread this information the company can follow the strategy that the competitor "Go Fucking Do It" did. Try to appear in newspaper, internet blog focusing in startup, blog related to entrepreneurship and also telegram groups in which there will be people interested in this type of bots in order to increase the spread.

The second step that the company has to overtake in the near future is to maintain the relation with the users and clients. For that purpose and seeing the good results obtained from the validation survey, new surveys and personal contact with the users is going to be really important for the future of the company.

The obsession of the company has to be to cover all the problems, necessities and inquiries that the users and potential users have because that is the path to increase the engagement with them.

5. CHAPTER 5: CONCLUSIONS

5.1 INTRODUCTION

In this chapter are exposed the conclusions taken out from the development of the project during the academic year 2019-2020. These conclusions are split in two, firstly, the technical conclusion around the project are exposed and after that some personal conclusions and knowledge learned during the time employ in the project.

5.2 PROJECT CONCLUSIONS

After detecting a problem in the society, developed a solution to this problem, validate this solution with the market and creating a business around this concept, some important conclusions are drawn from it.

1. It has been analyzed a problem in the actual society, in particular in some segments of the population. This problem is related with the laziness, disorganization and lack of willpower that are retaining people from achieving their personal challenges in their lives.
2. To solve this problem a software solution has been designed. This software is a Telegram Bot, which serves as a conversational tool that help the users to create challenges, judge other users challenge, see the user actual challenges or finally, challenge someone else. This Bot implements a money motivation system in which the person who want to develop the challenge bet an amount of money that will be lost if the user does not reach the challenge created. This is sustained because it is proved that the main motivator nowadays is money for most of the society.
3. A MVP (Minimal Viable Product) has been created and tested with potential clients. Each time it has been tested, the company has pivot with the feedback from the users and the MVP has been improved. For instance, adding an option to challenge someone else using the product or adding also an option to see the challenges that an user has with the bot.
4. Using the last MVP, which is explained in detail in this document, the company has validated the idea spreading a promotional video explaining how the product works, and also a survey to give some feedback that was used to validate the idea and find

new ways to satisfy the necessities of these potential users in the future, adapting the product.

5. Around the product created and validated, a business strategy has been created. For that purpose, the BCM (Business Canvas Model) has been used. With this tool the main agents and the relation with the company are explained briefly and clearly.
6. Finally, some financial previsions have been performed in order to see the profitability of the idea created. To this effect some models have been created:
    a. Monthly revenue model for the next three based in these parameters (users, conversion rate, monthly transaction per user, average bet and percentage of revenue from lost bet)
    b. Cost model split in three different expenses by nature (salaries, operating expenses, marketing expenses)
    c. Profitability analysis based on the monthly profit and the accumulated cash flow
    d. Scenario Analysis
    e. Two sensitivity analysis based on 4 revenues parameters

    From this analysis and models, it is expected that the project created is going to be profitable in the long term being the most important advantage its scalability which will lead to abnormal returns when the company reach a base of clients around (20k-40k users) and achieve engagement with them.

5.3 PERSONAL CONCLUSIONS

The project has been realized from September ’19 to August ’20. The type of work realized in it can be split in:

- 60% technical, which involves the time employ in program the product in Python as well as creating the CRM and the implementation of the database to the Bot created).
- 30% of work has been employ creating the business around the product. Inside this type of work are:
    - Developing the Business Model Canvas
    - Creating financial models

- Creating the promotional video and survey
- Validating the idea

- 10% writing the document and doing the final and intermediate presentations.

After completing all this work, I have learned so much during all this time performing the project and the most important thing I have learned are listed below:

1. Program using a new code language, Python. **Learning Python** has been an incredible satisfaction because I have been learning it using online material from another programmers, searching and asking other people in social network as Telegram and Github and always with the help of my director, Pablo Zulaica, who was essential to the development of the project. Finally, I believe that I have developed a new skill and that I will be able to do projects like this one in a few time thanks to this project, which is so important because Python is becoming more and more demanding in engineers because it is a essential tool to analyze huge amount of data more accurate and in less time.
2. Learn the **use of the API's** to combine the use of different apps. This have been used to implement the Telegram API to the code in order to create the Bot and also the Airtable API to implement the database to the product.
3. Uses and how to **create a Telegram Bot** since scratch. For me was completely new because I did not know that this type of tools was being used by companies, however, there are millions of bot created in Telegram and there are so many uses in which they can be used. The most important advantage of them is its scalability because once they are programmed you do not need to spend more resources in them.
4. The **BMC (Business Model Canvas)** philosophy in which are based the majority of the startups worldwide. With this tool well used you can created a solid business and in less time, that can be presented to investor reducing the time and being more concrete in what are the important things around the company that you want to create.
5. How to **validate an idea** in the market and **analyze the data** and feedback received from it. Using this information **to pivot around the initial idea** and perform a best suited solution to the potential clients.

6. **Create a financial planning** base on parameter that can be estimated or compared with potential competitors or other companies with similar strategies. Also created some sensitivity analysis to decide in which parameter focus.

Finally, apart from this skills and knowledge, personally this project has helped me in improving my searching, learning and analyzing skills that have been so important during my degree and also in the master's degree.

6.BIBLIOGRAFÍA

2014. 29 de May. https://levels.io/debriefing-go-fucking-do-it/.

Laura. 2020. *Bitdegree.* 23 de January. https://www.bitdegree.org/tutorials/telegram-bots/#What_is_a_Telegram_bot.

Treuberg, Matthias. 2020. *Creative minds.* 13 de Mayo. https://www.cminds.com/5-best-practices-using-telegram-bots/.

7.ANEXO A

7.1 INTERACTION WITH SDGs

The SDGs (Sustainable Development Goals) are a set of objectives that have been designed in order to create a better and more sustainable future for the entire society. Therefore, this project should be align with some of these objectives because the final goal of a professional is to contribute to a create a better environment and society.

In this section is going to be exposed the relation of the project with the SGDs. Firstly the 17 goals are going to be listed and after that the ones related to the project will be explained in detail with the relation that can be primary (the project has direct impact in the achievement of the goal) or secondary (the project is somehow related to the goal).

The SGDs are the following ones:

1. No poverty
2. Zero hanger
3. Good health and well-being
4. Quality education
5. Gender equality
6. Clear water and sanitation
7. Affordable and clean energy
8. Decent work and economic growth
9. Industry innovation and infrastructure
10. Reduced inequalities
11. Sustainable cities and communities
12. Responsible consumption and production
13. Climate action
14. Life below water
15. Life on land
16. Peace, justice and strong institutions
17. Partnerships for the goals

The goals are split in three categories: Biosphere, Society and Economy. Because of the nature of the project that is related to the society and because is also a way of generating profitability, the goals that are related are inside the categories of society and economy. However, also goals in the Biosphere category are related.

The main goal that are primary related to the project are:

- **Economy:** Industry innovation and infrastructure

Moreover, some other goals are also related but not as directly as the first ones:

- **Society:** No poverty, Good health and well-being, good education
- **Biosphere:** Climate action

PRIMARY RELATIONS

1. **No poverty and climate action**

No poverty: "More than 700 million people, or 10 per cent of the world population, still live in extreme poverty today, struggling to fulfil the most basic needs like health, education, and access to water and sanitation, to name a few. The majority of people living on less than $1.90 a day live in sub-Saharan Africa. Worldwide, the poverty rate in rural areas is 17.2 per cent—more than three times higher than in urban areas."

Climate action: "Climate change is affecting every country on every continent. It is disrupting national economies and affecting lives. Weather patterns are changing, sea levels are rising, and weather events are becoming more extreme. Saving lives and livelihoods requires urgent action to address both the pandemic and the climate emergency."

One of the main characteristics of the project is the monetization of the business in which part of the revenues are not directed to the company, but to a social cause that can be elected by the user.

The social causes that are mainly demanded by the potential market that have been used to validate the idea are related to the eradication of the poverty in the world, helping people who is suffering and on the other hand, social causes related to the conservation of the environment and therefore related to the goal of climate action.

It is no possible to obtain an accurate quantification of the contribution of the project to both causes. Nevertheless, using the financial planning presented in the previous sections an estimation of the aid can be obtained.

The **expected scenario** used a 50% of the transaction as revenue **and the other 50% as contribution to social causes**, therefore the expected **contribution** to causes that can be related to the no poverty goal or the climate action goal after the first three years of the project is **225,501.94 $.** Of course is a small amount and it will not be enough to achieve the goals but is a way to contribute to it and by the contribution of more small amounts like this one both objectives will be achieved.

2. Industry innovation and infrastructure

"Inclusive and sustainable industrialization, together with innovation and infrastructure, can unleash dynamic and competitive economic forces that generate employment and income. They play a key role in introducing and promoting new technologies, facilitating international trade and enabling the efficient use of resources.

However, the world still has a long way to go to fully tap this potential. Least developed countries, in particular, need to accelerate the development of their manufacturing sector if they are to meet the 2030 target, and scale up investment in scientific research and innovation. "

In the project it is presented a technological infrastructure in which several organizations can be sustained. The main advantage of the innovation presented using Telegram Bots, the one created in the project, or other Bots, is the scalability that they provide because they reduce the needs from human capital that only have to perform mechanical activities which can be done by a machine.

By increasing the facility of creating profitable companies reducing the cost of them will encourage more people to create new business which will create a lot more sustainable and impacting employments in compare of the loss jobs that will be change by the implantation of this innovation technology.

The main obstacle will be in the least developed countries in which the access to Internet is not as common as it is in Europe or America. For that reason, more investment are necessary in relation to this goal to achieve the final objective.

SECONDARY RELATIONS

1. Good health and well-being & Good education

Good health and well being: "Ensuring healthy lives and promoting well-being at all ages is essential to sustainable development. Currently, the world is facing a global health crisis unlike any other — COVID-19 is spreading human suffering, destabilizing the global *economy and upending the lives of billions of people around the globe. "*

Good education; "Education enables upward socioeconomic mobility and is a key to escaping poverty. Over the past decade, major progress was made towards increasing access to education and school enrollment rates at all levels, particularly for girls. Nevertheless, about 260 million children were still out of school in 2018 — nearly one fifth of the global population in that age group. And more than half of all children and adolescents worldwide are not meeting minimum proficiency standards in reading and mathematics. "

The project is not primary focus in promoting health or quality education as it is stated in the goals of the SDGs. However, the project tries to promote and encourage both aspect in a less intrinsic way.

The company and the product developed in this project is not directed to improve the health in the least developed world, but, by using the product the users will decide to take challenges in which they will try to achieve healthy habits and promote a healthy society by increasing sports activities, having a healthy diet that finally will help to have a better health as a society.

On the other hand, using the product will not improve the education as a whole, however, it will be useful to encourage the users of the product to pursue challenges related with their education and in that way it will help to improve the education and responsibility of the society in a less deep way.

8. ANEXO B

8.1 PYTHON CODE

```
import logging, json

from telegram import (ReplyKeyboardMarkup, ReplyKeyboardRemove, InlineKeyboardButton, InlineKeyboardMarkup)
from telegram.ext import (Updater, CommandHandler, MessageHandler, Filters,
                          ConversationHandler, CallbackQueryHandler)

from airtable.airtable import Airtable
from datetime import date
from datetime import datetime

# Enable logging
logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s', level=logging.INFO)

logger = logging.getLogger(__name__)

DECISION, CHALLENGENAME, JUDGENAME, JUDGEMAIL, JUDGE, JUDGE1, AMOUNT, DUEDATE, USERMAIL, ADDITIONAL, DATE, FINAL, EXISTING, FINAL1, REVIEW,
DATE1, DUEDATE1, START, VERCHALLENGE = range(19)

# Configuration of Airtable
base_id = 'appQlPzd1zYwknW0b'
key = 'key9nBwO3xeJ3OcEj'
tabla = 'All'
tabla1 = 'Users'
tabla2 = 'Challenges'
tabla3 = 'Judges'
tabla4 = 'Variables'
tabla5 = 'Desafiance'
airtable = Airtable(base_id, tabla, key)
airtable1 = Airtable(base_id, tabla1, key)
airtable2 = Airtable(base_id, tabla2, key)
airtable3 = Airtable(base_id, tabla3, key)
airtable4 = Airtable(base_id, tabla4, key)
airtable5 = Airtable(base_id, tabla5, key)
colCode = 'Code'
colChatid = 'User ID'
colDecision = 'Challenger or not'
colChallengename = 'Challenge name'
colUsername = 'User name'
colJudgename = 'Judge name'
colEmailJudge = 'Email judge'
colUserEmail = 'User email'
colDueDate = 'Due date'
colAmount = 'Amount'
colStatus = 'Status'
colJudgeID = 'JudgeID'
colNamejuez = 'Name'
colIdioma = 'Idioma'
colExistingJudge = 'EligeExisting'
colChange = 'Change'
colNuevoUser = 'NuevoUser'
colMonth = 'Month'
colDay = 'Day'
colCode1 = 'Code1'
colChallenge = 'Challenge'


## PARTE 1
def start(update, context):
    keyboard = [[InlineKeyboardButton("Español", callback_data='Español'),
                 InlineKeyboardButton("English", callback_data='English')]]

    reply_markup = InlineKeyboardMarkup(keyboard)

    update.message.reply_text('Welcome! Please, select your language',
                              reply_markup=reply_markup)

    return START


def start1(update, context):
    idioma = update.callback_query.data
    print("El usuario ha elegido el idioma {}".format(idioma))
    # Actualizamos codigo del challenge
    numero = buscarcode()
    code = int(numero) + 1
    code1 = str(code)
    verify = airtable.search('Code', code1)
    reto = 0
    while (verify != []):
        code = buscarcode()
        code = code + 1
        code1 = str(code)
        verify = airtable.search('Code', code1)

    # Nuevo user o no
    chat_id1 = str(update.effective_chat.id)
    verify1 = airtable1.search('User ID', chat_id1)
    if verify1 == []:
        nuevouser = str(0)
        nuevouser1 = 0
        print("El usuario es NUEVO")
    else:
        ultimoemail = buscaruseremails(chat_id1)
        numerousers = len(ultimoemail)
        verify2 = ultimoemail[numerousers - 1]
        print("Ya conocemos a este usuario")
        if (verify1 != []) and (verify2 != "NA"):
            nuevouser = str(1)
            nuevouser1 = 1
            ultimo = buscarduedates(chat_id1, reto)
            numerochallenges = len(ultimo)
            verify = ultimo[numerochallenges - 1]
            codes = buscarcode2(chat_id1)
            numerochallenges = len(codes)
            ultimocode = codes[numerochallenges - 1]
            while verify == "NA":
                print("Este usuario no terminó su último reto")
                airtable2.delete_by_field('Code', ultimocode)
                verify3 = airtable3.search("Code", ultimocode)
                if verify3 != []:
                    airtable3.delete_by_field("Code", ultimocode)
                ultimo = buscarduedates(chat_id1, reto)
                numerochallenges = len(ultimo)
                verify = ultimo[numerochallenges - 1]
        else:
            print("Este usuario no termino su primer reto")
            nuevouser = str(0)
            nuevouser1 = 0
            chat_id1 = str(update.effective_chat.id)
            airtable1.delete_by_field('User ID', chat_id1)
            codes = buscarcode2(chat_id1)
            if codes != [0]:
                numerochallenges = len(codes)
                ultimocode = codes[numerochallenges - 1]
                airtable2.delete_by_field('Code', ultimocode)
                verify3 = airtable3.search('Code', ultimocode)
                if verify3 != []:
                    airtable3.delete_by_field("Code", ultimocode)
    chat_id1 = str(update.effective_chat.id)
    name = update.effective_chat.first_name
    record = {'Code': code1}
    airtable.insert(record)
    rellenarairtable(code1)
    airtable4.insert(record)
    rellenarairtable4(code1)
    record1 = {colChatid: chat_id1}
    airtable.update_by_field('Code', code1, record1)
    actualizarTABLAVARIABLES(chat_id1, colNuevoUser, nuevouser)
    actualizarTABLATODO(chat_id1, colUsername, name)
    actualizarTABLATODO(chat_id1, colIdioma, idioma)
    if idioma == "English":
        keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                     InlineKeyboardButton("Judge", callback_data='Judge')],
                    [InlineKeyboardButton("My Challenges", callback_data='Review'),
                     InlineKeyboardButton("Challenge", callback_data='Challenge')]]
        keyboard1 = [[InlineKeyboardButton("Create", callback_data='Create'),
                      InlineKeyboardButton("Judge", callback_data='Judge'),
                      InlineKeyboardButton("Challenge", callback_data='Challenge')]]

        reply_markup = InlineKeyboardMarkup(keyboard)
        reply_markup1 = InlineKeyboardMarkup(keyboard1)
        if nuevouser1 == 0:
            chat_id1 = str(update.effective_chat.id)
            context.bot.send_message(chat_id1,
                                     text='Hello {}, I am Bobs and I am here to help you to become the best of yourself'.format(
                                         name))
            context.bot.send_message(chat_id1,
                                     text='What do you wanna do today? \n\n-Select *Create* if you wanna create a new own challenge. \n-
Select *Judge* if you received a code to judge another one challenge. \n-Select *Challenge* if you wanna defiance someone to do a challenge
that you are going to create or if someone have desafiance you ',
                                     parse_mode="Markdown", reply_markup=reply_markup1)
        else:
            chat_id1 = str(update.effective_chat.id)
            context.bot.send_message(chat_id1, text='Hi {}, how are you doing?'.format(name))
            context.bot.send_message(chat_id1,
                                     text='What do you wanna do today? \n\n-Select *Create* if you wanna create a new own challenge. \n-
Select *Judge* if you received a code to judge another one challenge. \n-Select *Review* if you wanna see the status of your current and past 
challenges \n-Select *Challenge* if you wanna defiance someone to do a challenge that you are going to create or if someone have desafiance 
you ',
                                     parse_mode="Markdown", reply_markup=reply_markup)
        return DECISION
    elif idioma == "Español":
        keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                     InlineKeyboardButton("Juzgar", callback_data='Judge')],
                    [InlineKeyboardButton("Mis Retos", callback_data='Review'),
                     InlineKeyboardButton("Desafío", callback_data='Challenge')]]
        keyboard1 = [[InlineKeyboardButton("Crear", callback_data='Create'),
                      InlineKeyboardButton("Juzgar", callback_data='Judge'),
                      InlineKeyboardButton("Desafío", callback_data='Challenge')]]

        reply_markup = InlineKeyboardMarkup(keyboard)
        reply_markup1 = InlineKeyboardMarkup(keyboard1)

        if nuevouser1 == 0:
            chat_id1 = str(update.effective_chat.id)
            context.bot.send_message(chat_id1,
                                     text='Hola {}, mi nombre es Bobs y estoy aquí para ayudarte a cumplir tus retos.'.format(
                                         name))
            context.bot.send_message(chat_id1,
                                     text='¿Que quieres hacer hoy? \n\n-Selecciona *Crear* si quieres crear un nuevo reto personal. \n-
Selecciona *Juzgar* si has recibido un código para determinar si otra persona ha realizado su reto o no. \n-Selecciona *Desafío* si quieres 
retar a alguien a hacer un reto que tu vas a crear o si alguien te ha retado a ti ',
                                     parse_mode="Markdown", reply_markup=reply_markup1)
        else:
            chat_id1 = str(update.effective_chat.id)
            context.bot.send_message(chat_id1, text='Hello {}. ¿Cómo estás?'.format(name))
            context.bot.send_message(chat_id1,
                                     text='¿Que quieres hacer hoy? \n\n-Selecciona *Crear* si quieres crear un nuevo reto personal. \n-
Selecciona *Juzgar* si has recibido un código para determinar si otra persona ha realizado su reto o no. \n-Selecciona *Mis Retos* si quieres 
ver el estado de tus retos actuales y pasados. \n-Selecciona *Desafío* si quieres retar a alguien a hacer un reto que tu vas a crear o si 
alguien te ha retado a ti ',
                                     parse_mode="Markdown", reply_markup=reply_markup)
        return DECISION


def start2(update, context):
    idioma = update.message.text
    name = update.effective_chat.first_name
    print("El usuario ha elegido el idioma {}".format(idioma))
    reto = 0
    if idioma == "español" or idioma == "english":
        numero = 1
        code = int(code) + numero
        code1 = str(code)

        # Actualizamos codigo del challenge
        verify = airtable.search('Code', code1)
        while (verify != []):
            code = buscarcode()
            code = code + numero
            code1 = str(code)
            verify = airtable.search('Code', code1)

        # Nuevo user o no
        chat_id1 = str(update.effective_chat.id)
        verify1 = airtable1.search('User ID', chat_id1)
        if verify1 == []:
            nuevouser = str(0)
            nuevouser1 = 0
            print("El usuario es NUEVO")
        else:
            ultimoemail = buscaruseremails(chat_id1)
            numerousers = len(ultimoemail)
            verify2 = ultimoemail[numerousers - 1]
            print("Ya conocemos a este usuario")
            if (verify1 != []) and (verify2 != "NA"):
                nuevouser = str(1)
                nuevouser1 = 1
                ultimo = buscarduedates(chat_id1, reto)
                numerochallenges = len(ultimo)
                verify = ultimo[numerochallenges - 1]
                codes = buscarcode2(chat_id1)
                numerochallenges = len(codes)
                ultimocode = codes[numerochallenges - 1]
                while verify == "NA":
                    print("Este usuario no terminó su último reto")
                    airtable2.delete_by_field('Code', ultimocode)
                    verify3 = airtable3.search("Code", ultimocode)
                    if verify3 != []:
                        airtable3.delete_by_field("Code", ultimocode)
                    ultimo = buscarduedates(chat_id1, reto)
                    numerochallenges = len(ultimo)
                    verify = ultimo[numerochallenges - 1]
            else:
                print("Este usuario no termino su primer reto")
                nuevouser = str(0)
                nuevouser1 = 0
                chat_id1 = str(update.effective_chat.id)
                airtable1.delete_by_field('User ID', chat_id1)
                codes = buscarcode2(chat_id1)
                if codes != [0]:
                    numerochallenges = len(codes)
                    ultimocode = codes[numerochallenges - 1]
                    airtable2.delete_by_field('Code', ultimocode)
                    verify3 = airtable3.search('Code', ultimocode)
                    if verify3 != []:
                        airtable3.delete_by_field("Code", ultimocode)

        chat_id1 = str(update.effective_chat.id)
        name = update.effective_chat.first_name
        record = {'Code': code1}
        airtable.insert(record)
        rellenarairtable(code1)
        airtable4.insert(record)
        rellenarairtable4(code1)
        record1 = {colChatid: chat_id1}
        airtable.update_by_field('Code', code1, record1)
        actualizarTABLAVARIABLES(chat_id1, colNuevoUser, nuevouser)
        actualizarTABLATODO(chat_id1, colUsername, name)
        actualizarTABLATODO(chat_id1, colIdioma, idioma)

        if idioma == "English":
            keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                         InlineKeyboardButton("Judge", callback_data='Judge')],
                        [InlineKeyboardButton("My Challenges", callback_data='Review'),
                         InlineKeyboardButton("Challenge", callback_data='Challenge')]]
            keyboard1 = [[InlineKeyboardButton("Create", callback_data='Create'),
                          InlineKeyboardButton("Judge", callback_data='Judge'),
                          InlineKeyboardButton("Challenge", callback_data='Challenge')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            reply_markup1 = InlineKeyboardMarkup(keyboard1)
            if nuevouser1 == 0:
                chat_id1 = str(update.effective_chat.id)
                context.bot.send_message(chat_id1,
                                         text='Hello {}, I am Bobs and I am here to help you to become the best of yourself'.format(
                                             name))
                context.bot.send_message(chat_id1,
                                         text='What do you wanna do today? \n\n-Select *Create* if you wanna create a new own challenge. \n-
Select *Judge* if you received a code to judge another one challenge. \n-Select *Challenge* if you wanna defiance someone to do a challenge
that you are going to create or if someone have desafiance you ',
                                         parse_mode="Markdown", reply_markup=reply_markup1)
            else:
                chat_id1 = str(update.effective_chat.id)
                context.bot.send_message(chat_id1, text='Hi {}, how are you doing?'.format(name))
                context.bot.send_message(chat_id1,
                                         text='What do you wanna do today? \n\n-Select *Create* if you wanna create a new own challenge. \n-
Select *Judge* if you received a code to judge another one challenge. \n-Select *Review* if you wanna see the status of your current and past
challenges \n-Select *Challenge* if you wanna defiance someone to do a challenge that you are going to create or if someone have desafiance
you ',
                                         parse_mode="Markdown", reply_markup=reply_markup)
            return DECISION
        elif idioma == "Español":
            keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                         InlineKeyboardButton("Juzgar", callback_data='Judge')],
                        [InlineKeyboardButton("Mis Retos", callback_data='Review'),
                         InlineKeyboardButton("Desafío", callback_data='Challenge')]]
            keyboard1 = [[InlineKeyboardButton("Crear", callback_data='Create'),
                          InlineKeyboardButton("Juzgar", callback_data='Judge'),
                          InlineKeyboardButton("Desafío", callback_data='Challenge')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            reply_markup1 = InlineKeyboardMarkup(keyboard1)

            if nuevouser1 == 0:
                chat_id1 = str(update.effective_chat.id)
                context.bot.send_message(chat_id1,
                                         text='Hola {}, mi nombre es Bobs y estoy aquí para ayudarte a cumplir tus retos.'.format(
                                             name))
                context.bot.send_message(chat_id1,
                                         text='¿Que quieres hacer hoy? \n\n-Selecciona *Crear* si quieres crear un nuevo reto personal. \n-
Selecciona *Juzgar* si has recibido un código para determinar si otra persona ha realizado su reto o no. \n-Selecciona *Desafío* si quieres
retar a alguien a hacer un reto que tu vas a crear o si alguien te ha retado a ti ',
                                        parse_mode="Markdown", reply_markup=reply_markup1)
            else:
                chat_id1 = str(update.effective_chat.id)
                context.bot.send_message(chat_id1, text='Hello {}. ¿Cómo estás?'.format(name))
                context.bot.send_message(chat_id1,
                                        text='¿Que quieres hacer hoy? \n\n-Selecciona *Crear* si quieres crear un nuevo reto personal. \n-
Selecciona *Juzgar* si has recibido un código para determinar si otra persona ha realizado su reto o no. \n-Selecciona *Mis Retos* si quieres
ver el estado de tus retos actuales y pasados. \n-Selecciona *Desafío* si quieres retar a alguien a hacer un reto que tu vas a crear o si
alguien te ha retado a ti ',
                                        parse_mode="Markdown", reply_markup=reply_markup)
            return DECISION
    else:
        keyboard = [[InlineKeyboardButton("Español", callback_data='Español'),
                     InlineKeyboardButton("English", callback_data='Ingles')]]

        reply_markup = InlineKeyboardMarkup(keyboard)
        update.message.reply_text(
            'Sorry, I could not understand you. Select the languages pressing in the buttons below this message \n'
            'Perdona no te he entendido. Por favor, selecciona el idioma de los botones que aparecen debajo de este mensaje',
            reply_markup=reply_markup)

        return START


def decision(update, context):
    chat_id1 = str(update.effective_chat.id)
    name = update.effective_chat.first_name
    # Verificamos que es está registrado como USER (por si se restaura la base de datos)
    verify1 = airtable1.search('User ID', chat_id1)
    if verify1 == []:
        nuevouser = str(0)
        nuevouser1 = 0
        actualizarTABLAVARIABLES(chat_id1, colNuevoUser, nuevouser)
    else:
        nuevouser = str(1)
        nuevouser1 = 1
        actualizarTABLAVARIABLES(chat_id1, colNuevoUser, nuevouser)
        ultimoemail = buscaruseremails(chat_id1)
        numerousers = len(ultimoemail)
        verify2 = ultimoemail[numerousers - 1]
        if (verify1 != []) and (verify2 != "NA"):
            nuevouser = str(1)
            actualizarTABLAVARIABLES(chat_id1, colNuevoUser, nuevouser)
        else:
            airtable1.delete_by_field('User ID', chat_id1)
            nuevouser = str(0)
            actualizarTABLAVARIABLES(chat_id1, colNuevoUser, nuevouser)
            codes = buscarcode2(chat_id1)
            numerochallenges = len(codes)
            ultimocode = codes[numerochallenges - 1]
            airtable2.delete_by_field('Code', ultimocode)
            verify3 = airtable3.search("Code", ultimocode)
            if verify3 != []:
                airtable3.delete_by_field("Code", ultimocode)

    query = update.callback_query.data
    print("El usuario quiere {}".format(query))

    if query == "Create":
        chat_id1 = str(update.effective_chat.id)
        name = update.effective_chat.first_name
        codigoanterior = buscarcodeanterior(chat_id1)
        reto = str(0)
        actualizarTABLAVARIABLES(chat_id1, colChallenge, reto)
        if nuevouser == 1:
            chat_id1 = str(update.effective_chat.id)
            ultimo = buscarduedates(chat_id1, reto)
            verify = ultimo[len(ultimo) - 1]
            codes = buscarcode2(chat_id1)
            ultimocode = codes[len(codes) - 1]
            while verify == "NA":
                airtable2.delete_by_field('Code', ultimocode)
                verify3 = airtable3.search("Code", ultimocode)
                if verify3 != []:
                    airtable3.delete_by_field("Code", ultimocode)
                ultimo = buscarduedates(chat_id1, reto)
                verify = ultimo[len(ultimo) - 1]
                codes = buscarcode2(chat_id1)
                ultimocode = codes[len(codes) - 1]
        record2 = {'Code': codigoanterior}
        airtable2.insert(record2)
        rellenarairtable2(chat_id1)
        chat_id1 = str(update.effective_chat.id)
        if nuevouser1 == 0:
            record3 = {'User ID': chat_id1}
            airtable1.insert(record3)
            rellenarairtable1(chat_id1)
            actualizarTABLAUSERS(chat_id1, colUsername, name)
            idioma = buscaridioma1(chat_id1)
            print("Idioma {} actualizado".format(name))
            actualizarTABLAUSERS(chat_id1, colIdioma, idioma)
            actualizarTABLATODO(chat_id1, colDecision, 'Yes')
            actualizarTABLACHALLENGES(chat_id1, colChatid, chat_id1)
            actualizarTABLACHALLENGES(chat_id1, colUsername, name)
            if idioma == "English":
                context.bot.send_message(chat_id1, text='Great {}. Lets do it!'.format(name))
                context.bot.send_message(chat_id1,
                                         text='Describe briefly the challenge goal. Use the Menu above if you wanna change what to do.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text='Perfecto {}. Vamos a ello!'.format(name))
                context.bot.send_message(chat_id1,
                                         text='Describe brevemente en que consiste el reto. Usa el Menu de arriba si quieres cambiar lo que
hacer.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            return CHALLENGENAME
        else:
            idioma = buscaridioma1(chat_id1)
            actualizarTABLAUSERS(chat_id1, colIdioma, idioma)
            actualizarTABLATODO(chat_id1, colDecision, 'Yes')
            actualizarTABLACHALLENGES(chat_id1, colChatid, chat_id1)
            actualizarTABLACHALLENGES(chat_id1, colUsername, name)
            if idioma == "English":
                context.bot.send_message(chat_id1, text='Great {}. Lets do it!'.format(name))
                context.bot.send_message(chat_id1,
                                         text='Describe briefly the challenge goal. Use the Menu above if you wanna change what to do.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text='Perfecto {}. Vamos a ello!'.format(name))
                context.bot.send_message(chat_id1,
                                         text='Describe brevemente en que consiste el reto. Usa el Menu de arriba si quieres cambiar lo que
hacer.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))

            return CHALLENGENAME

    elif query == "Judge":
        chat_id1 = str(update.effective_chat.id)
        idioma = buscaridioma1(chat_id1)
        actualizarTABLATODO(chat_id1, colDecision, 'No')
        reto = str(0)
        actualizarTABLAVARIABLES(chat_id1, colChallenge, reto)
        if idioma == "English":
            context.bot.send_message(chat_id1,
                                     text='I see! Please, send the code of the challenge that you want to judge. Use the Menu above if you'
                                          ' wanna change what to do. ',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
        elif idioma == "Español":
            context.bot.send_message(chat_id1,
                                     text='Bien, envíame el código del reto que quieres juzgar. Si quieres cambiar que hacer utiliza de
nuevom el Menu anterior. ',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
        return JUDGE
    elif query == "Review":
        chat_id1 = str(update.effective_chat.id)
        idioma = buscaridioma1(chat_id1)
        reto = str(0)
        ultimo = buscarduedates(chat_id1, reto)
        verify = ultimo[len(ultimo) - 1]
        codes = buscarcode2(chat_id1)
        ultimocode = codes[len(codes) - 1]
        actualizarTABLAVARIABLES(chat_id1, colChallenge, reto)
        while verify == "NA":
            airtable2.delete_by_field('Code', ultimocode)
            verify3 = airtable3.search("Code", ultimocode)
            if verify3 != []:
                airtable3.delete_by_field("Code", ultimocode)
            ultimo = buscarduedates(chat_id1, reto)
            verify = ultimo[len(ultimo) - 1]
            codes = buscarcode2(chat_id1)
            ultimocode = codes[len(codes) - 1]
        chat_id1 = str(update.effective_chat.id)
        challenge = buscarchallenges(chat_id1, reto)
        numchallenge = len(challenge)
        actualizarTABLATODO(chat_id1, colDecision, 'No')
        if numchallenge == 1:
            keyboard = [[InlineKeyboardButton(challenge[0], callback_data='Challenge1')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            if idioma == "English":
                context.bot.send_message(chat_id1,
                                         text='You have the following challenges. See its descriptions and status pressing in its name'
                                              '. Otherwise, change what you want to do using the previous MENU',
                                         reply_markup=reply_markup)
            elif idioma == "Español":
                context.bot.send_message(chat_id1,
                                         text='Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si quieres
cambiar lo que hacer utiliza el Menu anterior',
                                         reply_markup=reply_markup)
            return REVIEW
        elif numchallenge == 2:
            keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                         InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            if idioma == "English":
                context.bot.send_message(chat_id1,
                                         text='You have the following challenges. See its descriptions and status pressing in its name'
                                              '. Otherwise, change what you want to do using the previous MENU',
                                         reply_markup=reply_markup)
            elif idioma == "Español":
                context.bot.send_message(chat_id1,
                                         text='Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si quieres
cambiar lo que hacer utiliza el Menu anterior',
                                         reply_markup=reply_markup)
            return REVIEW
        elif numchallenge == 3:
            keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                         InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2'),
                         InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            if idioma == "English":
                context.bot.send_message(chat_id1,
                                         text='You have the following challenges. See its descriptions and status pressing in its name'
                                              '. Otherwise, change what you want to do using the previous MENU',
                                         reply_markup=reply_markup)
            elif idioma == "Español":
                context.bot.send_message(chat_id1,
                                         text='Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si quieres
cambiar lo que hacer utiliza el Menu anterior',
                                         reply_markup=reply_markup)
            return REVIEW
        elif numchallenge == 4:
            keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                         InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')],
                        [InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3'),
                         InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            if idioma == "English":
                context.bot.send_message(chat_id1,
                                         text='You have the following challenges. See its descriptions and status pressing in its name'
                                              '. Otherwise, change what you want to do using the previous MENU',
                                         reply_markup=reply_markup)
            elif idioma == "Español":
                context.bot.send_message(chat_id1,
                                         text='Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si quieres
cambiar lo que hacer utiliza el Menu anterior',
                                         reply_markup=reply_markup)
            return REVIEW
        elif numchallenge == 5:
            keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                         InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2'),
                         InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3')],
                        [InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4'),
                         InlineKeyboardButton(challenge[numchallenge - 5], callback_data='Challenge5')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            if idioma == "English":
                context.bot.send_message(chat_id1,
                                         text='You have the following challenges. See its descriptions and status pressing in its name'
                                              '. Otherwise, change what you want to do using the previous MENU',
                                         reply_markup=reply_markup)
            elif idioma == "Español":
                context.bot.send_message(chat_id1,
                                         text='Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si quieres
cambiar lo que hacer utiliza el Menu anterior',
                                         reply_markup=reply_markup)
            return REVIEW
        elif numchallenge >= 6:
            keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                         InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')],
                        [InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3'),
                         InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4')],
                        [InlineKeyboardButton(challenge[numchallenge - 5], callback_data='Challenge5'),
                         InlineKeyboardButton(challenge[numchallenge - 6], callback_data='Challenge6')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            if idioma == "English":
                context.bot.send_message(chat_id1,
                                         text='You have the following challenges. See its descriptions and status pressing in its name'
                                              '. Otherwise, change what you want to do using the previous MENU',
                                         reply_markup=reply_markup)
            elif idioma == "Español":
                context.bot.send_message(chat_id1,
                                         text='Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si quieres 
cambiar lo que hacer utiliza el Menu anterior',
                                         reply_markup=reply_markup)
            return REVIEW
    elif query == "Challenge":
        chat_id1 = str(update.effective_chat.id)
        idioma = buscaridioma1(chat_id1)
        reto = str(1)
        actualizarTABLAVARIABLES(chat_id1, colChallenge, reto)
        if idioma == "English":
            keyboard = [[InlineKeyboardButton("Create", callback_data='Retar'),
                         InlineKeyboardButton("See Challenge", callback_data='Retado')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1,
                                     text='Select *Create* if you want to challenge. However, if someone have challenged you press *See 
Challenge* to see the challenge features and decided if accept it or not',
                                     parse_mode="Markdown", reply_markup=reply_markup)
        elif idioma == "Español":
            keyboard = [[InlineKeyboardButton("Crear Reto", callback_data='Retar'),
                         InlineKeyboardButton("Ver Reto", callback_data='Retado')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1,
                                     text='Selecciona *Crear Reto* si lo que quieres es retar tu a otra persona. \nSi te han retado a ti 
pulsa sobre *Ver Reto* para ver las características de este y ver si aceptarlo o no',
                                     parse_mode="Markdown", reply_markup=reply_markup)
        return DECISION
    elif query == "Retar":
        chat_id1 = str(update.effective_chat.id)
        name = update.effective_chat.first_name
        reto = buscarvariable(chat_id1, colChallenge)
        codigoanterior = buscarcodeanterior(chat_id1)
        if nuevouser == 1:
            ultimo = buscarduedates(chat_id1, reto)
            verify = ultimo[len(ultimo) - 1]
            codes = buscarcode2(chat_id1)
            ultimocode = codes[len(codes) - 1]
            while verify == "NA":
                airtable2.delete_by_field('Code', ultimocode)
                verify3 = airtable3.search("Code", ultimocode)
                if verify3 != []:
                    airtable3.delete_by_field("Code", ultimocode)
                ultimo = buscarduedates(chat_id1, reto)
                verify = ultimo[len(ultimo) - 1]
                codes = buscarcode2(chat_id1)
                ultimocode = codes[len(codes) - 1]
        record2 = {'Code': codigoanterior}
        airtable5.insert(record2)
        rellenarairtable5(codigoanterior)
        if nuevouser1 == 0:
            record3 = {'User ID': chat_id1}
            airtable1.insert(record3)
            rellenarairtable1(chat_id1)
            actualizarTABLAUSERS(chat_id1, colUsername, name)
            idioma = buscaridioma1(chat_id1)
            print("Idioma {} actualizado".format(name))
            actualizarTABLAUSERS(chat_id1, colIdioma, idioma)
            actualizarTABLATODO(chat_id1, colDecision, 'Yes')
            actualizarTABLADEFIANCE(chat_id1, colJudgeID, chat_id1)
            if idioma == "English":
                context.bot.send_message(chat_id1, text='Great {}. Lets do it!'.format(name))
                context.bot.send_message(chat_id1,
                                         text='Describe briefly the challenge goal. Use the Menu above if you wanna change what to do.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text='Perfecto {}. Vamos a ello!'.format(name))
                context.bot.send_message(chat_id1,
                                         text='Describe brevemente en que consiste el reto. Usa el Menu de arriba si quieres cambiar lo que 
hacer.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            return CHALLENGENAME
        else:
            idioma = buscaridioma1(chat_id1)
            actualizarTABLAUSERS(chat_id1, colIdioma, idioma)
            actualizarTABLATODO(chat_id1, colDecision, 'Yes')
            actualizarTABLADEFIANCE(chat_id1, colJudgeID, chat_id1)

            if idioma == "English":
                context.bot.send_message(chat_id1, text='Great {}. Lets do it!'.format(name))
                context.bot.send_message(chat_id1,
                                         text='Describe briefly the challenge goal. Use the Menu above if you wanna change what to do.',
                                  reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text='Perfecto {}. Vamos a ello!'.format(name))
                context.bot.send_message(chat_id1,
                                  text='Describe brevemente en que consiste el reto. Usa el Menu de arriba si quieres cambiar lo que
hacer.',
                                  reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            return CHALLENGENAME
    elif query == "Retado":
        chat_id1 = str(update.effective_chat.id)
        name = update.effective_chat.first_name
        reto = buscarvariable(chat_id1, colChallenge)
        codigoanterior = buscarcodeanterior(chat_id1)
        if nuevouser == 1:
            ultimo = buscarduedates(chat_id1, reto)
            verify = ultimo[len(ultimo) - 1]
            codes = buscarcode2(chat_id1)
            ultimocode = codes[len(codes) - 1]
            while verify == "NA":
                airtable2.delete_by_field('Code', ultimocode)
                verify3 = airtable3.search("Code", ultimocode)
                if verify3 != []:
                    airtable3.delete_by_field("Code", ultimocode)
                ultimo = buscarduedates(chat_id1, reto)
                verify = ultimo[len(ultimo) - 1]
                codes = buscarcode2(chat_id1)
                ultimocode = codes[len(codes) - 1]
        if nuevouser1 == 0:
            record3 = {'User ID': chat_id1}
            airtable1.insert(record3)
            rellenarairtable1(chat_id1)
            actualizarTABLAUSERS(chat_id1, colUsername, name)
            idioma = buscaridioma1(chat_id1)
            print("Idioma {} actualizado".format(name))
            actualizarTABLAUSERS(chat_id1, colIdioma, idioma)
            actualizarTABLATODO(chat_id1, colDecision, 'Yes')
            if idioma == "English":
                context.bot.send_message(chat_id1, text='Great {}. Lets do it!'.format(name))
                context.bot.send_message(chat_id1,
                                  text='Send me the code of the challenge that have been sent to you ',
                                  reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text='Perfecto {}. Vamos a ello!'.format(name))
                context.bot.send_message(chat_id1,
                                  text='Envíame el código del reto que te han enviado',
                                  reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            return VERCHALLENGE
        else:
            idioma = buscaridioma1(chat_id1)
            actualizarTABLAUSERS(chat_id1, colIdioma, idioma)
            actualizarTABLATODO(chat_id1, colDecision, 'Yes')

            if idioma == "English":
                context.bot.send_message(chat_id1, text='Great {}. Lets do it!'.format(name))
                context.bot.send_message(chat_id1,
                                  text='Send me the code of the challenge that have been sent to you ',
                                  reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text='Perfecto {}. Vamos a ello!'.format(name))
                context.bot.send_message(chat_id1,
                                  text='Envíame el código del reto que te han enviado',
                                  reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            return VERCHALLENGE


def decision1(update, context):
    chat_id1 = str(update.effective_chat.id)
    idioma = buscaridioma1(chat_id1)
    text = update.message.text
    verify1 = airtable1.search('User ID', chat_id1)
    if verify1 == []:
        nuevouser = str(0)
        nuevouser1 = 0
        actualizarTABLAVARIABLES(chat_id1, colNuevoUser, nuevouser)
    else:
        ultimoemail = buscaruseremails(chat_id1)
        verify2 = ultimoemail[len(ultimoemail) - 1]
        if (verify1 != []) and (verify2 != "NA"):
            nuevouser = str(1)
            nuevouser1 = 1
            actualizarTABLAVARIABLES(chat_id1, colNuevoUser, nuevouser)
        else:
            airtable1.delete_by_field('User ID', chat_id1)
            nuevouser = str(0)
            nuevouser1 = 0
            actualizarTABLAVARIABLES(chat_id1, colNuevoUser, nuevouser)
            codes = buscarcode2(chat_id1)
            numerochallenges = len(codes)
            ultimocode = codes[numerochallenges - 1]
            airtable2.delete_by_field('Code', ultimocode)
            verify3 = airtable3.search("Code", ultimocode)
            if verify3 != []:
                airtable3.delete_by_field("Code", ultimocode)

    if text == "Create" or text == "create":
        print("El usuario quiere {}".format(text))
        chat_id1 = str(update.effective_chat.id)
        name = update.effective_chat.first_name
        codigoanterior = buscarcodeanterior(chat_id1)
        reto = buscarvariable(chat_id1, colChallenge)
        if nuevouser1 != 0:
            chat_id1 = str(update.effective_chat.id)
            ultimo = buscarduedates(chat_id1, reto)
            numerochallenges = len(ultimo)
            verify = ultimo[numerochallenges - 1]
            codes = buscarcode2(chat_id1)
            numerochallenges = len(codes)
            ultimocode = codes[numerochallenges - 1]
            while verify == "NA":
                airtable2.delete_by_field('Code', ultimocode)
                verify3 = airtable3.search("Code", ultimocode)
                if verify3 != []:
                    airtable3.delete_by_field("Code", ultimocode)
                ultimo = buscarduedates(chat_id1, reto)
                verify = ultimo[len(ultimo) - 1]
                codes = buscarcode2(chat_id1)
                ultimocode = codes[len(codes) - 1]
        record2 = {'Code': codigoanterior}
        airtable2.insert(record2)
        rellenarairtable2(chat_id1)

        chat_id1 = str(update.effective_chat.id)
        if nuevouser1 == 0:
            record3 = {'User ID': chat_id1}
            airtable1.insert(record3)
            rellenarairtable1(chat_id1)
            actualizarTABLAUSERS(chat_id1, colUsername, name)
            actualizarTABLAUSERS(chat_id1, colIdioma, idioma)
            actualizarTABLATODO(chat_id1, colDecision, 'Yes')
            actualizarTABLACHALLENGES(chat_id1, colChatid, chat_id1)
            if idioma == "English":
                update.message.reply_text('Great {}. Lets do it!'.format(name))
                update.message.reply_text(
                    'Describe briefly the challenge goal. Use the Menu above if you wanna change what to do.',
                    reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text('Perfecto {}. Vamos a ello'.format(name))
                update.message.reply_text(
                    'Describe brevemente en que consiste el reto. Usa el Menu de arriba si quieres cambiar lo que hacer.',
                    reply_markup=ReplyKeyboardRemove(remove_keyboard=True))

            return CHALLENGENAME
        else:
            actualizarTABLATODO(chat_id1, colDecision, 'Yes')
            actualizarTABLACHALLENGES(chat_id1, colChatid, chat_id1)

            if idioma == "English":
                update.message.reply_text('Great {}. Lets do it!'.format(name))
                update.message.reply_text(
                    'Describe briefly the challenge goal. Use the Menu above if you wanna change what to do.',
                    reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text('Perfecto {}. Vamos a ello'.format(name))
                update.message.reply_text(
                    'Describe brevemente en que consiste el reto. Usa el Menu de arriba si quieres cambiar lo que hacer.',
                    reply_markup=ReplyKeyboardRemove(remove_keyboard=True))

            return CHALLENGENAME

    elif text == "Judge" or text == "judge":
        print("El usuario quiere {}".format(text))
        actualizarTABLATODO(chat_id1, colDecision, 'No')
        chat_id1 = str(update.effective_chat.id)
        if idioma == "English":
            update.message.reply_text(
                'I see! Please, send the code of the challenge that you want to judge. Use the Menu above if you'
                ' wanna change what to do. ', reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
        elif idioma == "Español":
            update.message.reply_text(
                'Bien, envíame el código del reto que quieres juzgar. Si quieres cambiar que hacer utiliza de nuevom el Menu anterior. ',
                reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
        return JUDGE
    elif text == "My challenges" or text == "my challenges":
        print("El usuario quiere {}".format(text))
        chat_id1 = str(update.effective_chat.id)
        reto = buscarvariable(chat_id1, colChallenge)
        ultimo = buscarduedates(chat_id1, reto)
        numerochallenges = len(ultimo)
        verify = ultimo[numerochallenges - 1]
        codes = buscarcode2(chat_id1)
        numerochallenges = len(codes)
        ultimocode = codes[numerochallenges - 1]
        while verify == "NA":
            airtable2.delete_by_field('Code', ultimocode)
            verify3 = airtable3.search("Code", ultimocode)
            if verify3 != []:
                airtable3.delete_by_field("Code", ultimocode)
            ultimo = buscarduedates(chat_id1, reto)
            verify = ultimo[len(ultimo) - 1]
            codes = buscarcode2(chat_id1)
            ultimocode = codes[len(codes) - 1]
        chat_id1 = str(update.effective_chat.id)
        challenge = buscarchallenges(chat_id1, reto)
        numchallenge = len(challenge)
        actualizarTABLATODO(chat_id1, colDecision, 'No')
        if numchallenge == 1:
            keyboard = [[InlineKeyboardButton(challenge[0], callback_data='Challenge1')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            if idioma == "English":
                update.message.reply_text(
                    'You have the following challenges. See its descriptions and status pressing in its name'
                    '. Otherwise, change what you want to do using the previous MENU',
                    reply_markup=reply_markup)
            elif idioma == "Español":
                update.message.reply_text(
                    'Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si quieres cambiar lo que hacer 
utiliza el Menu anterior',
                    reply_markup=reply_markup)
            return REVIEW
        elif numchallenge == 2:
            keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                         InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            if idioma == "English":
                update.message.reply_text(
                    'You have the following challenges. See its descriptions and status pressing in its name'
                    '. Otherwise, change what you want to do using the previous MENU',
                    reply_markup=reply_markup)
            elif idioma == "Español":
                update.message.reply_text(
                    'Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si quieres cambiar lo que hacer 
utiliza el Menu anterior',
                    reply_markup=reply_markup)
            return REVIEW
        elif numchallenge == 3:
            keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                         InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2'),
                         InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            if idioma == "English":
                update.message.reply_text(
                    'You have the following challenges. See its descriptions and status pressing in its name'
                    '. Otherwise, change what you want to do using the previous MENU',
                    reply_markup=reply_markup)
            elif idioma == "Español":
                update.message.reply_text(
                    'Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si quieres cambiar lo que hacer 
utiliza el Menu anterior',
                    reply_markup=reply_markup)
            return REVIEW
        elif numchallenge == 4:
            keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                         InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')],
                        [InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3'),
                         InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            if idioma == "English":
                update.message.reply_text(
                    'You have the following challenges. See its descriptions and status pressing in its name'
                    '. Otherwise, change what you want to do using the previous MENU',
                    reply_markup=reply_markup)
            elif idioma == "Español":
                update.message.reply_text(
                    'Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si quieres cambiar lo que hacer 
utiliza el Menu anterior',
                reply_markup=reply_markup)
            return REVIEW
        elif numchallenge == 5:
            keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                         InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2'),
                         InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3')],
                        [InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4'),
                         InlineKeyboardButton(challenge[numchallenge - 5], callback_data='Challenge5')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            if idioma == "English":
                update.message.reply_text(
                    'You have the following challenges. See its descriptions and status pressing in its name'
                    '. Otherwise, change what you want to do using the previous MENU',
                    reply_markup=reply_markup)
            elif idioma == "Español":
                update.message.reply_text(
                    'Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si quieres cambiar lo que hacer 
utiliza el Menu anterior',
                    reply_markup=reply_markup)
            return REVIEW
        elif numchallenge >= 6:
            keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                         InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')],
                        [InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3'),
                         InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4')],
                        [InlineKeyboardButton(challenge[numchallenge - 5], callback_data='Challenge5'),
                         InlineKeyboardButton(challenge[numchallenge - 6], callback_data='Challenge6')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            if idioma == "English":
                update.message.reply_text(
                    'You have the following challenges. See its descriptions and status pressing in its name'
                    '. Otherwise, change what you want to do using the previous MENU',
                    reply_markup=reply_markup)
            elif idioma == "Español":
                update.message.reply_text(
                    'Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si quieres cambiar lo que hacer 
utiliza el Menu anterior',
                    reply_markup=reply_markup)
            return REVIEW
    elif text == "Challenge" or "challenge":
        chat_id1 = str(update.effective_chat.id)
        idioma = buscaridioma1(chat_id1)
        reto = str(1)
        actualizarTABLAVARIABLES(chat_id1, colChallenge, reto)
        if idioma == "English":
            keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                         InlineKeyboardButton("See Challenge", callback_data='Retado')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1,
                                     text='Select *Create* if you want to challenge. However, if someone have challenged you press *See 
Challenge* to see the challenge features and decided if accept it or not',
                                     parse_mode="Markdown", reply_markup=reply_markup)
        elif idioma == "Español":
            keyboard = [[InlineKeyboardButton("Crear Reto", callback_data='Create'),
                         InlineKeyboardButton("Ver Reto", callback_data='Retado')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1,
                                     text='Selecciona *Crear Reto* si lo que quieres es retar tu a otra persona. \nSi te han retado a ti 
pulsa sobre *Ver Reto* para ver las características de este y ver si aceptarlo o no',
                                     parse_mode="Markdown", reply_markup=reply_markup)
        return DECISION
    elif text == "Retar":
        chat_id1 = str(update.effective_chat.id)
        name = update.effective_chat.first_name
        codigoanterior = buscarcodeanterior(chat_id1)
        reto = buscarvariable(chat_id1, colChallenge)
        if nuevouser == 1:
            ultimo = buscarduedates(chat_id1, reto)
            verify = ultimo[len(ultimo) - 1]
            codes = buscarcode2(chat_id1)
            ultimocode = codes[len(codes) - 1]
            while verify == "NA":
                airtable2.delete_by_field('Code', ultimocode)
                verify3 = airtable3.search("Code", ultimocode)
                if verify3 != []:
                    airtable3.delete_by_field("Code", ultimocode)
                ultimo = buscarduedates(chat_id1, reto)
                verify = ultimo[len(ultimo) - 1]
                codes = buscarcode2(chat_id1)
                ultimocode = codes[len(codes) - 1]
        record2 = {'Code': codigoanterior}
        airtable5.insert(record2)
        rellenarairtable5(codigoanterior)
        if nuevouser1 == 0:
            record3 = {'User ID': chat_id1}
            airtable1.insert(record3)
            rellenarairtable1(chat_id1)
            actualizarTABLAUSERS(chat_id1, colUsername, name)
            idioma = buscaridioma1(chat_id1)
            print("Idioma {} actualizado".format(name))
            actualizarTABLAUSERS(chat_id1, colIdioma, idioma)
            actualizarTABLATODO(chat_id1, colDecision, 'Yes')
            actualizarTABLADEFIANCE(chat_id1, colJudgeID, chat_id1)
            if idioma == "English":
                context.bot.send_message(chat_id1, text='Great {}. Lets do it!'.format(name))
                context.bot.send_message(chat_id1,
                                         text='Describe briefly the challenge goal. Use the Menu above if you wanna change what to do.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text='Perfecto {}. Vamos a ello!'.format(name))
                context.bot.send_message(chat_id1,
                                         text='Describe brevemente en que consiste el reto. Usa el Menu de arriba si quieres cambiar lo que
hacer.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            return CHALLENGENAME
        else:
            idioma = buscaridioma1(chat_id1)
            actualizarTABLAUSERS(chat_id1, colIdioma, idioma)
            actualizarTABLATODO(chat_id1, colDecision, 'Yes')
            actualizarTABLADEFIANCE(chat_id1, colJudgeID, chat_id1)

            if idioma == "English":
                context.bot.send_message(chat_id1, text='Great {}. Lets do it!'.format(name))
                context.bot.send_message(chat_id1,
                                         text='Describe briefly the challenge goal. Use the Menu above if you wanna change what to do.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text='Perfecto {}. Vamos a ello!'.format(name))
                context.bot.send_message(chat_id1,
                                         text='Describe brevemente en que consiste el reto. Usa el Menu de arriba si quieres cambiar lo que
hacer.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            return CHALLENGENAME
    else:
        if nuevouser == 0:
            keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                         InlineKeyboardButton("Judge", callback_data='Judge')]]
        elif nuevouser == 1:
            keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                         InlineKeyboardButton("Judge", callback_data='Judge'),
                         InlineKeyboardButton("My Challenges", callback_data='Review')]]

        reply_markup = InlineKeyboardMarkup(keyboard)

        if idioma == "English":
            update.message.reply_text(
                'Sorry you should decide what to do. Please, use the Menu below',
                reply_markup=reply_markup1)
        elif idioma == "Español":
            update.message.reply_text(
                'Lo siento tienes que decidir que quieres hacer. Please, use the Menu below',
                reply_markup=reply_markup)

        return DECISION


def decision2(update, context):
    chat_id1 = str(update.effective_chat.id)
    query = update.callback_query.data
    idioma = buscaridiomauser(chat_id1)
    change = int(buscarvariable(chat_id1, colChange))
    reto = buscarvariable(chat_id1, colChallenge)
    if change == 0:
        print("Un usuario que ya interactuó con nosostros hace tiempo quiere {}.".format(query))
        verify1 = airtable1.search('User ID', chat_id1)
        # Verificación ningún problema con basedatos
        if verify1 == []:
            nuevouser = str(0)
            nuevouser1 = 0
        else:
            ultimoemail = buscaruseremails(chat_id1)
            numerousers = len(ultimoemail)
            verify2 = ultimoemail[numerousers - 1]
            if (verify1 != []) and (verify2 != "NA"):
                nuevouser = str(1)
                nuevouser1 = 1
            else:
                airtable1.delete_by_field('User ID', chat_id1)
                nuevouser = str(0)
                nuevouser1 = 0
            codes = buscarcode2(chat_id1)
            numerochallenges = len(codes)
            ultimocode = codes[numerochallenges - 1]
            airtable2.delete_by_field('Code', ultimocode)
            verify3 = airtable3.search("Code", ultimocode)
            if verify3 != []:
                airtable3.delete_by_field("Code", ultimocode)
        # Nuevocodigo
        numero = int(buscarcodeanterior(chat_id1))
        code = numero + 1
        code1 = str(code)
        verify = airtable.search('Code', code1)
        while (verify != []):
            code = buscarcode()
            code = code + numero
            code1 = str(code)
            verify = airtable.search('Code', code1)

        chat_id1 = str(update.effective_chat.id)
        name = update.effective_chat.first_name
        record = {'Code': code1}
        airtable.insert(record)
        rellenarairtable(code1)
        airtable4.insert(record)
        rellenarairtable4(code1)
        record1 = {colChatid: chat_id1}
        airtable.update_by_field('Code', code1, record1)
        actualizarTABLAVARIABLES(chat_id1, colNuevoUser, nuevouser)
        actualizarTABLATODO(chat_id1, colUsername, name)
        actualizarTABLATODO(chat_id1, colIdioma, idioma)

        if query == "Create":
            chat_id1 = str(update.effective_chat.id)
            name = update.effective_chat.first_name
            codigoanterior = buscarcodeanterior(chat_id1)
            reto = buscarvariable(chat_id1, colChallenge)
            if nuevouser1 != 0:
                chat_id1 = str(update.effective_chat.id)
                ultimo = buscarduedates(chat_id1, reto)
                numerochallenges = len(ultimo)
                verify = ultimo[numerochallenges - 1]
                codes = buscarcode2(chat_id1)
                numerochallenges = len(codes)
                ultimocode = codes[numerochallenges - 1]
                while verify == "NA":
                    airtable2.delete_by_field('Code', ultimocode)
                    if verify3 != []:
                        airtable3.delete_by_field("Code", ultimocode)
                    ultimo = buscarduedates(chat_id1, reto)
                    verify = ultimo[len(ultimo) - 1]
                    codes = buscarcode2(chat_id1)
                    ultimocode = codes[len(codes) - 1]
            record2 = {'Code': codigoanterior}
            airtable2.insert(record2)
            rellenarairtable2(chat_id1)

            chat_id1 = str(update.effective_chat.id)
            if nuevouser1 == 0:
                record3 = {'User ID': chat_id1}
                airtable1.insert(record3)
                rellenarairtable1(chat_id1)
                actualizarTABLAUSERS(chat_id1, colUsername, name)
                actualizarTABLAUSERS(chat_id1, colIdioma, idioma)
                actualizarTABLATODO(chat_id1, colDecision, 'Yes')
                actualizarTABLACHALLENGES(chat_id1, colChatid, chat_id1)
                if idioma == "English":
                    context.bot.send_message(chat_id1, text='Great {}. Lets do it!'.format(name))
                    context.bot.send_message(chat_id1,
                                             text='Describe briefly the challenge goal. Use the Menu above if you wanna change what to do.',
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                elif idioma == "Español":
                    context.bot.send_message(chat_id1, text='Perfecto {}. Vamos a ello!'.format(name))
                    context.bot.send_message(chat_id1,
                                             text='Describe brevemente en que consiste el reto. Usa el Menu de arriba si quieres cambiar lo
que hacer.',
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                return CHALLENGENAME
            else:
                actualizarTABLATODO(chat_id1, colDecision, 'Yes')
                actualizarTABLACHALLENGES(chat_id1, colChatid, chat_id1)

                if idioma == "English":
                    context.bot.send_message(chat_id1, text='Great {}. Lets do it!'.format(name))
                    context.bot.send_message(chat_id1,
                                             text='Describe briefly the challenge goal. Use the Menu above if you wanna change what to do.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                elif idioma == "Español":
                    context.bot.send_message(chat_id1, text='Perfecto {}. Vamos a ello!'.format(name))
                    context.bot.send_message(chat_id1,
                                         text='Describe brevemente en que consiste el reto. Usa el Menu de arriba si quieres cambiar lo
que hacer.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                return CHALLENGENAME
        elif query == "Judge":
            actualizarTABLATODO(chat_id1, colDecision, 'No')
            chat_id1 = str(update.effective_chat.id)
            if idioma == "English":
                context.bot.send_message(chat_id1,
                                     text='I see! Please, send the code of the challenge that you want to judge. Use the Menu above if
you'
                                          ' wanna change what to do. ',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1,
                                     text='Bien, envíame el código del reto que quieres juzgar. Si quieres cambiar que hacer utiliza de
nuevom el Menu anterior. ',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            return JUDGE
        elif query == "Review":
            chat_id1 = str(update.effective_chat.id)
            reto = buscarvariable(chat_id1, colChallenge)
            ultimo = buscarduedates(chat_id1, reto)
            numerochallenges = len(ultimo)
            verify = ultimo[numerochallenges - 1]
            codes = buscarcode2(chat_id1)
            numerochallenges = len(codes)
            ultimocode = codes[numerochallenges - 1]
            while verify == "NA":
                airtable2.delete_by_field('Code', ultimocode)
                if verify3 != []:
                    airtable3.delete_by_field("Code", ultimocode)
                ultimo = buscarduedates(chat_id1, reto)
                verify = ultimo[len(ultimo) - 1]
                codes = buscarcode2(chat_id1)
                ultimocode = codes[len(codes) - 1]
            chat_id1 = str(update.effective_chat.id)
            challenge = buscarchallenges(chat_id1, reto)
            numchallenge = len(challenge)
            actualizarTABLATODO(chat_id1, colDecision, 'No')
            if numchallenge == 1:
                keyboard = [[InlineKeyboardButton(challenge[0], callback_data='Challenge1')]]
                reply_markup = InlineKeyboardMarkup(keyboard)
                if idioma == "English":
                    context.bot.send_message(chat_id1,
                                         text='You have the following challenges. See its descriptions and status pressing in its name'
                                              '. Otherwise, change what you want to do using the previous MENU',
                                         reply_markup=reply_markup)
                elif idioma == "Español":
                    context.bot.send_message(chat_id1,
                                         text='Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si
quieres cambiar lo que hacer utiliza el Menu anterior',
                                         reply_markup=reply_markup)
                return REVIEW
            elif numchallenge == 2:
                keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                             InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')]]
                reply_markup = InlineKeyboardMarkup(keyboard)
                if idioma == "English":
                    context.bot.send_message(chat_id1,
                                         text='You have the following challenges. See its descriptions and status pressing in its name'
                                              '. Otherwise, change what you want to do using the previous MENU',
                                         reply_markup=reply_markup)
                elif idioma == "Español":
                    context.bot.send_message(chat_id1,
                                         text='Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si
quieres cambiar lo que hacer utiliza el Menu anterior',
                                         reply_markup=reply_markup)
                return REVIEW
            elif numchallenge == 3:
                keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                             InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2'),
                             InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3')]]
                reply_markup = InlineKeyboardMarkup(keyboard)
                if idioma == "English":
                    context.bot.send_message(chat_id1,
                                         text='You have the following challenges. See its descriptions and status pressing in its name'
                                              '. Otherwise, change what you want to do using the previous MENU',
                                         reply_markup=reply_markup)
                elif idioma == "Español":
                    context.bot.send_message(chat_id1,
                                             text='Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si 
quieres cambiar lo que hacer utiliza el Menu anterior',
                                             reply_markup=reply_markup)
                return REVIEW
            elif numchallenge == 4:
                keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                             InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')],
                            [InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3'),
                             InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4')]]
                reply_markup = InlineKeyboardMarkup(keyboard)
                if idioma == "English":
                    context.bot.send_message(chat_id1,
                                             text='You have the following challenges. See its descriptions and status pressing in its name'
                                                  '. Otherwise, change what you want to do using the previous MENU',
                                             reply_markup=reply_markup)
                elif idioma == "Español":
                    context.bot.send_message(chat_id1,
                                             text='Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si 
quieres cambiar lo que hacer utiliza el Menu anterior',
                                             reply_markup=reply_markup)
                return REVIEW
            elif numchallenge == 5:
                keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                             InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2'),
                             InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3')],
                            [InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4'),
                             InlineKeyboardButton(challenge[numchallenge - 5], callback_data='Challenge5')]]
                reply_markup = InlineKeyboardMarkup(keyboard)
                if idioma == "English":
                    context.bot.send_message(chat_id1,
                                             text='You have the following challenges. See its descriptions and status pressing in its name'
                                                  '. Otherwise, change what you want to do using the previous MENU',
                                             reply_markup=reply_markup)
                elif idioma == "Español":
                    context.bot.send_message(chat_id1,
                                             text='Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si 
quieres cambiar lo que hacer utiliza el Menu anterior',
                                             reply_markup=reply_markup)
                return REVIEW
            elif numchallenge >= 6:
                keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                             InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')],
                            [InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3'),
                             InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4')],
                            [InlineKeyboardButton(challenge[numchallenge - 5], callback_data='Challenge5'),
                             InlineKeyboardButton(challenge[numchallenge - 6], callback_data='Challenge6')]]
                reply_markup = InlineKeyboardMarkup(keyboard)
                if idioma == "English":
                    context.bot.send_message(chat_id1,
                                             text='You have the following challenges. See its descriptions and status pressing in its name'
                                                  '. Otherwise, change what you want to do using the previous MENU',
                                             reply_markup=reply_markup)
                elif idioma == "Español":
                    context.bot.send_message(chat_id1,
                                             text='Tienes los siguientes retos. Presiona sobre el nombre de del reto que quieras ver. Si 
quieres cambiar lo que hacer utiliza el Menu anterior',
                                             reply_markup=reply_markup)
                return REVIEW
        elif query == "Challenge":
            chat_id1 = str(update.effective_chat.id)
            idioma = buscaridioma1(chat_id1)
            reto = str(1)
            actualizarTABLAVARIABLES(chat_id1, colChallenge, reto)
            if idioma == "English":
                keyboard = [[InlineKeyboardButton("Create", callback_data='Retar'),
                             InlineKeyboardButton("See Challenge", callback_data='Retado')]]
                reply_markup = InlineKeyboardMarkup(keyboard)
                context.bot.send_message(chat_id1,
                                         text='Select *Create* if you want to challenge. However, if someone have challenged you press *See 
Challenge* to see the challenge features and decided if accept it or not',
                                         parse_mode="Markdown", reply_markup=reply_markup)
            elif idioma == "Español":
                keyboard = [[InlineKeyboardButton("Crear Reto", callback_data='Retar'),
                             InlineKeyboardButton("Ver Reto", callback_data='Retado')]]
                reply_markup = InlineKeyboardMarkup(keyboard)
                context.bot.send_message(chat_id1,
                                         text='Selecciona *Crear Reto* si lo que quieres es retar tu a otra persona. \nSi te han retado a ti 
pulsa sobre *Ver Reto* para ver las características de este y ver si aceptarlo o no',
                                         parse_mode="Markdown", reply_markup=reply_markup)
            return DECISION
        elif query == "Retar":
            chat_id1 = str(update.effective_chat.id)
            name = update.effective_chat.first_name
            reto = buscarvariable(chat_id1, colChallenge)
            codigoanterior = buscarcodeanterior(chat_id1)
            if nuevouser == 1:
                ultimo = buscarduedates(chat_id1, reto)
                verify = ultimo[len(ultimo) - 1]
                codes = buscarcode2(chat_id1)
                ultimocode = codes[len(codes) - 1]
                while verify == "NA":
                    airtable2.delete_by_field('Code', ultimocode)
                    verify3 = airtable3.search("Code", ultimocode)
                    if verify3 != []:
                        airtable3.delete_by_field("Code", ultimocode)
                    ultimo = buscarduedates(chat_id1, reto)
                    verify = ultimo[len(ultimo) - 1]
                    codes = buscarcode2(chat_id1)
                    ultimocode = codes[len(codes) - 1]
            record2 = {'Code': codigoanterior}
            airtable5.insert(record2)
            rellenarairtable5(codigoanterior)
            if nuevouser1 == 0:
                record3 = {'User ID': chat_id1}
                airtable1.insert(record3)
                rellenarairtable1(chat_id1)
                actualizarTABLAUSERS(chat_id1, colUsername, name)
                idioma = buscaridioma1(chat_id1)
                print("Idioma {} actualizado".format(name))
                actualizarTABLAUSERS(chat_id1, colIdioma, idioma)
                actualizarTABLATODO(chat_id1, colDecision, 'Yes')
                actualizarTABLADEFIANCE(chat_id1, colJudgeID, chat_id1)
                if idioma == "English":
                    context.bot.send_message(chat_id1, text='Great {}. Lets do it!'.format(name))
                    context.bot.send_message(chat_id1,
                                             text='Describe briefly the challenge goal. Use the Menu above if you wanna change what to do.',
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                elif idioma == "Español":
                    context.bot.send_message(chat_id1, text='Perfecto {}. Vamos a ello!'.format(name))
                    context.bot.send_message(chat_id1,
                                             text='Describe brevemente en que consiste el reto. Usa el Menu de arriba si quieres cambiar lo
que hacer.',
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                return CHALLENGENAME
            else:
                idioma = buscaridioma1(chat_id1)
                actualizarTABLAUSERS(chat_id1, colIdioma, idioma)
                actualizarTABLATODO(chat_id1, colDecision, 'Yes')
                actualizarTABLADEFIANCE(chat_id1, colJudgeID, chat_id1)

                if idioma == "English":
                    context.bot.send_message(chat_id1, text='Great {}. Lets do it!'.format(name))
                    context.bot.send_message(chat_id1,
                                             text='Describe briefly the challenge goal. Use the Menu above if you wanna change what to do.',
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                elif idioma == "Español":
                    context.bot.send_message(chat_id1, text='Perfecto {}. Vamos a ello!'.format(name))
                    context.bot.send_message(chat_id1,
                                             text='Describe brevemente en que consiste el reto. Usa el Menu de arriba si quieres cambiar lo
que hacer.',
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                return CHALLENGENAME
        elif query == "Retado":
            chat_id1 = str(update.effective_chat.id)
            name = update.effective_chat.first_name
            reto = buscarvariable(chat_id1, colChallenge)
            codigoanterior = buscarcodeanterior(chat_id1)
            if nuevouser == 1:
                ultimo = buscarduedates(chat_id1, reto)
                verify = ultimo[len(ultimo) - 1]
                codes = buscarcode2(chat_id1)
                ultimocode = codes[len(codes) - 1]
                while verify == "NA":
                    airtable2.delete_by_field('Code', ultimocode)
                    verify3 = airtable3.search("Code", ultimocode)
                    if verify3 != []:
                        airtable3.delete_by_field("Code", ultimocode)
                    ultimo = buscarduedates(chat_id1, reto)
                    verify = ultimo[len(ultimo) - 1]
                    codes = buscarcode2(chat_id1)
                    ultimocode = codes[len(codes) - 1]
            if nuevouser1 == 0:
                record3 = {'User ID': chat_id1}
                airtable1.insert(record3)
                rellenarairtable1(chat_id1)
                actualizarTABLAUSERS(chat_id1, colUsername, name)
                idioma = buscaridioma1(chat_id1)
                print("Idioma {} actualizado".format(name))
                actualizarTABLAUSERS(chat_id1, colIdioma, idioma)
                actualizarTABLATODO(chat_id1, colDecision, 'Yes')
                if idioma == "English":
                    context.bot.send_message(chat_id1, text='Great {}. Lets do it!'.format(name))
                    context.bot.send_message(chat_id1,
                                             text='Send me the code of the challenge that have been sent to you ',
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                elif idioma == "Español":
                    context.bot.send_message(chat_id1, text='Perfecto {}. Vamos a ello!'.format(name))
                    context.bot.send_message(chat_id1,
                                             text='Envíame el código del reto que te han enviado',
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                return VERCHALLENGE
            else:
                idioma = buscaridioma1(chat_id1)
                actualizarTABLAUSERS(chat_id1, colIdioma, idioma)
                actualizarTABLATODO(chat_id1, colDecision, 'Yes')

                if idioma == "English":
                    context.bot.send_message(chat_id1, text='Great {}. Lets do it!'.format(name))
                    context.bot.send_message(chat_id1,
                                             text='Send me the code of the challenge that have been sent to you ',
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                elif idioma == "Español":
                    context.bot.send_message(chat_id1, text='Perfecto {}. Vamos a ello!'.format(name))
                    context.bot.send_message(chat_id1,
                                             text='Envíame el código del reto que te han enviado',
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                return VERCHALLENGE
    elif change == 1:
        print("El usuario quiere cambiar el/la {} del reto que está creando".format(query))
        chat_id1 = str(update.effective_chat.id)
        nuevouser = int(buscarvariable(chat_id1, colNuevoUser))
        if query == "Name":
            if idioma == "English":
                context.bot.send_message(chat_id1,
                                         text='Describe briefly the challenge goal. Use the Menu above if you wanna change what to do.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1,
                                         text='Describe brevemente en que consiste el reto. Usa el Menu de arriba si quieres cambiar lo que
hacer.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            return CHALLENGENAME
        elif query == "Amount":
            keyboard = [['5€', '10€', '15€', ], ['20€', '25€', 'Other']]
            if idioma == "English":
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Select the quantity that you wanna bet from the options below or send me the amount directly',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Selecciona la cantidad que quieres apostar de las opciones de debajo o envíame la cantidad',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            return AMOUNT
        elif query == "Duedate":
            if idioma == "English":
                keyboard = [['January', 'February', 'March', 'April'], ['May', 'June', 'July', 'August'],
                            ['September', 'October', 'November', 'December']]
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Select the month',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                keyboard = [['Enero', 'Febrero', 'Marzo', 'Abril'], ['Mayo', 'Junio', 'Julio', 'Agosto'],
                            ['Septiembre', 'Octubre', 'Noviembre', 'Diciembre']]
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Selecciona el mes',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            return DATE
        elif query == "Judgename":
            if nuevouser == 1:
                chat_id1 = str(update.effective_chat.id)
                prueba = buscarjueces(chat_id1)
                jueces = buscarjuecesmail(chat_id1)
                longitud = len(prueba)
                if longitud == 1:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1')],
                                [InlineKeyboardButton("New", callback_data='New')]]
                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Selecciona uno de ellos o pulsa Nuevo si quieres
un nuevo juez para esto reto.',
                                                 reply_markup=reply_markup)
                    return EXISTING
                elif longitud == 2:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                                 InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2')],
                                [InlineKeyboardButton("New", callback_data='New')]]
                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Selecciona uno de ellos o pulsa Nuevo si quieres
un nuevo juez para esto reto.',
                                                 reply_markup=reply_markup)
                    return EXISTING
                elif longitud == 3:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                                 InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2'),
                                 InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3')],
                                [InlineKeyboardButton("New", callback_data='New')]]
                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Selecciona uno de ellos o pulsa Nuevo si quieres
un nuevo juez para esto reto.',
                                                 reply_markup=reply_markup)
                    return EXISTING
                elif longitud == 4:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                                 InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2')],
                                [InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3'),
                                 InlineKeyboardButton(prueba[longitud - 4], callback_data='Judge4')],
                                [InlineKeyboardButton("New", callback_data='New')]]
                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Selecciona uno de ellos o pulsa Nuevo si quieres
un nuevo juez para esto reto.',
                                                 reply_markup=reply_markup)
                    return EXISTING
                elif longitud == 5:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                                 InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2'),
                                 InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3')],
                                [InlineKeyboardButton(prueba[longitud - 4], callback_data='Judge4'),
                                 InlineKeyboardButton(prueba[longitud - 5], callback_data='Judge5')],
                                [InlineKeyboardButton("New", callback_data='New')]]
                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Selecciona uno de ellos o pulsa Nuevo si quieres
un nuevo juez para esto reto.',
                                                 reply_markup=reply_markup)
                    return EXISTING
                elif longitud >= 6:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                                 InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2')],
                                [InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3'),
                                 InlineKeyboardButton(prueba[longitud - 4], callback_data='Judge4')],
                                [InlineKeyboardButton(prueba[longitud - 5], callback_data='Judge5'),
                                 InlineKeyboardButton(prueba[longitud - 6], callback_data='Judge6')],
                                [InlineKeyboardButton("New", callback_data='New')]]
                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Selecciona uno de ellos o pulsa Nuevo si quieres
un nuevo juez para esto reto.',
                                                 reply_markup=reply_markup)
                    return EXISTING

            elif nuevouser == 0:
                chat_id1 = str(update.effective_chat.id)
                code = buscarcodeanterior(chat_id1)
                airtable3.delete_by_field('Code', code)
                record = {'Code': code}
                airtable3.insert(record)
                rellenarairtable3(chat_id1)
                if idioma == "English":
                    context.bot.send_message(chat_id1,
                                             text='Who is going to be the judge?, send me his/her name please ',
                                             reply_markup=ReplyKeyboardRemove(
                                                 remove_keyboard=True))
                elif idioma == "Español":
                    context.bot.send_message(chat_id1, text='¿Quién va a ser el juez?, envíame su nombre por favor. ',
                                             reply_markup=ReplyKeyboardRemove(
                                                 remove_keyboard=True))
                return JUDGENAME
        elif query == "JudgeEmail":
            if reto == "0":
                if idioma == "English":
                    context.bot.send_message(chat_id1, text='Provide me the judge email, please',
                                             reply_markup=ReplyKeyboardRemove(
                                                 remove_keyboard=True))
                elif idioma == "Español":
                    context.bot.send_message(chat_id1, text='Enviame el correo del juez por favor',
                                             reply_markup=ReplyKeyboardRemove(
                                                 remove_keyboard=True))
                return JUDGEMAIL
            elif reto == "1":
                if idioma == "English":
                    context.bot.send_message(chat_id1, text='Provide me your email, please',
                                             reply_markup=ReplyKeyboardRemove(
                                                 remove_keyboard=True))
                elif idioma == "Español":
                    context.bot.send_message(chat_id1, text='Envíame tu correo por favor',
                                             reply_markup=ReplyKeyboardRemove(
                                                 remove_keyboard=True))
                return USERMAIL
        elif query == "UserEmail":
            if idioma == "English":
                context.bot.send_message(chat_id1, text=
                'Could you provide me your email? You will receive an email confirming the challenge and future updates.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text=
                '¿Puedes darme tu email? Recibirás un mail confirmando la creación de tu reto y sus actualizaciones.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            return USERMAIL


##FIN PARTE 1


## ITINERARIO CREAR RETO
def challengename(update, context):
    chat_id1 = str(update.effective_chat.id)
    text = update.message.text
    idioma = buscaridiomauser(chat_id1)
    codigoanterior = buscarcodeanterior(chat_id1)
    reto = buscarvariable(chat_id1, colChallenge)
    print("next")
    print(reto)
    if reto == "0":
        actualizarTABLACHALLENGES(chat_id1, colChallengename, text)
    elif reto == "1":
        actualizarTABLADEFIANCE(chat_id1, colChallengename, text)
    nuevouser = int(buscarvariable(chat_id1, colNuevoUser))
    change = int(buscarvariable(chat_id1, colChange))
    if change == 1:
        print("Nombre del reto cambiado")
        if reto == "0":
            nombresretos = buscarchallenges(chat_id1, reto)
            long = len(nombresretos)
            cantidadretos = buscaramounts(chat_id1, reto)
            fechasretos = buscarduedates(chat_id1, reto)
            juecesretos = buscarjueces1(chat_id1)
            juecesmails = buscarjuecesmail1(chat_id1, reto)
            usermails = buscaruseremails(chat_id1)
            if idioma == "English":
                reply_keyboard = [['Change'], ['OK']]
                context.bot.send_message(chat_id1, text='Thank you!')
                context.bot.send_message(chat_id1,
                                         text='You have created this challenge \n\n  *Name 🎫 :* {}  \n *Amount 💰 :* {} \n *DueDate 📅 :*
{} \n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                update.message.reply_text(
                    '\n\n'
                    ' If there is any mistake, press the change button. Otherwise press Ok button.',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                reply_keyboard = [['Cambiar'], ['OK']]
                context.bot.send_message(chat_id1, text='Gracias!!')
                context.bot.send_message(chat_id1,
                                         text='Has creado el siguiente reto: \n\n  *Name 🎫 :* {}  \n *Amount 💰 :* {} \n *DueDate 📅 :* {}
\n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                update.message.reply_text(
                    '\n\n'
                    ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        elif reto == "1":
            nombresretos = buscarchallenges(chat_id1, reto)
            long = len(nombresretos)
            cantidadretos = buscaramounts(chat_id1, reto)
            fechasretos = buscarduedates(chat_id1, reto)
            usermails = buscaruseremails(chat_id1)
            if idioma == "English":
                reply_keyboard = [['Change'], ['OK']]
                context.bot.send_message(chat_id1, text='Thank you!')
                context.bot.send_message(chat_id1,
                                         text='You have created this challenge \n\n*Name 🎫 :* {}  \n*Amount 💰 :* {} \n*DueDate 📅 :* {}
\n*JudgeEmail 📩 :* {} '.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                update.message.reply_text(
                    '\n\n'
                    ' If there is any mistake, press the change button. Otherwise press Ok button.',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                reply_keyboard = [['Cambiar'], ['OK']]
                context.bot.send_message(chat_id1, text='Gracias!!')
                context.bot.send_message(chat_id1,
                                         text='Has creado el siguiente reto: \n\n*Name 🎫 :* {}  \n*Amount 💰 :* {} \n*DueDate 📅 :* {}
\n*JudgeEmail 📩 :* {}'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                update.message.reply_text(
                    '\n\n'
                    ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        return FINAL
    else:
        print("okaa")
        print(reto)
        if reto == "0":
            print("El nombre del reto es {}".format(text))
            if nuevouser == 0:
                record = {'Code': codigoanterior}
                airtable3.insert(record)
                rellenarairtable3(chat_id1)
                if idioma == "English":
                    update.message.reply_text('Who is going to be the judge?, send me his/her name please ',
                                              reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                elif idioma == "Español":
```

```
                update.message.reply_text('¿Quién va a ser el juez?, envíame su nombre por favor',
                                          reply_markup=ReplyKeyboardRemove(remove_keyboard=True))

            return JUDGENAME
        else:
            actualizarTABLATODO(chat_id1, colChallengename, text)
            if nuevouser == 1:
                print("Tiene jueces anteriores")
                chat_id1 = str(update.effective_chat.id)
                prueba = buscarjueces(chat_id1)
                longitud = len(prueba)
                if longitud == 1:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1')],
                                [InlineKeyboardButton("New", callback_data='New')]]
                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for
this challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Selecciona uno de ellos o pulsa Nuevo si
quieres un nuevo juez para esto reto.',
                                                 reply_markup=reply_markup)
                    return EXISTING
                elif longitud == 2:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                                 InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2')],
                                [InlineKeyboardButton("New", callback_data='New')]]
                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for
this challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Selecciona uno de ellos o pulsa Nuevo si
quieres un nuevo juez para esto reto.',
                                                 reply_markup=reply_markup)
                    return EXISTING
                elif longitud == 3:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                                 InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2'),
                                 InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3')],
                                [InlineKeyboardButton("New", callback_data='New')]]
                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for
this challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Selecciona uno de ellos o pulsa Nuevo si
quieres un nuevo juez para esto reto.',
                                                 reply_markup=reply_markup)
                    return EXISTING
                elif longitud == 4:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                                 InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2')],
                                [InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3'),
                                 InlineKeyboardButton(prueba[longitud - 4], callback_data='Judge4')],
                                [InlineKeyboardButton("New", callback_data='New')]]
                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for
this challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Selecciona uno de ellos o pulsa Nuevo si
quieres un nuevo juez para esto reto.',
                                                 reply_markup=reply_markup)
                    return EXISTING
                elif longitud == 5:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                                 InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2'),
                                 InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3')],
                                   [InlineKeyboardButton(prueba[longitud - 4], callback_data='Judge4'),
                                    InlineKeyboardButton(prueba[longitud - 5], callback_data='Judge5')],
                                   [InlineKeyboardButton("New", callback_data='New')]]
                       if idioma == "English":
                           reply_markup = InlineKeyboardMarkup(keyboard)
                           context.bot.send_message(chat_id1,
                                                    text='You have one judge before. Select it, or press NEW if you want a new judge for
this challenge.',
                                                    reply_markup=reply_markup)
                       elif idioma == "Español":
                           reply_markup = InlineKeyboardMarkup(keyboard)
                           context.bot.send_message(chat_id1,
                                                    text='Has utilizado los siguientes jueces. Selecciona uno de ellos o pulsa Nuevo si
quieres un nuevo juez para esto reto.',
                                                    reply_markup=reply_markup)
                       return EXISTING
                   elif longitud >= 6:
                       keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                                    InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2')],
                                   [InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3'),
                                    InlineKeyboardButton(prueba[longitud - 4], callback_data='Judge4')],
                                   [InlineKeyboardButton(prueba[longitud - 5], callback_data='Judge5'),
                                    InlineKeyboardButton(prueba[longitud - 6], callback_data='Judge6')],
                                   [InlineKeyboardButton("New", callback_data='New')]]

                       if idioma == "English":
                           reply_markup = InlineKeyboardMarkup(keyboard)
                           context.bot.send_message(chat_id1,
                                                    text='You have one judge before. Select it, or press NEW if you want a new judge for
this challenge.',
                                                    reply_markup=reply_markup)
                       elif idioma == "Español":
                           reply_markup = InlineKeyboardMarkup(keyboard)
                           context.bot.send_message(chat_id1,
                                                    text='Has utilizado los siguientes jueces. Selecciona uno de ellos o pulsa Nuevo si
quieres un nuevo juez para esto reto.',
                                                    reply_markup=reply_markup)
                       return EXISTING
               elif nuevouser == 0:
                   print("No tiene jueces anteriores")
                   record = {'Code': codigoanterior}
                   airtable3.insert(record)
                   rellenarairtable3(chat_id1)
                   if idioma == "English":
                       update.message.reply_text('Who is going to be the judge?, send me his/her name please ',
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                   elif idioma == "Español":
                       update.message.reply_text('¿Quién va a ser el juez?, envíame su nombre por favor',
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))

                   return JUDGENAME
        elif reto == "1":
            print("El usuario va a ingresar la cantidad")
            if idioma == "English":
                keyboard = [['5€', '10€', '15€', ], ['20€', '25€', 'Other']]
                update.message.reply_text(
                    '\n\n'
                    'Select the quantity that you wanna bet from the options below or send me the amount directly',
                    reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                keyboard = [['5€', '10€', '15€', ], ['20€', '25€', 'Other']]
                update.message.reply_text(
                    '\n\n'
                    'Selecciona la cantidad que quieres apostar de las opciones de abajo o envíame la cantidad directamente',
                    reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            return AMOUNT


def existingjudge(update, context):
    chat_id1 = str(update.effective_chat.id)
    existing = str(1)
    actualizarTABLAVARIABLES(chat_id1, colExistingJudge, existing)
    idioma = buscaridiomauser(chat_id1)
    query = update.callback_query.data
    change = int(buscarvariable(chat_id1, colChange))
    reto = buscarvariable(chat_id1, colChallenge)
    if change == 0:
        keyboard = [['5€', '10€', '15€', ], ['20€', '25€', 'Other']]
        if query == "Judge1":
            # actualizo airtable de Challenges y All con nombre y email del judge elegido
            nombrejuez = buscarjueces1(chat_id1)
            long = len(nombrejuez)
            print(long)
            emailjuez = buscarjuecesmail1(chat_id1, reto)
            print("El nombre del juez es {}".format(nombrejuez[long - 2]))
            print("El email del juez es {}".format(emailjuez[long - 2]))
            actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 2])
            actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 2])
            actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 2])
            actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 2])
            if idioma == "English":
                context.bot.send_message(chat_id1, text=
                'How many money do you want to bet?')
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Select the quantity from the options below or send me the amount directly',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text=
                '¿Cuánto dinero quieres apostar?')
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Selecciona la cantidad de las opciones de debajo o envíame la cantidad',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))

            return AMOUNT
        elif query == "Judge2":
            # actualizo airtable de Challenges y All con nombre y email del judge elegido
            nombrejuez = buscarjueces1(chat_id1)
            long = len(nombrejuez)
            emailjuez = buscarjuecesmail1(chat_id1, reto)
            print("El nombre del juez es {}".format(nombrejuez[long - 3]))
            print("El email del juez es {}".format(emailjuez[long - 3]))
            actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 3])
            actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 3])
            actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 3])
            actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 3])

            if idioma == "English":
                context.bot.send_message(chat_id1, text=
                'How many money do you want to bet?')
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Select the quantity from the options below or send me the amount directly',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text=
                '¿Cuánto dinero quieres apostar?')
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Selecciona la cantidad de las opciones de debajo o envíame la cantidad',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))

            return AMOUNT
        elif query == "Judge3":
            # actualizo airtable de Challenges y All con nombre y email del judge elegido
            nombrejuez = buscarjueces1(chat_id1)
            long = len(nombrejuez)
            emailjuez = buscarjuecesmail1(chat_id1, reto)
            print("El nombre del juez es {}".format(nombrejuez[long - 4]))
            print("El email del juez es {}".format(emailjuez[long - 4]))
            actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 4])
            actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 4])
            actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 4])
            actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 4])
            if idioma == "English":
                context.bot.send_message(chat_id1, text=
                'How many money do you want to bet?')
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Select the quantity from the options below or send me the amount directly',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text=
                '¿Cuánto dinero quieres apostar?')
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Selecciona la cantidad de las opciones de debajo o envíame la cantidad',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))

            return AMOUNT
        elif query == "Judge4":
            # actualizo airtable de Challenges y All con nombre y email del judge elegido
            nombrejuez = buscarjueces1(chat_id1)
            long = len(nombrejuez)
            emailjuez = buscarjuecesmail1(chat_id1, reto)
            print("El nombre del juez es {}".format(nombrejuez[long - 5]))
            print("El email del juez es {}".format(emailjuez[long - 5]))
            actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 5])
            actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 5])
            actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 5])
            actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 5])
            if idioma == "English":
                context.bot.send_message(chat_id1, text=
                'How many money do you want to bet?')
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Select the quantity from the options below or send me the amount directly',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text=
                '¿Cuánto dinero quieres apostar?')
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Selecciona la cantidad de las opciones de debajo o envíame la cantidad',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))

            return AMOUNT
        elif query == "Judge5":
            # actualizo airtable de Challenges y All con nombre y email del judge elegido
            nombrejuez = buscarjueces1(chat_id1)
            long = len(nombrejuez)
            emailjuez = buscarjuecesmail1(chat_id1, reto)
            print("El nombre del juez es {}".format(nombrejuez[long - 6]))
            print("El email del juez es {}".format(emailjuez[long - 6]))
            actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 6])
            actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 6])
            actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 6])
            actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 6])
            if idioma == "English":
                context.bot.send_message(chat_id1, text=
                'How many money do you want to bet?')
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Select the quantity from the options below or send me the amount directly',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text=
                '¿Cuánto dinero quieres apostar?')
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Selecciona la cantidad de las opciones de debajo o envíame la cantidad',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))

            return AMOUNT
        elif query == "Judge6":
            # actualizo airtable de Challenges y All con nombre y email del judge elegido
            nombrejuez = buscarjueces1(chat_id1)
            long = len(nombrejuez)
            emailjuez = buscarjuecesmail1(chat_id1, reto)
            print("El nombre del juez es {}".format(nombrejuez[long - 7]))
            print("El email del juez es {}".format(emailjuez[long - 7]))
            actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 7])
            actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 7])
            actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 7])
            actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 7])
            if idioma == "English":
                context.bot.send_message(chat_id1, text=
                'How many money do you want to bet?')
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Select the quantity from the options below or send me the amount directly',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text=
                '¿Cuánto dinero quieres apostar?')
                context.bot.send_message(chat_id1, text=
                '\n\n'
                'Selecciona la cantidad de las opciones de debajo o envíame la cantidad',
                                         reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))

            return AMOUNT
        elif query == "New":
            existing = str(0)
            actualizarTABLAVARIABLES(chat_id1, colExistingJudge, existing)
            code = buscarcodeanterior(chat_id1)
            record = {'Code': code}
            airtable3.insert(record)
            rellenarairtable3(chat_id1)
            if idioma == "English":
                context.bot.send_message(chat_id1, text='Who is going to be the judge?, send me his/her name please ',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text='¿Quién va a ser el juez?, envíame su nombre por favor ',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            return JUDGENAME
    elif change == 1:
        reto = buscarvariable(chat_id1, reto)
        chat_id1 = str(update.effective_chat.id)
        if query == "Judge1":
            # actualizo airtable de Challenges y All con nombre y email del judge elegido
            nombrejuez = buscarjueces1(chat_id1)
            long = len(nombrejuez)
            emailjuez = buscarjuecesmail1(chat_id1, reto)
            print("Se cambia el nombre del juez a {}".format(nombrejuez[long - 2]))
            print("Se cambia el email del juez a {}".format(emailjuez[long - 2]))
            actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 2])
            actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 2])
            actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 2])
            actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 2])
            nombresretos = buscarchallenges(chat_id1, reto)
            long = len(nombresretos)
            cantidadretos = buscaramounts(chat_id1, reto)
            fechasretos = buscarduedates(chat_id1, reto)
            juecesretos = buscarjueces1(chat_id1)
            juecesmails = buscarjuecesmail1(chat_id1, reto)
            usermails = buscaruseremails(chat_id1)
            if idioma == "English":
                reply_keyboard = [['Change'], ['OK']]
                context.bot.send_message(chat_id1, text='Thank you!')
                context.bot.send_message(chat_id1,
                                         text='You have created this challenge \n\n  *Name 🏷️ :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :*
{} \n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text=
                                         '\n\n'
                                         ' If there is any mistake, press the change button. Otherwise press Ok button.',
                                         reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                reply_keyboard = [['Cambiar'], ['OK']]
                context.bot.send_message(chat_id1, text='Gracias!!')
                context.bot.send_message(chat_id1,
                                         text='Has creado el siguiente reto: \n\n  *Name 🏷️ :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :* {}
\n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text=
                                         '\n\n'
                                         ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                                         reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

            return FINAL
        elif query == "Judge2":
            # actualizo airtable de Challenges y All con nombre y email del judge elegido
            nombrejuez = buscarjueces1(chat_id1)
            long = len(nombrejuez)
            emailjuez = buscarjuecesmail1(chat_id1, reto)
            print("Se cambia el nombre del juez a {}".format(nombrejuez[long - 3]))
            print("Se cambia el email del juez a {}".format(emailjuez[long - 3]))
            actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 3])
            actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 3])
            actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 3])
            actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 3])
            nombresretos = buscarchallenges(chat_id1, reto)
            long = len(nombresretos)
            cantidadretos = buscaramounts(chat_id1, reto)
            fechasretos = buscarduedates(chat_id1, reto)
            juecesretos = buscarjueces1(chat_id1)
            juecesmails = buscarjuecesmail1(chat_id1, reto)
            usermails = buscaruseremails(chat_id1)
            if idioma == "English":
                reply_keyboard = [['Change'], ['OK']]
                context.bot.send_message(chat_id1, text='Thank you!')
                context.bot.send_message(chat_id1,
                                         text='You have created this challenge \n\n  *Name 🏷️ :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :*
{} \n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text=
                                         '\n\n'
                                         ' If there is any mistake, press the change button. Otherwise press Ok button.',
                                         reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                reply_keyboard = [['Cambiar'], ['OK']]
                context.bot.send_message(chat_id1, text='Gracias!!')
                context.bot.send_message(chat_id1,
                                         text='Has creado el siguiente reto: \n\n  *Name 🎟 :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :* {}
\n *Judge 🧑‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text=
                                         '\n\n'
                                         ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                                         reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

            return FINAL
        elif query == "Judge3":
            # actualizo airtable de Challenges y All con nombre y email del judge elegido
            nombrejuez = buscarjueces1(chat_id1)
            long = len(nombrejuez)
            emailjuez = buscarjuecesmail1(chat_id1, reto)
            print("Se cambia el nombre del juez a {}".format(nombrejuez[long - 4]))
            print("Se cambia el email del juez a {}".format(emailjuez[long - 4]))
            actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 4])
            actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 4])
            actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 4])
            actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 4])
            nombresretos = buscarchallenges(chat_id1, reto)
            long = len(nombresretos)
            cantidadretos = buscaramounts(chat_id1, reto)
            fechasretos = buscarduedates(chat_id1, reto)
            juecesretos = buscarjueces1(chat_id1)
            juecesmails = buscarjuecesmail1(chat_id1, reto)
            usermails = buscaruseremails(chat_id1)

            if idioma == "English":
                reply_keyboard = [['Change'], ['OK']]
                context.bot.send_message(chat_id1, text='Thank you!')
                context.bot.send_message(chat_id1,
                                         text='You have created this challenge \n\n  *Name 🎟 :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :*
{} \n *Judge 🧑‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text=
                                         '\n\n'
                                         ' If there is any mistake, press the change button. Otherwise press Ok button.',
                                         reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                reply_keyboard = [['Cambiar'], ['OK']]
                context.bot.send_message(chat_id1, text='Gracias!!')
                context.bot.send_message(chat_id1,
                                         text='Has creado el siguiente reto: \n\n  *Name 🎟 :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :* {}
\n *Judge 🧑‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text=
                                         '\n\n'
                                         ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                                         reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

            return FINAL
        elif query == "Judge4":
            # actualizo airtable de Challenges y All con nombre y email del judge elegido
            nombrejuez = buscarjueces1(chat_id1)
            long = len(nombrejuez)
            emailjuez = buscarjuecesmail1(chat_id1, reto)
            print("Se cambia el nombre del juez a {}".format(nombrejuez[long - 5]))
            print("Se cambia el email del juez a {}".format(emailjuez[long - 5]))
            actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 5])
            actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 5])
            actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 5])
            actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 5])
            nombresretos = buscarchallenges(chat_id1, reto)
            long = len(nombresretos)
            cantidadretos = buscaramounts(chat_id1, reto)
            fechasretos = buscarduedates(chat_id1, reto)
            juecesretos = buscarjueces1(chat_id1)
            juecesmails = buscarjuecesmail1(chat_id1, reto)
            usermails = buscaruseremails(chat_id1)
            if idioma == "English":
                reply_keyboard = [['Change'], ['OK']]
                context.bot.send_message(chat_id1, text='Thank you!')
                context.bot.send_message(chat_id1,
                                         text='You have created this challenge \n\n  *Name 🎫 :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :*
{} \n *Judge 👨⚖:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text=
                                         '\n\n'
                                         ' If there is any mistake, press the change button. Otherwise press Ok button.',
                                         reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                reply_keyboard = [['Cambiar'], ['OK']]
                context.bot.send_message(chat_id1, text='Gracias!!')
                context.bot.send_message(chat_id1,
                                         text='Has creado el siguiente reto: \n\n  *Name 🎫 :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :* {}
\n *Judge 👨⚖:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text=
                                         '\n\n'
                                         ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                                         reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

            return FINAL
        elif query == "Judge5":
            # actualizo airtable de Challenges y All con nombre y email del judge elegido
            nombrejuez = buscarjueces1(chat_id1)
            long = len(nombrejuez)
            emailjuez = buscarjuecesmail1(chat_id1, reto)
            print("Se cambia el nombre del juez a {}".format(nombrejuez[long - 6]))
            print("Se cambia el email del juez a {}".format(emailjuez[long - 6]))
            actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 6])
            actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 6])
            actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 6])
            actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 6])
            nombresretos = buscarchallenges(chat_id1, reto)
            long = len(nombresretos)
            cantidadretos = buscaramounts(chat_id1, reto)
            fechasretos = buscarduedates(chat_id1, reto)
            juecesretos = buscarjueces1(chat_id1)
            juecesmails = buscarjuecesmail1(chat_id1, reto)
            usermails = buscaruseremails(chat_id1)
            if idioma == "English":
                reply_keyboard = [['Change'], ['OK']]
                context.bot.send_message(chat_id1, text='Thank you!')
                context.bot.send_message(chat_id1,
                                         text='You have created this challenge \n\n  *Name 🎫 :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :*
{} \n *Judge 👨⚖:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text=
                                         '\n\n'
                                         ' If there is any mistake, press the change button. Otherwise press Ok button.',
                                         reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                reply_keyboard = [['Cambiar'], ['OK']]
                context.bot.send_message(chat_id1, text='Gracias!!')
                context.bot.send_message(chat_id1,
                                         text='Has creado el siguiente reto: \n\n  *Name 🎫 :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :* {}
\n *Judge 👨⚖:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text=
                                         '\n\n'
                                         ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                                         reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

            return FINAL
        elif query == "Judge6":
            # actualizo airtable de Challenges y All con nombre y email del judge elegido
            nombrejuez = buscarjueces1(chat_id1)
            long = len(nombrejuez)
            emailjuez = buscarjuecesmail1(chat_id1, reto)
            print("Se cambia el nombre del juez a {}".format(nombrejuez[long - 7]))
            print("Se cambia el email del juez a {}".format(emailjuez[long - 7]))
            actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 7])
            actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 7])
            actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 7])
            actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 7])
            nombresretos = buscarchallenges(chat_id1, reto)
            long = len(nombresretos)
            cantidadretos = buscaramounts(chat_id1, reto)
            fechasretos = buscarduedates(chat_id1, reto)
            juecesretos = buscarjueces1(chat_id1)
            juecesmails = buscarjuecesmail1(chat_id1, reto)
            usermails = buscaruseremails(chat_id1)

            if idioma == "English":
                reply_keyboard = [['Change'], ['OK']]
                context.bot.send_message(chat_id1, text='Thank you!')
                context.bot.send_message(chat_id1,
                                         text='You have created this challenge \n\n  *Name 🏷 :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :*
{} \n *Judge 🧑‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text=
                                         '\n\n'
                                         ' If there is any mistake, press the change button. Otherwise press Ok button.',
                                         reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                reply_keyboard = [['Cambiar'], ['OK']]
                context.bot.send_message(chat_id1, text='Gracias!!')
                context.bot.send_message(chat_id1,
                                         text='Has creado el siguiente reto: \n\n  *Name 🏷 :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :* {}
\n *Judge 🧑‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                         parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text=
                                         '\n\n'
                                         ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                                         reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

            return FINAL
        elif query == "New":
            existing = str(0)
            actualizarTABLAVARIABLES(chat_id1, colExistingJudge, existing)
            code = buscarcodeanterior(chat_id1)
            verify = airtable3.search('Code', code)
            if verify != []:
                airtable3.delete_by_field('Code', code)
            record = {'Code': code}
            airtable3.insert(record)
            rellenarairtable3(chat_id1)
            if idioma == "English":
                context.bot.send_message(chat_id1, text='Who is going to be the judge?, send me his/her name please ',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text='¿Quién va a ser el juez?, enviame su nombre por favor ',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            return JUDGENAME


def existingjudge1(update, context):
    chat_id1 = str(update.effective_chat.id)
    idioma = buscaridiomauser(chat_id1)
    existing = str(1)
    actualizarTABLAVARIABLES(chat_id1, colExistingJudge, existing)
    query = update.message.text
    reto = buscarvariable(chat_id1, colChallenge)
    if query == "Judge1":
        # actualizo airtable de Challenges y All con nombre y email del judge elegido
        nombrejuez = buscarjueces1(chat_id1)
        long = len(nombrejuez)
        emailjuez = buscarjuecesmail1(chat_id1, reto)
        print("El nombre del juez es {}".format(nombrejuez[long - 2]))
        print("El email del juez es {}".format(emailjuez[long - 2]))
        actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 2])
        actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 2])
        actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 2])
        actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 2])
        if idioma == "English":
            reply_keyboard = [['Insert Quantity']]
            context.bot.send_message(chat_id1, text=
            'How many money do you want to bet?',
                                     reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        elif idioma == "Español":
            reply_keyboard = [['Insertar Cantidad']]

            context.bot.send_message(chat_id1, text=
            '¿Cuánto dinero quieres apostar?',
                                     reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

        return ADDITIONAL
    elif query == "Judge2":
        # actualizo airtable de Challenges y All con nombre y email del judge elegido
        nombrejuez = buscarjueces1(chat_id1)
        long = len(nombrejuez)
        emailjuez = buscarjuecesmail1(chat_id1, reto)
        print("El nombre del juez es {}".format(nombrejuez[long - 3]))
        print("El email del juez es {}".format(emailjuez[long - 3]))
        actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 3])
        actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 3])
        actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 3])
        actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 3])

        if idioma == "English":
            reply_keyboard = [['Insert Quantity']]

            context.bot.send_message(chat_id1, text=
            'How many money do you want to bet?',
                                     reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        elif idioma == "Español":
            reply_keyboard = [['Insertar Cantidad']]

            context.bot.send_message(chat_id1, text=
            '¿Cuánto dinero quieres apostar?',
                                     reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

        return ADDITIONAL
    elif query == "Judge3":
        # actualizo airtable de Challenges y All con nombre y email del judge elegido
        nombrejuez = buscarjueces1(chat_id1)
        long = len(nombrejuez)
        emailjuez = buscarjuecesmail1(chat_id1, reto)
        print("El nombre del juez es {}".format(nombrejuez[long - 4]))
        print("El email del juez es {}".format(emailjuez[long - 4]))
        actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 4])
        actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 4])
        actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 4])
        actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 4])
        if idioma == "English":
            reply_keyboard = [['Insert Quantity']]

            context.bot.send_message(chat_id1, text=
            'How many money do you want to bet?',
                                     reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        elif idioma == "Español":
            reply_keyboard = [['Insertar Cantidad']]

            context.bot.send_message(chat_id1, text=
            '¿Cuánto dinero quieres apostar?',
                                     reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

        return ADDITIONAL
    elif query == "Judge4":
        # actualizo airtable de Challenges y All con nombre y email del judge elegido
        nombrejuez = buscarjueces1(chat_id1)
        long = len(nombrejuez)
        emailjuez = buscarjuecesmail1(chat_id1, reto)
        print("El nombre del juez es {}".format(nombrejuez[long - 5]))
        print("El email del juez es {}".format(emailjuez[long - 5]))
        actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 5])
        actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 5])
        actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 5])
        actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 5])
        if idioma == "English":
            reply_keyboard = [['Insert Quantity']]

            context.bot.send_message(chat_id1, text=
            'How many money do you want to bet?',
                                     reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        elif idioma == "Español":
            reply_keyboard = [['Insertar Cantidad']]

            context.bot.send_message(chat_id1, text=
            '¿Cuánto dinero quieres apostar?',
                                     reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        return ADDITIONAL
    elif query == "Judge5":
        # actualizo airtable de Challenges y All con nombre y email del judge elegido
        nombrejuez = buscarjueces1(chat_id1)
        long = len(nombrejuez)
        emailjuez = buscarjuecesmail1(chat_id1, reto)
        print("El nombre del juez es {}".format(nombrejuez[long - 6]))
        print("El email del juez es {}".format(emailjuez[long - 6]))
        actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 6])
        actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 6])
        actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 6])
        actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 6])
        if idioma == "English":
            reply_keyboard = [['Insert Quantity']]

            context.bot.send_message(chat_id1, text=
            'How many money do you want to bet?',
                                     reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        elif idioma == "Español":
            reply_keyboard = [['Insertar Cantidad']]

            context.bot.send_message(chat_id1, text=
            '¿Cuánto dinero quieres apostar?',
                                     reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

        return ADDITIONAL
    elif query == "Judge6":
        # actualizo airtable de Challenges y All con nombre y email del judge elegido
        nombrejuez = buscarjueces1(chat_id1)
        long = len(nombrejuez)
        emailjuez = buscarjuecesmail1(chat_id1, reto)
        print("El nombre del juez es {}".format(nombrejuez[long - 7]))
        print("El email del juez es {}".format(emailjuez[long - 7]))
        actualizarTABLATODO(chat_id1, colJudgename, nombrejuez[long - 7])
        actualizarTABLATODO(chat_id1, colEmailJudge, emailjuez[long - 7])
        actualizarTABLACHALLENGES(chat_id1, colJudgename, nombrejuez[long - 7])
        actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjuez[long - 7])
        if idioma == "English":
            reply_keyboard = [['Insert Quantity']]

            context.bot.send_message(chat_id1, text=
            'How many money do you want to bet?',
                                     reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        elif idioma == "Español":
            reply_keyboard = [['Insertar Cantidad']]

            context.bot.send_message(chat_id1, text=
            '¿Cuánto dinero quieres apostar?',
                                     reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

        return ADDITIONAL
    elif query == "New":
        existing = str(0)
        actualizarTABLAVARIABLES(chat_id1, colExistingJudge, existing)
        code = buscarcodeanterior(chat_id1)
        record = {'Code': code}
        airtable3.insert(record)
        rellenarairtable3(chat_id1)
        if idioma == "English":
            context.bot.send_message(chat_id1, text='Who is going to be the judge?, send me his/her name please ',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
        elif idioma == "Español":
            context.bot.send_message(chat_id1, text='¿Quién va a ser el juez?, send me his/her name please ',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))

        return JUDGENAME
    else:
        chat_id1 = str(update.effective_chat.id)
        prueba = buscarjueces(chat_id1)
        longitud = len(prueba)
        if longitud == 1:
            keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1')],
                        [InlineKeyboardButton("New", callback_data='New')]]
            if idioma == "English":
                reply_markup = InlineKeyboardMarkup(keyboard)
                context.bot.send_message(chat_id1,
                                         text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                         reply_markup=reply_markup)
            elif idioma == "Español":
                reply_markup = InlineKeyboardMarkup(keyboard)
                context.bot.send_message(chat_id1,
                                         text='Has utilizado los siguientes jueces. Select it, or press NEW if you want a new judge for this
challenge.',
                                         reply_markup=reply_markup)
            return EXISTING
        elif longitud == 2:
            keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                         InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2')],
                        [InlineKeyboardButton("New", callback_data='New')]]
            if idioma == "English":
                reply_markup = InlineKeyboardMarkup(keyboard)
                context.bot.send_message(chat_id1,
                                         text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                         reply_markup=reply_markup)
            elif idioma == "Español":
                reply_markup = InlineKeyboardMarkup(keyboard)
                context.bot.send_message(chat_id1,
                                         text='Has utilizado los siguientes jueces. Select it, or press NEW if you want a new judge for this
challenge.',
                                         reply_markup=reply_markup)
            return EXISTING
        elif longitud == 3:
            keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                         InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2'),
                         InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3')],
                        [InlineKeyboardButton("New", callback_data='New')]]
            if idioma == "English":
                reply_markup = InlineKeyboardMarkup(keyboard)
                context.bot.send_message(chat_id1,
                                         text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                         reply_markup=reply_markup)
            elif idioma == "Español":
                reply_markup = InlineKeyboardMarkup(keyboard)
                context.bot.send_message(chat_id1,
                                         text='Has utilizado los siguientes jueces. Select it, or press NEW if you want a new judge for this
challenge.',
                                         reply_markup=reply_markup)
            return EXISTING
        elif longitud == 4:
            keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                         InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2')],
                        [InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3'),
                         InlineKeyboardButton(prueba[longitud - 4], callback_data='Judge4')],
                        [InlineKeyboardButton("New", callback_data='New')]]
            if idioma == "English":
                reply_markup = InlineKeyboardMarkup(keyboard)
                context.bot.send_message(chat_id1,
                                         text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                         reply_markup=reply_markup)

            elif idioma == "Español":
                reply_markup = InlineKeyboardMarkup(keyboard)
                context.bot.send_message(chat_id1,
                                         text='Has utilizado los siguientes jueces. Select it, or press NEW if you want a new judge for this
challenge.',
                                         reply_markup=reply_markup)
            return EXISTING
        elif longitud == 4:
            keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                         InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2'),
                         InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3')],
                        [InlineKeyboardButton(prueba[longitud - 4], callback_data='Judge4'),
                         InlineKeyboardButton(prueba[longitud - 5], callback_data='Judge5')],
                        [InlineKeyboardButton("New", callback_data='New')]]
            if idioma == "English":
                reply_markup = InlineKeyboardMarkup(keyboard)
                context.bot.send_message(chat_id1,
                                         text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                         reply_markup=reply_markup)
            elif idioma == "Español":
                reply_markup = InlineKeyboardMarkup(keyboard)
                context.bot.send_message(chat_id1,
                                         text='Has utilizado los siguientes jueces. Select it, or press NEW if you want a new judge for this
challenge.',
                                         reply_markup=reply_markup)
            return EXISTING
        elif longitud >= 6:
            keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                         InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2')],
                        [InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3'),
                         InlineKeyboardButton(prueba[longitud - 4], callback_data='Judge4')],
                        [InlineKeyboardButton(prueba[longitud - 5], callback_data='Judge5'),
                         InlineKeyboardButton(prueba[longitud - 6], callback_data='Judge6')],
                        [InlineKeyboardButton("New", callback_data='New')]]
            if idioma == "English":
                reply_markup = InlineKeyboardMarkup(keyboard)
                context.bot.send_message(chat_id1,
                                         text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                         reply_markup=reply_markup)
            elif idioma == "Español":
                reply_markup = InlineKeyboardMarkup(keyboard)
                context.bot.send_message(chat_id1,
                                         text='Has utilizado los siguientes jueces. Select it, or press NEW if you want a new judge for this
challenge.',
                                         reply_markup=reply_markup)
            return EXISTING


def judgename(update, context):
    chat_id1 = str(update.effective_chat.id)
    text = update.message.text
    idioma = buscaridiomauser(chat_id1)
    print("El nombre del juez es {}.".format(text))
    actualizarTABLATODO(chat_id1, colJudgename, text)
    actualizarTABLACHALLENGES(chat_id1, colJudgename, text)
    actualizarTABLAJUDGES(chat_id1, colNamejuez, text)
    if idioma == "English":
        update.message.reply_text('Provide me the judge email, please',
                                  reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
    elif idioma == "Español":
        update.message.reply_text('Por favor, enviame el email del juez',
                                  reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
    return JUDGEMAIL


def judgemail(update, context):
    text = update.message.text
    chat_id1 = str(update.effective_chat.id)
    idioma = buscaridiomauser(chat_id1)
    existing = int(buscarvariable(chat_id1, colExistingJudge))
    change = int(buscarvariable(chat_id1, colChange))
    reto = buscarvariable(chat_id1, colChallenge)
    if ("@" in text) and ("." in text):
        a = 1
    else:
        a = 0
    if a == 0:
        if idioma == "English":
            context.bot.send_message(chat_id1, text='Sorry, this format is not accepted. ')
            update.message.reply_text('Please, verify that you have sent a valid email and resend it to me.')
        elif idioma == "Español":
            context.bot.send_message(chat_id1, text='Sorry, this format is not accepted. ')
            update.message.reply_text('Please, verify that you have sent a valid email and resend it to me.')
        return JUDGEMAIL
    else:
        actualizarTABLATODO(chat_id1, colEmailJudge, text)
        actualizarTABLACHALLENGES(chat_id1, colEmailJudge, text)
        if change == 0:
            print("El email del juez es {}.".format(text))
            chat_id1 = str(update.effective_chat.id)
            emails = todosjuecesemails()
            if (text in emails):
                code = buscarcodeanterior(chat_id1)
                airtable3.delete_by_field('Code', code)
            else:
                actualizarTABLAJUDGES(chat_id1, colEmailJudge, text)
            if idioma == "English":
                if existing == 0 and reto == "0":
                    reply_keyboard = [['Insert Quantity'], ['Return']]

                    context.bot.send_message(chat_id1, text=
                    'How many money do you want to bet?',
                                             reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif existing == 1 or reto == "1":
                    reply_keyboard = [['Insert Quantity']]

                    context.bot.send_message(chat_id1, text=
                    'How many money do you want to bet?',
                                             reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                if existing == 0 and reto == "0":
                    reply_keyboard = [['Insertar Cantidad'], ['Volver Atrás']]

                    context.bot.send_message(chat_id1, text=
                    '¿Cuánto dinero quieres apostar?',
                                             reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif existing == 1 or reto == "1":
```

```
                reply_keyboard = [['Insertar Cantidad']]

                context.bot.send_message(chat_id1, text=
                '¿Cuánto dinero quieres apostar?',
                                         reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

            return ADDITIONAL
        elif change == 1:
            reto = buscarvariable(chat_id1, colChallenge)
            print("Email del juez cambiado")
            chat_id1 = str(update.effective_chat.id)
            emails = todosjuecesemails()
            existing = int(buscarvariable(chat_id1, colExistingJudge))
            if (text in emails):
                print("El email ya esta en la base de datos por tanto este juez ya lo tenemos registrado")
                if existing == 0:
                    code = buscarcodeanterior(chat_id1)
                    airtable3.delete_by_field('Code', code)
            else:
                actualizarTABLAJUDGES(chat_id1, colEmailJudge, text)

            nombresretos = buscarchallenges(chat_id1, reto)
            long = len(nombresretos)
            cantidadretos = buscaramounts(chat_id1, reto)
            fechasretos = buscarduedates(chat_id1, reto)
            juecesretos = buscarjueces1(chat_id1)
            juecesmails = buscarjuecesmail1(chat_id1, reto)
            usermails = buscaruseremails(chat_id1)

            if idioma == "English":
                reply_keyboard = [['Change'], ['OK']]
                context.bot.send_message(chat_id1, text='Thank you!')
                context.bot.send_message(chat_id1,
                                     text='You have created this challenge \n\n  *Name 🏷 :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :*
{} \n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                         nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                         juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                     parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                update.message.reply_text(
                    '\n\n'
                    ' If there is any mistake, press the change button. Otherwise press Ok button.',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                reply_keyboard = [['Cambiar'], ['OK']]
                context.bot.send_message(chat_id1, text='Gracias!!')
                context.bot.send_message(chat_id1,
                                     text='Has creado el siguiente reto: \n\n  *Name 🏷 :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :* {}
\n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                         nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                         juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                     parse_mode="Markdown", reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                update.message.reply_text(
                    '\n\n'
                    ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

            return FINAL


def metercantidad(update, context):
    chat_id1 = str(update.effective_chat.id)
    idioma = buscaridiomauser(chat_id1)
    existing = int(buscarvariable(chat_id1, colExistingJudge))
    nuevouser = int(buscarvariable(chat_id1, colNuevoUser))
    if existing == 0:
        print("Se ha equivocado en el email")
        if idioma == "English":
            update.message.reply_text('Provide me the judge email, please',
                                      reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
        elif idioma == "Español":
            update.message.reply_text('Por favor, enviame el email del juez',
                                      reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
        return JUDGEMAIL
    if existing == 1:
        print("Se ha equivocado eligiendo juez")
        if nuevouser == 0:
            record = {'Code': codigoanterior}
            airtable3.insert(record)
            rellenarairtable3(chat_id1)
            if idioma == "English":
                update.message.reply_text('Who is going to be the judge?, send me his/her name please ',
                                          reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text('¿Quién va a ser el juez?, send me his/her name please ',
                                          reply_markup=ReplyKeyboardRemove(remove_keyboard=True))

            return JUDGENAME

        else:
            if nuevouser == 1:
                print("Tiene jueces anteriores")
                chat_id1 = str(update.effective_chat.id)
                prueba = buscarjueces(chat_id1)
                longitud = len(prueba)
                if longitud == 1:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1')],
                                [InlineKeyboardButton("New", callback_data='New')]]
                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Select it, or press NEW if you want a new judge
for this challenge.',
                                                 reply_markup=reply_markup)
                    return EXISTING
                elif longitud == 2:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                                 InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2')],
                                [InlineKeyboardButton("New", callback_data='New')]]
                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Select it, or press NEW if you want a new judge
for this challenge.',
                                                 reply_markup=reply_markup)
                    return EXISTING
                elif longitud == 3:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                                 InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2'),
                                 InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3')],
                                [InlineKeyboardButton("New", callback_data='New')]]
                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Select it, or press NEW if you want a new judge
for this challenge.',
                                                 reply_markup=reply_markup)
                    return EXISTING
                elif longitud == 4:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                                 InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2')],
                                [InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3'),
                                 InlineKeyboardButton(prueba[longitud - 4], callback_data='Judge4')],
                                [InlineKeyboardButton("New", callback_data='New')]]
                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Select it, or press NEW if you want a new judge
for this challenge.',
                                                 reply_markup=reply_markup)
                    return EXISTING
                elif longitud == 5:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                                 InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2'),
                                 InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3')],
                                [InlineKeyboardButton(prueba[longitud - 4], callback_data='Judge4'),
                                 InlineKeyboardButton(prueba[longitud - 5], callback_data='Judge5')],
                                [InlineKeyboardButton("New", callback_data='New')]]
                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Select it, or press NEW if you want a new judge
for this challenge.',
                                                 reply_markup=reply_markup)
                    return EXISTING
                elif longitud >= 6:
                    keyboard = [[InlineKeyboardButton(prueba[longitud - 1], callback_data='Judge1'),
                                 InlineKeyboardButton(prueba[longitud - 2], callback_data='Judge2')],
                                [InlineKeyboardButton(prueba[longitud - 3], callback_data='Judge3'),
                                 InlineKeyboardButton(prueba[longitud - 4], callback_data='Judge4')],
                                [InlineKeyboardButton(prueba[longitud - 5], callback_data='Judge5'),
                                 InlineKeyboardButton(prueba[longitud - 6], callback_data='Judge6')],
                                [InlineKeyboardButton("New", callback_data='New')]]

                    if idioma == "English":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='You have one judge before. Select it, or press NEW if you want a new judge for this
challenge.',
                                                 reply_markup=reply_markup)
                    elif idioma == "Español":
                        reply_markup = InlineKeyboardMarkup(keyboard)
                        context.bot.send_message(chat_id1,
                                                 text='Has utilizado los siguientes jueces. Select it, or press NEW if you want a new judge
for this challenge.',
                                                 reply_markup=reply_markup)
                    return EXISTING
            elif nuevouser == 0:
                print("No tiene jueces anteriores")
                record = {'Code': codigoanterior}
                airtable3.insert(record)
                rellenarairtable3(chat_id1)
                if idioma == "English":
                    update.message.reply_text('Who is going to be the judge?, send me his/her name please ',
                                              reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                elif idioma == "Español":
                    update.message.reply_text('¿Quién va a ser el juez?, send me his/her name please ',
                                              reply_markup=ReplyKeyboardRemove(remove_keyboard=True))

                return JUDGENAME


def amount1(update, context):
    text = update.message.text
    chat_id1 = str(update.effective_chat.id)
    idioma = buscaridiomauser(chat_id1)
    change = int(buscarvariable(chat_id1, colChange))
    reto = buscarvariable(chat_id1, colChallenge)
    if text == "Insert Quantity" or text == "Insertar Cantidad":
        print("El usuario va a ingresar la cantidad")
        if idioma == "English":
            keyboard = [['5€', '10€', '15€', ], ['20€', '25€', 'Other']]
            update.message.reply_text(
                '\n\n'
                'Select the quantity that you wanna bet from the options below or send me the amount directly',
                reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
        elif idioma == "Español":
            keyboard = [['5€', '10€', '15€', ], ['20€', '25€', 'Other']]
            update.message.reply_text(
                '\n\n'
                'Selecciona la cantidad que quieres apostar de las opciones de abajo o envíame la cantidad directamente',
                reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
        return AMOUNT
    else:
        chat_id1 = str(update.effective_chat.id)
        try:
            text = int(text)
            text1 = str(text) + str("€")
            actualizarTABLATODO(chat_id1, colAmount, text1)
            if reto == "0":
                actualizarTABLACHALLENGES(chat_id1, colAmount, text1)
            elif reto == "1":
                actualizarTABLADEFIANCE(chat_id1, colAmount, text1)
            if change == 0:
                print("El usuario quiere apostar {}".format(text1))
                print("El usuario quiere apostar {}".format(text1))
                if idioma == "English":
                    reply_keyboard = [['Insert Date'], ['Return']]

                    update.message.reply_text(
                        '\n\n'
                        'When must the challenge be done?',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif idioma == "Español":
                    reply_keyboard = [['Insertar Fecha'], ['Volver Atrás']]

                    update.message.reply_text(
                        '\n\n'
                        '¿Cuándo debe de estar hecho el reto?',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

                return DUEDATE
            elif change == 1:
                print("Cantidad apostada cambiada")
                if reto == "0":
                    nombresretos = buscarchallenges(chat_id1, reto)
                    long = len(nombresretos)
                    cantidadretos = buscaramounts(chat_id1, reto)
                    fechasretos = buscarduedates(chat_id1, reto)
                    juecesretos = buscarjueces1(chat_id1)
                    juecesmails = buscarjuecesmail1(chat_id1, reto)
                    usermails = buscaruseremails(chat_id1)
                    if idioma == "English":
                        reply_keyboard = [['Change'], ['OK']]
                        context.bot.send_message(chat_id1, text='Thank you!')
                        context.bot.send_message(chat_id1,
                                                 text='You have created this challenge \n\n  *Name 🏷️ :* {}  \n *Amount 💸 :* {} \n *DueDate
📅 :* {} \n *Judge 🧑‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1],
                                                     fechasretos[long - 1], juecesretos[long - 1],
                                                     juecesmails[long - 1], usermails[0]),
                                                 parse_mode="Markdown",
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' If there is any mistake, press the change button. Otherwise press Ok button.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                    elif idioma == "Español":
                        reply_keyboard = [['Cambiar'], ['OK']]
                        context.bot.send_message(chat_id1, text='Gracias!!')
                        context.bot.send_message(chat_id1,
                                                 text='Has creado el siguiente reto: \n\n  *Name 🏷️ :* {}  \n *Amount 💸 :* {} \n *DueDate
📅 :* {} \n *Judge 🧑‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1],
                                                     fechasretos[long - 1], juecesretos[long - 1],
                                                     juecesmails[long - 1], usermails[0]),
                                                 parse_mode="Markdown",
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif reto == "1":
                    nombresretos = buscarchallenges(chat_id1, reto)
                    long = len(nombresretos)
                    cantidadretos = buscaramounts(chat_id1, reto)
                    fechasretos = buscarduedates(chat_id1, reto)
                    usermails = buscaruseremails(chat_id1)
                    if idioma == "English":
                        reply_keyboard = [['Change'], ['OK']]
                        context.bot.send_message(chat_id1, text='Thank you!')
                        context.bot.send_message(chat_id1,
                                                 text='You have created this challenge \n\n*Name 🏷️ :* {}  \n*Amount 💸 :* {} \n*DueDate 📅
:* {} \n*JudgeEmail 📩 :* {} '.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1],
                                                     fechasretos[long - 1], usermails[0]),
                                                 parse_mode="Markdown",
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' If there is any mistake, press the change button. Otherwise press Ok button.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                    elif idioma == "Español":
                        reply_keyboard = [['Cambiar'], ['OK']]
                        context.bot.send_message(chat_id1, text='Gracias!!')
                        context.bot.send_message(chat_id1,
                                                 text='Has creado el siguiente reto: \n\n*Name 🏷️ :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :*
{} \n*JudgeEmail 📩 :* {}'.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1],
                                                fechasretos[long - 1], usermails[0]),
                                            parse_mode="Markdown",
                                            reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                    update.message.reply_text(
                        '\n\n'
                        ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                return FINAL
        except ValueError:
            prueba2 = text.find("€")
            if prueba2 == -1:
                print("El usuario ha metido una cantidad invalida")
                update.message.reply_text('Sorry, I cannot storage text',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                if change == 0:
                    if idioma == "English":
                        if reto == "0":
                            reply_keyboard = [['Insert Quantity'], ['Return']]
                        elif reto == "1":
                            reply_keyboard = [['Insert Quantity']]

                        context.bot.send_message(chat_id1, text=
                        'How many money do you want to bet?',
                                            reply_markup=ReplyKeyboardMarkup(reply_keyboard,
                                                                             one_time_keyboard=True))
                    elif idioma == "Español":
                        if reto == "0":
                            reply_keyboard = [['Insert Quantity'], ['Volver Atrás']]
                        elif reto == "1":
                            reply_keyboard = [['Insertar Cantidad']]

                        context.bot.send_message(chat_id1, text=
                        '¿Cuánto dinero quieres apostar?',
                                            reply_markup=ReplyKeyboardMarkup(reply_keyboard,
                                                                             one_time_keyboard=True))
                    return ADDITIONAL
                elif change == 1:
                    if idioma == "English":
                        keyboard = [['5€', '10€', '15€', ], ['20€', '25€', 'Other']]
                        update.message.reply_text(
                            '\n\n'
                            'Select the quantity that you wanna bet from the options below or send me the amount directly',
                            reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
                    elif idioma == "Español":
                        keyboard = [['5€', '10€', '15€', ], ['20€', '25€', 'Other']]
                        update.message.reply_text(
                            '\n\n'
                            'Selecciona la cantidad que quieres apostar de las opciones de abajo o envíame la cantidad directamente',
                            reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
                    return AMOUNT
            elif prueba2 != -1:
                text1 = str(text)
                prueba = text1.find("€")
                if prueba == -1:
                    text1 = str(text) + str("€")
                elif prueba != -1:
                    text1 = str(text)
                if reto == "1":
                    actualizarTABLADEFIANCE(chat_id1, colAmount, text1)
                elif reto == "0":
                    actualizarTABLACHALLENGES(chat_id1, colAmount, text1)
                print("El usuario quiere apostar {}".format(text1))
                if change == 0:
                    if idioma == "English":
                        reply_keyboard = [['Insert Date'], ['Return']]

                        update.message.reply_text(
                            '\n\n'
                            'When must the challenge be done?',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                    elif idioma == "Español":
                        reply_keyboard = [['Insertar Fecha'], ['Volver Atrás']]

                        update.message.reply_text(
                            '\n\n'
                            '¿Cuándo debe de estar hecho el reto?',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

                    return DUEDATE
                elif change == 1:
                    print("El usuario ha cambiado la apuesta a {}".format(text1))
                    nombresretos = buscarchallenges(chat_id1, reto)
                    long = len(nombresretos)
                    cantidadretos = buscaramounts(chat_id1, reto)
                    fechasretos = buscarduedates(chat_id1, reto)
                    juecesretos = buscarjueces1(chat_id1)
                    juecesmails = buscarjuecesmail1(chat_id1, reto)
                    usermails = buscaruseremails(chat_id1)
                    if idioma == "English":
                        reply_keyboard = [['Change'], ['OK']]
                        context.bot.send_message(chat_id1, text='Thank you!')
                        context.bot.send_message(chat_id1,
                                                 text='You have created this challenge \n\n  *Name 🎟️ :* {}  \n *Amount 💸 :* {} \n *DueDate
    📅 :* {} \n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1],
                                                     fechasretos[long - 1], juecesretos[long - 1],
                                                     juecesmails[long - 1], usermails[0]),
                                                 parse_mode="Markdown",
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' If there is any mistake, press the change button. Otherwise press Ok button.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                    elif idioma == "Español":
                        reply_keyboard = [['Cambiar'], ['OK']]
                        context.bot.send_message(chat_id1, text='Gracias!!')
                        context.bot.send_message(chat_id1,
                                                 text='Has creado el siguiente reto: \n\n  *Name 🎟️ :* {}  \n *Amount 💸 :* {} \n *DueDate
    📅 :* {} \n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1],
                                                     fechasretos[long - 1], juecesretos[long - 1],
                                                     juecesmails[long - 1], usermails[0]),
                                                 parse_mode="Markdown",
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

                    return FINAL
            else:
                text1 = str(text)
                prueba = text1.find("€")
                if prueba == -1:
                    text1 = str(text) + str("€")
                elif prueba != -1:
                    text1 = str(text)
                if reto == "1":
                    actualizarTABLADEFIANCE(chat_id1, colAmount, text1)
                elif reto == "0":
                    actualizarTABLACHALLENGES(chat_id1, colAmount, text1)
                print("El usuario quiere apostar {}".format(text1))
                if change == 0:
                    if idioma == "English":
                        reply_keyboard = [['Insert Date'], ['Return']]

                        update.message.reply_text(
                            '\n\n'
                            'When must the challenge be done?',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                    elif idioma == "Español":
                        reply_keyboard = [['Insertar Fecha'], ['Volver Atrás']]

                        update.message.reply_text(
                            '\n\n'
                            '¿Cuándo debe de estar hecho el reto?',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

                    return DUEDATE
                elif change == 1:
                    print("El usuario ha cambiado la apuesta a {}".format(text1))
                    nombresretos = buscarchallenges(chat_id1, reto)
                    long = len(nombresretos)
                    cantidadretos = buscaramounts(chat_id1, reto)
                    fechasretos = buscarduedates(chat_id1, reto)
                    juecesretos = buscarjueces1(chat_id1)
                    juecesmails = buscarjuecesmail1(chat_id1, reto)
                    usermails = buscaruseremails(chat_id1)
                    if idioma == "English":
                        reply_keyboard = [['Change'], ['OK']]
                        context.bot.send_message(chat_id1, text='Thank you!')
                        context.bot.send_message(chat_id1,
                                                 text='You have created this challenge \n\n  *Name 🎟️ :* {}  \n *Amount 💸 :* {} \n *DueDate
    📅 :* {} \n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1],
                                                     fechasretos[long - 1], juecesretos[long - 1],
                                                     juecesmails[long - 1], usermails[0]),
                                                 parse_mode="Markdown",
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' If there is any mistake, press the change button. Otherwise press Ok button.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                    elif idioma == "Español":
                        reply_keyboard = [['Cambiar'], ['OK']]
                        context.bot.send_message(chat_id1, text='Gracias!!')
                        context.bot.send_message(chat_id1,
                                                 text='Has creado el siguiente reto: \n\n  *Name 🎫 :* {}  \n *Amount 💸 :* {} \n *DueDate
📅 :* {} \n *Judge 👨⚖:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1],
                                                     fechasretos[long - 1], juecesretos[long - 1],
                                                     juecesmails[long - 1], usermails[0]),
                                                 parse_mode="Markdown",
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

                    return FINAL


def amount(update, context):
    text = update.message.text
    chat_id1 = str(update.effective_chat.id)
    idioma = buscaridiomauser(chat_id1)
    change = int(buscarvariable(chat_id1, colChange))
    reto = buscarvariable(chat_id1, colChallenge)
    if change == 1:
        if (text == "Other"):
            if idioma == "English":
                update.message.reply_text('Please, send me the amount that you want to bet in €.',
                                          reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text('Envíame la cantidad que quieres apostar en €.',
                                          reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            return ADDITIONAL
        else:
            chat_id1 = str(update.effective_chat.id)
            if reto == "0":
                actualizarTABLACHALLENGES(chat_id1, colAmount, text)
            elif reto == "1":
                actualizarTABLADEFIANCE(chat_id1, colAmount, text)
            print("El usuario ha cambiado su apuesta a {}".format(text))
            if reto == "0":
                nombresretos = buscarchallenges(chat_id1)
                long = len(nombresretos)
                cantidadretos = buscaramounts(chat_id1)
                fechasretos = buscarduedates(chat_id1)
                juecesretos = buscarjueces1(chat_id1)
                juecesmails = buscarjuecesmail1(chat_id1)
                usermails = buscaruseremails(chat_id1)
                if idioma == "English":
                    reply_keyboard = [['Change'], ['OK']]
                    context.bot.send_message(chat_id1, text='Thank you!')
                    context.bot.send_message(chat_id1,
                                             text='You have created this challenge \n\n  *Name 🎫 :* {}  \n *Amount 💸 :* {} \n *DueDate 📅
:* {} \n *Judge 👨⚖:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                 nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                                 juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                             parse_mode="Markdown",
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                    update.message.reply_text(
                        '\n\n'
                        ' If there is any mistake, press the change button. Otherwise press Ok button.',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif idioma == "Español":
                    reply_keyboard = [['Cambiar'], ['OK']]
                    context.bot.send_message(chat_id1, text='Gracias!!')
                    context.bot.send_message(chat_id1,
                                             text='Has creado el siguiente reto: \n\n  *Name 🎫 :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :*
{} \n *Judge 👨⚖:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                 nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                                 juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                             parse_mode="Markdown",
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                    update.message.reply_text(
                        '\n\n'
                        ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                return FINAL
            elif reto == "1":
                nombresretos = buscarchallenges(chat_id1, reto)
                long = len(nombresretos)
                cantidadretos = buscaramounts(chat_id1, reto)
                fechasretos = buscarduedates(chat_id1, reto)
                usermails = buscaruseremails(chat_id1)
                if idioma == "English":
                    reply_keyboard = [['Change'], ['OK']]
                    context.bot.send_message(chat_id1, text='Thank you!')
                    context.bot.send_message(chat_id1,
                                             text='You have created this challenge \n\n*Name 🎫 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :*
{} \n*JudgeEmail 📩 :* {} '.format(
                                                 nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                                 usermails[0]),
                                             parse_mode="Markdown",
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                    update.message.reply_text(
                        '\n\n'
                        ' If there is any mistake, press the change button. Otherwise press Ok button.',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif idioma == "Español":
                    reply_keyboard = [['Cambiar'], ['OK']]
                    context.bot.send_message(chat_id1, text='Gracias!!')
                    context.bot.send_message(chat_id1,
                                             text='Has creado el siguiente reto: \n\n*Name 🎫 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :* {}
\n*JudgeEmail 📩 :* {}'.format(
                                                 nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                                 usermails[0]),
                                             parse_mode="Markdown",
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                    update.message.reply_text(
                        '\n\n'
                        ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            return FINAL
    else:
        if (text == "Other"):
            if idioma == "English":
                update.message.reply_text('Please, send me the amount that you want to bet in €.',
                                          reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text('Envíame la cantidad que quieres apostar en €.',
                                          reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            return ADDITIONAL
        elif (text != "Other"):
            chat_id1 = str(update.effective_chat.id)
            if reto == "0":
                actualizarTABLACHALLENGES(chat_id1, colAmount, text)
            elif reto == "1":
                actualizarTABLADEFIANCE(chat_id1, colAmount, text)
            if (change == 0):
                if idioma == "English":
                    reply_keyboard = [['Insert Date'], ['Return']]

                    update.message.reply_text(
                        '\n\n'
                        'When must the challenge be done?',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif idioma == "Español":
                    reply_keyboard = [['Insertar Fecha'], ['Volver Atrás']]

                    update.message.reply_text(
                        '\n\n'
                        '¿Cuándo debe de estar hecho el reto?',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

                return DUEDATE
            else:
                chat_id1 = str(update.effective_chat.id)
                print("El usuario ha cambiado su apuesta a {}".format(text))
                nombresretos = buscarchallenges(chat_id1, reto)
                long = len(nombresretos)
                cantidadretos = buscaramounts(chat_id1, reto)
                fechasretos = buscarduedates(chat_id1, reto)
                juecesretos = buscarjueces1(chat_id1)
                juecesmails = buscarjuecesmail1(chat_id1, reto)
                usermails = buscaruseremails(chat_id1)
                if idioma == "English":
                    reply_keyboard = [['Change'], ['OK']]
                    context.bot.send_message(chat_id1, text='Thank you!')
                    context.bot.send_message(chat_id1,
                                             text='You have created this challenge \n\n  *Name 🎫 :* {}  \n *Amount 💸 :* {} \n *DueDate 📅
:* {} \n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                 nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                                 juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                             parse_mode="Markdown",
```

```python
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                    update.message.reply_text(
                        '\n\n'
                        ' If there is any mistake, press the change button. Otherwise press Ok button.',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif idioma == "Español":
                    reply_keyboard = [['Cambiar'], ['OK']]
                    context.bot.send_message(chat_id1, text='Gracias!!')
                    context.bot.send_message(chat_id1,
                                             text='Has creado el siguiente reto: \n\n  *Name 🏷 :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :*
{} \n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                 nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                                 juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                             parse_mode="Markdown",
                                             reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                    update.message.reply_text(
                        '\n\n'
                        ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

                return FINAL


def additional(update, context):
    chat_id1 = str(update.effective_chat.id)
    text = update.message.text
    idioma = buscaridiomauser(chat_id1)
    reto = buscarvariable(chat_id1, colChallenge)
    try:
        text = int(text)
    except ValueError:
        prueba2 = text.find("€")
        if prueba2 == -1:
            context.bot.send_message(chat_id1, text='Sorry, I cannot storage text',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            update.message.reply_text('Please, send me again the quantity as a number instead of a text',
                                      reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            return ADDITIONAL
        elif prueba2 != -1:
            text1 = str(text)
            prueba = text1.find("€")
            if prueba == -1:
                text1 = str(text) + str("€")
            elif prueba != -1:
                text1 = str(text)
            actualizarTABLATODO(chat_id1, colAmount, text1)
            if reto == "0":
                actualizarTABLACHALLENGES(chat_id1, colAmount, text1)
            elif reto == "1":
                actualizarTABLADEFIANCE(chat_id1, colAmount, text1)
            if change == 0:
                print("El usuario quiere apostar {}".format(text1))
                if idioma == "English":
                    reply_keyboard = [['Insert Date'], ['Return']]

                    update.message.reply_text(
                        '\n\n'
                        'When must the challenge be done?',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif idioma == "Español":
                    reply_keyboard = [['Insertar Fecha'], ['Volver Atrás']]

                    update.message.reply_text(
                        '\n\n'
                        '¿Cuándo debe de estar hecho el reto?',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

                return DUEDATE
            elif change == 1:
                print("El usuario ha cambiado su apuesta a {}".format(text))
                if reto == "0":
                    nombresretos = buscarchallenges(chat_id1, reto)
                    long = len(nombresretos)
                    cantidadretos = buscaramounts(chat_id1, reto)
                    fechasretos = buscarduedates(chat_id1, reto)
                    juecesretos = buscarjueces1(chat_id1)
                    juecesmails = buscarjuecesmail1(chat_id1, reto)
                    usermails = buscaruseremails(chat_id1)
                    if idioma == "English":
                        reply_keyboard = [['Change'], ['OK']]
                        context.bot.send_message(chat_id1, text='Thank you!')
                        context.bot.send_message(chat_id1,
                                                 text='You have created this challenge \n\n  *Name 🏷 :* {}  \n *Amount 💸 :* {} \n *DueDate
📅 :* {} \n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                    nombresretos[long - 1], cantidadretos[long - 1],
                                                    fechasretos[long - 1], juecesretos[long - 1],
                                                    juecesmails[long - 1], usermails[0]),
                                                parse_mode="Markdown",
                                                reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' If there is any mistake, press the change button. Otherwise press Ok button.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                    elif idioma == "Español":
                        reply_keyboard = [['Cambiar'], ['OK']]
                        context.bot.send_message(chat_id1, text='Gracias!!')
                        context.bot.send_message(chat_id1,
                                                text='Has creado el siguiente reto: \n\n  *Name 🎟 :* {}  \n *Amount 💸 :* {} \n *DueDate
📅 :* {} \n *Judge 👨⚖:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                    nombresretos[long - 1], cantidadretos[long - 1],
                                                    fechasretos[long - 1], juecesretos[long - 1],
                                                    juecesmails[long - 1], usermails[0]),
                                                parse_mode="Markdown",
                                                reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif reto == "1":
                    nombresretos = buscarchallenges(chat_id1, reto)
                    long = len(nombresretos)
                    cantidadretos = buscaramounts(chat_id1, reto)
                    fechasretos = buscarduedates(chat_id1, reto)
                    juecesmails = buscarjuecesmail1(chat_id1, reto)
                    if idioma == "English":
                        reply_keyboard = [['Change'], ['OK']]
                        context.bot.send_message(chat_id1, text='Thank you!')
                        context.bot.send_message(chat_id1,
                                                text='You have created this challenge \n\n  *Name 🎟 :* {}  \n *Amount 💸 :* {} \n *DueDate
📅 :* {} \n *Judge 👨⚖:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                    nombresretos[long - 1], cantidadretos[long - 1],
                                                    fechasretos[long - 1], juecesmails[long - 1]),
                                                parse_mode="Markdown",
                                                reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' If there is any mistake, press the change button. Otherwise press Ok button.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                    elif idioma == "Español":
                        reply_keyboard = [['Cambiar'], ['OK']]
                        context.bot.send_message(chat_id1, text='Gracias!!')
                        context.bot.send_message(chat_id1,
                                                text='Has creado el siguiente reto: \n\n  *Name 🎟 :* {}  \n *Amount 💸 :* {} \n *DueDate
📅 :* {} \n *Judge 👨⚖:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                    nombresretos[long - 1], cantidadretos[long - 1],
                                                    fechasretos[long - 1], juecesmails[long - 1]),
                                                parse_mode="Markdown",
                                                reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                return FINAL

        else:
            text1 = str(text)
            prueba = text1.find("€")
            if prueba == -1:
                text1 = str(text) + str("€")
            elif prueba != -1:
                text1 = str(text)
            actualizarTABLATODO(chat_id1, colAmount, text1)
            if reto == "0":
                actualizarTABLACHALLENGES(chat_id1, colAmount, text1)
            elif reto == "1":
                actualizarTABLADEFIANCE(chat_id1, colAmount, text1)
            if change == 0:
                print("El usuario quiere apostar {}".format(text1))
                if idioma == "English":
                    reply_keyboard = [['Insert Date'], ['Return']]

                    update.message.reply_text(
                        '\n\n'
                        'When must the challenge be done?',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif idioma == "Español":
                    reply_keyboard = [['Insertar Fecha'], ['Volver Atrás']]
                    update.message.reply_text(
                        '\n\n'
                        '¿Cuándo debe de estar hecho el reto?',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

                return DUEDATE
            elif change == 1:
                print("El usuario ha cambiado su apuesta a {}".format(text))
                if reto == "0":
                    nombresretos = buscarchallenges(chat_id1, reto)
                    long = len(nombresretos)
                    cantidadretos = buscaramounts(chat_id1, reto)
                    fechasretos = buscarduedates(chat_id1, reto)
                    juecesretos = buscarjueces1(chat_id1)
                    juecesmails = buscarjuecesmail1(chat_id1, reto)
                    usermails = buscaruseremails(chat_id1)
                    if idioma == "English":
                        reply_keyboard = [['Change'], ['OK']]
                        context.bot.send_message(chat_id1, text='Thank you!')
                        context.bot.send_message(chat_id1,
                                                 text='You have created this challenge \n\n  *Name 🏷️ :* {}  \n *Amount 💸 :* {} \n *DueDate 📅
    :* {} \n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                                     juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                                 parse_mode="Markdown",
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' If there is any mistake, press the change button. Otherwise press Ok button.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                    elif idioma == "Español":
                        reply_keyboard = [['Cambiar'], ['OK']]
                        context.bot.send_message(chat_id1, text='Gracias!!')
                        context.bot.send_message(chat_id1,
                                                 text='Has creado el siguiente reto: \n\n  *Name 🏷️ :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :*
    {} \n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                                     juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                                 parse_mode="Markdown",
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif reto == "1":
                    nombresretos = buscarchallenges(chat_id1, reto)
                    long = len(nombresretos)
                    cantidadretos = buscaramounts(chat_id1, reto)
                    fechasretos = buscarduedates(chat_id1, reto)
                    juecesmails = buscarjuecesmail1(chat_id1, reto)
                    if idioma == "English":
                        reply_keyboard = [['Change'], ['OK']]
                        context.bot.send_message(chat_id1, text='Thank you!')
                        context.bot.send_message(chat_id1,
                                                 text='You have created this challenge \n\n  *Name 🏷️ :* {}  \n *Amount 💸 :* {} \n *DueDate 📅
    :* {} \n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                                     juecesmails[long - 1]),
                                                 parse_mode="Markdown",
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' If there is any mistake, press the change button. Otherwise press Ok button.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                    elif idioma == "Español":
                        reply_keyboard = [['Cambiar'], ['OK']]
                        context.bot.send_message(chat_id1, text='Gracias!!')
                        context.bot.send_message(chat_id1,
                                                 text='Has creado el siguiente reto: \n\n  *Name 🏷️ :* {}  \n *Amount 💸 :* {} \n *DueDate 📅 :*
    {} \n *Judge 👨‍⚖️:* {} \n  *JudgeEmail 📩 :* {}  \n *UserEmail 👤 :* {}  \n  Has been created'.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                                     juecesmails[long - 1]),
                                                 parse_mode="Markdown",
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                return FINAL


    def meterfecha(update, context):
        chat_id1 = str(update.effective_chat.id)
        idioma = buscaridiomauser(chat_id1)
        print("El usuario quiere VOLVER ha elegir la cantidad porque se ha equivocado")
        if idioma == "English":
            keyboard = [['5€', '10€', '15€', ], ['20€', '25€', 'Other']]
            update.message.reply_text(
                '\n\n'
                'Select the quantity that you wanna bet from the options below or send me the amount directly',
                reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
        elif idioma == "Español":
            keyboard = [['5€', '10€', '15€', ], ['20€', '25€', 'Other']]
            update.message.reply_text(
                '\n\n'
                'Selecciona la cantidad que quieres apostar de las opciones de abajo o envíame la cantidad directamente',
                reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
        return AMOUNT


    def date(update, context):
        text = update.message.text
        chat_id1 = str(update.effective_chat.id)
        idioma = buscaridiomauser(chat_id1)
        change = buscarvariable(chat_id1, colChange)
        if text == "Insert Date" or text == "Insertar Fecha" or (text == "/Duedate" and (change) == 1):
            if idioma == "English":
                keyboard = [['January', 'February', 'March', 'April'], ['May', 'June', 'July', 'August'],
                            ['September', 'October', 'November', 'December']]
                update.message.reply_text(
                    '\n\n'
                    'Select the month',
                    reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                keyboard = [['Enero', 'Febrero', 'Marzo', 'Abril'], ['Mayo', 'Junio', 'Julio', 'Agosto'],
                            ['Septiembre', 'Octubre', 'Noviembre', 'Diciembre']]
                update.message.reply_text(
                    '\n\n'
                    'Seleccione el mes',
                    reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            return DATE
        else:
            if idioma == "English":
                reply_keyboard = [['Insert Date'], ['Return']]

                update.message.reply_text(
                    '\n\n'
                    'When must the challenge be done?',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                reply_keyboard = [['Insertar Fecha'], ['Volver Atrás']]

                update.message.reply_text(
                    '\n\n'
                    '¿Cuándo debe de estar hecho el reto?',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

            return DUEDATE


    def month(update, context):
        text = update.message.text
        month = text
        chat_id1 = str(update.effective_chat.id)
        idioma = buscaridiomauser(chat_id1)
        actualizarTABLAVARIABLES(chat_id1, colMonth, text)
        vals31 = ["January", "March", "May", "July", "August", "October", "December", "Enero", "Marzo", "Mayo", "Julio",
                  "Agosto", "Octubre", "Diciembre"]
        vals30 = ["April", "June", "September", "November", "Abril", "Junio", "Septiembre", "Noviembre"]
        vals28 = ["February", "Febrero"]

        if month in vals31:
            reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                              ['15', '16', '17', '18', '19', '20', '21'],
                              ['22', '23', '24', '25', '26', '27', '28'], ['29', '30', '31', '_', '_', '_', '_']]
            if idioma == "English":
                update.message.reply_text(
                    '\n\n'
                    'Select the day',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text(
                    '\n\n'
                    'Selecciona el día',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            return DATE1
        elif month in vals30:
            reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                              ['15', '16', '17', '18', '19', '20', '21'],
                              ['22', '23', '24', '25', '26', '27', '28'], ['29', '30', '_', '_', '_', '_', '_']]

            if idioma == "English":
                update.message.reply_text(
                    '\n\n'
                    'Select the day',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text(
                    '\n\n'
                    'Selecciona el día',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            return DATE1
        elif month in vals28:
            reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                              ['15', '16', '17', '18', '19', '20', '21'],
                              ['22', '23', '24', '25', '26', '27', '28']]

            if idioma == "English":
                update.message.reply_text(
                    '\n\n'
                    'Select the day',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text(
                    '\n\n'
                    'Selecciona el día',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            return DATE1


    def month1(update, context):
        text = update.message.text
        chat_id1 = str(update.effective_chat.id)
        idioma = buscaridiomauser(chat_id1)
        vals31 = ["January", "March", "May", "July", "August", "October", "December", "january", "march", "may", "july",
                  "august", "october", "december", "Enero", "Marzo", "Mayo", "Julio", "Agosto", "Octubre", "Diciembre"]
        vals30 = ["April", "June", "September", "November", "april", "june", "september", "november", "Abril", "Junio",
                  "Septiembre", "Noviembre"]
        vals28 = ["February", "february", "Febrero"]

        if text in vals31:
            month = text
            actualizarTABLAVARIABLES(chat_id1, colMonth, month)
            reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                              ['15', '16', '17', '18', '19', '20', '21'],
                              ['22', '23', '24', '25', '26', '27', '28'], ['29', '30', '31', '_', '_', '_', '_']]

            if idioma == "English":
                update.message.reply_text(
                    '\n\n'
                    'Select the day',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text(
                    '\n\n'
                    'Selecciona el día',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            return DATE1
        elif text in vals30:
            month = text
            actualizarTABLAVARIABLES(chat_id1, colMonth, month)
            reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                              ['15', '16', '17', '18', '19', '20', '21'],
                              ['22', '23', '24', '25', '26', '27', '28'], ['29', '30', '_', '_', '_', '_', '_']]

            if idioma == "English":
                update.message.reply_text(
                    '\n\n'
                    'Select the day',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text(
                    '\n\n'
                    'Selecciona el día',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            return DATE1
        elif text in vals28:
            month = text
            actualizarTABLAVARIABLES(chat_id1, colMonth, month)
            reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                              ['15', '16', '17', '18', '19', '20', '21'],
                              ['22', '23', '24', '25', '26', '27', '28']]

            if idioma == "English":
                update.message.reply_text(
                    '\n\n'
                    'Select the day',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text(
                    '\n\n'
                    'Selecciona el día',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            return DATE1
        else:
            if idioma == "English":
                keyboard = [['January', 'February', 'March', 'April'], ['May', 'June', 'July', 'August'],
                            ['September', 'October', 'November', 'December']]
                update.message.reply_text(
                    '\n\n'
                    'Select the month',
                    reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                keyboard = [['Enero', 'Febrero', 'Marzo', 'Abril'], ['Mayo', 'Junio', 'Julio', 'Agosto'],
                            ['Septiembre', 'Octubre', 'Noviembre', 'Diciembre']]
                update.message.reply_text(
                    '\n\n'
                    'Seleccione el mes',
                    reply_markup=ReplyKeyboardMarkup(keyboard, one_time_keyboard=True))
            return DATE


    def day(update, context):
        chat_id1 = str(update.effective_chat.id)
        idioma = buscaridiomauser(chat_id1)
        month = buscarvariable(chat_id1, colMonth)
        vals28 = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19',
                  '20', '21', '22', '23', '24', '25', '26', '27', '28']
        vals30 = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19',
                  '20', '21', '22', '23', '24', '25', '26', '27', '28', '29', '30']
        vals31 = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19',
                  '20', '21', '22', '23', '24', '25', '26', '27', '28', '29', '30', '31']
        text = update.message.text
        actualizarTABLAVARIABLES(chat_id1, colDay, text)
        if month == "January" or month == "March" or month == "May" or month == "July" or month == "August" or month == "October" or month ==
    "December" or month == "january" \
            or month == "march" or month == "may" or month == "july" or month == "august" or month == "october" or month == "december" or month
    == "Enero" or month == "Marzo" \
            or month == "Mayo" or month == "Julio" or month == "Agosto" or month == "Octubre" or month == "Diciembre":
            a = 1
        else:
            if month == "April" or month == "June" or month == "September" or month == "November" or month == "april" or month == "june" or month
    == "september" \
                or month == "november" or month == "Abril" or month == "Junio" or month == "Septiembre" or month == "Noviembre":
                a = 2
            else:
                if month == "February" or month == "february" or month == "Febrero":
                    a = 3
        if (text in vals31) and (text in vals30) and (text in vals28):
            b = 1
        else:
            if (text in vals31) and (text in vals30) and ((text in vals28) == False):
                b = 2
            else:
                if (text in vals31) and ((text in vals30) == False):
                    b = 3
                else:
                    b = 4
        if b == 2 and a == 3:
            reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                              ['15', '16', '17', '18', '19', '20', '21'],
                              ['22', '23', '24', '25', '26', '27', '28']]

            if idioma == "English":
                update.message.reply_text(
                    '\n\n'
                    'Select the day',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text(
                    '\n\n'
                    'Selecciona el día',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            return DATE1
        elif b == 3 and a == 2:
            reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                              ['15', '16', '17', '18', '19', '20', '21'],
                              ['22', '23', '24', '25', '26', '27', '28'], ['29', '30', '_', '_', '_', '_', '_']]
            if idioma == "English":
                update.message.reply_text(
                    '\n\n'
                    'Select the day',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text(
                    '\n\n'
                    'Selecciona el día',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            return DATE1
        elif b == 4:
            if a == 1:
                reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                                  ['15', '16', '17', '18', '19', '20', '21'],
                                  ['22', '23', '24', '25', '26', '27', '28'], ['29', '30', '31', '_', '_', '_', '_']]

                if idioma == "English":
                    update.message.reply_text(
                        '\n\n'
                        'Select the day',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif idioma == "Español":
                    update.message.reply_text(
                        '\n\n'
                        'Selecciona el día',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                return DATE1
            elif a == 2:
                reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                                  ['15', '16', '17', '18', '19', '20', '21'],
                                  ['22', '23', '24', '25', '26', '27', '28'], ['29', '30', '_', '_', '_', '_', '_']]

                if idioma == "English":
                    update.message.reply_text(
                        '\n\n'
                        'Select the day',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif idioma == "Español":
                    update.message.reply_text(
                        '\n\n'
                        'Selecciona el día',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                return DATE1
            elif a == 3:
                reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                                  ['15', '16', '17', '18', '19', '20', '21'],
                                  ['22', '23', '24', '25', '26', '27', '28']]

                if idioma == "English":
                    update.message.reply_text(
                        '\n\n'
                        'Select the day',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif idioma == "Español":
                    update.message.reply_text(
                        '\n\n'
                        'Selecciona el día',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                return DATE1
        else:
            reply_keyboard = [['2020'], ['2021'], ['2022'], ['2023']]

            if idioma == "English":
                update.message.reply_text(
                    '\n\n'
                    'Select the year',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text(
                    '\n\n'
                    'Selecciona el año',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            return DUEDATE1


    def day1(update, context):
        chat_id1 = str(update.effective_chat.id)
        idioma = buscaridiomauser(chat_id1)
        month = buscarvariable(chat_id1, colMonth)
        vals28 = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19',
                  '20', '21', '22', '23', '24', '25', '26', '27', '28']
        vals30 = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19',
                  '20', '21', '22', '23', '24', '25', '26', '27', '28', '29', '30']
        vals31 = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19',
                  '20', '21', '22', '23', '24', '25', '26', '27', '28', '29', '30', '31']
        text = update.message.text
        actualizarTABLAVARIABLES(chat_id1, colDay, text)
        if month == "January" or month == "March" or month == "May" or month == "July" or month == "August" or month == "October" or month == 
    "December" or month == "january" \
            or month == "march" or month == "may" or month == "july" or month == "august" or month == "october" or month == "december" or month 
    == "Enero" or month == "Marzo" \
            or month == "Mayo" or month == "Julio" or month == "Agosto" or month == "Octubre" or month == "Diciembre":
            a = 1
        else:
            if month == "April" or month == "June" or month == "September" or month == "November" or month == "april" or month == "june" or month 
    == "september" \
                or month == "november" or month == "Abril" or month == "Junio" or month == "Septiembre" or month == "Noviembre":
                a = 2
            else:
                if month == "February" or month == "february" or month == "Febrero":
                    a = 3
        if (text in vals31) and (text in vals30) and (text in vals28):
            b = 1
        else:
            if (text in vals31) and (text in vals30) and ((text in vals28) == False):
                b = 2
            else:
                if (text in vals31) and ((text in vals30) == False) and ((text in vals28) == False):
                    b = 3
                else:
                    b = 4
        if (b == 2 or b == 3) and a == 3:
            reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                              ['15', '16', '17', '18', '19', '20', '21'],
                              ['22', '23', '24', '25', '26', '27', '28']]

            if idioma == "English":
                update.message.reply_text(
                    '\n\n'
                    'Select the day',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text(
                    '\n\n'
                    'Selecciona el día',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            return DATE1
        elif b == 3 and a == 2:
            reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                              ['15', '16', '17', '18', '19', '20', '21'],
                              ['22', '23', '24', '25', '26', '27', '28'], ['29', '30', '_', '_', '_', '_', '_']]

            if idioma == "English":
                update.message.reply_text(
                    '\n\n'
                    'Select the day',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text(
                    '\n\n'
                    'Selecciona el día',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            return DATE1
        elif b == 4:
            if a == 1:
                reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                                  ['15', '16', '17', '18', '19', '20', '21'],
                                  ['22', '23', '24', '25', '26', '27', '28'], ['29', '30', '31', '_', '_', '_', '_']]

                if idioma == "English":
                    update.message.reply_text(
                        '\n\n'
                        'Select the day',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif idioma == "Español":
                    update.message.reply_text(
                        '\n\n'
                        'Selecciona el día',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                return DATE1
            elif a == 2:
                reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                                  ['15', '16', '17', '18', '19', '20', '21'],
                                  ['22', '23', '24', '25', '26', '27', '28'], ['29', '30', '_', '_', '_', '_', '_']]

                if idioma == "English":
                    update.message.reply_text(
                        '\n\n'
                        'Select the day',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif idioma == "Español":
                    update.message.reply_text(
                        '\n\n'
                        'Selecciona el día',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                return DATE1
            elif a == 3:
                reply_keyboard = [['1', '2', '3', '4', '5', '6', '7'], ['8', '9', '10', '11', '12', '13', '14'],
                                  ['15', '16', '17', '18', '19', '20', '21'],
                                  ['22', '23', '24', '25', '26', '27', '28']]

                if idioma == "English":
                    update.message.reply_text(
                        '\n\n'
                        'Select the day',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif idioma == "Español":
                    update.message.reply_text(
                        '\n\n'
                        'Selecciona el día',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                return DATE1
        else:
            reply_keyboard = [['2020'], ['2021'], ['2022'], ['2023']]

            if idioma == "English":
                update.message.reply_text(
                    '\n\n'
                    'Select the year',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                update.message.reply_text(
                    '\n\n'
                    'Selecciona el año',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            return DUEDATE1


    def year(update, context):
        text = update.message.text
        chat_id1 = str(update.effective_chat.id)
        idioma = buscaridiomauser(chat_id1)
        change = buscarvariable(chat_id1, colChange)
        month = buscarvariable(chat_id1, colMonth)
        day = buscarvariable(chat_id1, colDay)
        reto = buscarvariable(chat_id1, colChallenge)
        if text == "/UserEmail" and (change) == 1:
            if idioma == "English":
                update.message.reply_text(
                    'Could you provide me your email? You will receive an email confirming the challenge and future updates.',
                    reply_markup=ReplyKeyboardRemove(
                        remove_keyboard=True))
                return USERMAIL
            elif idioma == "Español":
                update.message.reply_text(
                    'Por favor, enviame tu email. Recibirás un correo confirmando que has creado el reto y futuras actualizaciones de este.',
                    reply_markup=ReplyKeyboardRemove(
                        remove_keyboard=True))
                return USERMAIL
        else:
            if month == "January" or month == "january" or month == "Enero":
                month = '01'
            elif month == "February" or month == "february" or month == "Febrero":
                month = '02'
            elif month == "March" or month == "march" or month == "Marzo":
                month = '03'
            elif month == "April" or month == "april" or month == "Abril":
                month = '04'
            elif month == "May" or month == "may" or month == "Mayo":
                month = '05'
            elif month == "June" or month == "june" or month == "Junio":
                month = '06'
            elif month == "July" or month == "july" or month == "Julio":
                month = '07'
            elif month == "August" or month == "august" or month == "Agosto":
                month = '08'
            elif month == "September" or month == "september" or month == "Septiembre":
                month = '09'
            elif month == "October" or month == "october" or month == "Octubre":
                month = '10'
            elif month == "November" or month == "november" or month == "Noviembre":
                month = '11'
            elif month == "December" or month == "december" or month == "Diciembre":
                month = '12'
            today = datetime.today()

            date = day + "/" + month + "/" + text
            date1 = datetime.strptime(date, "%d/%m/%Y")

            if date1 < today:
                print("El usuario ha elegido una fecha pasada")
                if idioma == "English":
                    reply_keyboard = [['Insert Date'], ['Return']]

                    update.message.reply_text(
                        '\n\n'
                        'Sorry, the date selected has passed. Please select a future date',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                elif idioma == "Español":
                    reply_keyboard = [['Insertar Fecha'], ['Volver Atrás']]

                    update.message.reply_text(
                        '\n\n'
                        'Lo siento, la fecha seleccionada ya ha pasado. Por favor, seleccione una fecha futura.',
                        reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

                return DUEDATE
            else:
                print("El usuario ha elegido la siguiente fecha {}".format(date))
                chat_id1 = str(update.effective_chat.id)
                actualizarTABLATODO(chat_id1, colDueDate, date)
                existe = buscaruseremails(chat_id1)
                print(existe)
                if reto == "0":
                    actualizarTABLACHALLENGES(chat_id1, colDueDate, date)
                elif reto == "1":
                    actualizarTABLADEFIANCE(chat_id1, colDueDate, date)
                if existe[0] == "NA":
                    if idioma == "English":
                        update.message.reply_text(
                            'Could you provide me your email? You will receive an email confirming the challenge and future updates.',
                            reply_markup=ReplyKeyboardRemove(
                                remove_keyboard=True))
                        return USERMAIL
                    elif idioma == "Español":
                        update.message.reply_text(
                            'Por favor, enviame tu email. Recibirás un correo confirmando que has creado el reto y futuras actualizaciones de
    este.',
                            reply_markup=ReplyKeyboardRemove(
                                remove_keyboard=True))
                        return USERMAIL
                else:
                    actualizarTABLADEFIANCE(chat_id1, colEmailJudge, existe[0])
                    if reto == "1":
                        nombresretos = buscarchallenges(chat_id1, reto)
                        long = len(nombresretos)
                        cantidadretos = buscaramounts(chat_id1, reto)
                        fechasretos = buscarduedates(chat_id1, reto)
                        juecesmails = buscarjuecesmail1(chat_id1, reto)
                        print(
                            "El usuario ha creado el siguiente reto Name: {} Amount: {} DueDate:{} JudgeEmail: {} ".format(
                                nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                juecesmails[long - 1]))
                        if idioma == "English":
                            reply_keyboard = [['Change'], ['OK']]
                            context.bot.send_message(chat_id1, text='Thank you!')
                            context.bot.send_message(chat_id1,
                                                     text='You have created this challenge \n\n*Name 🏷 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅
    :* {} \n*JudgeEmail 📩 :* {} \n\nHas been created'.format(
                                                         nombresretos[long - 1], cantidadretos[long - 1],
                                                         fechasretos[long - 1],
                                                         juecesmails[long - 1]),
                                                     parse_mode="Markdown",
                                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                            update.message.reply_text(
                                '\n\n'
                                ' If there is any mistake, press the change button. Otherwise press Ok button.',
                                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                        elif idioma == "Español":
                            reply_keyboard = [['Cambiar'], ['OK']]
                            context.bot.send_message(chat_id1, text='Gracias!!')
                            context.bot.send_message(chat_id1,
                                                     text='Has creado el siguiente reto: \n\n*Name 🏷 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :*
    {} \n*JudgeEmail 📩 :* {} \n\nHas been created'.format(
                                                         nombresretos[long - 1], cantidadretos[long - 1],
                                                         fechasretos[long - 1],
                                                         juecesmails[long - 1]),
                                                    parse_mode="Markdown",
                                                    reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                            update.message.reply_text(
                                '\n\n'
                                ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

                        return FINAL
                    else:
                        nombresretos = buscarchallenges(chat_id1, reto)
                        long = len(nombresretos)
                        cantidadretos = buscaramounts(chat_id1, reto)
                        fechasretos = buscarduedates(chat_id1, reto)
                        juecesretos = buscarjueces1(chat_id1)
                        juecesmails = buscarjuecesmail1(chat_id1, reto)
                        usermails = buscaruseremails(chat_id1)
                        print(
                            "El usuario ha creado el siguiente reto Name: {} Amount: {} DueDate:{} Judge: {} JudgeEmail: {} UserEmail : {}
".format(
                                nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                juecesretos[long - 1], juecesmails[long - 1], usermails[0]))
                        if idioma == "English":
                            reply_keyboard = [['Change'], ['OK']]
                            context.bot.send_message(chat_id1, text='Thank you!')
                            context.bot.send_message(chat_id1,
                                                    text='You have created this challenge \n\n*Name 🎟 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅
:* {} \n*Judge 👨⚖:* {} \n*JudgeEmail 📩 :* {}  \n*UserEmail 👤 :* {}  \n\nHas been created'.format(
                                                        nombresretos[long - 1], cantidadretos[long - 1],
                                                        fechasretos[long - 1], juecesretos[long - 1],
                                                        juecesmails[long - 1], usermails[0]),
                                                    parse_mode="Markdown",
                                                    reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                            update.message.reply_text(
                                '\n\n'
                                ' If there is any mistake, press the change button. Otherwise press Ok button.',
                                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                        elif idioma == "Español":
                            reply_keyboard = [['Cambiar'], ['OK']]
                            context.bot.send_message(chat_id1, text='Gracias!!')
                            context.bot.send_message(chat_id1,
                                                    text='Has creado el siguiente reto: \n\n*Name 🎟 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :*
{} \n*Judge 👨⚖:* {} \n*JudgeEmail 📩 :* {}  \n*UserEmail 👤 :* {}  \n\nHas been created'.format(
                                                        nombresretos[long - 1], cantidadretos[long - 1],
                                                        fechasretos[long - 1], juecesretos[long - 1],
                                                        juecesmails[long - 1], usermails[0]),
                                                    parse_mode="Markdown",
                                                    reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                            update.message.reply_text(
                                '\n\n'
                                ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

                        return FINAL


def year1(update, context):
    text = update.message.text
    chat_id1 = str(update.effective_chat.id)
    idioma = buscaridiomauser(chat_id1)
    month = buscarvariable(chat_id1, colMonth)
    reto = buscarvariable(chat_id1, colChallenge)
    if text == "2020" or text == "2021" or text == "2022" or text == "2023":
        if month == "January" or month == "january" or month == "Enero":
            month = '01'
        elif month == "February" or month == "february" or month == "Febrero":
            month = '02'
        elif month == "March" or month == "march" or month == "Marzo":
            month = '03'
        elif month == "April" or month == "april" or month == "Abril":
            month = '04'
        elif month == "May" or month == "may" or month == "Mayo":
            month = '05'
        elif month == "June" or month == "june" or month == "Junio":
            month = '06'
        elif month == "July" or month == "july" or month == "Julio":
            month = '07'
        elif month == "August" or month == "august" or month == "Agosto":
            month = '08'
        elif month == "September" or month == "september" or month == "Septiembre":
            month = '09'
        elif month == "October" or month == "october" or month == "Octubre":
            month = '10'
        elif month == "November" or month == "november" or month == "Noviembre":
            month = '11'
        elif month == "December" or month == "december" or month == "Diciembre":
            month = '12'
        today = datetime.today()
        date = day + "/" + month + "/" + text
        date1 = datetime.strptime(date, "%d/%m/%Y")

        if date1 < today:
            print("El usuario ha elegido una fecha pasada")
            if idioma == "English":
                reply_keyboard = [['Insert Date'], ['Return']]

                update.message.reply_text(
                    '\n\n'
                    'Sorry, the date selected has passed. Please select a future date',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                reply_keyboard = [['Insertar Fecha'], ['Volver Atrás']]

                update.message.reply_text(
                    '\n\n'
                    'Lo siento, la fecha seleccionada ya ha pasado. Por favor, seleccione una fecha futura.',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            return DUEDATE
        else:
            print("El usuario ha elegido la siguiente fecha {}".format(date))
            chat_id1 = str(update.effective_chat.id)
            actualizarTABLATODO(chat_id1, colDueDate, date)
            existe = str(buscaruseremails(chat_id1))
            if reto == "0":
                actualizarTABLACHALLENGES(chat_id1, colDueDate, date)

            elif reto == "1":
                actualizarTABLADEFIANCE(chat_id1, colDueDate, date)

            if (existe == "['NA']"):
                if idioma == "English":
                    update.message.reply_text(
                        'The last step. Could you provide me your email? You will receive an email confirming the challenge and future 
updates.',
                        reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                elif idioma == "Español":
                    update.message.reply_text(
                        'El último paso.¿Puedes, por favor, enviarme tu email? Recibirás un email confirmando el reto que se ha creado el 
reto y futuras actualizaciones',
                        reply_markup=ReplyKeyboardRemove(remove_keyboard=True))

                return USERMAIL
            elif existe != "NA":
                actualizarTABLADEFIANCE(chat_id1, colEmailJudge, existe[0])
                if reto == "1":
                    nombresretos = buscarchallenges(chat_id1, reto)
                    long = len(nombresretos)
                    cantidadretos = buscaramounts(chat_id1, reto)
                    fechasretos = buscarduedates(chat_id1, reto)
                    juecesmails = buscarjuecesmail1(chat_id1, reto)
                    print(
                        "El usuario ha creado el siguiente reto Name: {} Amount: {} DueDate:{} JudgeEmail: {} ".format(
                            nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                            juecesmails[long - 1]))
                    if idioma == "English":
                        reply_keyboard = [['Change'], ['OK']]
                        context.bot.send_message(chat_id1, text='Thank you!')
                        context.bot.send_message(chat_id1,
                                                 text='You have created this challenge \n\n*Name 🏷️ :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 
:* {} \n*JudgeEmail 📩 :* {} \n\nHas been created'.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1],
                                                     fechasretos[long - 1],
                                                     juecesmails[long - 1]),
                                                 parse_mode="Markdown",
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' If there is any mistake, press the change button. Otherwise press Ok button.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                    elif idioma == "Español":
                        reply_keyboard = [['Cambiar'], ['OK']]
                        context.bot.send_message(chat_id1, text='Gracias!!')
                        context.bot.send_message(chat_id1,
                                                 text='Has creado el siguiente reto: \n\n*Name 🏷️ :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :* 
{} \n*JudgeEmail 📩 :* {} \n\nHas been created'.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1],
                                                     fechasretos[long - 1],
                                                     juecesmails[long - 1]),
                                                 parse_mode="Markdown",
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

                    return FINAL
                else:
                    nombresretos = buscarchallenges(chat_id1, reto)
                    long = len(nombresretos)
                    cantidadretos = buscaramounts(chat_id1, reto)
                    fechasretos = buscarduedates(chat_id1, reto)
                    juecesretos = buscarjueces1(chat_id1)
                    juecesmails = buscarjuecesmail1(chat_id1, reto)
                    usermails = buscaruseremails(chat_id1)
                    print(
                        "El usuario ha creado el siguiente reto Name: {} Amount: {} DueDate:{} Judge: {} JudgeEmail: {} UserEmail : {}
".format(
                            nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                            juecesretos[long - 1], juecesmails[long - 1], usermails[0]))
                    if idioma == "English":
                        reply_keyboard = [['Change'], ['OK']]
                        context.bot.send_message(chat_id1, text='Thank you!')
                        context.bot.send_message(chat_id1,
                                                 text='You have created this challenge \n\n*Name 🏷️ :* {}  \n*Amount 💸 :* {} \n*DueDate 📅
:* {} \n*Judge 👨‍⚖️:* {} \n*JudgeEmail 📩 :* {}  \n*UserEmail 👤 :* {}  \n\nHas been created'.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1],
                                                     fechasretos[long - 1], juecesretos[long - 1],
                                                     juecesmails[long - 1], usermails[0]),
                                                 parse_mode="Markdown",
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' If there is any mistake, press the change button. Otherwise press Ok button.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
                    elif idioma == "Español":
                        reply_keyboard = [['Cambiar'], ['OK']]
                        context.bot.send_message(chat_id1, text='Gracias!!')
                        context.bot.send_message(chat_id1,
                                                 text='Has creado el siguiente reto: \n\n*Name 🏷️ :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :*
{} \n*Judge 👨‍⚖️:* {} \n*JudgeEmail 📩 :* {}  \n*UserEmail 👤 :* {}  \n\nHas been created'.format(
                                                     nombresretos[long - 1], cantidadretos[long - 1],
                                                     fechasretos[long - 1], juecesretos[long - 1],
                                                     juecesmails[long - 1], usermails[0]),
                                                 parse_mode="Markdown",
                                                 reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                        update.message.reply_text(
                            '\n\n'
                            ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                            reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

                    return FINAL
    else:
        print("El usuario ha elegido un año no valido")
        reply_keyboard = [['2020'], ['2021'], ['2022'], ['2023']]
        if idioma == "English":
            update.message.reply_text(
                '\n\n'
                'Select the year',
                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        elif idioma == "Español":
            update.message.reply_text(
                '\n\n'
                'Selecciona el año',
                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        return DUEDATE1


def usermail(update, context):
    text = update.message.text
    chat_id1 = str(update.effective_chat.id)
    idioma = buscaridiomauser(chat_id1)
    reto = buscarvariable(chat_id1, colChallenge)
    if ("@" in text) and ("." in text):
        a = 1
    else:
        a = 0
    if a == 0:
        if idioma == "English":
            context.bot.send_message(chat_id1, text='Sorry, this format is not accepted. ',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            update.message.reply_text('Please, verify that you have sent a valid email and resend it to me.',
                                      reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
        elif idioma == "Español":
            context.bot.send_message(chat_id1, text='Lo siento, este formato no se acepta. ',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            update.message.reply_text('Por favor, verifica que el correo está bien y reenviamelo.',
                                      reply_markup=ReplyKeyboardRemove(remove_keyboard=True))

        return USERMAIL
    else:
        print("El email del usuario es {}".format(text))
        actualizarTABLATODO(chat_id1, colUserEmail, text)
        actualizarTABLAUSERS(chat_id1, colUserEmail, text)
        if reto == "1":
            actualizarTABLADEFIANCE(chat_id1, colEmailJudge, text)
            nombresretos = buscarchallenges(chat_id1, reto)
            long = len(nombresretos)
            cantidadretos = buscaramounts(chat_id1, reto)
            fechasretos = buscarduedates(chat_id1, reto)
            juecesmails = buscarjuecesmail1(chat_id1, reto)
            print(
                "El usuario ha creado el siguiente reto Name: {} Amount: {} DueDate:{} JudgeEmail: {} ".format(
                    nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                    juecesmails[long - 1]))
            if idioma == "English":
                reply_keyboard = [['Change'], ['OK']]
                context.bot.send_message(chat_id1, text='Thank you!')
                context.bot.send_message(chat_id1,
                                         text='You have created this challenge \n\n*Name 🎫 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :* {} 
\n*JudgeEmail 📩 :* {} \n\nHas been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesmails[long - 1]),
                                         parse_mode="Markdown",
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                update.message.reply_text(
                    '\n\n'
                    ' If there is any mistake, press the change button. Otherwise press Ok button.',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
            elif idioma == "Español":
                reply_keyboard = [['Cambiar'], ['OK']]
                context.bot.send_message(chat_id1, text='Gracias!!')
                context.bot.send_message(chat_id1,
                                         text='Has creado el siguiente reto: \n\n*Name 🎫 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :* {} 
\n*JudgeEmail 📩 :* {} \n\nHas been created'.format(
                                             nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                             juecesmails[long - 1]),
                                         parse_mode="Markdown",
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                update.message.reply_text(
                    '\n\n'
                    ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                    reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

            return FINAL
        elif reto == "2":
            nombre = buscarchallenges(chat_id1, "0")
            cantidad = buscaramounts(chat_id1, "0")
            duedate = buscarduedates(chat_id1, "0")
            emailjudge = buscarjuecesmail1(chat_id1, "0")
            usermail = text
            actualizarTABLAUSERS(chat_id1, colUserEmail, usermail)
            print(
                "El usuario ha aceptado el siguiente reto Name: {} Amount: {} DueDate:{} JudgeEmail: {} UserEmail : {} ".format(
                    nombre, cantidad, duedate,
                    emailjudge, usermail))
            if idioma == "English":
                context.bot.send_message(chat_id1,
                                         text='You have accepted the following challenge: \n\n*Description:* {} \n*Amount:* {} \n*DueDate:*{} 
\n*JudgeEmail:* {} \n*UserEmail:* {}'.format(
                                             nombre, cantidad, duedate,
                                             emailjudge, usermail),
                                         parse_mode="Markdown",
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text='Thank you for you confidence in us. We are sure that you are going to perform the challenge 
💪.',

                                         reply_markup=ReplyKeyboardRemove(
                                             remove_keyboard=True))
                keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                             InlineKeyboardButton("Judge", callback_data='Judge')],
                            [InlineKeyboardButton("My Challenges", callback_data='Review'),
                             InlineKeyboardButton("Challenge", callback_data='Challenge')]]

                reply_markup = InlineKeyboardMarkup(keyboard)
                update.message.reply_text(
                    'The following MENU will be useful if you want to create a new challenge, judge or review your existing challenges in the 
future. \n\nLooking forward to hearing from you. ',
```

```python
                reply_markup=reply_markup)
        elif idioma == "Español":
            context.bot.send_message(chat_id1,
                                     text='Has aceptado el siguiente reto: \n\n*Description:* {} \n*Amount:* {} \n*DueDate:*{}
\n*JudgeEmail:* {} \n*UserEmail:* {}'.format(
                                         nombre, cantidad, duedate, emailjudge, usermail),
                                     parse_mode="Markdown",
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            context.bot.send_message(chat_id1,
                                     text='Gracias por tu confianza en nosotros. Estamos seguros de que conseguirá el reto 💪.',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                         InlineKeyboardButton("Juzgar", callback_data='Judge')],
                        [InlineKeyboardButton("Mis Retos", callback_data='Review'),
                         InlineKeyboardButton("Desafío", callback_data='Challenge')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            update.message.reply_text(
                'El siguiente Menu te será de ayuda para cuando quieras Crear, Juzgar o Revisar tus retos. \n Esperamos verte pronto por
aquí',
                reply_markup=reply_markup)
        return FINAL1
    else:
        nombresretos = buscarchallenges(chat_id1, reto)
        long = len(nombresretos)
        cantidadretos = buscaramounts(chat_id1, reto)
        fechasretos = buscarduedates(chat_id1, reto)
        juecesretos = buscarjueces1(chat_id1)
        juecesmails = buscarjuecesmail1(chat_id1, reto)
        usermails = buscaruseremails(chat_id1)
        print(
            "El usuario ha creado el siguiente reto Name: {} Amount: {} DueDate:{} Judge: {} JudgeEmail: {} UserEmail : {} ".format(
                nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1], juecesretos[long - 1],
                juecesmails[long - 1], usermails[0]))
        if idioma == "English":
            reply_keyboard = [['Change'], ['OK']]
            context.bot.send_message(chat_id1, text='Thank you!')
            context.bot.send_message(chat_id1,
                                     text='You have created this challenge \n\n*Name 🎫 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :* {}
\n*Judge 👨‍⚖️:* {} \n*JudgeEmail 📩 :* {}  \n*UserEmail 👤 :* {}  \n\nHas been created'.format(
                                         nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                         juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                     parse_mode="Markdown",
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            update.message.reply_text(
                '\n\n'
                ' If there is any mistake, press the change button. Otherwise press Ok button.',
                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        elif idioma == "Español":
            reply_keyboard = [['Cambiar'], ['OK']]
            context.bot.send_message(chat_id1, text='Gracias!!')
            context.bot.send_message(chat_id1,
                                     text='Has creado el siguiente reto: \n\n*Name 🎫 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :* {}
\n*Judge 👨‍⚖️:* {} \n*JudgeEmail 📩 :* {}  \n*UserEmail 👤 :* {}  \n\nHas been created'.format(
                                         nombresretos[long - 1], cantidadretos[long - 1], fechasretos[long - 1],
                                         juecesretos[long - 1], juecesmails[long - 1], usermails[0]),
                                     parse_mode="Markdown",
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            update.message.reply_text(
                '\n\n'
                ' Si hay algún error presiona el botón de cambiar. Si todo está bien presiona Ok.',
                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

        return FINAL


def final(update, context):
    query = update.message.text
    chat_id1 = str(update.effective_chat.id)
    idioma = buscaridiomauser(chat_id1)
    reto = buscarvariable(chat_id1, colChallenge)
    name = update.effective_chat.first_name
    code = buscarcode()
    challengename = buscarchallenges(chat_id1, reto)
    if query == "Change" or query == "Cambiar":
        change = str(1)
        actualizarTABLAVARIABLES(chat_id1, colChange, change)
        if reto == "0":
            if idioma == "English":
                keyboard = [[InlineKeyboardButton("Name", callback_data='Name'),
                             InlineKeyboardButton("Judge", callback_data='Judgename'),
                             InlineKeyboardButton("Judge Email", callback_data='JudgeEmail')],
                            [InlineKeyboardButton("Amount", callback_data='Amount'),
                             InlineKeyboardButton("DueDate", callback_data='Duedate'),
                             InlineKeyboardButton("User Email", callback_data='UserEmail')]]

                reply_markup = InlineKeyboardMarkup(keyboard)
                update.message.reply_text(
                    'Press the item you want to change before finish creating the challenge',
                    reply_markup=reply_markup)
            elif idioma == "Español":
                keyboard = [[InlineKeyboardButton("Nombre", callback_data='Name'),
                             InlineKeyboardButton("Juez", callback_data='Judgename'),
                             InlineKeyboardButton("Email juez", callback_data='JudgeEmail')],
                            [InlineKeyboardButton("Cantidad", callback_data='Amount'),
                             InlineKeyboardButton("Fecha", callback_data='Duedate'),
                             InlineKeyboardButton("Email usuario", callback_data='UserEmail')]]

                reply_markup = InlineKeyboardMarkup(keyboard)
                update.message.reply_text(
                    'Presiona el elemento que quieras modificar antes de terminar de crear el reto',
                    reply_markup=reply_markup)
            return FINAL1
        elif reto == "1":
            if idioma == "English":
                keyboard = [[InlineKeyboardButton("Name", callback_data='Name'),
                             InlineKeyboardButton("Judge Email", callback_data='JudgeEmail')],
                            [InlineKeyboardButton("Amount", callback_data='Amount'),
                             InlineKeyboardButton("DueDate", callback_data='Duedate')]]

                reply_markup = InlineKeyboardMarkup(keyboard)
                update.message.reply_text(
                    'Press the item you want to change before finish creating the challenge',
                    reply_markup=reply_markup)
            elif idioma == "Español":
                keyboard = [[InlineKeyboardButton("Nombre", callback_data='Name'),
                             InlineKeyboardButton("Email juez", callback_data='JudgeEmail')],
                            [InlineKeyboardButton("Cantidad", callback_data='Amount'),
                             InlineKeyboardButton("Fecha", callback_data='Duedate')]]

                reply_markup = InlineKeyboardMarkup(keyboard)
                update.message.reply_text(
                    'Presiona el elemento que quieras modificar antes de terminar de crear el reto',
                    reply_markup=reply_markup)
            return FINAL1
    elif query == "OK":
        chat_id1 = str(update.effective_chat.id)
        change = str(0)
        actualizarTABLAVARIABLES(chat_id1, colChange, change)
        if reto == "0":
            if idioma == "English":
                context.bot.send_message(chat_id1,
                                         text='Thank you for you confidence in us. We are sure that you are going to perform your challenge
💪.',
                                         reply_markup=ReplyKeyboardRemove(
                                             remove_keyboard=True))
                keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                             InlineKeyboardButton("Judge", callback_data='Judge')],
                            [InlineKeyboardButton("My Challenges", callback_data='Review'),
                             InlineKeyboardButton("Challenge", callback_data='Challenge')]]

                reply_markup = InlineKeyboardMarkup(keyboard)
                update.message.reply_text(
                    'The following MENU will be useful if you want to create a new challenge, judge or review your existing challenges in the
future. \n\nLooking forward to hearing from you. ',
                    reply_markup=reply_markup)
            elif idioma == "Español":
                context.bot.send_message(chat_id1,
                                         text='Gracias por tu confianza en nosotros. Estamos seguros de que conseguirá el reto 💪.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                             InlineKeyboardButton("Juzgar", callback_data='Judge')],
                            [InlineKeyboardButton("Mis Retos", callback_data='Review'),
                             InlineKeyboardButton("Desafío", callback_data='Challenge')]]

                reply_markup = InlineKeyboardMarkup(keyboard)
                update.message.reply_text(
                    'El siguiente Menu te será de ayuda para cuando quieras Crear, Juzgar o Revisar tus retos. \n Esperamos verte pronto por
aquí',
                    reply_markup=reply_markup)
            return FINAL1
        elif reto == "1":
            if idioma == "English":
                context.bot.send_message(chat_id1,
                                         text='Thank you for you confidence in us. You can now send the following message to the friend that
you are challenging',
                                         reply_markup=ReplyKeyboardRemove(
                                             remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text='Hello!! Your friend {} is challenging you to {}. Follow the next steps to see the complete 
challenge:'
                                              '\n- 1.Find @getyourchallenge in Telegram and start a conversation with him'
                                              '\n- 2.Select your language'
                                              '\n- 3.Select Desafiance-->See Challenge'
                                              '\n- 4.Send him the following code: {}'.format(name, str(
                                             challengename[len(challengename) - 1]), code),
                                         reply_markup=ReplyKeyboardRemove(
                                             remove_keyboard=True))
                keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                             InlineKeyboardButton("Judge", callback_data='Judge')],
                            [InlineKeyboardButton("My Challenges", callback_data='Review'),
                             InlineKeyboardButton("Challenge", callback_data='Challenge')]]

                reply_markup = InlineKeyboardMarkup(keyboard)
                update.message.reply_text(
                    'The following MENU will be useful if you want to create a new challenge, judge or review your existing challenges in the 
future. \n Looking forward to hearing from you. ',
                    reply_markup=reply_markup)
            elif idioma == "Español":
                context.bot.send_message(chat_id1,
                                         text='Gracias por tu confianza en nosotros.Ahora puedes enviarle el siguiente mensaje al amigo que 
estás retando.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text='Hola!! Tu amigo {} te está retando a {}. Sigue los siguientes pasos para ver el reto 
completo:'
                                              '\n- 1.Busca a @getyouchallenge_bot en Telegram e inicia una conversación con él'
                                              '\n- 2. Selecciona el Idioma'
                                              '\n- 3.Selecciona Desafio-->Ver Reto'
                                              '\n- 4.Envíale el siguiente código cuando te lo pida: {}'.format(name,
                                                                                                          str(
                                                                                                              challengename[
                                                                                                                  len(
                                                                                                                      challengename) - 
1]),
                                                                                                          code),
                                         reply_markup=ReplyKeyboardRemove(
                                             remove_keyboard=True))
                keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                             InlineKeyboardButton("Juzgar", callback_data='Judge')],
                            [InlineKeyboardButton("Mis Retos", callback_data='Review'),
                             InlineKeyboardButton("Desafío", callback_data='Challenge')]]

                reply_markup = InlineKeyboardMarkup(keyboard)
                update.message.reply_text(
                    'El siguiente Menu te será de ayuda para cuando quieras Crear, Juzgar o Revisar tus retos. \n Esperamos verte pronto por 
aquí',
                    reply_markup=reply_markup)
            return FINAL1
    else:
        if idioma == "English":
            reply_keyboard = [['Change'], ['OK']]
            update.message.reply_text(
                '\n\n'
                'Sorry you have to select the OK button or send it manually before starting again. Otherwise if you have made a mistake press 
change button',
                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        elif idioma == "Español":
            reply_keyboard = [['Cambiar'], ['OK']]
            update.message.reply_text(
                '\n\n'
                'Lo siento, tienes que pulsar el boton de Ok, o enviar ok manualmente. Si lo que quieres es cambiar algo del reto, presiona 
el botón de cambiar',
                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

        return FINAL1


def final1(update, context):
    query = update.message.text
    chat_id1 = str(update.effective_chat.id)
    idioma = buscaridiomauser(chat_id1)
    if query == "Change" or query == "Cambiar":
        change = str(1)
        actualizarTABLAVARIABLES(chat_id1, colChange, change)
        if idioma == "English":
            keyboard = [[InlineKeyboardButton("Name", callback_data='Name'),
                         InlineKeyboardButton("Judge", callback_data='Judgename'),
                         InlineKeyboardButton("Judge Email", callback_data='JudgeEmail')],
                        [InlineKeyboardButton("Amount", callback_data='Amount'),
                         InlineKeyboardButton("DueDate", callback_data='Duedate'),
                         InlineKeyboardButton("User Email", callback_data='UserEmail')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            update.message.reply_text(
                'Press the item you want to change before finish creating the challenge',
                reply_markup=reply_markup)
        elif idioma == "Español":
            keyboard = [[InlineKeyboardButton("Nombre", callback_data='Name'),
                         InlineKeyboardButton("Juez", callback_data='Judgename'),
                         InlineKeyboardButton("Email juez", callback_data='JudgeEmail')],
                        [InlineKeyboardButton("Cantidad", callback_data='Amount'),
                         InlineKeyboardButton("Fecha", callback_data='Duedate'),
                         InlineKeyboardButton("Email usuario", callback_data='UserEmail')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            update.message.reply_text(
                'Presiona el elemento que quieras modificar antes de terminar de crear el reto',
                reply_markup=reply_markup)
        return FINAL1
    elif query == "OK":
        chat_id1 = str(update.effective_chat.id)
        change = str(0)
        actualizarTABLAVARIABLES(chat_id1, colChange, change)
        if idioma == "English":
            context.bot.send_message(chat_id1,
                                     text='Thank you for you confidence in us. We are sure that you are going to perform your challenge.',
                                     reply_markup=ReplyKeyboardRemove(
                                         remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                         InlineKeyboardButton("Judge", callback_data='Judge'),
                         InlineKeyboardButton("My Challenges", callback_data='Review')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            update.message.reply_text(
                'The following MENU will be useful if you want to create a new challenge, judge or review your existing challenges in the
future. \n Looking forward to hearing from you. ',
                reply_markup=reply_markup)
        elif idioma == "Español":
            context.bot.send_message(chat_id1,
                                     text='Gracias por tu confianza en nosotros. Estamos seguros de que conseguirá el reto 💪.',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                         InlineKeyboardButton("Juzgar", callback_data='Judge'),
                         InlineKeyboardButton("Mis Retos", callback_data='Review')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            update.message.reply_text(
                'El siguiente Menu te será de ayuda para cuando quieras Crear, Juzgar o Revisar tus retos. \n Esperamos verte pronto por
aquí',
                reply_markup=reply_markup)
        return FINAL1
    else:
        if idioma == "English":
            reply_keyboard = [['Change'], ['OK']]
            update.message.reply_text(
                '\n\n'
                'Sorry you have to select the OK button or send it manually before starting again. Otherwise if you have made a mistake press
change button',
                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        elif idioma == "Español":
            reply_keyboard = [['Cambiar'], ['OK']]
            update.message.reply_text(
                '\n\n'
                'Lo siento, tienes que pulsar el boton de Ok, o enviar ok manualmente. Si lo que quieres es cambiar algo del reto, presiona
el botón de cambiar',
                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

        return FINAL1


# FIN DE ITINERARIO CREAR RETO

# ITINERARIO JUZGAR RETO
def judge(update, context):
    chat_id1 = str(update.effective_chat.id)
    text = update.message.text
    chat_id1 = str(update.effective_chat.id)
    idioma = buscaridioma1(chat_id1)
    code1 = text
    actualizarTABLAVARIABLES(chat_id1, colCode1, code1)
    vector = buscarcode2(chat_id1)
    if code1 in vector:
        juzgado = buscarstatus1(code1)
    else:
        juzgado = 1
    chat_id1 = str(update.effective_chat.id)
    verify = airtable2.search('Code', code1)
    if (verify != []) and juzgado == "NA":
        print("El usuario quiere juzar el reto con codigo: {}".format(text))
        actualizarTABLAJUDGES(chat_id1, colJudgeID, chat_id1)
        if idioma == "English":
            reply_keyboard = [['Achieve'], ['Not Achieve']]
            update.message.reply_text(
                '\n\n'
                'Did the user reach his/her goal?',
                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        elif idioma == "Español":
            reply_keyboard = [['Conseguido'], ['No Conseguido']]
            update.message.reply_text(
                '\n\n'
                '¿Ha conseguido el reto?',
                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

        return JUDGE1
    else:
        print("El usuario quiere juzgar un reto con un codigo incorrecto")
        if idioma == "English":
            context.bot.send_message(chat_id1, text='Incorrect code',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            update.message.reply_text(
                'Please, verify the code and resend it me. Otherwise,use the Menu above if you wanna change what to do.',
                reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
        elif idioma == "Español":
            context.bot.send_message(chat_id1, text='Código incorrecto',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            update.message.reply_text(
                'Por favor, verifica el código que has recibido y envíamelo de nuevo. Si quieres cambiar lo que hacer utiliza el primer
Menu.',
                reply_markup=ReplyKeyboardRemove(remove_keyboard=True))

        return JUDGE


def judge1(update, context):
    chat_id1 = str(update.effective_chat.id)
    text = update.message.text
    idioma = buscaridioma1(chat_id1)
    code1 = buscarvariable(chat_id1, colCode1)
    if (
        text == "Achieve" or text == "achieve" or text == "Not Achieve" or text == "not achieve" or text == "Not achieve" or text ==
"Conseguido" or text == "No Conseguido"):
        print("El juez ha dado el reto como {}".format(text))
        chat_id1 = str(update.effective_chat.id)
        actualizarTABLATODO(chat_id1, colStatus, text)
        record = record2 = {'Status': text}
        airtable2.update_by_field('Code', code1, record)
        if idioma == "English":
            update.message.reply_text('Thank you for your help', reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            context.bot.send_message(chat_id1,
                                     text='The challenge with code: {} \n Result: {} \n Has been sucessfully judged'.format(
                                         code1, text, reply_markup=ReplyKeyboardRemove(remove_keyboard=True)))

            keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                         InlineKeyboardButton("Judge", callback_data='Judge'),
                         InlineKeyboardButton("My Challenges", callback_data='Review')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            update.message.reply_text(
                'The following MENU will be useful if you want to create a new challenge, judge or review your existing challenges in the
future. \n Looking forward to hearing from you. ',
                reply_markup=reply_markup)
        elif idioma == "Español":
            update.message.reply_text('Gracias por tu ayuda', reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            context.bot.send_message(chat_id1,
                                     text='El reto con código: {} \n *Resultado 🏁:* {} \n Ha sido juzgado con éxito'.format(
                                         code1, text), parse_mode="Markdown",
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))

            keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                         InlineKeyboardButton("Juzgar", callback_data='Judge'),
                         InlineKeyboardButton("Mis Retos", callback_data='Review')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            update.message.reply_text(
                'El siguiente Menu te será de ayuda para cuando quieras Crear, Juzgar o Revisar tus retos. \n Esperamos verte pronto por
aquí',
                reply_markup=reply_markup)
        return FINAL1
    else:
        print("El juez ha elegido una opción incorrecta para juzgar el reto")
        if idioma == "English":
            reply_keyboard = [['Achieve'], ['Not Achieve']]
            update.message.reply_text(
                '\n\n'
                'Please select the result of the challenge from the options below.',
                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        elif idioma == "Español":
            reply_keyboard = [['Conseguido'], ['No Conseguido']]
            update.message.reply_text(
                '\n\n'
                'Selecciona el resultado del reto',
                reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))

        return JUDGE1


# FIN DE ITINERARIO JUZGAR RETO

# ITINERARIO VER MI RETO

def review(update, context):
    query = update.callback_query.data
    chat_id1 = str(update.effective_chat.id)
    idioma = buscaridioma1(chat_id1)
    reto = buscarvariable(chat_id1, colChallenge)
    vectorchallenge = ["Challenge1", "Challenge2", "Challenge3", "Challenge4", "Challenge5", "Challenge6", "Challenge7",
                       "Challenge8", "Challenge9", "Challenge10", "Challenge11", "Challenge12"]
    if query in vectorchallenge:
        challengenames = buscarchallenges(chat_id1, reto)
        numerochallenges = len(challengenames)
        challengeamounts = buscaramounts(chat_id1, reto)
        challengeduedates = buscarduedates(chat_id1, reto)
        challengejudges = buscarjueces1(chat_id1)
        challengejudgemails = buscarjuecesmail1(chat_id1, reto)
        challengeuseremails = buscaruseremails(chat_id1)
        challengestatus = buscarstatus(chat_id1)
    else:
        if query == "Create" or query == "create":
            print("El usuario cambia de ver retos a CREAR")
            codigoanterior = buscarcodeanterior(chat_id1)
            record2 = {'Code': codigoanterior}
            airtable2.insert(record2)
            rellenarairtable2(chat_id1)
            actualizarTABLATODO(chat_id1, colDecision, 'Yes')
            name = update.effective_chat.first_name
            print(idioma)
            if idioma == "English":
                context.bot.send_message(chat_id1, text='Great {}. Lets do it!'.format(name))
                context.bot.send_message(chat_id1,
                                         text='Describe briefly the challenge goal. Use the Menu above if you wanna change what to do.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1, text='Perfecto {}. Vamos a ello!'.format(name))
                context.bot.send_message(chat_id1,
                                         text='Describe brevemente en que consiste el reto. Usa el Menu de arriba si quieres cambiar lo que
hacer.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))

            return CHALLENGENAME
        elif query == "Judge" or query == "judge":
            print("El usuario cambia de ver retos a JUZGAR")
            print(idioma)
            actualizarTABLATODO(chat_id1, colDecision, 'No')
            if idioma == "English":
                context.bot.send_message(chat_id1,
                                         text='I see! Please, send the code of the challenge that you want to judge. Use the Menu above if
you wanna change what to do',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            elif idioma == "Español":
                context.bot.send_message(chat_id1,
                                         text='Por favor envíame el código del reto que quieres juzgar. Usa el Menu de arriba si quieres
cambiar lo que hacer',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))

            return JUDGE
        elif query == "My challenges" or query == "my challenges":
            chat_id1 = str(update.effective_chat.id)
            challenge = buscarchallenges(chat_id1, reto)
            numchallenge = len(challenge)
            actualizarTABLATODO(chat_id1, colDecision, 'No')
            if numchallenge == 1:
                keyboard = [[InlineKeyboardButton(challenge[0], callback_data='Challenge1')]]
                reply_markup = InlineKeyboardMarkup(keyboard)
                if idioma == "English":
                    context.bot.send_message(chat_id1,
                                             text='You have the following challenges. See its descriptions and status pressing in its name'
                                                  ' in the following MENU. Otherwise, change what you want to do using the previous MENU',
                                             reply_markup=reply_markup)
                elif idioma == "Español":
                    context.bot.send_message(chat_id1,
                                             text='Tienes los siguientes retos. See its descriptions and status pressing in its name'
                                                  ' in the following MENU. Otherwise, change what you want to do using the previous MENU',
                                             reply_markup=reply_markup)
                return REVIEW
            elif numchallenge == 2:
                keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                             InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')]]
                reply_markup = InlineKeyboardMarkup(keyboard)
                if idioma == "English":
                    context.bot.send_message(chat_id1,
                                             text='You have the following challenges. See its descriptions and status pressing in its name'
                                                  ' in the following MENU. Otherwise, change what you want to do using the previous MENU',
                                             reply_markup=reply_markup)
                elif idioma == "Español":
                    context.bot.send_message(chat_id1,
                                             text='Tienes los siguientes retos. See its descriptions and status pressing in its name'
                                                  ' in the following MENU. Otherwise, change what you want to do using the previous MENU',
                                             reply_markup=reply_markup)
                return REVIEW
            elif numchallenge == 3:
                keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                             InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2'),
                             InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3')]]
                reply_markup = InlineKeyboardMarkup(keyboard)
                if idioma == "English":
                    context.bot.send_message(chat_id1,
                                             text='You have the following challenges. See its descriptions and status pressing in its name'
                                                  ' in the following MENU. Otherwise, change what you want to do using the previous MENU',
                                             reply_markup=reply_markup)
                elif idioma == "Español":
                    context.bot.send_message(chat_id1,
                                             text='Tienes los siguientes retos. See its descriptions and status pressing in its name'
                                                  ' in the following MENU. Otherwise, change what you want to do using the previous MENU',
                                             reply_markup=reply_markup)
                return REVIEW
            elif numchallenge == 4:
                keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                             InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')],
                            [InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3'),
                             InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4')]]
                reply_markup = InlineKeyboardMarkup(keyboard)
                if idioma == "English":
                    context.bot.send_message(chat_id1,
                                             text='You have the following challenges. See its descriptions and status pressing in its name'
                                                  ' in the following MENU. Otherwise, change what you want to do using the previous MENU',
                                             reply_markup=reply_markup)
                elif idioma == "Español":
                    context.bot.send_message(chat_id1,
                                             text='Tienes los siguientes retos. See its descriptions and status pressing in its name'
                                                  ' in the following MENU. Otherwise, change what you want to do using the previous MENU',
                                             reply_markup=reply_markup)
                return REVIEW
            elif numchallenge == 5:
                keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                             InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2'),
                             InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3')],
                            [InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4'),
                             InlineKeyboardButton(challenge[numchallenge - 5], callback_data='Challenge5')]]
                reply_markup = InlineKeyboardMarkup(keyboard)
                if idioma == "English":
                    context.bot.send_message(chat_id1,
                                             text='You have the following challenges. See its descriptions and status pressing in its name'
                                                  ' in the following MENU. Otherwise, change what you want to do using the previous MENU',
                                             reply_markup=reply_markup)
                elif idioma == "Español":
                    context.bot.send_message(chat_id1,
                                             text='Tienes los siguientes retos. See its descriptions and status pressing in its name'
                                                  ' in the following MENU. Otherwise, change what you want to do using the previous MENU',
                                             reply_markup=reply_markup)
                return REVIEW
            elif numchallenge >= 6:
                keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                             InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')],
                            [InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3'),
                             InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4')],
                            [InlineKeyboardButton(challenge[numchallenge - 5], callback_data='Challenge5'),
                             InlineKeyboardButton(challenge[numchallenge - 6], callback_data='Challenge6')]]
                reply_markup = InlineKeyboardMarkup(keyboard)
                if idioma == "English":
                    context.bot.send_message(chat_id1,
                                             text='You have the following challenges. See its descriptions and status pressing in its name'
                                                  ' in the following MENU. Otherwise, change what you want to do using the previous MENU',
                                             reply_markup=reply_markup)
```

```
                elif idioma == "Español":
                    context.bot.send_message(chat_id1,
                                             text='Tienes los siguientes retos. See its descriptions and status pressing in its name'
                                                  ' in the following MENU. Otherwise, change what you want to do using the previous MENU',
                                             reply_markup=reply_markup)
                return REVIEW
    if query == "Challenge1":
        if idioma == "English":
            chat_id1 = str(update.effective_chat.id)
            context.bot.send_message(chat_id1,
                                     text='*STATE OF THE CHALLENGE:*\n\n*Name 🏷 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :* {} \n*Judge
👨‍⚖️:* {} \n*JudgeEmail 📩 :* {}  \n*Status 🏁:* {} '.format(
                                         challengenames[numerochallenges - 1], challengeamounts[numerochallenges - 1],
                                         challengeduedates[numerochallenges - 1], challengejudges[numerochallenges - 1],
                                         challengejudgemails[numerochallenges - 1],
                                         challengestatus[numerochallenges - 1])
                                     , parse_mode='Markdown', reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                         InlineKeyboardButton("Judge", callback_data='Judge')],
                        [InlineKeyboardButton("My Challenges", callback_data='Review'),
                         InlineKeyboardButton("Challenge", callback_data='Challenge')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1, text=
            'The following MENU will be useful if you want to create a new challenge, judge or review your existing challenges in the future.
\n Looking forward to hearing from you. ',
                                     reply_markup=reply_markup)
            return FINAL1
        elif idioma == "Español":
            chat_id1 = str(update.effective_chat.id)
            context.bot.send_message(chat_id1,
                                     text='*ESTADO DEL RETO:*\n\n*Nombre 🏷 :* {}  \n*Cantidad 💸 :* {} \n*Fecha 📅 :* {} \n*Juez 👨‍⚖️:* {}
\n*Email Juez 📩 :* {}  \n*Estado 🏁:* {} '.format(
                                         challengenames[numerochallenges - 1], challengeamounts[numerochallenges - 1],
                                         challengeduedates[numerochallenges - 1], challengejudges[numerochallenges - 1],
                                         challengejudgemails[numerochallenges - 1],
                                         challengestatus[numerochallenges - 1])
                                     , parse_mode='Markdown', reply_markup=ReplyKeyboardRemove(remove_keyboard=True))

            keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                         InlineKeyboardButton("Juzgar", callback_data='Judge')],
                        [InlineKeyboardButton("Mis Retos", callback_data='Review'),
                         InlineKeyboardButton("Desafío", callback_data='Challenge')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1, text=
            'El siguiente Menu te será de ayuda para cuando quieras Crear, Juzgar o Revisar tus retos. \n Esperamos verte pronto por aquí',
                                     reply_markup=reply_markup)
        return FINAL1
    elif query == "Challenge2":
        if idioma == "English":
            context.bot.send_message(chat_id1,
                                     text='*STATE OF THE CHALLENGE:*\n\n*Name 🏷 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :* {} \n*Judge
👨‍⚖️:* {} \n*JudgeEmail 📩 :* {}  \n*Status 🏁:* {} '.format(
                                         challengenames[numerochallenges - 2], challengeamounts[numerochallenges - 2],
                                         challengeduedates[numerochallenges - 2], challengejudges[numerochallenges - 2],
                                         challengejudgemails[numerochallenges - 2],
                                         challengestatus[numerochallenges - 2])
                                     , parse_mode='Markdown', reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                         InlineKeyboardButton("Judge", callback_data='Judge')],
                        [InlineKeyboardButton("My Challenges", callback_data='Review'),
                         InlineKeyboardButton("Challenge", callback_data='Challenge')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1, text=
            'The following MENU will be useful if you want to create a new challenge, judge or review your existing challenges in the future.
\n Looking forward to hearing from you. ',
                                     reply_markup=reply_markup)
            return FINAL1
        elif idioma == "Español":
            context.bot.send_message(chat_id1,
                                     text='*ESTADO DEL RETO:*\n\n*Nombre 🏷 :* {}  \n*Cantidad 💸 :* {} \n*Fecha 📅 :* {} \n*Juez 👨‍⚖️:* {}
\n*Email Juez 📩 :* {}  \n*Estado 🏁:* {} '.format(
                                         challengenames[numerochallenges - 2], challengeamounts[numerochallenges - 2],
                                         challengeduedates[numerochallenges - 2], challengejudges[numerochallenges - 2],
                                         challengejudgemails[numerochallenges - 2],
                                         challengestatus[numerochallenges - 2])
                                     , parse_mode='Markdown', reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                         InlineKeyboardButton("Juzgar", callback_data='Judge')],
                        [InlineKeyboardButton("Mis Retos", callback_data='Review'),
                         InlineKeyboardButton("Desafío", callback_data='Challenge')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1, text=
            'El siguiente Menu te será de ayuda para cuando quieras Crear, Juzgar o Revisar tus retos. \n Esperamos verte pronto por aquí',
                                     reply_markup=reply_markup)
        return FINAL1
    elif query == "Challenge3":
        if idioma == "English":
            context.bot.send_message(chat_id1,
                                     text='*STATE OF THE CHALLENGE:*\n\n*Name 🏷 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :* {} \n*Judge 
🧑‍⚖️:* {} \n*JudgeEmail 📩 :* {}  \n*Status 🏁:* {} '.format(
                                         challengenames[numerochallenges - 2], challengeamounts[numerochallenges - 2],
                                         challengeduedates[numerochallenges - 2], challengejudges[numerochallenges - 2],
                                         challengejudgemails[numerochallenges - 2],
                                         challengestatus[numerochallenges - 2])
                                     , parse_mode='Markdown', reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                         InlineKeyboardButton("Judge", callback_data='Judge')],
                        [InlineKeyboardButton("My Challenges", callback_data='Review'),
                         InlineKeyboardButton("Challenge", callback_data='Challenge')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1, text=
            'The following MENU will be useful if you want to create a new challenge, judge or review your existing challenges in the future.
\n Looking forward to hearing from you. ',
                                     reply_markup=reply_markup)
            return FINAL1
        elif idioma == "Español":
            context.bot.send_message(chat_id1,
                                     text='*ESTADO DEL RETO:*\n\n*Nombre 🏷 :* {}  \n*Cantidad 💸 :* {} \n*Fecha 📅 :* {} \n*Juez 🧑‍⚖️:* {}
\n*Email Juez 📩 :* {}  \n*Estado 🏁:* {} '.format(
                                         challengenames[numerochallenges - 3], challengeamounts[numerochallenges - 3],
                                         challengeduedates[numerochallenges - 3], challengejudges[numerochallenges - 3],
                                         challengejudgemails[numerochallenges - 3],
                                         challengestatus[numerochallenges - 3])
                                     , parse_mode='Markdown', reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                         InlineKeyboardButton("Juzgar", callback_data='Judge')],
                        [InlineKeyboardButton("Mis Retos", callback_data='Review'),
                         InlineKeyboardButton("Desafío", callback_data='Challenge')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1, text=
            'El siguiente Menu te será de ayuda para cuando quieras Crear, Juzgar o Revisar tus retos. \n Esperamos verte pronto por aquí',
                                     reply_markup=reply_markup)
        return FINAL1
    elif query == "Challenge4":
        if idioma == "English":
            context.bot.send_message(chat_id1,
                                     text='*STATE OF THE CHALLENGE:*\n\n*Name 🏷 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :* {} \n*Judge 
🧑‍⚖️:* {} \n*JudgeEmail 📩 :* {}  \n*Status 🏁:* {} '.format(
                                         challengenames[numerochallenges - 4], challengeamounts[numerochallenges - 4],
                                         challengeduedates[numerochallenges - 4], challengejudges[numerochallenges - 4],
                                         challengejudgemails[numerochallenges - 4],
                                         challengestatus[numerochallenges - 4])
                                     , parse_mode='Markdown', reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                         InlineKeyboardButton("Judge", callback_data='Judge')],
                        [InlineKeyboardButton("My Challenges", callback_data='Review'),
                         InlineKeyboardButton("Challenge", callback_data='Challenge')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1, text=
            'The following MENU will be useful if you want to create a new challenge, judge or review your existing challenges in the future.
\n Looking forward to hearing from you. ',
                                     reply_markup=reply_markup)

            return FINAL1
        elif idioma == "Español":
            context.bot.send_message(chat_id1,
                                     text='*ESTADO DEL RETO:*\n\n*Nombre 🏷 :* {}  \n*Cantidad 💸 :* {} \n*Fecha 📅 :* {} \n*Juez 🧑‍⚖️:* {}
\n*Email Juez 📩 :* {}  \n*Estado 🏁:* {} '.format(
                                         challengenames[numerochallenges - 4], challengeamounts[numerochallenges - 4],
                                         challengeduedates[numerochallenges - 4], challengejudges[numerochallenges - 4],
                                         challengejudgemails[numerochallenges - 4],
                                         challengestatus[numerochallenges - 4])
                                     , parse_mode='Markdown', reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                         InlineKeyboardButton("Juzgar", callback_data='Judge')],
                        [InlineKeyboardButton("Mis Retos", callback_data='Review'),
                         InlineKeyboardButton("Desafío", callback_data='Challenge')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1, text=
            'El siguiente Menu te será de ayuda para cuando quieras Crear, Juzgar o Revisar tus retos. \n Esperamos verte pronto por aquí',
                                reply_markup=reply_markup)
        return FINAL1
    elif query == "Challenge5":
        if idioma == "English":
            context.bot.send_message(chat_id1,
                                text='*STATE OF THE CHALLENGE:*\n\n*Name 🏷 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :* {} \n*Judge 
👨‍⚖️:* {} \n*JudgeEmail 📩 :* {}  \n*Status 🏁:* {} '.format(
                                    challengenames[numerochallenges - 5], challengeamounts[numerochallenges - 5],
                                    challengeduedates[numerochallenges - 5], challengejudges[numerochallenges - 5],
                                    challengejudgemails[numerochallenges - 5],
                                    challengestatus[numerochallenges - 5])
                                , parse_mode='Markdown', reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                         InlineKeyboardButton("Judge", callback_data='Judge')],
                        [InlineKeyboardButton("My Challenges", callback_data='Review'),
                         InlineKeyboardButton("Challenge", callback_data='Challenge')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1, text=
            'The following MENU will be useful if you want to create a new challenge, judge or review your existing challenges in the future. 
\n Looking forward to hearing from you. ',
                                reply_markup=reply_markup)
            return FINAL1
        elif idioma == "Español":
            context.bot.send_message(chat_id1,
                                text='*ESTADO DEL RETO:*\n\n*Nombre 🏷 :* {}  \n*Cantidad 💸 :* {} \n*Fecha 📅 :* {} \n*Juez 👨‍⚖️:* {} 
\n*Email Juez 📩 :* {}  \n*Estado 🏁:* {} '.format(
                                    challengenames[numerochallenges - 5], challengeamounts[numerochallenges - 5],
                                    challengeduedates[numerochallenges - 5], challengejudges[numerochallenges - 5],
                                    challengejudgemails[numerochallenges - 5],
                                    challengestatus[numerochallenges - 5])
                                , parse_mode='Markdown', reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                         InlineKeyboardButton("Juzgar", callback_data='Judge')],
                        [InlineKeyboardButton("Mis Retos", callback_data='Review'),
                         InlineKeyboardButton("Desafío", callback_data='Challenge')]]
            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1, text=
            'El siguiente Menu te será de ayuda para cuando quieras Crear, Juzgar o Revisar tus retos. \n Esperamos verte pronto por aquí',
                                reply_markup=reply_markup)
        return FINAL1
    elif query == "Challenge6":
        if idioma == "English":
            context.bot.send_message(chat_id1,
                                text='*STATE OF THE CHALLENGE:*\n\n*Name 🏷 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :* {} \n*Judge 
👨‍⚖️:* {} \n*JudgeEmail 📩 :* {}  \n*Status 🏁:* {} '.format(
                                    challengenames[numerochallenges - 6], challengeamounts[numerochallenges - 6],
                                    challengeduedates[numerochallenges - 6], challengejudges[numerochallenges - 6],
                                    challengejudgemails[numerochallenges - 6],
                                    challengestatus[numerochallenges - 6])
                                , parse_mode='Markdown', reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                         InlineKeyboardButton("Judge", callback_data='Judge')],
                        [InlineKeyboardButton("My Challenges", callback_data='Review'),
                         InlineKeyboardButton("Challenge", callback_data='Challenge')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1, text=
            'The following MENU will be useful if you want to create a new challenge, judge or review your existing challenges in the future. 
\n Looking forward to hearing from you. ',
                                reply_markup=reply_markup)

            return FINAL1
        elif idioma == "Español":
            context.bot.send_message(chat_id1,
                                text='*ESTADO DEL RETO:*\n\n*Nombre 🏷 :* {}  \n*Cantidad 💸 :* {} \n*Fecha 📅 :* {} \n*Juez 👨‍⚖️:* {} 
\n*Email Juez 📩 :* {}  \n*Estado 🏁:* {} '.format(
                                    challengenames[numerochallenges - 6], challengeamounts[numerochallenges - 6],
                                    challengeduedates[numerochallenges - 6], challengejudges[numerochallenges - 6],
                                    challengejudgemails[numerochallenges - 6],
                                    challengestatus[numerochallenges - 6])
                                , parse_mode='Markdown', reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                         InlineKeyboardButton("Juzgar", callback_data='Judge')],
                        [InlineKeyboardButton("Mis Retos", callback_data='Review'),
                         InlineKeyboardButton("Desafío", callback_data='Challenge')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            context.bot.send_message(chat_id1, text=
            'El siguiente Menu te será de ayuda para cuando quieras Crear, Juzgar o Revisar tus retos. \n Esperamos verte pronto por aquí',
                                reply_markup=reply_markup)
        return FINAL1
```

```
def review1(update, context):
    chat_id1 = str(update.effective_chat.id)
    idioma = buscaridioma1(chat_id1)
    reto = buscarvariable(chat_id1, colChallenge)
    challenge = buscarchallenges(chat_id1, reto)
    numchallenge = len(challenge)
    actualizarTABLATODO(chat_id1, colDecision, 'No')
    if numchallenge == 1:
        keyboard = [InlineKeyboardButton(challenge[numchallenge], callback_data='Challenge1')]
        reply_markup = InlineKeyboardMarkup(keyboard)
        if idioma == "English":
            context.bot.send_message(chat_id1, text=
            'You have this challenges. See its description and status pressing in its name in the following MENU. Otherwise, change what you
want to do using the previous MENU',
                                     reply_markup=reply_markup)
        elif idioma == "Español":
            context.bot.send_message(chat_id1, text=
            'Tienes el siguiente reto. Presiona sobre el nombre para ver las características del reto y su estado. Si quieres cambiar lo que
hacer utiliza el Menu anterior',
                                     reply_markup=reply_markup)
        return REVIEW
    elif numchallenge == 2:
        keyboard = [InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                    InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')]
        reply_markup = InlineKeyboardMarkup(keyboard)
        if idioma == "English":
            context.bot.send_message(chat_id1, text=
            'You have this challenges. See its description and status pressing in its name in the following MENU. Otherwise, change what you
want to do using the previous MENU',
                                     reply_markup=reply_markup)
        elif idioma == "Español":
            context.bot.send_message(chat_id1, text=
            'Tienes los siguientes retos. Presiona sobre el nombre para ver las características del reto y su estado. Si quieres cambiar lo
que hacer utiliza el Menu anterior',
                                     reply_markup=reply_markup)
        return REVIEW
    elif numchallenge == 3:
        keyboard = [InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                    InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2'),
                    InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3')]
        reply_markup = InlineKeyboardMarkup(keyboard)
        if idioma == "English":
            context.bot.send_message(chat_id1, text=
            'You have this challenges. See its description and status pressing in its name in the following MENU. Otherwise, change what you
want to do using the previous MENU',
                                     reply_markup=reply_markup)
        elif idioma == "Español":
            context.bot.send_message(chat_id1, text=
            'Tienes los siguientes retos. Presiona sobre el nombre para ver las características del reto y su estado. Si quieres cambiar lo
que hacer utiliza el Menu anterior',
                                     reply_markup=reply_markup)
        return REVIEW
    elif numchallenge == 4:
        keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                     InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')],
                    [InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3'),
                     InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4')]]
        reply_markup = InlineKeyboardMarkup(keyboard)
        if idioma == "English":
            context.bot.send_message(chat_id1, text=
            'You have this challenges. See its description and status pressing in its name in the following MENU. Otherwise, change what you
want to do using the previous MENU',
                                     reply_markup=reply_markup)
        elif idioma == "Español":
            context.bot.send_message(chat_id1, text=
            'Tienes los siguientes retos. Presiona sobre el nombre para ver las características del reto y su estado. Si quieres cambiar lo
que hacer utiliza el Menu anterior',
                                     reply_markup=reply_markup)
        return REVIEW
    elif numchallenge == 4:
        keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                     InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')],
                    [InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3'),
                     InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4')]]
        reply_markup = InlineKeyboardMarkup(keyboard)
        if idioma == "English":
            context.bot.send_message(chat_id1, text=
            'You have this challenges. See its description and status pressing in its name in the following MENU. Otherwise, change what you
want to do using the previous MENU',
                                     reply_markup=reply_markup)
        elif idioma == "Español":
            context.bot.send_message(chat_id1, text=
            'Tienes los siguientes retos. Presiona sobre el nombre para ver las características del reto y su estado. Si quieres cambiar lo
que hacer utiliza el Menu anterior',
                                     reply_markup=reply_markup)
```

```python
        return REVIEW
    elif numchallenge == 5:
        keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                     InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2'),
                     InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3')],
                    [InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4'),
                     InlineKeyboardButton(challenge[numchallenge - 5], callback_data='Challenge5')]]
        reply_markup = InlineKeyboardMarkup(keyboard)
        if idioma == "English":
            context.bot.send_message(chat_id1, text=
            'You have this challenges. See its description and status pressing in its name in the following MENU. Otherwise, change what you
want to do using the previous MENU',
                                      reply_markup=reply_markup)
        elif idioma == "Español":
            context.bot.send_message(chat_id1, text=
            'Tienes los siguientes retos. Presiona sobre el nombre para ver las características del reto y su estado. Si quieres cambiar lo
que hacer utiliza el Menu anterior',
                                      reply_markup=reply_markup)
        return REVIEW
    elif numchallenge >= 6:
        keyboard = [[InlineKeyboardButton(challenge[numchallenge - 1], callback_data='Challenge1'),
                     InlineKeyboardButton(challenge[numchallenge - 2], callback_data='Challenge2')],
                    [InlineKeyboardButton(challenge[numchallenge - 3], callback_data='Challenge3'),
                     InlineKeyboardButton(challenge[numchallenge - 4], callback_data='Challenge4')],
                    [InlineKeyboardButton(challenge[numchallenge - 5], callback_data='Challenge5'),
                     InlineKeyboardButton(challenge[numchallenge - 6], callback_data='Challenge6')]]
        reply_markup = InlineKeyboardMarkup(keyboard)
        if idioma == "English":
            context.bot.send_message(chat_id1, text=
            'You have this challenges. See its description and status pressing in its name in the following MENU. Otherwise, change what you
want to do using the previous MENU',
                                      reply_markup=reply_markup)
        elif idioma == "Español":
            context.bot.send_message(chat_id1, text=
            'Tienes los siguientes retos. Presiona sobre el nombre para ver las características del reto y su estado. Si quieres cambiar lo
que hacer utiliza el Menu anterior',
                                      reply_markup=reply_markup)
        return REVIEW


# ACEPTAR O DENEGAR RETO

def verreto(update, context):
    text = update.message.text
    print("El usuario está viendo si aceptar el reto {}".format(text))
    chat_id2 = update.effective_chat.id
    codigoanterior = buscarcodeanterior(chat_id2)
    record = {'User ID': str(chat_id2)}
    airtable5.update_by_field('Code', text, record)
    record = {'Code1': text}
    airtable4.update_by_field('Code', codigoanterior, record)
    codigos = buscarcodigos()
    idioma = buscaridioma1(chat_id2)
    if text in codigos:
        nombre = buscarchallenge2(text)
        cantidad = buscaramounts2(text)
        duedate = buscarduedates2(text)
        emailjudge = buscarjuecesmails2(text)
        if idioma == "English":
            reply_keyboard = [['Take it'], ['Leave it']]
            context.bot.send_message(chat_id2, text='Thank you!')
            context.bot.send_message(chat_id2,
                                      text='You have been challenged to \n\n*Description 🎫 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :* {}
\n*JudgeEmail 📩 :* {} '.format(
                                          nombre, cantidad, duedate,
                                          emailjudge),
                                      parse_mode="Markdown",
                                      reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            update.message.reply_text(
                '\n\n'
                'Select *Take it* if you wanna do it 💪 or otherwise press *Leave it*. If you take the challenge you will have the
opportunity to add another judge if you want',
                parse_mode="Markdown"
                , reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        elif idioma == "Español":
            reply_keyboard = [['Aceptar reto'], ['Denegar reto']]
            context.bot.send_message(chat_id2, text='Gracias!')
            context.bot.send_message(chat_id2,
                                      text='Has sido retado a \n\n*Description 🎫 :* {}  \n*Amount 💸 :* {} \n*DueDate 📅 :* {} \n*JudgeEmail
📩 :* {} '.format(
                                          nombre, cantidad, duedate,
                                          emailjudge),
                                      parse_mode="Markdown",
                                      reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            update.message.reply_text(
                '\n\n'
                'Selecciona *Aceptar reto* si quieres hacer el reto 💪 o sino presiona *Denegar reto*. Si decides aceptar el reto podrás
elegir un juez adicional si quieres.',
                parse_mode="Markdown"
                , reply_markup=ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True))
        return VERCHALLENGE
    else:
        if idioma == "English":
            context.bot.send_message(chat_id1, text='Incorrect code.'.format(name))
            context.bot.send_message(chat_id1,
                                     text='Send me the code of the challenge that have been sent to you ',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
        elif idioma == "Español":
            context.bot.send_message(chat_id1, text='Código incorrecto'.format(name))
            context.bot.send_message(chat_id1,
                                     text='Envíame el código del reto que te han enviado',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
        return VERCHALLENGE


def finalretado(update, context):
    text = update.message.text
    chat_id1 = str(update.effective_chat.id)
    name = update.effective_chat.first_name
    idioma = buscaridioma1(chat_id1)
    codigoreto = buscarvariable(chat_id1, colCode1)
    nuevouser = buscarvariable(chat_id1, colNuevoUser)
    print(nuevouser)
    codigoanterior = buscarcodeanterior(chat_id1)
    print("El usuario ha {} el reto".format(text))
    if text == "Aceptar reto" or text == "Take it":
        retado = "2"
        actualizarTABLAVARIABLES(chat_id1, colChallenge, retado)
        nombre = buscarchallenge2(codigoreto)
        cantidad = buscaramounts2(codigoreto)
        duedate = buscarduedates2(codigoreto)
        emailjudge = buscarjuecesmails2(codigoreto)
        record = {'Code': codigoanterior}
        airtable2.insert(record)
        actualizarTABLACHALLENGES(chat_id1, colChallengename, nombre)
        actualizarTABLACHALLENGES(chat_id1, colAmount, cantidad)
        actualizarTABLACHALLENGES(chat_id1, colDueDate, duedate)
        actualizarTABLACHALLENGES(chat_id1, colEmailJudge, emailjudge)
        actualizarTABLACHALLENGES(chat_id1, colChatid, chat_id1)
        actualizarTABLACHALLENGES(chat_id1, colUsername, name)
        actualizarTABLACHALLENGES(chat_id1, colStatus, "NA")
        actualizarTABLACHALLENGES(chat_id1, colJudgename, "NA")
        airtable5.delete_by_field('Code', codigoreto)
        if nuevouser == "0":
            if idioma == "English":
                update.message.reply_text(
                    'Could you provide me your email? You will receive an email confirming the challenge and future updates.',
                    reply_markup=ReplyKeyboardRemove(
                        remove_keyboard=True))
                return USERMAIL
            elif idioma == "Español":
                update.message.reply_text(
                    'Por favor, enviame tu email. Recibirás un correo confirmando que has creado el reto y futuras actualizaciones de este.',
                    reply_markup=ReplyKeyboardRemove(
                        remove_keyboard=True))
                return USERMAIL
        else:
            usermail = buscaruseremails(chat_id1)
            print(
                "El usuario ha aceptado el siguiente reto Name: {} Amount: {} DueDate:{} JudgeEmail: {} UserEmail : {} ".format(
                    nombre, cantidad, duedate,
                    emailjudge, usermail))
            if idioma == "English":
                context.bot.send_message(chat_id1,
                                         text='You have accepted the following challenge: \n\n*Description:* {} \n*Amount:* {} \n*DueDate:*{}
\n*JudgeEmail:* {} \n*UserEmail:* {}'.format(
                                             nombre, cantidad, duedate,
                                             emailjudge, usermail),
                                         parse_mode="Markdown",
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text='Thank you for you confidence in us. We are sure that you are going to perform the challenge
💪.',
                                         reply_markup=ReplyKeyboardRemove(
                                             remove_keyboard=True))
                keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                             InlineKeyboardButton("Judge", callback_data='Judge')],
                            [InlineKeyboardButton("My Challenges", callback_data='Review'),
                             InlineKeyboardButton("Challenge", callback_data='Challenge')]]
                reply_markup = InlineKeyboardMarkup(keyboard)
                update.message.reply_text(
                    'The following MENU will be useful if you want to create a new challenge, judge or review your existing challenges in the
future. \n Looking forward to hearing from you. ',
                    reply_markup=reply_markup)
            elif idioma == "Español":
                context.bot.send_message(chat_id1,
                                         text='Has aceptado el siguiente reto: \n\n*Description:* {} \n*Amount:* {} \n*DueDate:*{}
\n*JudgeEmail:* {} \n*UserEmail:* {}'.format(
                                             nombre, cantidad, duedate, emailjudge, usermail),
                                         parse_mode="Markdown",
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                context.bot.send_message(chat_id1,
                                         text='Gracias por tu confianza en nosotros. Estamos seguros de que conseguirá el reto 💪.',
                                         reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
                keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                             InlineKeyboardButton("Juzgar", callback_data='Judge')],
                            [InlineKeyboardButton("Mis Retos", callback_data='Review'),
                             InlineKeyboardButton("Desafío", callback_data='Challenge')]]

                reply_markup = InlineKeyboardMarkup(keyboard)
                update.message.reply_text(
                    'El siguiente Menu te será de ayuda para cuando quieras Crear, Juzgar o Revisar tus retos. \n Esperamos verte pronto por
aquí',
                    reply_markup=reply_markup)
            return FINAL1
    else:
        airtable5.delete_by_field('Code', codigoreto)
        if idioma == "English":
            context.bot.send_message(chat_id1,
                                     text='We hope to see you again to pursue another challenge that suit you more 💪.',
                                     reply_markup=ReplyKeyboardRemove(
                                         remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                         InlineKeyboardButton("Judge", callback_data='Judge')],
                        [InlineKeyboardButton("My Challenges", callback_data='Review'),
                         InlineKeyboardButton("Challenge", callback_data='Challenge')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            update.message.reply_text(
                'The following MENU will be useful if you want to create a new challenge, judge or review your existing challenges in the
future.',
                reply_markup=reply_markup)
        elif idioma == "Español":
            context.bot.send_message(chat_id1,
                                     text='Esperamos verte de nuevo por aquí para conseguir un reto que se ajuste más a tus objetivos 💪.',
                                     reply_markup=ReplyKeyboardRemove(remove_keyboard=True))
            keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                         InlineKeyboardButton("Juzgar", callback_data='Judge')],
                        [InlineKeyboardButton("Mis Retos", callback_data='Review'),
                         InlineKeyboardButton("Desafío", callback_data='Challenge')]]

            reply_markup = InlineKeyboardMarkup(keyboard)
            update.message.reply_text(
                'El siguiente Menu te será de ayuda para cuando quieras Crear, Juzgar o Revisar tus retos.',
                reply_markup=reply_markup)
        return FINAL1


def buscarcodigos():
    prueba22 = airtable5.get_all(view='Grid view', fields=['Code'])
    numerochallenges = len(prueba22)
    x = json.dumps(prueba22)
    y = json.loads(x)
    i = 0
    while (i < (numerochallenges)):
        resultado = (y[i]["fields"]["Code"])
        if i == 0:
            vector = [resultado]
            i = i + 1
        else:
            vector.insert(i, resultado)
            i = i + 1
    return vector


def buscarchallenge2(code):
    code1 = "'{}'".format(code)
    prueba22 = airtable5.get_all(view='Grid view', fields=['Challenge name'],
                                 formula="{{Code}}={valor}".format(valor=code1))
    x = json.dumps(prueba22)
    y = json.loads(x)
    resultado = (y[0]["fields"]["Challenge name"])
    print(resultado)
    return resultado


def buscaramounts2(code):
    prueba22 = airtable5.get_all(view='Grid view', fields=['Amount'],
                                 formula="{{Code}}={valor}".format(valor=code))
    numerochallenges = len(prueba22)
    x = json.dumps(prueba22)
    y = json.loads(x)
    resultado = (y[numerochallenges - 1]["fields"]["Amount"])
    return resultado


def buscarduedates2(code):
    prueba22 = airtable5.get_all(view='Grid view', fields=['Due date'],
                                 formula="{{Code}}={valor}".format(valor=code))
    numerochallenges = len(prueba22)
    x = json.dumps(prueba22)
    y = json.loads(x)
    resultado = (y[numerochallenges - 1]["fields"]["Due date"])
    return resultado


def buscarjuecesmails2(code):
    prueba22 = airtable5.get_all(view='Grid view', fields=['Email judge'],
                                 formula="{{Code}}={valor}".format(valor=code))
    numerochallenges = len(prueba22)
    x = json.dumps(prueba22)
    y = json.loads(x)
    resultado = (y[numerochallenges - 1]["fields"]["Email judge"])
    return resultado


# RELLENAR AIRTABLE
def rellenarairtable(code):
    record1 = {'User ID': "NA"}
    record2 = {'Challenger or not': "NA"}
    record3 = {'Challenge name': "NA"}
    record4 = {'User name': "NA"}
    record5 = {'Judge name': "NA"}
    record6 = {'Email judge': "NA"}
    record7 = {'User email': "NA"}
    record8 = {'Due date': "NA"}
    record9 = {'Amount': "NA"}
    record10 = {'Status': "NA"}
    record11 = {'Idioma': "NA"}
    airtable.update_by_field('Code', code, record1)
    airtable.update_by_field('Code', code, record2)
    airtable.update_by_field('Code', code, record3)
    airtable.update_by_field('Code', code, record4)
    airtable.update_by_field('Code', code, record5)
    airtable.update_by_field('Code', code, record6)
    airtable.update_by_field('Code', code, record7)
    airtable.update_by_field('Code', code, record8)
    airtable.update_by_field('Code', code, record9)
    airtable.update_by_field('Code', code, record10)
    airtable.update_by_field('Code', code, record11)
    return


def rellenarairtable1(chat_id1):
    argumento = "NA"
    actualizarTABLAUSERS(chat_id1, colUsername, argumento)
    actualizarTABLAUSERS(chat_id1, colUserEmail, argumento)
    actualizarTABLAUSERS(chat_id1, colIdioma, argumento)
    return


def rellenarairtable2(chat_id1):
    argumento = "NA"
    actualizarTABLACHALLENGES(chat_id1, colChatid, argumento)
    actualizarTABLACHALLENGES(chat_id1, colChallengename, argumento)
    actualizarTABLACHALLENGES(chat_id1, colJudgename, argumento)
    actualizarTABLACHALLENGES(chat_id1, colEmailJudge, argumento)
    actualizarTABLACHALLENGES(chat_id1, colDueDate, argumento)
    actualizarTABLACHALLENGES(chat_id1, colAmount, argumento)
    actualizarTABLACHALLENGES(chat_id1, colStatus, argumento)
    actualizarTABLACHALLENGES(chat_id1, colUsername, argumento)
    return


def rellenarairtable3(chat_id1):
    argumento = "NA"
    actualizarTABLAJUDGES(chat_id1, colJudgeID, argumento)
    actualizarTABLAJUDGES(chat_id1, colNamejuez, argumento)
    actualizarTABLAJUDGES(chat_id1, colEmailJudge, argumento)
    return


def rellenarairtable4(code1):
    record1 = {colExistingJudge: str(0)}
    record2 = {colChange: str(0)}
    record3 = {colNuevoUser: str(0)}
    record4 = {colMonth: str(0)}
    record5 = {colDay: str(0)}
    record6 = {colCode1: str(0)}
    record7 = {colChallenge: str(0)}
    airtable4.update_by_field('Code', code1, record1)
    airtable4.update_by_field('Code', code1, record2)
    airtable4.update_by_field('Code', code1, record3)
    airtable4.update_by_field('Code', code1, record4)
    airtable4.update_by_field('Code', code1, record5)
    airtable4.update_by_field('Code', code1, record6)
    airtable4.update_by_field('Code', code1, record7)
    return


def rellenarairtable5(code1):
    record1 = {'User ID': "NA"}
    record3 = {'Challenge name': "NA"}
    record5 = {'JudgeID': "NA"}
    record6 = {'Email judge': "NA"}
    record8 = {'Due date': "NA"}
    record9 = {'Amount': "NA"}
    airtable5.update_by_field('Code', code1, record1)
    airtable5.update_by_field('Code', code1, record9)
    airtable5.update_by_field('Code', code1, record3)
    airtable5.update_by_field('Code', code1, record5)
    airtable5.update_by_field('Code', code1, record6)
    airtable5.update_by_field('Code', code1, record8)
    return


# BUSCAR EN AIRTABLE

def buscarcode():
    excodes = airtable.get_all(view='Grid view', fields=['Code'])
    longitud = len(excodes)
    posicion = int(longitud) - 1
    x = json.dumps(excodes)
    y = json.loads(x)
    excode = int(y[posicion]["fields"]["Code"])
    return excode


def buscarcodeanterior(chat_id2):
    print("estoy")
    chat_id1 = "'{}'".format(chat_id2)
    print(chat_id1)
    prueba22 = airtable.get_all(view='Grid view', fields=['Code'],
                                formula="{{User ID}}={valor}".format(valor=chat_id1))
    numerochallenges = len(prueba22)
    x = json.dumps(prueba22)
    y = json.loads(x)
    i = 0
    while (i < (numerochallenges)):
        resultado = (y[i]["fields"]["Code"])
        if i == 0:
            vector = [resultado]
            i = i + 1
        else:
            vector.insert(i, resultado)
            i = i + 1
    vector1 = vector[numerochallenges - 1]
    print(vector1)
    return vector1


def buscarcode2(chat_id2):
    chat_id1 = "'{}'".format(chat_id2)
    prueba22 = airtable2.get_all(view='Grid view', fields=['Code'],
                                 formula="{{User ID}}={valor}".format(valor=chat_id1))
    numerochallenges = len(prueba22)
    x = json.dumps(prueba22)
    y = json.loads(x)
    i = 0
    if (numerochallenges == 0):
        vector = [0]
    else:
        while (i < (numerochallenges)):
            resultado = (y[i]["fields"]["Code"])
            if i == 0:
                vector = [resultado]
                i = i + 1
            else:
                vector.insert(i, resultado)
                i = i + 1
    return vector


def buscaridiomauser(chat_id2):
    chat_id1 = "'{}'".format(chat_id2)
    prueba22 = airtable1.get_all(view='Grid view', fields=['Idioma'],
                                 formula="{{User ID}}={valor}".format(valor=chat_id1))
    numerochallenges = len(prueba22)
    x = json.dumps(prueba22)
    y = json.loads(x)
    i = 0
    while (i < (numerochallenges)):
        resultado = (y[i]["fields"]["Idioma"])
        if i == 0:
            vector = [resultado]
            i = i + 1
        else:
            vector.insert(i, resultado)
            i = i + 1
    vector1 = vector[0]
    return vector1


def buscaridioma1(chat_id2):
    chat_id1 = "'{}'".format(str(chat_id2))
    print(chat_id1)
    prueba22 = airtable.get_all(view='Grid view', fields=['Idioma'],
                                formula="{{User ID}}={valor}".format(valor=chat_id1))
    numerochallenges = len(prueba22)
    x = json.dumps(prueba22)
    y = json.loads(x)
    i = 0
    while (i < (numerochallenges)):
        resultado = (y[i]["fields"]["Idioma"])
        if i == 0:
            vector = [resultado]
            i = i + 1
        else:
            vector.insert(i, resultado)
            i = i + 1
    vector1 = vector[numerochallenges - 1]
    return vector1


def buscarjueces(chat_id2):
    chat_id1 = "'{}'".format(chat_id2)
    prueba11 = airtable2.get_all(view='Grid view', fields=['Judge name'],
                                 formula="{{User ID}}={valor}".format(valor=chat_id1))
    numerojudges = len(prueba11)
    x = json.dumps(prueba11)
    y = json.loads(x)
    i = 0
    while (i < (numerojudges - 1)):
        resultado = (y[i]["fields"]["Judge name"])
        if i == 0:
            vector = [resultado]
            i = i + 1
        else:
            vector.insert(i, resultado)
            i = i + 1

    res = []
    [res.append(x) for x in vector if x not in res]
    return res


def buscarjuecesmail(chat_id2):
    chat_id1 = "'{}'".format(chat_id2)
    prueba11 = airtable2.get_all(view='Grid view', fields=['Email judge'],
                                 formula="{{User ID}}={valor}".format(valor=chat_id1))
    numerojudges = len(prueba11)
    x = json.dumps(prueba11)
    y = json.loads(x)
    i = 0
    while (i < (numerojudges - 1)):
        resultado = (y[i]["fields"]["Email judge"])
        if i == 0:
            vector = [resultado]
            i = i + 1
        else:
            vector.insert(i, resultado)
            i = i + 1

    res = []
    [res.append(x) for x in vector if x not in res]
    return res


def buscarchallenges(chat_id1, challenge):
    chat_id2 = "'{}'".format(chat_id1)
    if challenge == "0" or challenge == 0:
        prueba22 = airtable2.get_all(view='Grid view', fields=['Challenge name'],
                                     formula="{{User ID}}={valor}".format(valor=chat_id2))
    elif challenge == "1" or challenge == 1:
        prueba22 = airtable5.get_all(view='Grid view', fields=['Challenge name'],
                                     formula="{{JudgeID}}={valor}".format(valor=chat_id2))
    numerochallenges = len(prueba22)
    x = json.dumps(prueba22)
    y = json.loads(x)
    i = 0
    while (i < (numerochallenges)):
        resultado = (y[i]["fields"]["Challenge name"])
        if i == 0:
            vector = [resultado]
            i = i + 1
        else:
            vector.insert(i, resultado)
            i = i + 1
    return vector


def buscarjueces1(chat_id2):
    chat_id1 = "'{}'".format(chat_id2)
    prueba11 = airtable2.get_all(view='Grid view', fields=['Judge name'],
                                 formula="{{User ID}}={valor}".format(valor=chat_id1))
    numerojudges = len(prueba11)
    x = json.dumps(prueba11)
    y = json.loads(x)
    i = 0
    while (i < (numerojudges)):
        resultado = (y[i]["fields"]["Judge name"])
        if i == 0:
            vector = [resultado]
            i = i + 1
        else:
            vector.insert(i, resultado)
            i = i + 1
    return vector


def buscarjuecesmail1(chat_id2, challenge):
    chat_id1 = "'{}'".format(chat_id2)

    if challenge == "0" or challenge == 0:
        prueba11 = airtable2.get_all(view='Grid view', fields=['Email judge'],
                                     formula="{{User ID}}={valor}".format(valor=chat_id2))
    elif challenge == "1" or challenge == 1:
        prueba11 = airtable5.get_all(view='Grid view', fields=['Email judge'],
                                     formula="{{JudgeID}}={valor}".format(valor=chat_id2))
    numerojudges = len(prueba11)
    x = json.dumps(prueba11)
    y = json.loads(x)
    i = 0
    while (i < (numerojudges)):
        resultado = (y[i]["fields"]["Email judge"])
        if i == 0:
            vector = [resultado]
            i = i + 1
        else:
            vector.insert(i, resultado)
            i = i + 1
    return vector


def todosjuecesemails():
    prueba11 = airtable3.get_all(view='Grid view', fields=['Email judge'])
    x = json.dumps(prueba11)
    y = json.loads(x)
    i = 0
    while (i < (len(prueba11))):
        resultado = (y[i]["fields"]["Email judge"])
        if i == 0:
            vector = [resultado]
            i = i + 1
        else:
            vector.insert(i, resultado)
            i = i + 1
    return vector


def buscaramounts(chat_id2, challenge):
    chat_id1 = "'{}'".format(chat_id2)
    if challenge == "0" or challenge == 0:
        prueba22 = airtable2.get_all(view='Grid view', fields=['Amount'],
                                     formula="{{User ID}}={valor}".format(valor=chat_id1))
    elif challenge == "1" or challenge == 1:
        prueba22 = airtable5.get_all(view='Grid view', fields=['Amount'],
                                     formula="{{JudgeID}}={valor}".format(valor=chat_id1))

    numerochallenges = len(prueba22)
    x = json.dumps(prueba22)
    y = json.loads(x)
    i = 0
    while (i < (numerochallenges)):
        resultado = (y[i]["fields"]["Amount"])
        if i == 0:
            vector = [resultado]
            i = i + 1
        else:
            vector.insert(i, resultado)
            i = i + 1
    return vector


def buscarduedates(chat_id2, challenge):
    chat_id1 = "'{}'".format(chat_id2)
    if challenge == "0" or challenge == 0:
        prueba22 = airtable2.get_all(view='Grid view', fields=['Due date'],
                                     formula="{{User ID}}={valor}".format(valor=chat_id1))
    elif challenge == "1" or challenge == 1:
        prueba22 = airtable5.get_all(view='Grid view', fields=['Due date'],
                                     formula="{{JudgeID}}={valor}".format(valor=chat_id1))
    numerochallenges = len(prueba22)
    x = json.dumps(prueba22)
    y = json.loads(x)
    i = 0
    while (i < (numerochallenges)):
        resultado = (y[i]["fields"]["Due date"])
        if i == 0:
            vector = [resultado]
            i = i + 1
        else:
            vector.insert(i, resultado)
            i = i + 1
    return vector


def buscaruseremails(chat_id2):
    chat_id1 = "'{}'".format(chat_id2)
    prueba22 = airtable1.get_all(view='Grid view', fields=['User email'],
                                 formula="{{User ID}}={valor}".format(valor=chat_id1))
    numerousers = len(prueba22)
    x = json.dumps(prueba22)
    y = json.loads(x)
    i = 0
    while (i < (numerousers)):
        resultado = (y[i]["fields"]["User email"])
        if i == 0:
            vector = [resultado]
            i = i + 1
        else:
            vector.insert(i, resultado)
            i = i + 1
    return vector


def buscarstatus(chat_id2):
    chat_id1 = "'{}'".format(chat_id2)
    prueba22 = airtable2.get_all(view='Grid view', fields=['Status'],
                                 formula="{{User ID}}={valor}".format(valor=chat_id1))
    numerochallenges = len(prueba22)
    x = json.dumps(prueba22)
    y = json.loads(x)
    i = 0
    while (i < (numerochallenges)):
        resultado = (y[i]["fields"]["Status"])
        if i == 0:
            vector = [resultado]
            i = i + 1
        else:
            vector.insert(i, resultado)
            i = i + 1
    return vector


def buscarstatus1(codigo):
    chat_id1 = "'{}'".format(codigo)
    prueba22 = airtable2.get_all(view='Grid view', fields=['Status'],
                                 formula="{{Code}}={valor}".format(valor=chat_id1))
    x = json.dumps(prueba22)
    y = json.loads(x)
    resultado = (y[0]["fields"]["Status"])
    return resultado


def buscarvariable(chat_id2, columna):
    chat_id1 = "'{}'".format(chat_id2)
    codigo = buscarcodeanterior(chat_id2)
    prueba22 = airtable4.get_all(view='Grid view', fields=[columna],
                                 formula="{{Code}}={valor}".format(valor=codigo))
    numerochallenges = len(prueba22)
    x = json.dumps(prueba22)
    y = json.loads(x)
    i = 0
    while (i < (numerochallenges)):
        resultado = (y[i]["fields"][columna])
        if i == 0:
            vector = [resultado]
            i = i + 1
        else:
            vector.insert(i, resultado)
            i = i + 1
    vector1 = vector[numerochallenges - 1]
    return vector1


## ACTUALIZAR BASE DE DATOS

def actualizarTABLATODO(identidad, columna, argumento):
    codigo = buscarcodeanterior(identidad)
    record = {columna: argumento}
    airtable.update_by_field('Code', codigo, record)
    return


def actualizarTABLAUSERS(identidad, columna, argumento):
    record = {columna: argumento}
    airtable1.update_by_field('User ID', identidad, record)
    return


def actualizarTABLACHALLENGES(identidad, columna, argumento):
    codigo = buscarcodeanterior(identidad)
    record = {columna: argumento}
    airtable2.update_by_field('Code', codigo, record)
    return


def actualizarTABLAJUDGES(identidad, columna, argumento):
    codigo = buscarcodeanterior(identidad)
    record = {columna: argumento}
    airtable3.update_by_field('Code', codigo, record)
    return


def actualizarTABLAVARIABLES(identidad, columna, argumento):
    codigo = buscarcodeanterior(identidad)
    record = {columna: argumento}
    airtable4.update_by_field('Code', codigo, record)
    return


def actualizarTABLADEFIANCE(identidad, columna, argumento):
    codigo = buscarcodeanterior(identidad)
    record = {columna: argumento}
    airtable5.update_by_field('Code', codigo, record)
    return


# FUNCIONES DE SALIDA
def cancel(update, context):
    chat_id1 = str(update.effective_chat.id)
    idioma = buscaridiomauser(chat_id1)
    codigoanterior = buscarcodeanterior()
    verify = airtable.search("Code", codigoanterior)
    verify2 = airtable2.search("Code", codigoanterior)
    verify3 = airtable3.search("Code", codigoanterior)
    if verify != []:
        airtable.delete_by_field("Code", codigoanterior)
        if verify2 != []:
            airtable2.delete_by_field("Code", codigoanterior)
            if verify3 != []:
                airtable3.delete_by_field("Code", codigoanterior)
    if idioma == "English":
        keyboard = [[InlineKeyboardButton("Create", callback_data='Create'),
                     InlineKeyboardButton("Judge", callback_data='Judge'),
                     InlineKeyboardButton("My Challenges", callback_data='Review')]]

        reply_markup = InlineKeyboardMarkup(keyboard)
        update.message.reply_text(
            '{}, no worries. You are in the main Menu, select the action you wanna do to start again.'.format(name),
            reply_markup=reply_markup)
    elif idioma == "Español":
        keyboard = [[InlineKeyboardButton("Crear", callback_data='Create'),
                     InlineKeyboardButton("Juzgar", callback_data='Judge'),
                     InlineKeyboardButton("Mis Retos", callback_data='Review')]]

        reply_markup = InlineKeyboardMarkup(keyboard)
        update.message.reply_text(
            '{}, no te preocupes. Estás en el Menu principal, selecciona la acción que quieres hacer para empezar de nuevo.'.format(
                name),
            reply_markup=reply_markup)

    return FINAL1


def error(update, context):
    """Log Errors caused by Updates."""
    logger.warning('Update "%s" caused error "%s"', update, context.error)


def main():
    while (1):
        # Create the Updater and pass it your bot's token.
        # Make sure to set use_context=True to use the new context based callbacks
        # Post version 12 this will no longer be necessary
        updater = Updater("1166595191:AAEaAaMB859vmxSK323OqQSm3mVE0b5p1l0", use_context=True)

        # Get the dispatcher to register handlers
        dp = updater.dispatcher

        # Add conversation handler with the states GENDER, PHOTO, LOCATION and BIO
        conv_handler = ConversationHandler(
            entry_points=[CommandHandler('start', start)],

            states={

                START: [CallbackQueryHandler(start1), MessageHandler(Filters.all, start2)],

                DECISION: [CallbackQueryHandler(decision), CommandHandler('cancel', cancel),
                           MessageHandler(Filters.all, decision1)],

                CHALLENGENAME: [CallbackQueryHandler(decision), CommandHandler('cancel', cancel),
                                MessageHandler(Filters.all, challengename)],

                AMOUNT: [CallbackQueryHandler(decision2),
                         MessageHandler(Filters.regex('^(5€|10€|15€|20€|25€|Other)$'), amount),
                         CommandHandler('cancel', cancel), MessageHandler(Filters.all, amount1)],

                DATE: [CallbackQueryHandler(decision2), MessageHandler(Filters.regex(
                    '^(January|February|March|April|May|June|July|August|September|October|November|December|'
                    'Enero|Febrero|Abril|Mayo|Junio|Julio|Agosto|Septiembre|Octubre|Noviembre|Diciembre)$'), month),
                       CommandHandler('cancel', cancel), MessageHandler(Filters.all, month1)],

                DATE1: [CallbackQueryHandler(decision2), MessageHandler(Filters.regex(
                    '^(1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|20|21|22|23|24|25|26|27|28|29|30|31)$'), day),
                        CommandHandler('cancel', cancel), MessageHandler(Filters.all, day1)],

                ADDITIONAL: [CallbackQueryHandler(decision2),
                             MessageHandler(Filters.regex('^(Insert Quantity|Insertar Cantidad)$'), amount1),
                             MessageHandler(Filters.regex('^(Return|Volver Atrás)$'), metercantidad),
                             CommandHandler('cancel', cancel), MessageHandler(Filters.all, amount1)],

                DUEDATE: [CallbackQueryHandler(decision2),
                          MessageHandler(Filters.regex('^(Insert Date|Insertar Fecha)$'), date),
                          MessageHandler(Filters.regex('^(Return|Volver Atrás)$'), meterfecha),
                          CommandHandler('cancel', cancel), MessageHandler(Filters.all, date)],
                DUEDATE1: [CallbackQueryHandler(decision2),
                           MessageHandler(Filters.regex('^(2020|2021|2022|2023|2024)$'), year),
                           CommandHandler('cancel', cancel), MessageHandler(Filters.all, year1)],

                USERMAIL: [CallbackQueryHandler(decision2), CommandHandler('cancel', cancel),
                           MessageHandler(Filters.all, usermail)],

                JUDGE: [CallbackQueryHandler(decision), CommandHandler('cancel', cancel),
                        MessageHandler(Filters.all, judge)],

                JUDGE1: [CallbackQueryHandler(decision2),
                         MessageHandler(Filters.regex('^(Achieve|Not Achieve|Conseguido|No Conseguido)$'), judge1),
                         CommandHandler('cancel', cancel), MessageHandler(Filters.all, judge1)],

                JUDGENAME: [CallbackQueryHandler(decision2), CommandHandler('cancel', cancel),
                            MessageHandler(Filters.all, judgename)],

                JUDGEMAIL: [CallbackQueryHandler(decision2), CommandHandler('cancel', cancel),
                            MessageHandler(Filters.all, judgemail)],

                FINAL: [CallbackQueryHandler(decision2), MessageHandler(Filters.regex('^(Change|OK|Cambiar)$'), final),
                        CommandHandler('cancel', cancel), MessageHandler(Filters.all, final1)],

                EXISTING: [CallbackQueryHandler(existingjudge), CommandHandler('cancel', cancel),
                           MessageHandler(Filters.all, existingjudge1)],

                FINAL1: [MessageHandler(Filters.regex('^(Change|OK|Cambiar)$'), final),
                         CommandHandler('cancel', cancel),
                         CallbackQueryHandler(decision2), MessageHandler(Filters.all, decision1)],

                REVIEW: [CallbackQueryHandler(review), CommandHandler('cancel', cancel),
                         MessageHandler(Filters.all, review1)],

                VERCHALLENGE: [CallbackQueryHandler(decision2),
                               MessageHandler(Filters.regex('^(Take it|Leave it|Aceptar reto|Denegar reto)$'),
                                              finalretado), CommandHandler('cancel', cancel),
                               MessageHandler(Filters.all, verreto)]

            },

            fallbacks=[CommandHandler('cancel', cancel)]
        )

        dp.add_handler(conv_handler)

        # log all errors
        dp.add_error_handler(error)

        # Start the Bot
        updater.start_polling()

        # Run the bot until you press Ctrl-C or the process receives SIGINT,
        # SIGTERM or SIGABRT. This should be used most of the time, since
        # start_polling() is non-blocking and will stop the bot gracefully.
        updater.idle()


if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        exit()
```