GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

CIBERHIGIENE: PLATAFORMA DE ENSEÑANZA DE CIBERSEGURIDAD BASADA EN LA GAMIFICACIÓN

Autor: Álvaro Prado Moreno

Director: Ramamoorthi, Lokesh

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Ciberhigiene: Plataforma de enseñanza de ciberseguridad basada en la gamificación

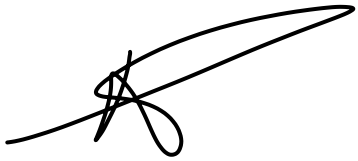
en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2021-2022 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.



Fdo.: Álvaro Prado Moreno Fecha: 31…/05…/ 2022

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

LokeshRamamoorthi

Fdo.: Lokesh Ramamoorthi Fecha: 31…/05…/2022

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

CIBERHIGIENE: PLATAFORMA DE ENSEÑANZA DE CIBERSEGURIDAD BASADA EN LA GAMIFICACIÓN

Autor: Álvaro Prado Moreno

Director: Ramamoorthi, Lokesh

Madrid

Agradecimientos

Me gustaría dedicarle esta sección a mi madre. Gracias a ella he conseguido superar los retos que estos 4 años han supuesto para mí. A pesar de las dificultades y sin importar la situación ella siempre ha estado para apoyarme en mis decisiones y darme fuerzas en los momentos más difíciles.

También me gustaría mencionar a mi hermano, el cual siempre ha sido para mí un modelo a seguir y una motivación continua para perseguir mis objetivos y marcarme nuevas metas en el ámbito académico y personal.

CIBERHIGIENE: PLATAFORMA DE ENSEÑANZA DE CIBERSEGURIDAD BASADA EN LA GAMIFICACIÓN

Autor: Prado Moreno, Álvaro.
Director: Ramamoorthi, Lokesh.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Este proyecto consiste en el desarrollo de una plataforma de enseñanza sobre conceptos relacionados con la ciberseguridad implementando técnicas de gamificación. La plataforma toma forma de una aplicación móvil desarrollada en Flutter válida para dispositivos iOS y Android. Se ha obtenido una versión alfa operativa y descargable.

Palabras clave: Gamificación, Ciberseguridad, Plataforma de enseñanza, Aplicación móvil,

Flutter, Firebase, Android, iOS

1. Introducción

En un mundo cada vez más dependiente de la tecnología, la ciberseguridad ha tomado un papel clave en la industria. Si se considera el crimen relacionado con la ciberseguridad como un país, este sería la tercera economía más grande a nivel mundial después de Estados Unidos y China.[1] Las economías de los países están experimentando grandes pérdidas como consecuencia de los ciberataques. Por ejemplo, Estados Unidos perdió 7 billones de dólares en 2021 debido a los ciberataques.[2] No obstante, lo más importante es que la mayor parte de los ataques están relacionados con errores humanos ya que un 95% de las filtraciones de datos se deben a errores humanos. [3] Con este proyecto se trata de abordar este problema.

2. Definición del proyecto

Este Proyecto consiste en el desarrollo de una plataforma de enseñanza. La plataforma enseña principalmente sobre conceptos relacionados con la ciberseguridad con el objetivo de que los usuarios acaben obteniendo una serie de conocimientos básicos relacionados con este ámbito. Además, se pretende que los usuarios terminen por desarrollar hábitos en su día a día, definidos en este proyecto como ciberhigiénicos, que les permitan estar más seguros a la hora de navegar en el ciberespacio, reducir su huella digital y mantener su información privada a salvo. Para conseguir este objetivo se han introducido técnicas de gamificación en la aplicación para motivar a los usuarios a completar los cursos ya que han demostrado su efectividad en otros casos.

3. Descripción del modelo/sistema/herramienta

En la Ilustración 1 se puede observar el esquema general que sigue la plataforma desarrollada. Los usuarios que utilizarían la aplicación son cualquier individuo con curiosidad acerca de la ciberseguridad, así como organizaciones que quieran tener acceso a la plataforma para formar a sus empleados y tener una plantilla formada en seguridad para evitar tener pérdidas monetarias o filtraciones de datos por errores humanos. Los

usuarios tendrían acceso a la plataforma por medio de la aplicación móvil desarrollada para dispositivos Android e iOS.

Las tres principales secciones de la aplicación vienen reflejadas como módulos. El módulo de los cursos hace referencia a los cursos que se ofrecen a los usuarios. Se trata de cursos breves de entre 5-10 preguntas con preguntas de tipo opción múltiple y rellenar los huecos. Los cursos pertenecen a una de las cuatro categorías definidas en la plataforma: Web, Dispositivos, Redes Sociales e Información. Cada curso ocupa una posición distinta en su categoría y tiene asociados diferentes puntos de experiencia e insignias. Además, un curso puede establecerse como destacado y otorgar el doble de puntos de experiencia o recomendado para que aparezca en el panel principal del usuario.

El módulo de perfil hace referencia a la información relativa al usuario y a las técnicas de gamificación incluidas en la plataforma. Los usuarios cuentan con un perfil en el que pueden observar qué insignias han logrado, qué cursos han completado o guardado y que avatares han conseguido. Cada vez que un usuario completa un curso, obtiene una insignia de reconocimiento en caso de responder a la mayoría de las preguntas bien. Además, se obtienen puntos de experiencia que hacen avanzar en un sistema de niveles. En caso de que un usuario suba de nivel, este obtiene un nuevo avatar que puede establecer como su imagen de perfil. Por último, El usuario puede observar su progreso en su perfil y modificar información relacionada con la cuenta.

En cuanto al módulo de administrador, este hace referencia a la sección de la aplicación destinada a los administradores de la plataforma. El administrador puede crear y eliminar cursos, ver el progreso de los usuarios y cambiar los cursos recomendado y destacado.

La capa de datos hace referencia a los servicios que se han utilizado de Firebase para el desarrollo. Se ha utilizado Firebase Authentication para controlar la autenticación de usuarios y Cloud Firestore para guardar información relativa a los usuarios y a los cursos en bases de datos no relacionales basadas en colecciones y documentos.

Por último, en la capa de negocio se puede observar que se mencionan al administrador y a los procuradores de contenido. Las organizaciones que obtuvieran acceso a la plataforma adoptarían ese papel de administradores para adaptar el contenido a sus necesidades. Por otro lado, esta plataforma necesita de unos procuradores de contenido que aporten información de calidad sobre conceptos de ciberseguridad.

Ilustración 1 Diagrama de la arquitectura de la plataforma

4. Resultados

En la Ilustración 2 se pueden observar algunas de las pantallas principales de la aplicación móvil. En las capturas se han reflejado pantallas pertenecientes a las secciones más relevantes de la aplicación. En la primera y en la segunda de las pantallas se puede observar la descripción de un curso y una pregunta de tipo rellenar los huecos de este. En la tercera de las pantallas se puede ver la pantalla principal del perfil de un usuario y en la cuarta el panel principal del administrador. Se puede observar cómo se ha logrado mostrar información proveniente de la base de datos de los cursos y de los usuarios.

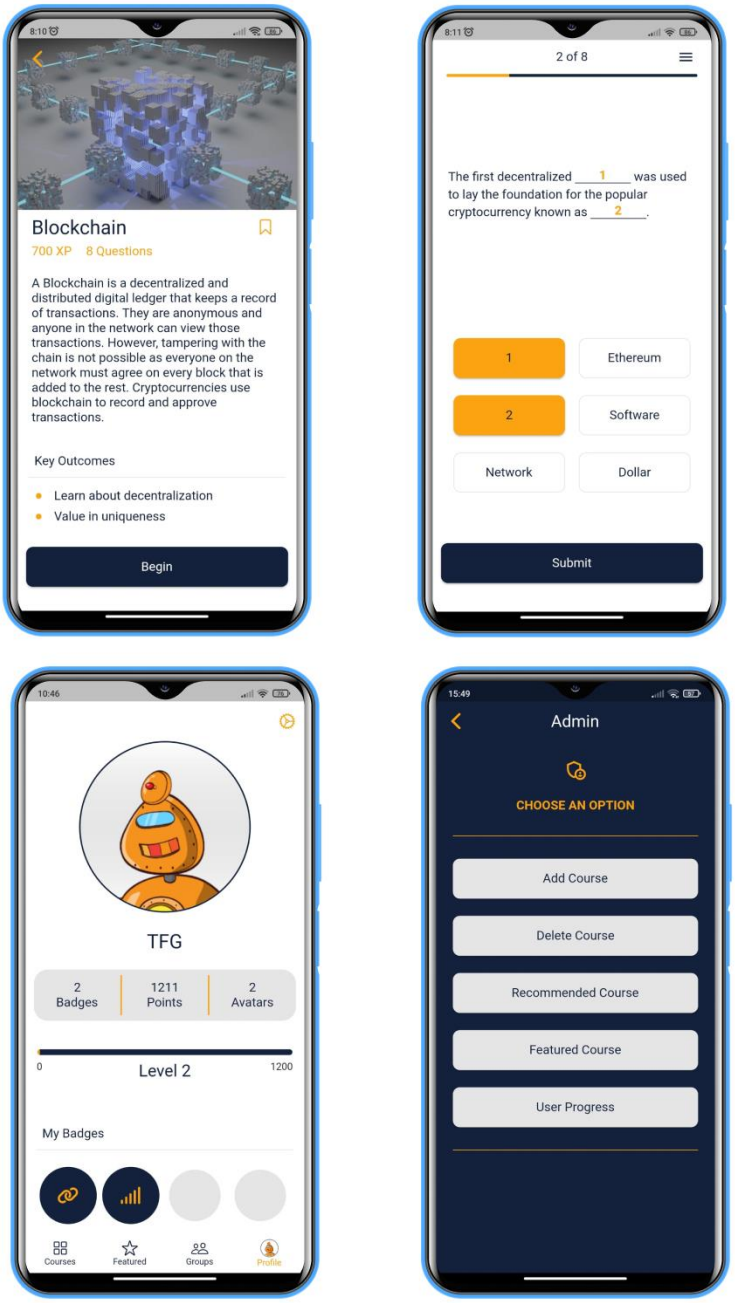


Ilustración 2 Principales páginas de la aplicación

5. Conclusiones

Como se ha justificado en la introducción, actualmente la ciberseguridad ocupa una posición clave en la mayoría de las industrias. No obstante, no todo el mundo recibe una educación acerca de aspectos relacionados con la misma. A pesar de que el mundo está cada vez más digitalizado y que dependemos de las tecnologías cada vez más en nuestro día a día tanto en el mundo laboral como en nuestra vida privada, no existe una formación genérica acerca de este ámbito, excluyendo titulaciones en cuyo programa sí se incluye educación de este tipo por su relación con la tecnología y consecuentemente la ciberseguridad. Es por ello que este proyecto, trata de ofrecer educación de calidad y atractiva sobre ciberseguridad.

Como consecuencia de la importancia que tiene la ciberseguridad, muchas organizaciones tratan de formar a sus empleados en ciertos aspectos relacionadas con ella. Véase los cursillos que se llevan a cabo en empresas para formar a los empleados sobre ciertos aspectos relacionados con la protección de datos o el uso de contraseñas seguras, así como los simulacros de phishing cada vez más comunes en diferentes industrias. No obstante, estos cursos son puntuales y no ofrecen una formación continua. Este proyecto trata de combatir este problema ofreciendo una plataforma con contenido dinámico que puede ser modificado por los administradores de la mismas y que, además incluye técnicas de gamificación para hacer la experiencia más atractiva y obtener mejores resultados en el aprendizaje.

6. Referencias

[1] S. Morg, Ed., "Cybercrime to cost the world $10.5 trillion annually by 2025," Cybercrime Magazine, 27-Mayo-2021. [En línea]. Disponible: https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/#:~:text=If%20it%20were%20measured%20as,after%20the%20U.S.%20and%20China. [Visitado: 22-Mayo-2022].

[2] Z. Muhammad, "95 percent of cybersecurity breaches are caused by human mistakes, World Economic Forum says," *Digital Information World*, 18-Enero-2022. [En línea]. Disponible: https://www.digitalinformationworld.com/2022/01/95-percent-of-cybersecurity-breaches.html. [Visitado: 22-Mayo-2022].

[3] FBI, "2021 internet crime report - ic3.gov," *IC3*, 2021. [En línea]. Disponible: https://www.ic3.gov/Media/PDF/AnnualReport/2021_IC3Report.pdf. [Visitado: 22-Mayo-2022].

CYBERHYGIENE: A GAMIFIED CYBERSECURITY LEARNING PLATFORM

Author: Prado Moreno, Álvaro
Supervisor: Ramamoorthi, Lokesh
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

This project consists of the development of a learning platform that teaches users about cybersecurity concepts using gamification techniques. The platform is deployed via a mobile application developed using Flutter. It is valid for both iOS and Android devices. An operating alpha version has been obtained during the development of the project. This version is available for download.

Keywords: Gamification, Cybersecurity, Learning Platform, Mobile application, Flutter, Firebase, Android, iOS

1. Introduction

Cybersecurity has taken a key position on society in a world that is relying increasingly on technology. If cybercrime is considered as another country, then it would be the third largest economy in the world after U.S and China.[1] World´s economies are experiencing large economic losses because of cyberattacks. For example, the U.S lost $7 billion to cyberattacks in 2021. [2] What is more is that 95% of the data breaches are due to human error.[3] This project tries to offer a solution to this problem.

2. Project definition

This project consists of the development of a learning platform. The platform teaches users about topics that are mainly related to cybersecurity. The objective is to make users acquire basic knowledge of this realm. Moreover, the project tries to make users develop daily habits, described in the project as cyberhygienic, that allow them to feel more secure when navigating in the cyberspace, reduce their digital footprint and maintain their private information secure. Several gamification strategies have been included to encourage users to complete the courses offered and consequently complete this objective. These techniques have been used because they have been proven effective in other cases.

3. Model/System/Tool description

Figure 1 shows the architecture of the platform that has been developed. Potential users of the platform are curious individuals that want to learn about cybersecurity and organizations that want to train their employees to get a cybersecurity educated workforce to avoid economic losses and data breaches caused by human error. Users would gain access to the platform through the mobile app developed for both Android and iOS devices.

The three main sections of the platform appear in the figure labeled as modules. The course module refers to the courses that are offered to the users in the platform. The courses are short quizzes with 5 to 10 questions each. The questions found on the quizzes

are Fill In The Blanks and Multiple Choice. 4 different categories can be found on the app: Web, Devices, Social Media and Information. Each course takes a specific position in its category and has different experience points and badges associated. Moreover, a course can be labeled as the featured one, so it grants two times its experience points when completed. It can also be labeled as the recommended course, so it appears on the Dashboard of the users.

The profile module refers to the information that is unique to the user and the different gamification techniques that have been included in the platform. Users can check the badges and the avatars that have earned in the app and the courses that have completed or saved. When a user completes a course successfully by answering to most of its questions correctly, he/she earns a badge. The user also earns experience points that make them go up in the level system. In case the user levels up, a new avatar is granted, and he/she can stablish it as the profile picture. Lastly the user can check his/her progress and modify the profile information.

The admin module refers to the section of the platform that is reserved to the platform administrators. The admin can create and delete courses, check the user's progress, and change the featured and the recommended course.

The data layer refers to the services used from Firebase in the development stage. Firebase Authentication has been used to create an authentication system and Cloud Firestore has been used to store data from the courses and the users in non-relational databases based on documents and collections.

Finally in the business layer, there are two different roles. The administrator role would be acquired by any organization interested on the platform so they can modify the content and customize it for their own purposes. On the other hand, the platform needs to have a relation with content curators that offer high quality cybersecurity content.

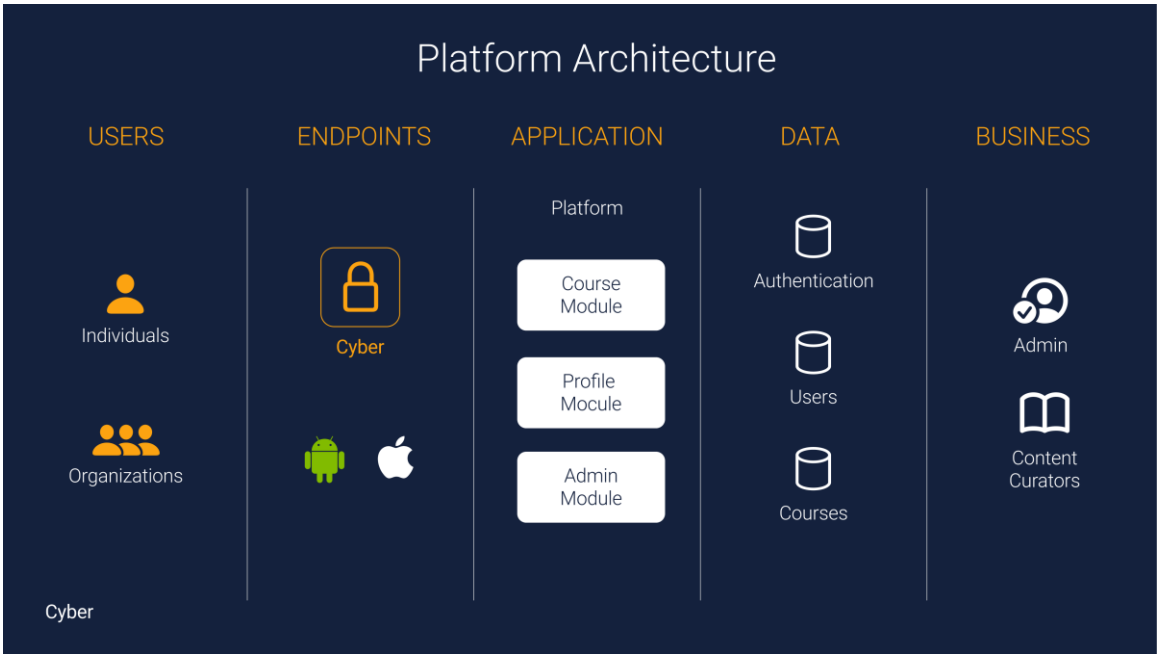


Figure 1 Platform Architecture Diagram

4. Outcomes

Figure 2 shows some of the most characteristic pages of the app. In the screenshots the reader can appreciate some of the main sections of the app. The first page shows the description of a course and the second one shows a Fill In The Blanks question that the user cand find when completing the course. The third page shows the main view of the profile and the fourth one shows the admin's dashboard. All the information shown regarding the courses and the user has been obtained from the databases.

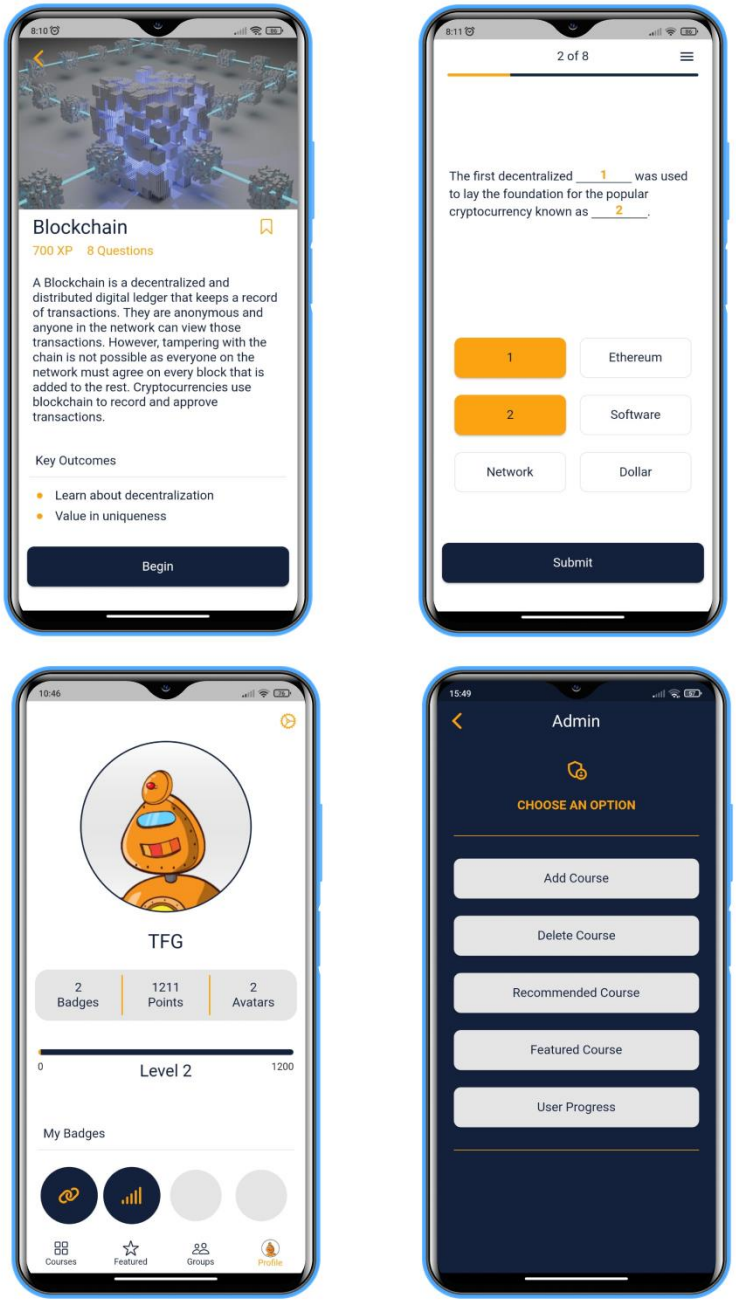


Figure 2 Pages of the main sections of the app

5. Conclusions

As stated in the introduction, nowadays cybersecurity plays a key role in the industry. However, not everyone has been educated about cybersecurity. Even though the world is becoming a more digitalized place and that we rely strongly on technology in our works and our private lives, there is not a universal education about cybersecurity. Excluding those majors and other programs related to technology and, consequently, related to cybersecurity. This is the reason why this project tries to offer engaging and quality cybersecurity content.

Various organizations are trying to educate their employees on cybersecurity because of its importance. Some examples are the courses that employees are required to take to learn about securing their data and having strong passwords, and the phishing drills that are becoming so common in the different industries. However, these courses and drills are just a onetime thing and do not offer a continuous way of learning as this project does. This project offers dynamic content that can be modified by the platform administrators, and it includes gamification techniques to make the experience an engaging and successful one.

6. References

[1] Vlassis, N.A.; Papakonstantinou, G.; Tsanakas, P. *Dynamic sensory probabilistic maps for mobile robot localization.* Source: Proceedings. 1998 IEEE / RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190) New York, NY, USA: IEEE, 1998.p.718-23 vol.2 of 3 vol. xliv+2010 pp. 11.

[2] Loeffler, B. “Cloud Computing: What is Infrastructure as a Service”, Microsoft Technet Magazine, October 211. https://technet.microsoft.com/en-us/magazine/hh509051.aspx

[3] Herrero Alcántara, T. “Big Data: ¿Moda u oportunidad de negocio para el emprendedor?”, Think Big, Octubre 2014. http://blogthinkbig.com/big-data-emprendedor/.

Índice de la memoria

Capítulo 1. Introducción 8

1.1 Introducción 8

1.2 Gamificación 10

1.3 Motivación del proyecto 11

Capítulo 2. Descripción de las Tecnologías 12

2.1 Fase de diseño 12

2.1.1 Proceso de diseño doble diamante 12

2.1.2 Miro 14

2.1.3 Figma 16

2.2 Fase de desarrollo 21

2.2.1 Flutter SDK 21

2.2.2 Dart 23

2.2.3 Android Studio 23

2.2.4 Firebase 25

2.2.5 Patrón de arquitectura 26

Capítulo 3. Estado de la Cuestión 28

3.1 Enseñanza en el ámbito de la ciberseguridad 28

3.1.1 Interland 28

3.1.2 Defend the crown 30

3.2 Enseñanza y Técnicas de gamificación 30

3.2.1 Duolingo 31

3.2.2 Kahoot 33

3.2.3 Quizlet 34

3.3 Tabla comparativa de características 37

Capítulo 4. Definición del Trabajo 39

4.1 Justificación 39

4.1.1 Efectividad de las técnicas de gamificación 40

4.1.2 Ciberataques: impacto económico 41

4.2 Objetivos 44

4.3 Metodología ... 46
4.4 Planificación y estimación económica ... 49
4.4.1 Planificación ... 49
4.4.2 Estimación económica ... 49

Capítulo 5. Sistema/Modelo Desarrollado ... 53
5.1 Diseño de la plataforma ... 53
5.1.1 Fase de descubrimiento ... 53
5.1.2 Fase de definición ... 59
5.1.3 Fase de desarrollo ... 66
5.1.4 Fase de entrega ... 71
5.2 Diagramas UML ... 82
5.2.1 diagrama de casos de uso ... 83
5.2.2 diagrama de clases ... 84
5.3 Fase de desarrollo ... 93
5.3.1 Autenticación ... 93
5.3.2 Registro ... 96
5.3.3 Flujo de los cursos ... 103
5.3.4 Curso Destacado ... 120
5.3.5 Perfil ... 122
5.3.6 Administrador ... 134

Capítulo 6. Análisis de Resultados ... 151
6.1 Capa de vista ... 151
6.2 Persistencia ... 152
6.3 Dispositivos Reales ... 153
6.3.1 Fase de desarrollo ... 154
6.3.2 Fase de entrega ... 155

Capítulo 7. Conclusiones y Trabajos Futuros ... 156

Capítulo 8. Bibliografía ... 158

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS ... 160

ANEXO II: encuesta sobre el uso de aplicaciones móvil ... 162

ANEXO III: Código del controlador "UserController"* *173

ANEXO IV: Código del controlador "ActiveUserController"*..... *181

ANEXO V: Código del controlador "CourseController"* *188

Índice de figuras

Figura 1 Esquema del modelo de Doble Diamante (Design Council) 13
Figura 2 Entorno de Trabajo de Miro 15
Figura 3 Diagrama de Flujo de Miro 16
Figura 4 Entorno de trabajo de Figma 17
Figura 5 Barra de herramientas de Figma 18
Figura 6 Funciones de diseño de Figma 19
Figura 7 Ventana de prototipado de Figma 20
Figura 8 Ventana de inspección de Figma 21
Figura 9 Ventana de desarrollo de Android Studio 24
Figura 10 Emulador de dispositivos de Android Studio 24
Figura 11 Esquema general de la base de datos del proyecto en Cloud Firestore 26
Figura 12 Diagrama del patrón de arquitectura de desarrollo MVC de la aplicación 27
Figura 13 Captura de pantalla de dinámica de juego de Interland (Interland) 30
Figura 14 Perfil de usuario en Duolingo (Duolingo) 32
Figura 15 Ejemplo de pregunta en Kahoot (Kahoot) 34
Figura 16 Ejemplo actividad combinar en Quizlet (Quizlet) 36
Figura 17 Media de ciberataques sufridos en 2021 ordenados por sector ("El estado de la Ciberseguridad en España. Post Pandemia: un camino inexplorado", Deloitte) 42
Figura 18 Porcentaje de los tipos de ciberataques sufridos en 2021 ("El estado de la Ciberseguridad en España. Post Pandemia: un camino inexplorado", Deloitte) 43
Figura 19 Diagrama de Gantt del proyecto 49
Figura 20 Gráfica de posicionamiento de mercado 57
Figura 21 Diagrama del Journey Map 65
Figura 22 Flujo de usuario I 69
Figura 23 Flujo de usuario II 70
Figura 24 Diagrama de información de la aplicación 71

Figura 25 Pantallas del prototipo de fidelidad baja I 73
Figura 26 Pantallas del prototipo de fidelidad baja II 73
Figura 27 Pantallas del prototipo de fidelidad baja III 74
Figura 28 Pantallas del prototipo de fidelidad baja IV 75
Figura 29 Pantallas del prototipo de fidelidad baja V 76
Figura 30 Pantallas del prototipo de fidelidad media I 77
Figura 31 Pantallas del prototipo de fidelidad media II 78
Figura 32 Pantallas del prototipo de fidelidad media III 78
Figura 33 Pantallas del prototipo de fidelidad media IV 79
Figura 34 Pantallas del prototipo de fidelidad media V 80
Figura 35 Pantallas del prototipo de fidelidad media VI 81
Figura 36 Pantallas del prototipo de fidelidad media VII 82
Figura 37 Diagrama de casos de uso I 83
Figura 38 Diagrama de casos de uso II 84
Figura 39 Diagrama de clases parte I 85
Figura 40 Diagrama de clases parte II 85
Figura 41 Splashscreen y login 93
Figura 42 Páginas para el registro de un nuevo usuario 96
Figura 43 Base de datos de autenticación 99
Figura 44 Colección de usuarios de Cloud Firestore 100
Figura 45 Fin del registro 100
Figura 46 Dashboard 107
Figura 47 Colección de cursos de Cloud Firestore 108
Figura 48 Colección del curso recomendado de Cloud Firestore 109
Figura 49 Cursos dentro de una categoría 110
Figura 50 Descripción del curso 111
Figura 51 Tipos de pregunta en un curso 114
Figura 52 Realimentación de una pregunta 117
Figura 53 Menú de opciones dentro del curso 118
Figura 54 Resumen del desempeño en el curso y progreso en la categoría 120

Figura 55 Colección para el curso destacado de Cloud Firestore 121
Figura 56 Curso destacado 122
Figura 57 Página principal del perfil 123
Figura 58 Insignias del usuario 129
Figura 59 Cursos del usuario 130
Figura 60 Avatares del usuario 132
Figura 61 Configuración del usuario 134
Figura 62 Panel de administrador 135
Figura 63 Comprobación de progreso de usuario 136
Figura 64 Cambiar curso recomendado 138
Figura 65 Borrar un curso 140
Figura 66 Añadir un curso parte I 142
Figura 67 Añadir un curso parte II 144
Figura 68 Añadir una pregunta 145
Figura 69 Añadir pregunta tipo opción múltiple 146
Figura 70 Añadir pregunta tipo rellenar los huecos 147
Figura 71 Emulador de dispositivos junto al IDE 152

Índice de tablas

Tabla 1 Comparativa de las características de los competidores..... 38
Tabla 2 Información incluida en el Lean UX Canvas 56
Tabla 3 Perfil de usuario..... 61
Tabla 4 Clasificación de características de la aplicación según el punto de vista del usuario 62
Tabla 5 Actitud del usuario en el escenario actual de aplicaciones de enseñanza I 64
Tabla 6 Actitud del usuario en el escenario actual de aplicaciones de enseñanza II..... 64
Tabla 7 Tabla MoSCoW..... 68

Capítulo 1. INTRODUCCIÓN

1.1 INTRODUCCIÓN

Cada vez más el uso y la dependencia de internet es más notable en nuestra sociedad. En nuestro día a día se utilizan diversas aplicaciones conectadas a Internet. Por un lado, se utilizan aplicaciones como Google Maps para encontrar la manera más rápida de llegar a un lugar concreto. Diversas redes sociales son utilizadas con distintos objetivos. Estas varían desde aplicaciones como LinkedIn para buscar empleo hasta aplicaciones como Facebook para saber del estado de familiares y amigos. También, en empleos e instituciones académicas se utilizan aplicaciones para facilitar el trabajo de empleados y estudiantes. Véase Moodle en el caso de ciertas universidades o Blackboard. Estos son unos de los muchos ejemplos de aplicaciones que un usuario medio podría llegar a utilizar en un único día.

A lo largo del día cuando los usuarios utilizan aplicaciones como las mencionadas, están dejando una huella digital, término conocido en inglés como “digital footprint”. Este término hace referencia al rastro de información que un usuario deja en Internet. El usuario puede estar dejando este rastro de manera consciente o inconsciente. Un ejemplo claro es el uso de las cookies en ciertas organizaciones. Las cookies guardan información sobre la sesión de un usuario en una web como los productos que ha comprado más veces o las secciones de una aplicación sobre las que muestra más interés. Además, muchas de estas organizaciones comparten esta información con terceros con objetivos comerciales [1].

En el año 2022 el número de usuarios que utiliza internet es de 4950 millones de personas lo cual representa un 62,5% de la población mundial [2]. Esto quiere decir que cada una de estas personas está dejando una huella digital. La información personal de los usuarios puede estar siendo utilizada para otros fines distintos a los que ellos tenían pensados. Esto supone un problema ya que se está atentando contra el derecho de privacidad de los usuarios los cuales desconocen el uso que se le está dando a su información personal, o bien porque

aceptaron las condiciones de uso de una aplicación sin leerlos o bien porque no se les ha notificado. La cantidad de usuarios utilizando Internet demuestra la magnitud del problema y la necesidad de implementar una solución.

Muchos usuarios tampoco hacen un uso adecuado de las redes sociales. Esto en muchas ocasiones, se debe a que el usuario desconoce las implicaciones que tiene el uso de ciertas redes sociales. No sabe realmente quién tiene acceso a esa información, dónde se almacena esa información o si es posible eliminarla de manera permanente. Por ejemplo, de acuerdo con la política de datos de ¨Instagram¨ (red social basada en compartir contenido multimedia en formato video o foto) el usuario acepta que la aplicación recoja información acerca de las personas con las que interactúa, la hora, la frecuencia y la duración de sus actividades en la aplicación, información sobre el dispositivo personal del usuario e incluso el número de tarjeta y otros datos bancarios en caso de realizar una compra dentro de la aplicación.

Por otro lado, los dispositivos que los usuarios utilizan para navegar en la red, así como los utilizados para guardar información como los USB o los discos duros son objeto de numerosos ataques. Para evitar que esto ocurra, debe de existir un conocimiento general sobre cómo proteger los dispositivos ante posibles ataques y como diferenciar un dispositivo seguro de uno inseguro. En muchas ocasiones, la manera de solucionar este problema es cambiando ciertos hábitos como no dejar un ordenador abierto o escribir una contraseña en un papel al alcance del dispositivo que se trata de proteger.

Por otro lado, además de los particulares también se debe de tener en cuenta que las empresas sufren al año miles de ciberataques y todavía no cuentan con una preparación suficiente en este ámbito. Las grandes empresas destinan una gran cantidad de su presupuesto en sus departamentos de ciberseguridad, así como en formación para sus empleados en este tema y en pagar dinero para solventar la situación generada por los ciberataques, es decir, en recuperar los sistemas dañados o ganar de nuevo acceso a la información encriptada mediante el pago a los atacantes. El 82% de las empresas tanto en España como a nivel global aumentaron su gasto en ciberseguridad en 2021. En este mismo año el 55% de las empresas no detuvo de manera eficaz estos ciberataques y además estos habían aumentado

un 32% respecto al año anterior. [3] En conclusión, estamos ante un problema con una clara tendencia al alza, de alta importancia por las pérdidas que supone a las empresas y frente al cual no están preparadas.

El uso de las redes sociales, la huella digital que se deja en Internet y la seguridad de los dispositivos y de la información son temas que deben de tenerse en cuenta en un mundo cada vez más digitalizado. Es por ello por lo que este proyecto trata de crear una plataforma de enseñanza sobre conceptos relacionados con la ciberseguridad. El objetivo del proyecto es crear una plataforma de enseñanza en la que el usuario aprenda conceptos sobre el uso de internet y la tecnología enfocados a la ciberseguridad. Esta plataforma se pone en práctica con una aplicación móvil y en cuanto al método utilizado para enseñar este consiste en someter al usuario a test sobre diversos temas en los que se le hacen preguntas para aprender de manera interactiva. El objetivo de este entorno de enseñanza es que el usuario entienda sobre los peligros presentes en Internet y que desarrolle costumbres ¨higiénicas¨ para evitar caer en ellos. Para ello la técnica empleada para garantizar que el aprendizaje es efectivo es la gamificación, explicada en el siguiente punto.

1.2 GAMIFICACIÓN

Un componente importante en esta plataforma de enseñanza es la gamificación, se pretende que el usuario interactúe lo más posible con la aplicación para poder completar el mayor número de test y así cambiar sus costumbres en el uso de internet por otras más seguras.

En cuanto a la gamificación, esta consiste en una técnica, estrategia y método basado en incorporar en elementos no jugables (como esta aplicación) características relacionadas con los juegos que los hacen atractivos. De esta manera se consigue un vínculo con el usuario de manera que se le incentiva a cambio de un comportamiento concreto.[4]

Con este objetivo, la plataforma incluye varios elementos característicos de los juegos para motivar este vínculo de incentivación con el usuario. Los tres elementos principales son: un mecanismo de niveles, uso de recompensas en forma emblemas y la posibilidad de obtener

avatares únicos si se avanza en el sistema de niveles. El usuario obtiene puntos de experiencia a la hora de completar los test para así obtener recompensas. A la hora de completar un curso el usuario obtendrá una insignia en caso de que la puntuación en dicha prueba sea mayor de un 50%. El usuario también obtiene fotos de perfil personalizadas cada vez que sube de nivel, la cual puede establecer como predeterminada en su perfil. Por otro lado, el o los administradores de la aplicación podrán seleccionar un módulo dentro de la aplicación como recomendado y otro como destacado. Esta funcionalidad se ha implementado para motivar al usuario a completar más cursos ya que el curso destacado incluye más puntos de experiencia y el recomendado es más visible.

1.3 MOTIVACIÓN DEL PROYECTO

Una vez se han mencionado todos los peligros que existen en Internet, así como la importancia que tiene la ciberseguridad para las empresas se procede a indicar la motivación de este proyecto.

Esta plataforma tiene como objetivo servir de ayuda tanto para particulares como para los empleados de las empresas para que tras completar los módulos presentes en la aplicación estos sean capaces de aprender sobre nuevos conceptos y consecuentemente desarrollar una serie de hábitos ciber saludables que reduzcan su vulnerabilidad en Internet. Esto supone que los usuarios de la plataforma no sufran de situaciones indeseables como robo de cuentas y contraseñas, suplantaciones de identidad o pérdida de información personal en manos de hackers. Por otro lado, supone un ahorro de dinero para las empresas que ya no deben destinar tanto presupuesto a cursos de preparación para los empleados. Además, estos cursos son puntuales mientras que la plataforma ofrece una manera de aprender continua mediante módulos actualizados e información proveniente de expertos en seguridad.

También es importante destacar que el contenido de la plataforma es dinámico y actualizable por aquellos que tengan derechos de administrador en la misma.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Respecto a este capítulo, cabe destacar que el desarrollo de este proyecto se ha llevado a cabo en dos fases claramente diferenciadas. Una primera fase de diseño y una segunda fase de desarrollo. Debido a la naturaleza del proyecto considero es conveniente separar la descripción de las tecnologías en función de la fase del proyecto para la cual se han empleado.

2.1 FASE DE DISEÑO

2.1.1 PROCESO DE DISEÑO DOBLE DIAMANTE

Para que el lector esté familiarizado y entienda en un futuro la explicación detallada de la fase de diseño, en este epígrafe se indica el método de diseño que se ha seguido. El método de diseño que se ha seguido se denomina Modelo de Doble Diamante.

El Modelo de Doble Diamante es un proceso de diseño de producto que aporta la perspectiva del consumidor de tal manera que al mismo tiempo que se diseña se trata de analizar si las necesidades del usuario están siendo cubiertas. Trata de crear un Producto Mínimo Viable (MVP) útil, sencillo de usar y de calidad en un tiempo limitado.[5]

Este modelo de diseño se estructura en 4 etapas consecutivas y lineales. Se denomina doble diamante ya que estas 4 etapas están englobadas en dos grandes secciones. Una primera sección creativa en la que se sitúan las etapas de Descubrimiento y Definición y una segunda sección (que hace referencia a ese segundo diamante) de prototipado. En esta última sección se encuentran las etapas de Desarrollo y Entrega. En la Figura 1 se muestra un diagrama que ilustra el proceso de diseño descrito.

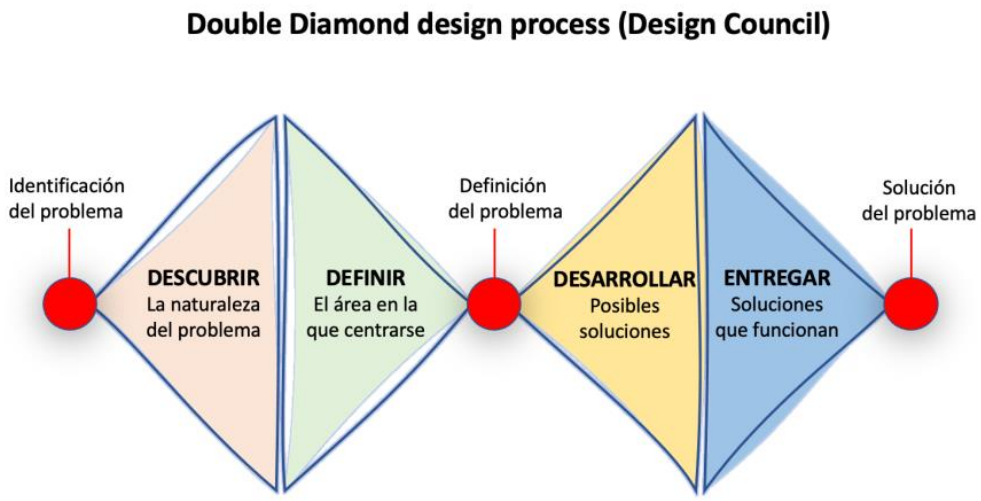


Figura 1 Esquema del modelo de Doble Diamante (Design Council)

A continuación, se indica en qué consiste cada una de las etapas.

- Fase de Descubrimiento: en esta fase se investiga acerca del problema que se quiere resolver. Se proponen varias ideas de cómo solucionarlo y se analizan cuáles serían los resultados tanto para los potenciales usuarios como para el negocio/idea. Para ello se lleva a cabo una interacción con el usuario. Se busca obtener información de este mediante investigación primaria (entrevistas y formularios) como investigación secundaria realizada por terceros. Además, para entender el contexto del problema se realiza un breve estudio de mercado para analizar qué competidores existen, qué ofrecen y cómo se posicionan en el mismo.
- Fase de Definición: en esta fase se reestructura toda la información recopilada en la fase anterior y se trata de sintetizar. Para ello, se organizan las conclusiones de la investigación de los usuarios en puntos que aportan valor al usuario o por el contrario se lo restan. Más adelante, se definen uno o varios modelos de un perfil de usuario arquetipo. Por último, se analizan las funcionalidades que se podrían incluir en el producto y cómo se relacionan estas con el usuario.
- Fase de desarrollo: en esta etapa se define la idea de producto. Inicialmente se realiza una tormenta de ideas sobre las funcionalidades que la aplicación puede implementar

para más adelantes separarlas por secciones en una tabla "MoSCoW (Must, Should, Could, Wont)". Esta tabla indica que funcionales deben implementarse, cuáles deberían o podrían implementarse y cuales no deberían incluirse. Más adelante, se describe el Producto Mínimo Viable que incluye aquellas funcionales que definen una versión básica de la idea. En esta fase también se incluye un diagrama de flujo sobre la experiencia básica del usuario en la aplicación, así como un diagrama mostrando la arquitectura básica de la información en la aplicación.

- Fase de Entrega: esta etapa consiste dar forma al producto mediante un prototipo. En esta fase se crean tres prototipos sirviendo cada uno de ellos como punto de partida para el siguiente. El prototipo de fidelidad baja se ha realizado en papel para este proyecto y en él se ha plasmado una idea general de las secciones que se querían incluir en la aplicación. Más adelante, este prototipo ha sido mejorado en sus versiones de fidelidad media y alta mediante la aplicación de diseño Figma, presentada en este aparatado de descripción de tecnologías.

2.1.2 MIRO

Miro es una herramienta web que te permite crear y guardar pizarras en blanco. La herramienta te ofrece la posibilidad de crear tablas comparativas, gráficos y demás diseños de manera rápida. Además, Miro ofrece una amplia biblioteca de plantillas que han sido de gran utilidad para el proyecto. Esta herramienta se ha utilizado durante toda la fase de diseño de la aplicación para el desarrollo de las cuatro etapas mencionadas del proceso de diseño Doble Diamante. Gracias a Miro se han podido separar las cuatro secciones en un archivo de la herramienta y se han utilizado plantillas como la tabla de posicionamiento de mercado para cumplir con los diferentes pasos del método mencionado.

En la Figura 2 se puede observar cual es el entorno de trabajo que ofrece la herramienta.

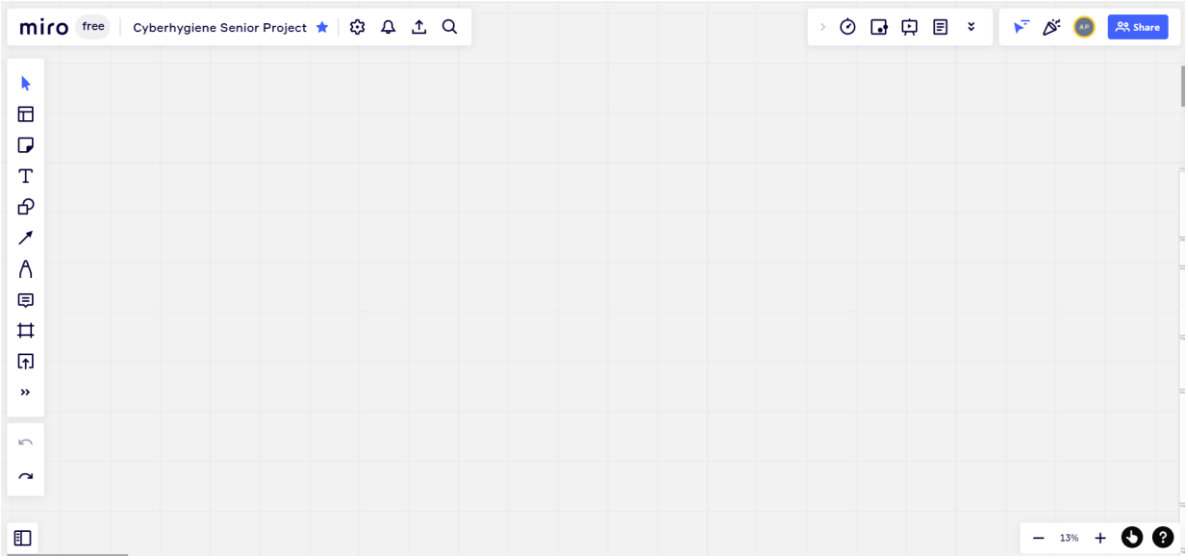


Figura 2 Entorno de Trabajo de Miro

A la izquierda pueden observarse las principales funcionalidades que ofrece Miro. Entre ellas cabe destacar, la posibilidad de crear marcos de distinto tamaño, crear áreas de texto y crear notas estilo post-it para pegar en la pantalla.

Por otro lado, en la barra de herramientas situada en la parte superior izquierda, se indica el nombre del proyecto en el que se está trabajando y se da la posibilidad de exportarlo en diversos formatos, entre ellos .pdf.

Por último, en la parte superior derecha de la pantalla se puede observar como también ofrece opciones de presentación, así como la opción de compartir el proyecto con un equipo para trabajar de manera simultánea.

En la Figura 3 se muestra la creación de un diagrama de flujo mediante la biblioteca de plantillas. Este mismo diagrama se ha utilizado en la fase de diseño del proyecto como punto de partida para definir el flujo de usuario en la aplicación.

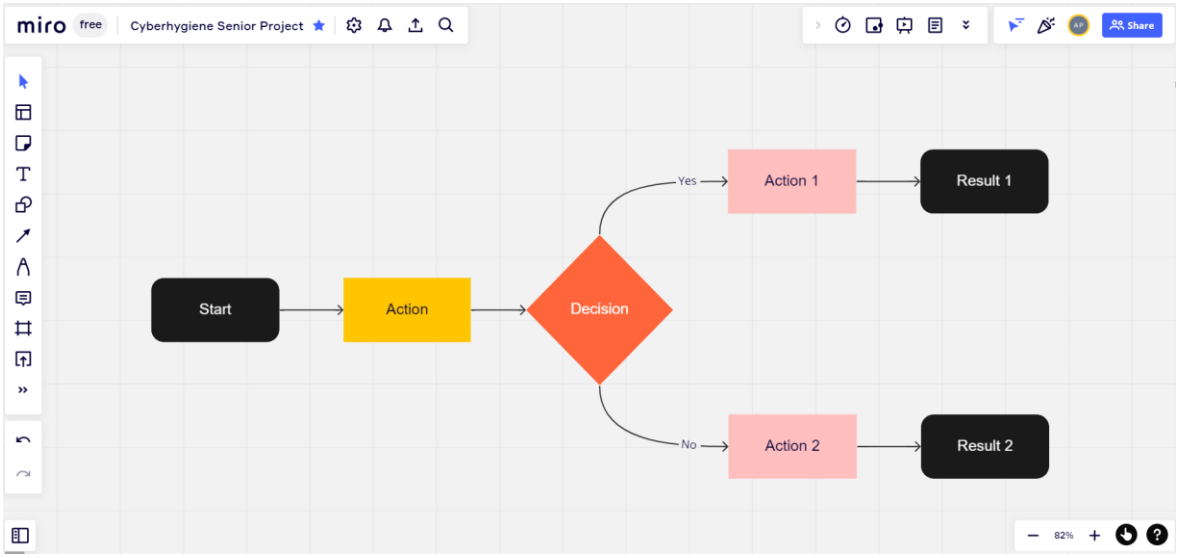


Figura 3 Diagrama de Flujo de Miro

2.1.3 FIGMA

Esta herramienta tiene una versión web y también permite la opción de instalarse como aplicación en el ordenador. Figma consiste en un editor de gráficos vectoriales y una herramienta de prototipado. Además, Figma es multiplataforma ya que opera en Windows, Mac y Linux. [6].

Figma tiene una versión gratuita con almacenamiento ilimitado en la nube y con la posibilidad de crear hasta 3 proyectos. En caso de demostrar pertenencia a una institución educativa y acreditar qué se es estudiante de esta, Figma ofrece un tipo de suscripción con más ventajas. Esta suscripción es la que se ha utilizado para el proyecto.

La herramienta se ha utilizado en el proyecto en la etapa final de la fase de diseño para poder crear un prototipo de fidelidad media y otro de fidelidad alta de la aplicación móvil. Figma es un software clave para el proyecto ya que permite obtener una visión clara de cuál va a ser la experiencia de usuario y la interfaz de la aplicación antes de comenzar con el desarrollo.

Este proyecto trata de crear una plataforma de aprendizaje basada en la gamificación entregada por medio de una aplicación móvil. De tal manera que la experiencia de usuario

en la aplicación o la interfaz son aspectos clave para lograr que el usuario esté cómodo utilizado la aplicación y las técnicas de gamificación no se vean mermadas por este aspecto.

En la Figura 4 se puede observar cómo es el entorno de trabajo de Figma.

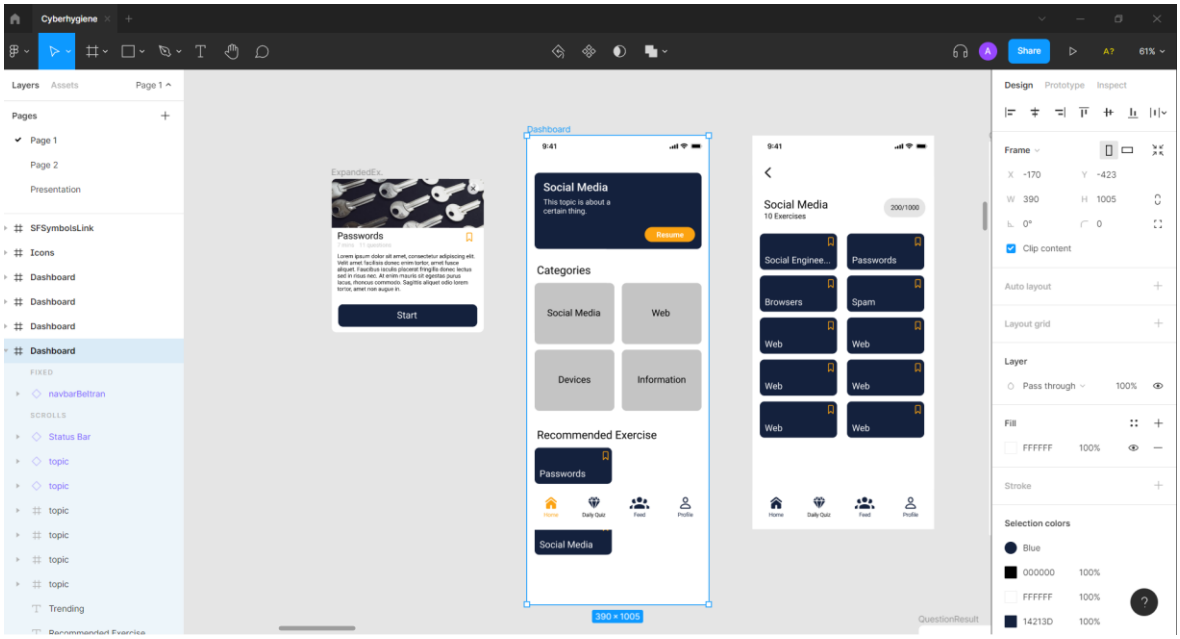


Figura 4 Entorno de trabajo de Figma

En la parte izquierda de la aplicación se puede apreciar una barra de herramientas que muestra las distintas páginas del proyecto. La mencionada barra de herramientas se muestra en la Figura 5 y, a continuación, se describe más detalladamente.

Como se puede observar, los proyectos en Figma constan de varias páginas. En esta sección se indica en qué página del proyecto se encuentra el usuario y además se indican los distintos componentes presentes en la misma. Un componente es la unidad básica de diseño de la herramienta. Un componente a su vez puede estar formado por varios componentes. Es decir, si se crea un marco y dentro del marco se coloca un texto, el texto forma parte del marco. Esto permite identificar de manera sencilla los principales componentes de la página que, en el caso de este proyecto son en su mayoría las diferentes páginas de la aplicación móvil.

También se puede observar que hay una pestaña denominada en inglés "Assets". En esta pestaña se pueden encontrar aquellos componentes a los que se les ha dado esta funcionalidad. Calificar a un componente de "Asset" implica que dicho componente queda definido y se puede utilizar en diferentes lugares del proyecto. Además, en un "Asset" se pueden definir diferentes estilos del componente para utilizar. Esto ha servido de gran utilidad para el proyecto. Por ejemplo, una vez se ha definido la curvatura y dimensiones de un botón en la aplicación, este se puede calificar como "Asset" y crear distintos estilos del botón cambiando únicamente los colores y el Texto mostrado en el mismo para posteriormente utilizar cada estilo en la ventana que corresponda.

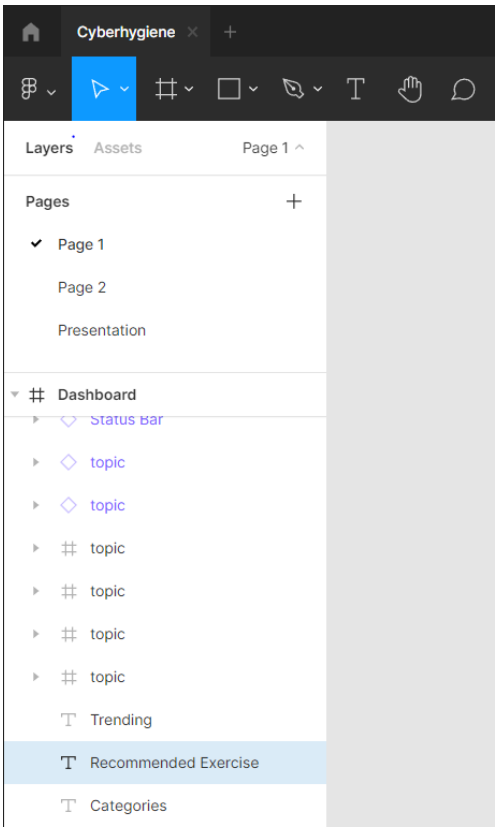


Figura 5 Barra de herramientas de Figma

En cuanto a la Figura 6, en ella se puede observar cual es el cuadro de opciones que se presenta cuando se tiene un componente seleccionado. En este caso el componente seleccionado es el marco que engloba toda la pantalla del "Dashboard" y la opción seleccionada en el cuadro de opciones es la de diseño. La ventana de diseño permite

seleccionar varias características del componente. Entre ellas, el color y las dimensiones. Cabe destacar que las opciones de diseño presentes varían en función del componente seleccionado. Por ejemplo, en caso de seleccionar un texto, también se ofrece la opción de elegir la familia y tamaño de fuente.

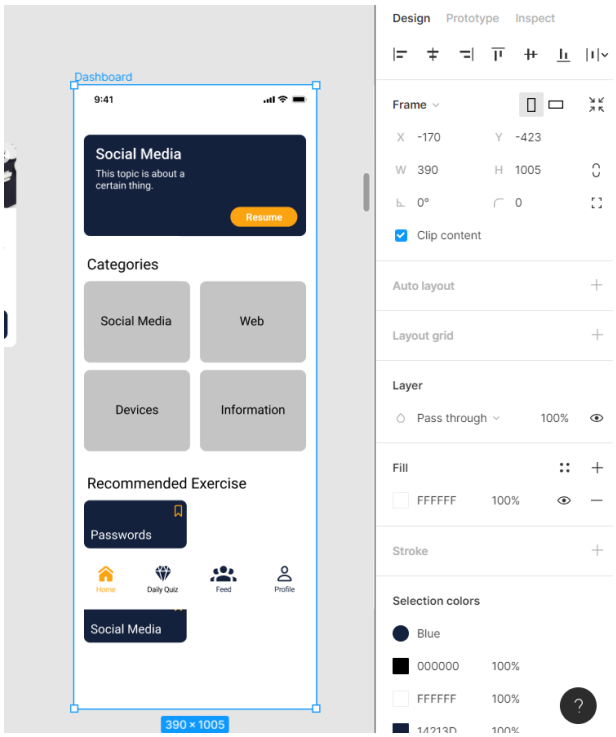


Figura 6 Funciones de diseño de Figma

Otra de las ventanas presentes en el cuadro de dialogo es la ventana de prototipado. En esta ventana se puede indicar qué ocurre cuando se lleva a cabo una acción determinada sobre un componente. En la Figura 7 se muestra la ventana de prototipado para el botón de continuar amarillo mostrado. Como puede observarse, en esta ventana se define que a la hora de pulsar dicho botón debe de aparecer el marco indicado por la flecha. Además, se pueden definir diversos atributos como si se desea abrir el marco como un cuadro de dialogo superpuesto a la ventana o navegar a este marco como si se tratase de una ventana nueva.

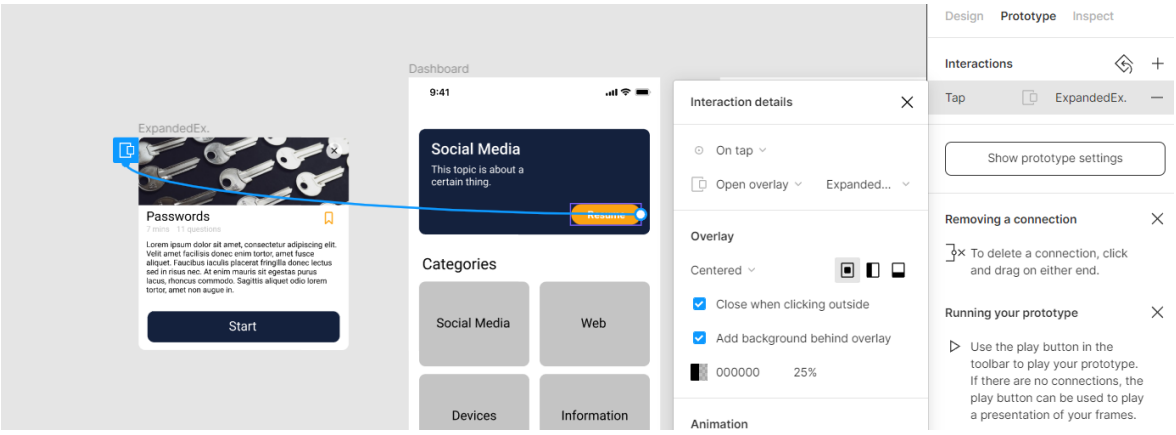


Figura 7 Ventana de prototipado de Figma

La última de las ventanas a la que se puede acceder desde un componente es la ventana de inspección. En esta ventana se muestra la información detallada del componente lo cual se ha utilizado en la aplicación para poder programar la interfaz de usuario de manera más sencilla. En la Figura 8 se muestra las características del contenedor “Social Media”. Además, Figma indica el código que se debe utilizar para programar la interfaz de dicho componente en CSS, Android e iOS. Debido a que el lenguaje de programación utilizado para la aplicación de este proyecto es Dart no se le ha dado uso a esta última funcionalidad mencionada.

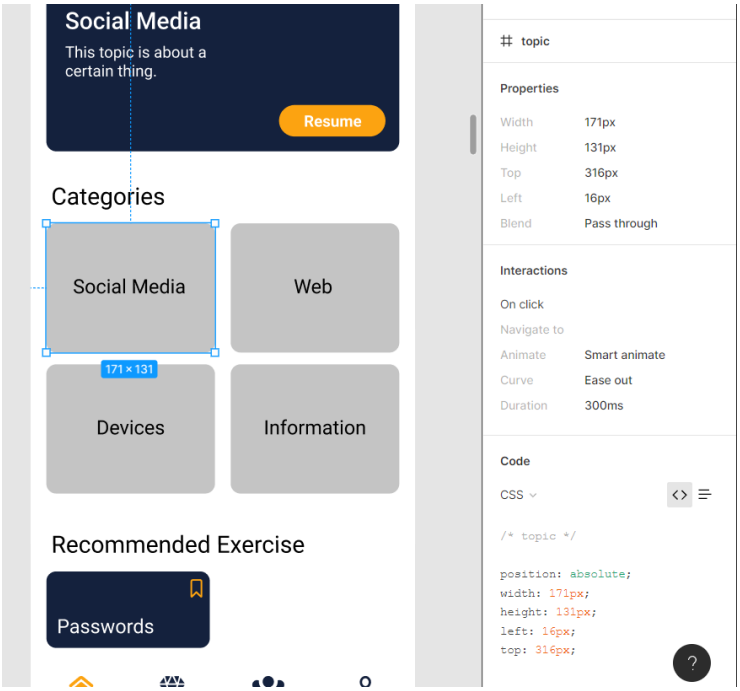


Figura 8 Ventana de inspección de Figma

2.2 *FASE DE DESARROLLO*

2.2.1 FLUTTER SDK

Para el desarrollo de la aplicación móvil del proyecto se ha utilizado Flutter como SDK. Flutter es un framework de código abierto desarrollado por Google que permite desarrollar aplicaciones nativas con un único código base.[7] Esto quiere decir que a partir de un único código se puede generar una aplicación con versión Web, iOS y Android. Se ha escogido este framework para el desarrollo del proyecto de tal manera que se pueda obtener una aplicación móvil que opere en Android y en iOS para así llegar a un mayor número de usuarios.

Flutter es un SDK (Software Development Kit) y un Framework. Al ser un SDK consta de herramientas que traducen el código fuente a código máquina para que funcione en las plataformas mencionadas. Por otro lado, al ser un framework ofrece una librería de Interfaz

de Usuario con una gran cantidad de componentes reutilizables como botones, cajas de texto y menús de navegación reutilizables por el desarrollador para sus propios objetivos.[7]

Algunas de las ventajas que ofrece Flutter además de la posibilidad de ofrecer a partir de un único código una aplicación iOS y Android, son una eficiencia mayor que otros frameworks al no necesitar comunicarse con componentes nativos y la funcionalidad de recarga en caliente ("hot reload"). Esta funcionalidad consiste en poder ver los cambios realizados en el código sin la necesidad de volver a iniciar la aplicación.[7]

Flutter como framework ofrece al desarrollador una clase denominada "Widget". Los widgets se caracterizan por ser la unidad básica de código utilizada en el desarrollo de la parte gráfica de la aplicación. Se trata de clases que pueden tener sus propios atributos y constan de un constructor. Lo que caracteriza a los widgets es la necesidad de implementar el método "build". Este método se ejecuta cada vez que se crea un widget en la aplicación y describe como este se muestra por pantalla.

La aplicación está formada por numerosos widgets que a su vez están formados por otros widgets. De tal manera que cuando se navega hacia una pantalla lo que ocurre en el código es que se está creando un widget y consecuentemente este está ejecutando su método ¨build¨ en el cual normalmente se crean otros widgets que actúan de manera semejante. Esto acaba generando lo que se denomina en Flutter como "Widget Tree", es decir, un árbol de jerarquía en el que se encuentran los diferentes widgets. Siendo el nodo origen la aplicación.

Los widgets se separan en dos grandes grupos: "Stateless Widgets" y "Stateful Widgets". Los segundos tienen la particularidad de tener otra clase asociada que es su estado. De tal manera que si un atributo del estado del widget cambia se puede ordenar al mismo que vuelva a ejecutar su método "build" para mostrar dichos cambios por pantalla o ejecutar ciertas líneas de código. Para poder llevar a cabo esta acción se utiliza el método "setState()".

Se ha llevado una descripción a alto nivel de algunas funcionalidades que ofrece Flutter como Framework para que el lector esté familiarizado y comprenda mejor el apartado de desarrollo de este proyecto.

2.2.2 DART

El lenguaje de programación que se utiliza en Flutter es Dart. Este lenguaje es un lenguaje orientado a objetos con análisis estático de tipo.

Una de las características más especiales de este lenguaje es el término "Null Safety". Este término hace referencia al hecho de que, en las últimas versiones de Dart, las variables no pueden contener el valor *Null* a no ser que se indique de manera explícita a la hora de declarar la variable.

En las siguientes líneas de código se muestra como declarar una variable que acepta *Null* como valor (nullable).

```
int aNonNullableInt = 0
int? aNullableInt = null
```

Esta característica de Dart hace que el lenguaje cuente con los siguientes operadores.

- Operador !: este operador se utiliza para indicar que se esta accediendo al valor de una variable nullable sabiendo que su valor no es null en el momento de acceso.
- Operador ??: este operador se utiliza para asignar un valor a una variable en caso de que esta tenga valor *Null*.

2.2.3 ANDROID STUDIO

Para el desarrollo de la aplicación móvil se ha utilizado Android Studio (A.S.) como IDE. Se ha elegido este IDE ya que soporta el SDK Flutter mencionado anteriormente. Otro de los motivos por los que se ha elegido A.S. como IDE es debido a que ofrece la posibilidad de añadir un emulador de un móvil Android.

En la Figura 9 se puede observar la vista principal del IDE. A la izquierda de la figura marcado con un 1 el lector puede observar la estructura de paquetes del proyecto. En la parte superior derecha marcada con el número 2 se puede observar una barra de herramientas que permite al desarrollador elegir el dispositivo sobre el que se corre o depura la aplicación.

Desde esta barra de herramientas también aparece la opción de llevar a cabo la recarga rápida mencionada anteriormente.

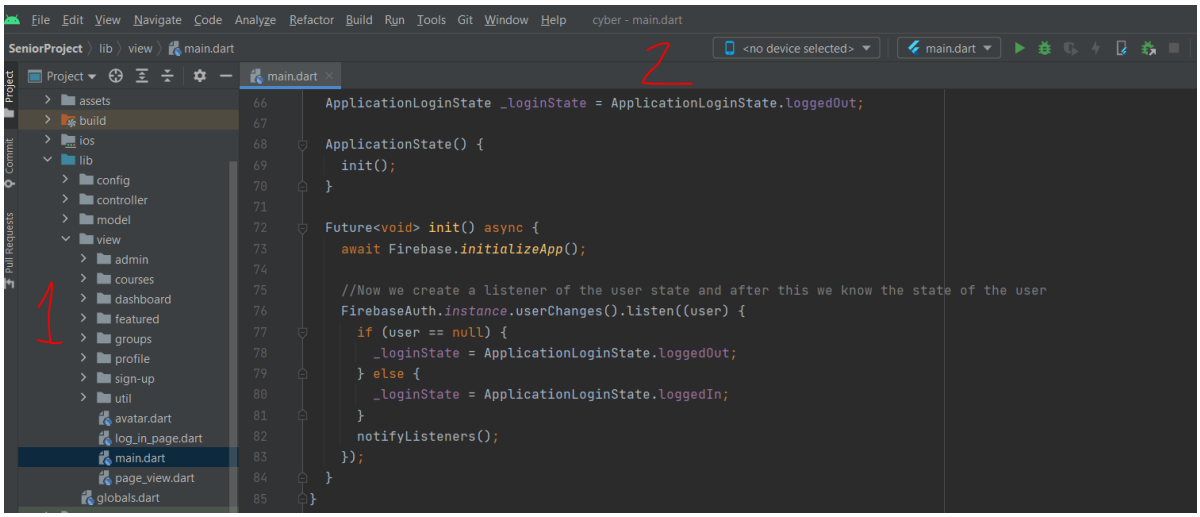


Figura 9 Ventana de desarrollo de Android Studio

A continuación, se muestra en la Figura 10 el emulador del dispositivo Android que se ha utilizado para el desarrollo del proyecto.

Figura 10 Emulador de dispositivos de Android Studio

2.2.4 FIREBASE

Firebase es una plataforma en la nube ofrecida por Google para el desarrollo de aplicaciones web y móvil.

Una de las herramientas que ofrece Firebase es su base de datos en tiempo real llamada Cloud Firestore. En este proyecto se ha utilizado esta base de datos para persistir la información relativa a los cursos, así como la información de los usuarios de la aplicación. Esta base de datos es No SQL, lo cual quiere decir que es una base de datos no relacional y almacena la información en formato JSON. La información dentro de Firebase se organiza en dos unidades básicas de almacenamiento: documentos y colecciones.

Cada documento en Firestore contiene un conjunto de pares clave-valor denominadas campos. Los valores que se pueden almacenar en un documento son los siguientes tipos de datos: String, number, boolean, null, timestamp, reference, array, map y geopoint. También existe la posibilidad de que uno de los campos del documento sea un objeto anidado, en este caso el objeto se almacena como mapa en formato JSON. [8]

Los documentos se almacenan en colecciones y cada documento debe de tener un identificador único el cual puede ser creado por la propia base de datos de manera automática como se ha hecho en este proyecto. Los documentos además de contener campos también pueden tener subcolecciones de otros documentos asociadas.

En la Figura 11 se muestra el esquema general de la base de datos que se ha utilizado para el proyecto. En ella se puede observar el documento de un curso. Este documento pertenece a la colección “courseCollection” y en la figura se pueden obsevar sus diferentes campos y la subcolección “questions” asociada al documento que contiene las preguntas del curso.

La otra herramienta que ofrece Firebase es Firebase Authentication. Se trata de un sistema de autenticación que permite el registro de usuarios y su identificación. Ofrece la posibilidad de utilizar cuentas en aplicaciones de terceros como Google o Facebook o crear una cuenta otorgando un email y una contraseña. Este último método es el que se ha utilizado en el proyecto.

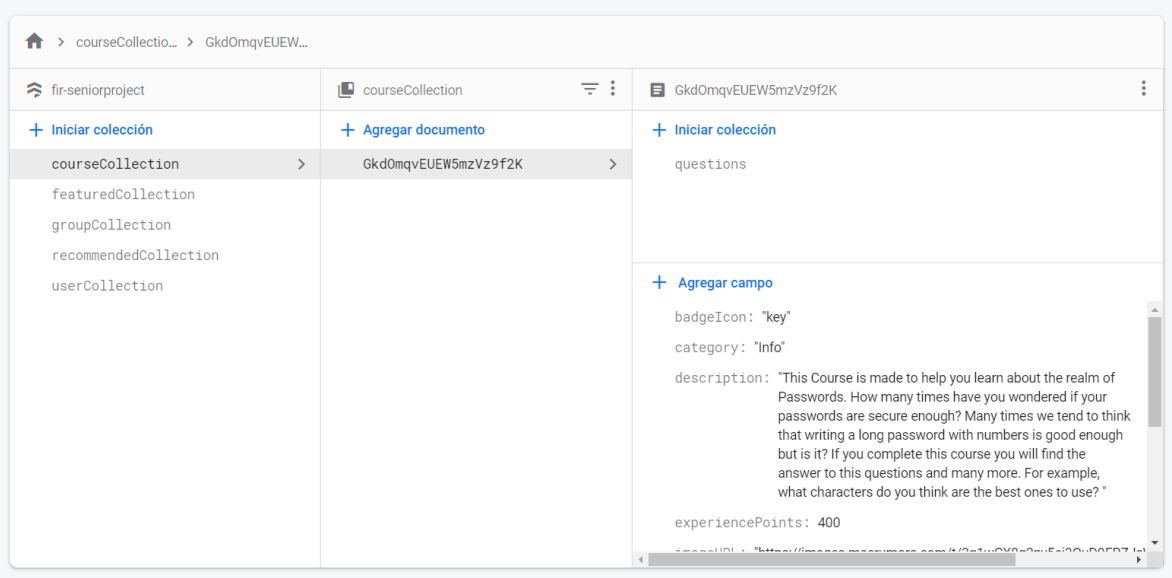


Figura 11 Esquema general de la base de datos del proyecto en Cloud Firestore

2.2.5 PATRÓN DE ARQUITECTURA

Para el diseño de la aplicación móvil se ha seguido el patrón de arquitectura conocido como MVC (Modelo, Vista, Controlador). Este patrón de arquitectura es una manera de separar la interfaz gráfica de la aplicación (Vista) del modelo de datos (Modelo) por medio del Controlador.

La capa de Modelo es aquella en la cual se guardan los datos del dominio, es decir, la información del sistema que se está desarrollando. Además, en esta capa se definen las funciones de transferencia de datos que permiten intercambiar la información con servicios de terceros, en este caso con Firestore. [9]

La capa de vista contiene todas las ventanas de la aplicación, es decir, la parte de interfaz de usuario. Es la manera que tiene el desarrollador de mostrar la información del modelo al usuario.[9] El usuario tiene también la posibilidad de modificar esta información mediante la interfaz.

La capa de controlador es la capa que actúa de interconexión entre el usuario y el sistema, es decir, en esta capa se define la lógica para conectar la capa de modelo con la de vista. Se

trata de un coordinador general del sistema.[9] En este proyecto en la capa de controlador se han definido las funciones para la comunicación con la base de datos y las funciones para la actualización de usuarios dentro de la aplicación.

En la arquitectura se podría haber identificado una capa de persistencia. No obstante, se ha decidido englobar todas las operaciones de coordinación de datos en la capa de controlador.

En la Figura 12 puede observarse un esquema simplificado de la arquitectura de aplicación. El esquema trata de ser auto explicativo, pero a continuación se indican algunos detalles que pueden pasar desapercibidos por el lector. El administrador de la aplicación puede acceder a una parte de la capa de Vista a la cual el usuario normal no tiene acceso. El controlador denominado en el esquema como ActiveUserController se utiliza para gestionar la información del usuario activo en la aplicación. De tal manera que tiene conexión con la vista y el modelo. No obstante, para persistir esos cambios en la base de datos lo hace a través del controlador UserController, sin tener acceso a la base de datos.

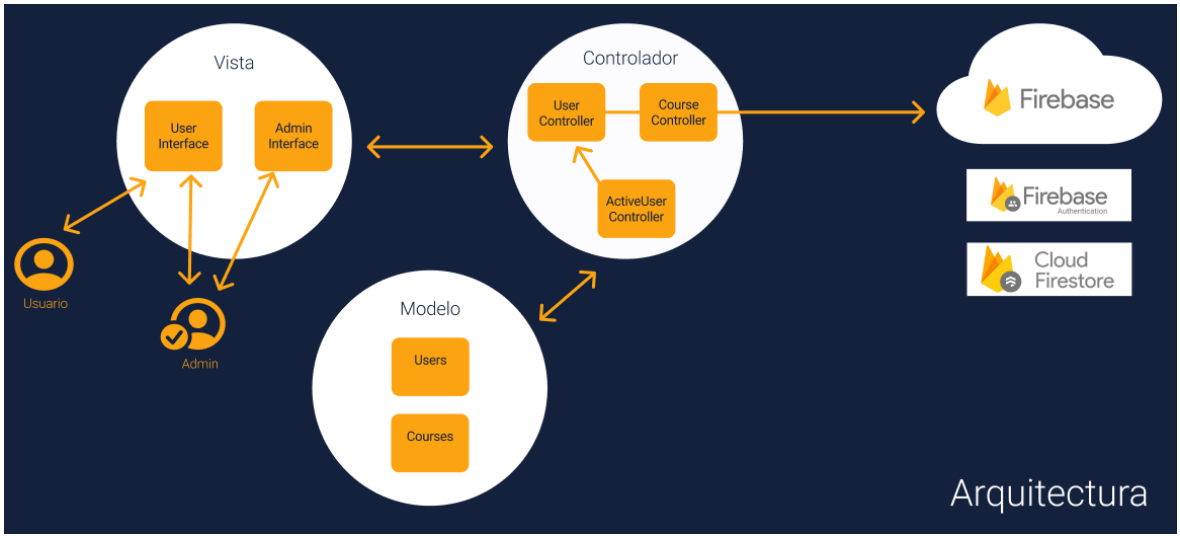


Figura 12 Diagrama del patrón de arquitectura de desarrollo MVC de la aplicación

Capítulo 3. ESTADO DE LA CUESTIÓN

Este capítulo cuenta con dos secciones diferenciadas. Por un lado, se analizan aquellas aplicaciones existentes que insertan técnicas de gamificación y, por otro lado, aquellas aplicaciones que enseñan sobre conceptos de ciberseguridad. Cada tipo de competidor se englobará en uno de estos grupos.

3.1 ENSEÑANZA EN EL ÁMBITO DE LA CIBERSEGURIDAD

3.1.1 INTERLAND

Interland es una aplicación web desarrollada por Google para enseñar sobre conceptos de ciberseguridad a un público mayormente joven. Consiste en un juego con 4 mundos distintos. Cada mundo en Interland tiene una dinámica de juego distinta y trata de enseñar al usuario sobre un aspecto particular de la ciberseguridad.

Interland trata de enseñar a los niños 5 fundamentos básicos:

1. Compartir con cuidado: Se enseña a los usuarios a saber qué compartir con conocidos y desconocidos en la red. Para ello se les indica que no deben compartir datos personales ni de familiares o amigos y que deben tratar las conversaciones en Internet como conversaciones cara a cara.
2. Mantenerse alerta: se muestra a los niños que en Internet no todo es siempre lo que parece para que estos sean capaces de diferenciar entre lo que es real y es falso.
3. Mantenerse seguro: se trata de que los niños entiendan que la seguridad y la privacidad son muy importantes en línea para que no dañen sus relaciones personales ni su reputación.
4. Ser amable: Internet es un medio poderoso para difundir comentarios tanto positivos como negativos. Se trata de que los niños tengan siempre en cuenta el concepto de tratar a los demás como les gustaría que les tratasen a ellos.

5. Ser valiente: este punto hace referencia a conseguir que los niños no tengan miedo en preguntar o informar a un adulto sobre situaciones en las que se puedan ver involucrados en la red.

Algunas de las características más relevantes de Interland se indican a continuación:

- Como ya se ha indicado Interland se centra en un público infantil.
- Solo tiene una única versión de la aplicación y esta es gratuita y en versión web.
- Implementa un sistema de puntos. Se van otorgando puntos a medida que se completan de manera correcta distintas acciones. No obstante, no se incluye un sistema de niveles
- También incluye un sistema de recompensa basado en trofeos que son otorgados al usuario cuando cumple un hito determinado.
- El contenido ofrecido por la aplicación es estático. Es decir, cada uno de los cuatro mundos con los que cuenta Interland implementan una dinámica de juego distinta en la que se muestra a los niños contenido sobre ciberseguridad, no obstante, este contenido es siempre el mismo.

En la Figura 13 el lector puede observar cómo es la interfaz de usuario del juego Interland, así como una de las preguntas que se hacen al usuario para que contextualice lo explicado en este apartado.

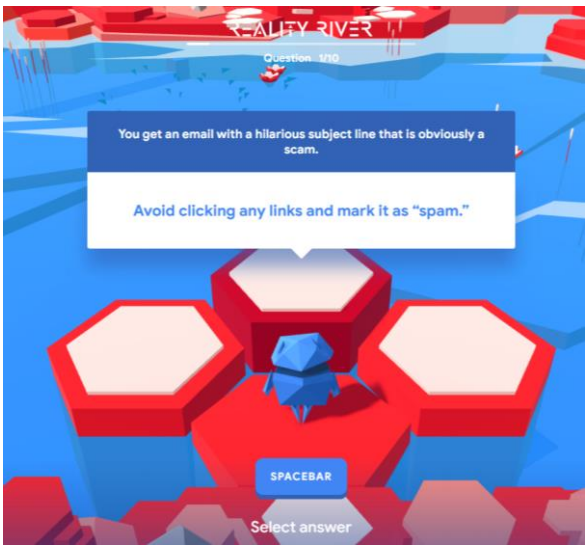


Figura 13 Captura de pantalla de dinámica de juego de Interland (Interland)

3.1.2 DEFEND THE CROWN

Defend the Crown se trata de un juego desarrollado en forma de aplicación móvil por petición de la agencia de Ciberseguridad y Seguridad de Infraestructuras de Estados Unidos. Esta agencia se asoció con el Laboratorio Nacional del Pacífico Noroeste para el desarrollo de la aplicación.

El objetivo por el que se ha desarrollado este juego ha sido el de enseñar a trabajadores de la agencia, así como a otros trabajadores federales y demás ciudadanos acerca de conceptos de ciberseguridad. Algunos de los temas sobre los que se educa en el juego son ciberataques, seguridad en IOT y seguridad en redes de computadores.

3.2 ENSEÑANZA Y TÉCNICAS DE GAMIFICACIÓN

Por otro lado, existe un conjunto de aplicaciones que implementan técnicas de gamificación para enseñar a sus usuarios sobre diversos contenidos. No obstante, estas aplicaciones no se centran en la ciberseguridad. Algunas de estas aplicaciones se describen a continuación con objetivo de que el lector entienda cómo se implementan las técnicas de gamificación en otros ámbitos

3.2.1 DUOLINGO

Duolingo es una aplicación con soporte web y móvil centrada en la enseñanza de idiomas implementando técnicas de gamificación.

De acuerdo con un estudio realizado por el propio Duolingo enfocado en las habilidades de compresión oral y escrita, los usuarios que alcanzan la sección 5 de los cursos de francés y español que ofrece la aplicación (nivel principiante) logran un desempeño similar en estas habilidades que estudiantes de universidad que han realizado 5 semestres de clases del mismo idioma.[10]

Por otro lado, en otro estudio realizado por la compañía enfocado en las habilidades conversacionales. 256 usuarios que llegaron hasta la mencionada sección 5 de los cursos de español y francés llegaron a ser capaces de adquirir habilidades conversacionales de nivel A2 respecto al MCER (Marco Común Europeo de Referencia) lo cual quiere decir que eran capaces de mantener una conversación sobre alguna situación de la vida cotidiana.[10]

Estos dos estudios muestran la efectividad que tienen las técnicas de gamificación a la hora de enseñar. Algunas de las técnicas de gamificación que Duolingo utiliza para enseñar son las siguientes. Esta aplicación implementa un sistema de puntos de experiencia que se van acumulando a medida que el usuario completa unidades. Cuando el usuario llega a un límite de puntos se le otorga un premio en forma de ´lingots´ que es la moneda utilizada en la tienda de la aplicación.

Por otro lado, las unidades se estructuran en secciones. Cuando el usuario completa una unidad con todas sus lecciones, gana una insignia del curso y se le permite pasar al siguiente, avanzando de esta manera en la sección. Este sistema de puntos, recompensas y cursos organizados en secciones, que dan al usuario la sensación de estar avanzando en un progreso lineal, son técnicas de gamificación que tratan de asemejar la experiencia del aprendizaje a la de un juego con el objetivo de que el usuario se sienta motivado y complete el mayor número de cursos posible.

En la Figura 14 se muestra el perfil de usuario de la aplicación web donde se puede observar la recompensa obtenida al haber llegado a los puntos de experiencia marcados, así como el conjunto de cursos que forman una sección.

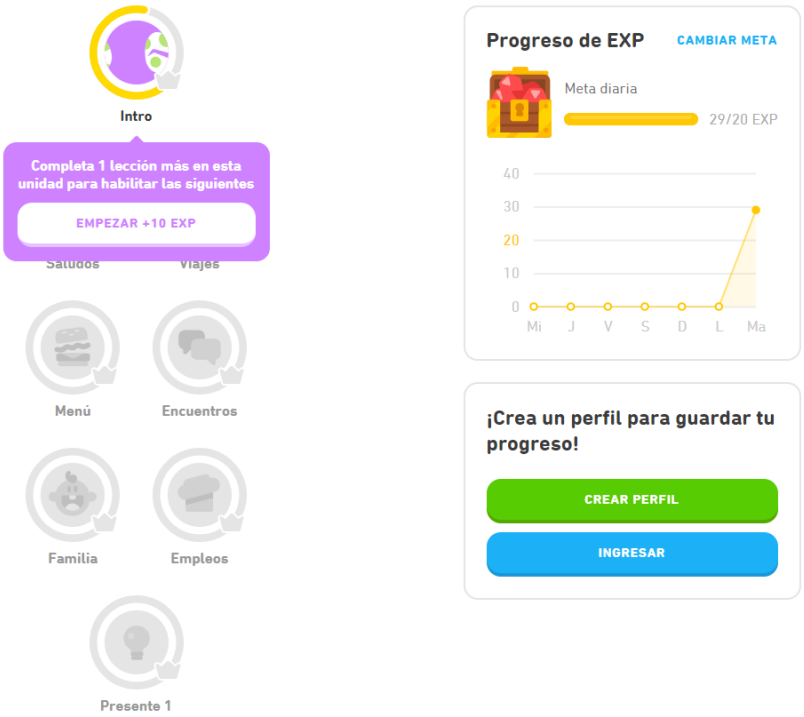


Figura 14 Perfil de usuario en Duolingo (Duolingo)

Algunas de las características que incluye esta aplicación son las siguientes:

- Sistema de puntos basado en EXP
- Unidades o cursos formados por lecciones y organizados en secciones
- Versión web y móvil. Existe una versión de pago y una versión gratuita. Con la versión de pago se eliminan los anuncios y los usuarios tienen la posibilidad de utilizar la aplicación sin conexión, así como de realizar lecciones para medir su progreso
- Distintos tipos de preguntas. Preguntas de opción múltiple, preguntas de respuesta múltiple, preguntas de rellenar los huecos en blanco y preguntas en las que hay que asociar distintas opciones con su correspondiente respuesta.

3.2.2 KAHOOT

Kahoot se trata de una aplicación con soporte web y móvil basada en la creación de tableros de juego para completarlos posteriormente. Esta aplicación no se centra en ningún tipo de contenido en especial si no que da la posibilidad a los usuarios de crear su propio contenido y compartirlo con el resto de los jugadores.

Kahoot también implementa ciertas técnicas de gamificación que hacen que la experiencia de usuario sea similar a un juego. A la hora de crear un Kahoot, el creador de contenido es proporcionado con un código PIN el cual comparte con los usuarios que él elija. Estos usuarios introducen el código en su aplicación y entran en el Kahoot. Cada usuario es identificado con un nombre y se comienza el Kahoot. Una vez se inicia el proceso los usuarios van respondiendo a distintas preguntas las cuales suelen ser estilo opción múltiple. Los usuarios obtienen puntos en función de si responden correctamente a la pregunta y en el tiempo que les toma hacerlo. Después de cada pregunta se muestra una clasificación de los usuarios. Además, si un usuario responde un número elevado de preguntas correctamente de manera seguida obtiene más puntos como forma de recompensa.

Esta manera de recompensar las respuestas rápidas y las rachas de respuestas correctas además de mostrar la clasificación de los usuarios en el Kahoot son técnicas de gamificación que hacen que el usuario tenga la sensación de estar en un juego en lugar de estar aprendiendo sobre conceptos mostrados en las preguntas. Además, la aplicación también incluye música que favorece a esta inmersión en la experiencia.

Un resumen de las principales características de la aplicación se indica a continuación:

- Aplicación con versión web y móvil
- Ofrece distintas versiones. Versión gratuita, Kahoot 360 Standard, Kahoot 360 Presenter y Kahoot 360 Pro. Cada versión ofrece un límite mayor de usuarios que pueden participar en un Kahoot y distintos tipos de preguntas.
- Incluye preguntas tipo: respuesta corta, opción múltiple y verdadero o falso

En la Figura 15 se puede observar un ejemplo de pregunta con opción múltiple. El contenido de la pregunta puede variar a criterio del creador de contenido.

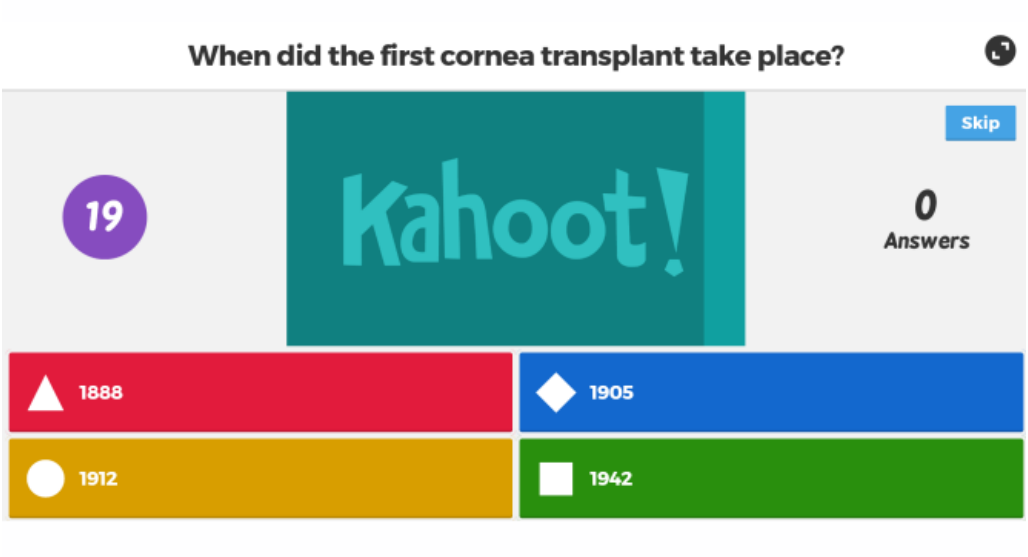


Figura 15 Ejemplo de pregunta en Kahoot (Kahoot)

3.2.3 QUIZLET

Quizlet es una aplicación web utilizada para la enseñanza de diversos conceptos. Al igual que Kahoot el contenido que se ofrece en Quizlet depende de los creadores de contenido. A la hora de crear una cuenta los usuarios pueden identificarse como profesores y crear contenido. Este contenido luego es compartido con alumnos para que realicen los cursos de manera presencial o telemática. Los creadores de contenido deben verificar su validez como tales indicando a que organización académica pertenecen o qué otro puesto ocupan el cual les habilita para crear contenido educativo. Otros usuarios pueden acceder a esos cursos y aprender de manera independiente. También pueden clonar cursos ya creados y modificarlos para crear nuevos cursos y añadir contenido a la aplicación en caso de tener una cuenta con dichos permisos.

Principalmente Quizlet se puede definir como una base de datos de cursos. Actualmente consta de más de 300 millones de secciones.[11] Los cursos principalmente están formados por tarjetas. Estas tarjetas cuentan con una palabra o concepto y al pulsarlas muestran una

descripción sobre dicha palabra. No obstante, Quizlet ofrece contenido también de otras maneras para que el usuario que esté tomando el curso pueda aprender de otras formas. A continuación, se describen las distintas categorías que ofrece Quizlet para que los usuarios pongan sus conocimientos a prueba.

Aprender: esta sección consiste en preguntas de estilo opción múltiple que el usuario va completando hasta obtener una puntuación general. Esta sección también incluye preguntas con formato imagen.

Ortografía: en este tipo de preguntas el usuario escucha una grabación del profesor pronunciando una palabra concreta y se le pide que la escriba correctamente.

Escribir: el usuario es presentado con una pregunta y debe de escribir la respuesta de manera breve.

Probar: se muestra al usuario un término con su correspondiente definición. Este deberá de responder con verdadero o falso para indicar si el término y la definición se corresponden.

Combinar: en una ventana se le muestran al usuario bloques con texto que contienen contenido relativo al curso que está tomando del cual ha aprendido al completar las tarjetas. El usuario debe de arrastrar los bloques de información para relacionarlos con otros. Si ambos bloques de información están relacionados, desaparecen. El objetivo es que desaparezcan todos los bloques en el menor tiempo posible.

Gravedad: esta sección es un minijuego. El usuario debe de derribar los bloques de texto que se acercan a un planeta para ganar puntos. Para ello debe de escribir en un cuadro de texto aquel término que se identifica con la definición presente en el bloque de texto.

En las distintas categorías de preguntas que Quizlet ofrece para poner a prueba los conocimientos del usuario, se encuentran varias técnicas de gamificación. Entre ellas cabe destacar un sistema de puntuación en el que se indica al usuario que preguntas le quedan por responder, cuales ha respondido correctamente y en cuales ha fallado. Además, a la hora de completar todas las preguntas, se otorga al usuario una puntuación basada en el desempeño

que ha tenido en la sección. En la sección "Combinar" hay un temporizador que anima al usuario a completar la actividad en el menor tiempo posible para así obtener una mejor puntuación. Una vez el usuario termina la prueba, se le indica cuál es su posición en un ranking en el que se encuentran el resto de los usuarios que también han participado. También se otorgan al usuario insignias reconociendo algún mérito como haber completado su primera sección de "Combinar". Además, en Quizlet las preguntas que son respondidas de manera incorrecta por el usuario reaparecen posteriormente para que sea más fácil para el mismo memorizar la respuesta.

En la Figura 16 se puede observar un ejemplo de actividad presente en la categoría "Combinar" para ilustrar al lector en la descripción dada.

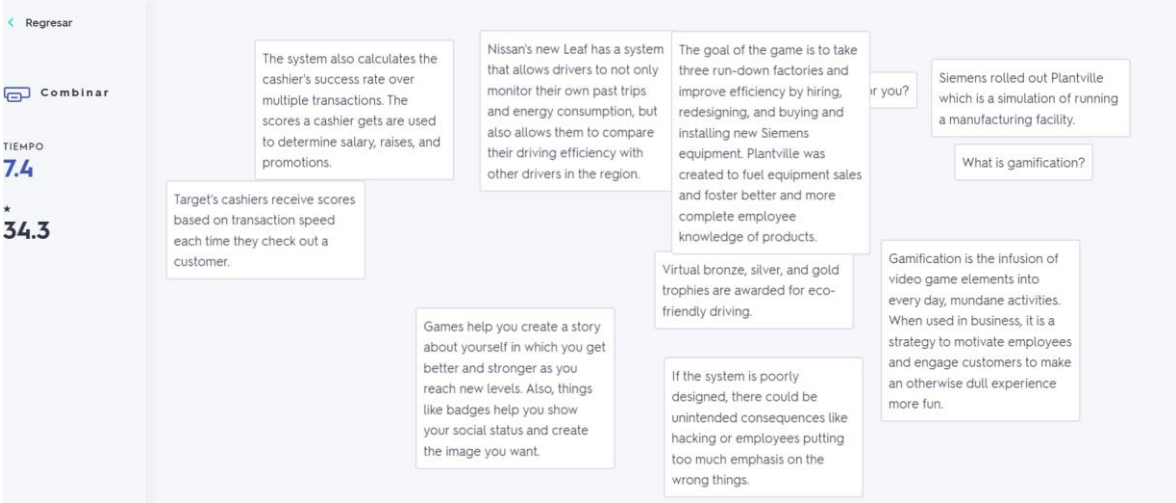


Figura 16 Ejemplo actividad combinar en Quizlet (Quizlet)

De acuerdo con un estudio realizado por la Universidad de Walailak las técnicas que utiliza quizlet resultan efectivas a la hora de enseñar.[12] El estudio se realizó a un grupo de estudiantes que estaban tomando un curso de inglés. Los estudiantes debían de aprender 500 palabras nuevas de vocabulario para mejorar su nivel. El estudio se prolongó durante 10 semanas. En las 5 primeras semanas los estudiantes no usaron Quizlet mientras que en las 5 últimas sí. Cada día los estudiantes completaban un pequeño examen para mostrar sus conocimientos. El desempeño de los estudiantes mejoró al utilizar la aplicación ya que demostraron ser capaces de aprender más palabras.

Las principales características que incluye la aplicación son:

- Aplicación web
- Una versión de pago para los creadores de contenido y una versión gratis para los consumidores de este. La versión de pago cuesta $34 al año y ofrece a los profesores la posibilidad de grabar su voz y subir imágenes a la plataforma entre otras cosas. Además, la versión de pago no cuenta con anuncios.
- Distintos estilos de preguntas para que los estudiantes pongan a prueba su nivel

3.3 TABLA COMPARATIVA DE CARACTERÍSTICAS

En esta sección se presenta una tabla comparativa con el objetivo de ilustrar qué características son comunes a las distintas aplicaciones y juegos descritos en este capítulo y cuales los diferencian. La tabla no contiene el juego Defend The Crown debido a que al tratarse de una aplicación desarrollada para la Agencia de Ciberseguridad y Seguridad de Infraestructuras de Estados Unidos cuenta con menos relevancia a la hora de analizar los competidores directos en el mercado por tener un público objetivo más específico.

En la Tabla 1 se presentan tanto características discutidas en el análisis de los competidores como características que no se han mencionado explícitamente en los anteriores párrafos.

Aplicación *Características*	*Duolingo*	*Kahoot*	*Internland*	*Quizlet*
Versión de pago	✓	✓	×	✓
Anuncios	✓	×	×	✓
Versión offline	✓	×	×	
Barra de Progreso	✓	✓	✓	✓

Avatar personalizable	✓	×	×	×
Sistema de puntos	✓	✓	✓	✓
Música de fondo	×	✓	✓	×
Temática cambiante	×	×	✓	×
Ranking	×	✓	×	✓
Insignias	✓	✓	✓	×
Realimentación	✓	×	✓	✓
Compartir	✓	×	✓	✓
Animaciones	✓	×	✓	✓
Creación de contenido	×	✓	×	✓
Tienda	✓	×	×	×
Foro	✓	×	×	✓

Tabla 1 Comparativa de las características de los competidores

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

En el capítulo anterior se han expuesto diversas aplicaciones que utilizan técnicas de gamificación para la enseñanza. En la descripción de dichas aplicaciones se han expuestos datos que demostraban la efectividad de dichas técnicas y cómo mediante la integración de características propias de videojuegos en sus plataformas consiguen un mayor desempeño en los alumnos. No obstante, ninguna de las aplicaciones que se han mencionado enseña sobre conceptos relacionados con la ciberseguridad. En el caso de Quizlet y Kahoot el contenido es variado y depende de los creadores de contenido. En cuanto a Duolingo el contenido es sobre idiomas.

Respecto a los otros competidores que son Defend The Crown e Interland, estas aplicaciones sí se centran en conceptos de ciberseguridad, pero son juegos y no plataformas de enseñanza. Además, el contenido que ofrecen es estático y no se ve sometido a actualizaciones. También, Interland se centra en un público mayormente infantil mientras que Defend The Crown tiene como principal público al Departamento de Ciberseguridad y Seguridad de Infraestructuras de Estados Unidos.

Es por ello por lo que este proyecto trata de ofrecer una plataforma única para aprender de manera divertida y atractiva acerca de contenidos de ciberseguridad. En esta plataforma el contenido es dinámico y puede ser cambiado por los administradores de esta. Los administradores pueden añadir cursos, borrarlos, ver el progreso de otros usuarios y establecer cursos recomendados o cursos destacados. Por tanto, se ofrece una plataforma especializada en ciberseguridad que incluye técnicas de gamificación implementando de esta manera las dos características ventajosas de ambos tipos de competidores. Además de incluir un contenido dinámico que puede ser actualizado a medida que surgen nuevos conceptos en este ámbito.

Por un lado, las técnicas de gamificación ayudan a los usuarios a sentirse más motivados para aprender y sumergen al usuario en una experiencia más interactiva. Por otro lado, el contenido técnico y dinámico de ciberseguridad logra que los usuarios aprendan conceptos nuevos sobre este ámbito que les ayuden a mantenerse a salvo de los posibles peligros presentes en Internet. La combinación de ambas características busca que los usuarios terminen por desarrollar ciertas costumbres calificadas en este proyecto como "ciberhigiénicas" en el ciberespacio que les hagan sentirse más seguros y realmente sirvan como técnicas para mantener sus datos a salvo.

Un ejemplo de la utilidad de esta plataforma es que los usuarios tras completar un curso centrado en contraseñas sepan diferenciar una contraseña fácilmente descifrable de una contraseña más segura. También los usuarios pueden aprender acerca de técnicas para mantener sus contraseñas seguras como la utilización de gestores de contraseñas o el hecho de nunca dejar escrita una contraseña.

4.1.1 EFECTIVIDAD DE LAS TÉCNICAS DE GAMIFICACIÓN

En este apartado se trata de justificar por qué se han implementado técnicas de gamificación en este proyecto y cuáles son los beneficios de estas. Las técnicas de gamificación han demostrado en diversos casos aportar un valor añadido a la enseñanza y a la productividad de aquellos procesos en los que se han introducido. A continuación, se presentan diversos casos.

La empresa de telecomunicaciones Cisco ha creado una plataforma de entrenamiento sobre redes sociales para sus empleados. Esta plataforma introduce tres tipos de certificación que los empleados pueden alcanzar: especialista, estratega y maestro. Esta plataforma además incluye cuatro certificaciones de más bajo nivel sobre Recursos Humanos, Comunicaciones Externas, Grupos de Trabajo y Ventas.

Gracias a la introducción de técnicas de gamificación Cisco ha logrado que en su plataforma se completasen más de 13.000 cursos y que participasen más de 650 empleados. [13]

La firma de consultoría Deloitte también ha puesto en práctica la gamificación en métodos de enseñanza. La empresa creó un curso para los ejecutivos senior con el objetivo de que estos aprendiesen sobre técnicas de liderazgo. Los ejecutivos no participaban activamente en el curso y Deloitte trató de resolver la situación mediante la ayuda de la empresa BadgeVille que introdujo en el curso sistemas de recompensa como insignias y símbolos de estatus, así como rankings en los que se mostraba que ejecutivos estaban participando activamente en el curso. Estos cambios resultaron en una reducción de un 50% en el tiempo que les tomaba a los ejecutivos completar los cursos, así como un aumento de un 46.6% en las veces que los usuarios volvían en el mismo día al curso.[13]

Por otro lado, la empresa Domino´s Pizza con el objetivo de aumentar sus ventas decidió crear una aplicación en 2015 llamada Pizza Hero. En esta aplicación los usuarios podían realizar sus pedidos online. No obstante, la empresa decidió implementar dos funcionalidades adicionales: agitar el móvil para que la aplicación eligiera una pizza por ellos y la posibilidad de que los usuarios crearan una pizza personalizada. Estos dos cambios supusieron un incremento de un 30% en sus ventas en Estados Unidos. [15]

Estos son algunos ejemplos de los muchos casos en los que la gamificación ha demostrado su efectividad tanto en el ámbito de la educación como en las ventas de determinados negocios. Es por ello por lo que en este proyecto se ha decidido implementar dichas técnicas para lograr una mayor tasa de conceptos aprendidos por usuario.

4.1.2 CIBERATAQUES: IMPACTO ECONÓMICO

Cada vez es más común que las empresas se vean sometidas a ciberataques que tratan de robar información estratégica de las mismas. Esta información está relacionada con proyectos que las empresas tienen pensado llevar a cabo, información sobre fusiones y adquisiciones de otras empresas y demás información privada. De acuerdo con un informe llevado a cabo por la empresa Deloitte en 2021 llamado “El estado de la Ciberseguridad en España. Post Pandemia: un camino inexplorado”, un 94% de las empresas españolas fueron sometidas a un ciberataque en este año. Además de esas empresas un 65% de las mismas afirma que han recibido más de un ataque y un 25% más de dos. No solo la mayoría de las

empresas son víctimas de ciberataques, sino que la situación va a peor a medida que pasa el tiempo. El porcentaje de ciberataques por empresa subió más de un 26% entre 2020 y 2021.

Cabe destacar que no todos los sectores son igualmente atacados. Los sectores de las Telecomunicaciones, Banca, Administración pública, Seguros y Fabricación son los más afectados. En la Figura 17 obtenida del mencionado informe de Deloitte se puede observar la media de ciberataques recibidos en 2021 por sector. Es importante destacar que estas empresas destinan una mayor cantidad de dinero a sus departamentos de ciberseguridad y tienen una mayor educación en este ámbito que otros sectores. No obstante, debido a la naturaleza de sus negocios son los sectores más atacados.

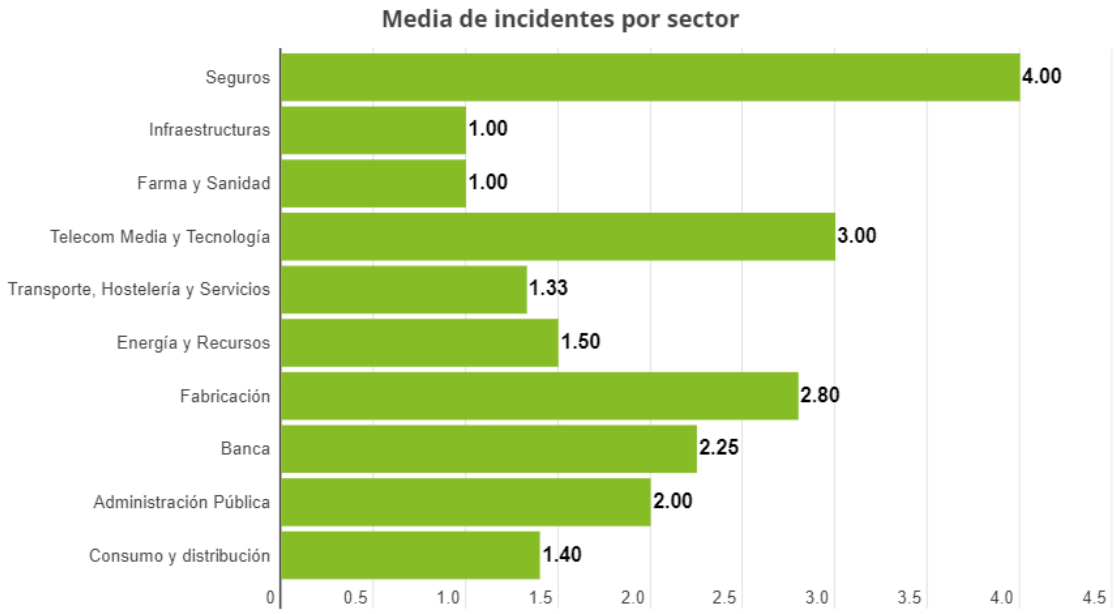


Figura 17 Media de ciberataques sufridos en 2021 ordenados por sector ("El estado de la Ciberseguridad en España. Post Pandemia: un camino inexplorado", Deloitte)

También existen amenazas más comunes que otras siendo el phishing el tipo de ciberataque más común seguido por ransomware y malware. El phishing es una técnica de ingeniería social que consiste en tratar de engañar a las personas mediante el envío de correos electrónicos y mensajes de texto con el objetivo de que estas respondan con información privada como su contraseña o algún tipo de identificador. El atacante se hace pasar por una empresa o servicio que la víctima utiliza y le pregunta por información personal. El

ransomware consiste en la codificación de los datos de un sistema. El atacante bloquea el sistema dejándolo inutilizado y pide dinero a la víctima para desbloquearlo. En cuanto al malware es un concepto que hace referencia a diversos tipos de ciberataques. En la Figura 18 extraída del citado informe se puede observar cuales son los ciberataques más comunes.

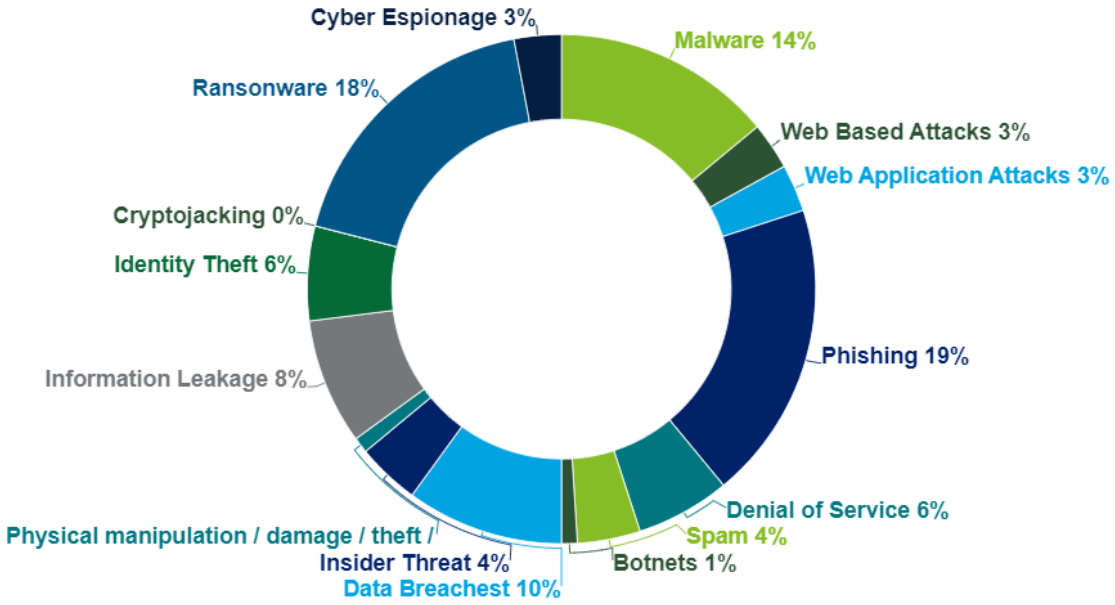


Figura 18 Porcentaje de los tipos de ciberataques sufridos en 2021 ("El estado de la Ciberseguridad en España. Post Pandemia: un camino inexplorado", Deloitte)

Después de haber analizado la situación en España respecto al número de ataques, los sectores más afectados y los tipos de ciberataques más comunes, también es importante destacar la cantidad de dinero perdido y qué se está haciendo al respecto. En 2021 las pérdidas para las pequeñas y medianas empresas (pymes) oscilaron entre 2000 y 5000 euros. En cuanto a las grandes empresas estas perdieron 3.6 millones de euros de media por causa de los ciberataques.[15]

A pesar de ello, un 55% de las grandes empresas no se protegen de manera adecuada ante los ciberataques y la mayoría piensan que adelantarse a esos ataques y estar siempre protegido es insostenible. No obstante, las empresas siguen aumentando su gasto en ciberseguridad que en 2021 fue superior a 2020. [16]

Debido a que cómo se ha mostrado el phishing es una de las formas de ciberataque más común. Muchas empresas incluyen campañas anuales en las que mandan a sus empleados correos simulando intentos de phishing para ver cómo reaccionan los mismos y de esta manera formarlos para posibles futuros ataques.

En esta sección se ha mostrado la dimensión de los problemas relacionados con la ciberseguridad. Se ha indicado cómo los ciberataques siguen en aumento y las pérdidas millonarias que suponen para algunos sectores. También se ha discutido acerca de cómo las empresas a pesar de seguir invirtiendo más dinero en ciberseguridad siguen sin estar bien protegidas. Este proyecto trata de ofrecer una solución a este problema. Las empresas en lugar de realizar campañas sobre phishing o cursos de formación de duración limitada podrían incluir esta plataforma en sus empresas. De esta manera los empleados estarían sometidos a un aprendizaje continuo sobre conceptos de ciberseguridad. Además, el tener una plataforma con cursos que se pueden volver a completar ofrece la posibilidad a las empresas de que sus empleados no olviden los conceptos aprendidos como podría estar pasando en estos cursos y campañas que se realizan de manera puntual en el tiempo.

Otra de las ventajas que ofrece este proyecto es la posibilidad de ofrecer contenido dinámico y preciso. En caso de que surjan nuevos tipos de ataques o que las técnicas utilizadas por los hackers para obtener información o ganar acceso a un sistema cambien, la plataforma podrá incluir nuevos cursos relativos a esto.

4.2 OBJETIVOS

Los objetivos principales de este proyecto son los siguientes

- **Desarrollar una aplicación móvil como medio para desplegar la plataforma**. Para ello se utilizará Flutter, el SDK ofrecido por Google con el objetivo de desarrollar una aplicación que sea válida para iOS y Android y así poder llegar a un mayor número de usuarios.

- **Insertar técnicas de gamificación en la plataforma**. La aplicación tendrá un sistema de niveles basado en puntos de experiencia. El usuario podrá subir de nivel mediante la acumulación de puntos de experiencia y como consecuencia ganará un avatar único. Para ganar puntos el usuario debe completar cursos. Los puntos otorgados serán relativos a las preguntas correctamente acertadas en un curso y serán únicos, es decir, si se repite un curso el usuario solo sumará los puntos relativos a las preguntas que ha contestado correctamente por primera vez. El usuario ganará un avatar único cada vez que suba de nivel, el cual podrá establecer como predeterminado. Además, cuando el usuario complete un curso, este ganará una insignia ligada a dicho curso. Existirá también un curso destacado que podrá ser modificado por el administrador de la plataforma que otorgará puntos extra.
- **Diseñar una base de datos** para guardar la información de los usuarios y de los cursos. Para ello se utilizará Cloud Firestore, una base de datos NoSQL ofrecida por Google para el diseño de aplicaciones móviles y web.
- **Diseñar un sistema de autenticación y autorización** de usuarios. Se desear diseñar un sistema de autenticación de usuarios para permitir el acceso a la plataforma para ello se utilizará Firebase Authentication, un servicio back-end ofrecido por Google con este objetivo. También se desea diferenciar a los usuarios por su rol que podrá ser normal o administrador.
- **Ofrecer contenido dinámico.** La plataforma debe de incluir contenido dinámico que sea modificable por el administrador de esta. El administrador tendrá la posibilidad de añadir cursos, eliminar cursos, cambiar el curso recomendado y cambiar el curso destacado. Otra de las funcionalidades del administrador es la de ver la información de otro usuario acerca de su progreso en la plataforma.

En cuanto a los objetivos secundarios del proyecto, estos son los siguientes:

- **Ofrecer la información por medio de cursos de 5-10 preguntas**. Uno de los objetivos del proyecto es que el usuario aprenda completando cursos directamente sin tener que realizar una tarea de investigación previa acerca del tema de los cursos. La aplicación ofrecerá pruebas de 5-10 preguntas. Las preguntas será de tipo opción

múltiple y rellenar los huecos. Cada pregunta tendrá una explicación asociada a la cual el usuario podrá acceder sin importar si ha respondido bien o mal.

- **Ofrecer una versión beta de la aplicación para su descarga**. Se pretende que al finalizar el proyecto exista una versión beta que los usuarios puedan descargar. Para Android se podría subir la aplicación a la tienda de Google, Play Store, y para iOS a la App Store.

4.3 METODOLOGÍA

A la hora de realizar el proyecto se ha seguido la siguiente metodología. El proyecto consta de dos partes diferenciadas, una parte de diseño inicial siguiendo el método del Doble Diamante, y una segunda parte de desarrollo en la que se da forma a la aplicación utilizando Flutter en el IDE Android Studio y utilizando los servicios ofrecidos por Firebase.

Como metodología de proyecto se ha utilizado una metodología híbrida. Por un lado, se han incluido características propias de la metodología Agile. Al principio de cada semana se han definido una serie de objetivos que debían cumplirse para finales de esta. A la hora de definir estos objetivos se ha tratado de no ser excesivamente ambicioso para no tener que acumular objetivos no completados en otros sprints. En caso de terminar con los objetivos definidos para dicho sprint se incluían más objetivos. Por el contrario, si los objetivos no podían ser cumplidos a tiempo se incluían en el sprint de la siguiente semana.

No obstante, la linealidad del proyecto ha forzado a incluir características propias de la metodología en cascada. La fase de desarrollo del proyecto no se podía empezar sin haber terminado la fase de diseño la cual terminaba con un prototipo de fidelidad alta de la aplicación que ha servido como punto inicial para el desarrollo de la aplicación en Flutter. Además, ciertas fases del diseño necesitaban terminar antes de que las siguientes comenzaran ya que servían de punto de partida para las futuras. Lo mismo ha ocurrido con la fase de desarrollo donde se ha decidido desarrollar cada una de las principales características de la aplicación por separado e ir integrándolas con el paso del tiempo.

La fase de diseño se compone de 4 partes bien diferenciadas. Una fase conocida como fase de descubrimiento en la cual se ha llevado a cabo un análisis de los competidores y una primera definición de las características a incluir en la aplicación. Tras realizar el análisis de los competidores se ha llevado a cabo un posicionamiento del producto que se ofrece en este proyecto dentro del mercado. Además, en esta fase se ha llevado a cabo una encuesta en Google Forms en la que se ha preguntado acerca de las características deseables en una aplicación de este tipo, así como características en general que los encuestados encontraban ventajosas en las aplicaciones que utilizaban. Esta encuesta se ha realizado a un total de 60 personas y los resultados han servido para continuar con la fase de diseño y saber qué características incluir en la aplicación.

Una vez terminada esta fase, se ha llevado a cabo la fase de definición. En ella utilizando la información de la encuesta, así como las conclusiones extraídas del análisis de los competidores se ha definido un perfil de usuario. Después de esto, se ha definido un tipo de usuario arquetípico que utilizaría la aplicación. Una vez definido un perfil de usuario con aquellas características que encuentra deseables y aquellas que no de acuerdo con la información de la encuesta se ha definido la situación actual de la industria y cómo se siente el usuario en ella. También se ha definido cuales serían las principales funcionalidades que el usuario llevaría a cabo en la aplicación. Después de esto se han definido los posibles problemas que se plantean en la situación actual y en las funcionalidades de la aplicación y una manera de resolverlos.

La tercera fase de diseño es la fase de desarrollo. En ella se define como va a ser el producto. Se vuelve a utilizar esa información del usuario definida en la fase anterior para encontrar características que le aportarían valor en la aplicación. Se definen características que son de obligada implementación, así como aquellas que deberían estar incluidas, podrían ser incluidas y las que no deben incluirse. Más adelante, se ha definido un producto mínimo viable (del inglés MVP) con aquellas funcionalidades mínimas que el producto tiene que incluir. Después, se ha definido el flujo principal de acciones que el usuario sería capaz de realizar en la aplicación, así como un esquema de la arquitectura de la información incluida en la aplicación en la que se han definido las distintas partes de esta.

La última fase es la fase de entrega. En ella se llevan a cabo tres prototipos de la aplicación. Uno de baja fidelidad a lápiz y papel. Más adelante se llevaron a cabo dos prototipos, uno de fidelidad media y otro de fidelidad alta utilizando la herramienta de diseño Figma.

Una vez terminada la fase de diseño, se comenzó con la fase de desarrollo. Con el objetivo de hacer más fácil esta parte se llevaron a cabo diversos diagramas UML. Se han realizado un diagrama de clases y un diagrama de casos de uso.

La fase de desarrollo se ha separado en 4 partes.

- Sistema de Autenticación. Lo primero que se ha llevado a cabo es el diseño de las páginas del log in y el sign up de la aplicación. Una vez diseñadas las ventanas, se han conectado con Firebase Authentication y se les ha otorgado funcionalidad. También se definió un primer modelo de usuario para guardar información en la base de datos.
- Flujo de Cursos. En esta parte del desarrollo se ha llevado a cabo la programación de todas las ventanas relativas a los cursos que se han definido en el prototipo de alta fidelidad. Una vez las ventanas estaban creadas se probaba su funcionalidad con información de prueba. Cuando se comprobó que funcionaban correctamente, se creó un curso falso siguiendo la estructura definida en el modelo para garantizar que la lógica de scripting funcionaba. Para finalizar se conectaron las páginas con Cloud Firestore para utilizar información real.
- Perfil de usuario. También se ha llevado a cabo una primera parte de diseño de las ventanas para más adelante incluir lógica y finalmente conectarlas con la capa de persistencia.
- Administrador. Siguiendo el mismo método que las 3 partes anteriores se han desarrollado las páginas necesarias para incluir la funcionalidad del administrador.

Una vez terminado el código de cada una de las partes se ha llevado a cabo una tarea de integración entre las diferentes partes para que los cambios se vieran reflejados en toda la aplicación.

4.4 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

4.4.1 PLANIFICACIÓN

En la Figura 19 se muestra un diagrama de Gantt que ilustra la planificación del proyecto con los pasos descritos en el epígrafe de metodología.

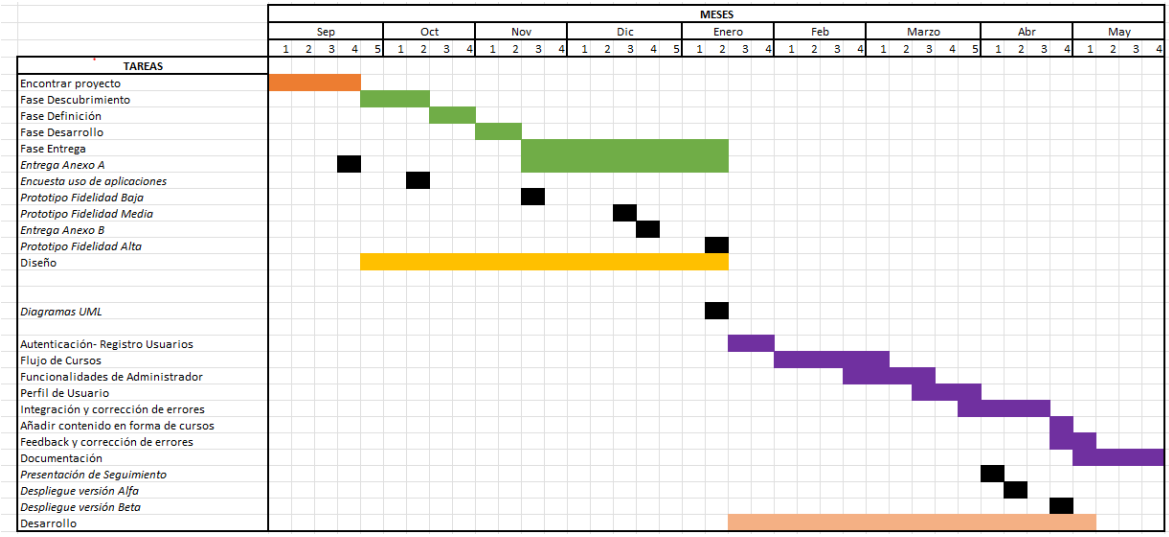


Figura 19 Diagrama de Gantt del proyecto

En el diagrama se pueden observar las dos principales fases del proyecto las cuales son la fase de diseño y la fase de desarrollo conectadas por la creación de los diagramas UML con el objetivo de facilitar la segunda parte del proyecto.

En el diagrama de Gantt se han indicado con letra cursiva y color negro aquellas tareas que son hitos. Al tratarse de hitos solo ocupan una celda. En este diagrama se puede observar esa planificación de estilo cascada que se ha comentado en el epígrafe de la metodología.

4.4.2 ESTIMACIÓN ECONÓMICA

En este apartado se analiza la estimación económica del proyecto. Para ello se deben analizar las diferentes tecnologías descritas en el capítulo 2 de esta memoria ya que ofrecen diferentes tipos de suscripciones.

Respecto a la fase de diseño se han utilizado mayoritariamente dos herramientas: Miro Board y Figma. Miro Board ofrece 4 tipos de suscripciones. Una suscripción gratuita, una suscripción de equipo ($8 al mes), una suscripción Business ($16 al mes) y una suscripción para organizaciones donde el precio varía en función del tipo de organización y sus necesidades. Para el desarrollo de este proyecto se ha utilizado la versión de estudiante la cual es gratuita siempre que se demuestre la condición de estudiante en una institución educativa. Esta versión incluye las características suficientes para el desarrollo de la fase de diseño de este proyecto. Algunas de estas características relevantes para el proyecto son las siguientes:

- Número ilimitado de tableros activos.
- Número ilimitado de acceso al tablero por medio de invitaciones con link.
- Tableros privados
- Exportación de archivos del tablero con alta resolución y sin marca de agua.

En cuanto a Figma, esta herramienta también ofrece varios tipos de suscripciones. Dentro de las suscripciones gratuitas existen dos subtipos. Una versión gratuita básica y una versión gratuita educativa. De nuevo para acceder a este último tipo de suscripción basta con demostrar la condición de estudiante en una institución educativa lo cual se ha hecho en este caso. Con la suscripción gratuita educativa tienes acceso a todas las funcionalidades que se tienen cuando se paga la suscripción de Figma Professional (12$ al mes). De esas funcionalidades, aquellas que han servido de utilidad para este proyecto han sido las siguientes:

- Acceso a proyectos privados y compartidos.
- Número ilimitado de archivos Figma.
- Acceso a proyectos de la comunidad Figma.

Figma también ofrece una suscripción para organizaciones ($540 al año). No obstante, esta suscripción ofrece funcionalidades que se escapan de los objetivos de este proyecto.

Respecto a la fase de desarrollo, se ha utilizado Android Studio como IDE y Flutter como SDK. Ambas herramientas no han supuesto ningún tipo de coste para el proyecto. A la hora de probar la aplicación se ha utilizado un simulador de AVD integrado en Android Studio que tampoco ha resultado en ningún gasto. Para pruebas de la aplicación en un dispositivo no simulado he utilizado mi móvil personal con sistema operativo Android. Se trata de un Xiaomi Redmi Note 8 cuyas especificaciones han sido suficientes para correr la aplicación desde Android Studio. Como máquina para el desarrollo de la aplicación en la cual se ha utilizado Android Studio se ha utilizado mi portátil personal. Un dell Inspiron 7000 con un procesador AMD Ryzen de séptima generación, 16GB de RAM y 512GB de memoria tipo SSD. Este portátil ha supuesto un costo de 1000$ y sí supone un costo para el desarrollo del proyecto debido a qué sin un ordenador de especificaciones similares no se puede desarrollar una aplicación móvil utilizando Android Studio como IDE junto con su simulador. Este ordenador se ha comprado con el objetivo mencionado para el proyecto.

En cuanto a la persistencia de los datos de la aplicación y al sistema de autenticación se han utilizado servicios ofrecidos por Firebase de Google. Para la persistencia de datos se ha utilizado Cloud Firestore y para la autenticación de usuarios de ha utilizado Firebase Authentication. Para utilizar ambos servicios se ha recurrido al plan de suscripción Stark. Este plan es gratuito y los servicios que ofrece cumplen con las demandas del proyecto. En lo que a Cloud Firestore se refiere, con este plan se permiten almacenar un máximo de 1 GiB de datos y realizar 20000 operaciones de escritura, 50000 operaciones de lectura y 20000 eliminaciones al día. Por otro lado, la autenticación de usuarios utilizando email y contraseña de Firebase Authentication (método utilizado en este proyecto) está incluida en el plan Spark. Este también incluye otros métodos de autenticación como la telefónica. Con este plan se pueden autenticar 1000 personas al mes en la región de Canadá, Estados Unidos e India y otras 1000 personas para el resto de los países. Firebase ofrece un plan llamado plan Blaze en el cual se describen los costes de los diferentes servicios una vez se han cubierto los límites marcados en el Plan Spark. Si la dimensión de este proyecto creciera en el futuro deberían incluirse dichos gastos variables o estudiar un plan de migración a otra plataforma.

No obstante, estos gastos quedan fueran de los objetivos descritos en este proyecto y se acercan más a una idea de negocio que al desarrollo de una versión beta de la aplicación.

Para poder desplegar la aplicación en la App Store y en Play Store se hace necesario obtener una cuenta de desarrollador iOS y una cuenta de desarrollador de Google. Para la primera deben abonarse $100 y para la segunda deben abonarse $50. Se trata de un pago único y no de una suscripción mensual. Con estas licencias se tiene la posibilidad de desplegar la aplicación en las mencionadas plataformas siempre que cumplan con sus políticas.

Por último, cabe destacar el gasto que hubiera supuesto contratar servicios profesionales para la fase de diseño y elaboración de los diagramas UML así como para la fase de desarrollo de la aplicación. Suponiendo un sueldo medio de 18000€ al año en España para un diseñador de UI/UX sin experiencia, esto equivaldría a unos 15€ por hora trabajada. La fase de diseño del proyecto junto con el diseño de los diagramas UML ocupa unas 16 semanas. Considerando una media de 2 horas de trabajo diarias y descontando fines de semana, supondría un coste de 2400€. Si se supone un coste similar para un desarrollador de aplicaciones Flutter en España sin experiencia, se obtiene un coste de 4800€ siguiendo un proceso similar y teniendo en cuenta el doble de horas trabajadas por la carga de trabajo de esta fase. No obstante, en el desarrollo del proyecto no se han contratado servicios de terceros, simplemente se incluye este párrafo con el objetivo de ilustrar al lector y mostrar un posible gasto que hubiera producido en caso de que el desarrollo no se hubiera llevado a cabo por el autor del informe.

En resumen, teniendo en cuenta que los gastos de desarrollo y diseño no se deben incluir ya que han sido llevados a cabo por el autor, que se ha podido acceder a varias suscripciones en su versión para estudiantes, y que la autenticación de usuarios y persistencia de datos no supone un coste; solo debe tenerse en cuenta el pago de las cuentas de desarrollador y el pago del portátil. Estos gastos suman un total de $1150 que en euros son unos 1090.20€.

Capítulo 5. SISTEMA/MODELO DESARROLLADO

En este capítulo se va a llevar a cabo una descripción detallada de la plataforma. Para ello se separará el mismo en tres epígrafes fundamentales. Un primer epígrafe dedicado a la fase de diseño de la plataforma, un segundo explicando la elaboración de los diagramas UML y un último epígrafe en el que se explica de manera detallada el desarrollo.

5.1 DISEÑO DE LA PLATAFORMA

Como se ha mencionado con anterioridad en esta documentación, el proceso de diseño que se ha llevado a cabo ha seguido la técnica del Doble Diamante explicada en detalle en el capítulo de descripción de las tecnologías. A continuación, se indicará con exactitud cual han sido las tareas que se han llevado a cabo en cada una de las fases del proceso de diseño ilustrando las explicaciones con figuras y tablas que muestren los resultados obtenidos.

Es importante que el lector tenga en cuenta que a continuación se van a describir características de la plataforma que se incluyeron en la fase de diseño y luego cambiaron a la hora de llevar a cabo el prototipado, así como la versión final de la aplicación. Muchas características y funcionalidades no se han llegado a implementar o han cambiado a medida que el proceso de diseño avanzaba.

5.1.1 FASE DE DESCUBRIMIENTO

5.1.1.1 Clientes y objetivos de negocio

Una vez surge la idea de llevar a cabo una plataforma de enseñanza acerca de conceptos de ciberseguridad incluyendo técnicas de gamificación el primero de los pasos es identificar los posibles clientes y los objetivos del negocio.

Los posibles clientes identificados en esta fase son los siguientes:

- Universidades

- Plataformas de enseñanza online
- Compañías concienciadas con la ciberseguridad
- Colegios
- Estudiantes independientes y curiosos

Una vez identificados los clientes, se indicaron los objetivos de la plataforma:

- Despertar curiosidad en los usuarios
- Descubrir conceptos relacionados con la ciberseguridad que son comúnmente desconocidos
- Ofrecer una manera eficiente de aprendizaje y a la vez divertida y atractiva
- Expandirse a más temáticas y no únicamente la ciberseguridad
- Hacer de internet un lugar más seguro
- Reducir el analfabetismo sobre ciberseguridad

5.1.1.2 Lean UX Canvas

La siguiente de las tareas llevada a cabo en esta fase fue la de rellenar el lienzo de experiencia de usuario definido por Jeff Gothelf un experto en el diseño de producto. En este lienzo se analizan los diversos problemas existentes en el mercado respecto al tema que se está analizando. Después se proponen diversas ideas para la solución y se analizan los impactos en el mercado y el valor añadido que aportaría a los usuarios. Es una forma eficiente de organizar las primeras ideas que se tienen sobre el producto que se quiere desarrollar.

En la Tabla 2 se trata de incluir la información necesaria para ilustrar el lienzo.

Tarea	*Resultado*
Problema de negocio. ¿Qué negocio necesita ayuda?	• Educación • Educación en línea • Ciberseguridad

Resultados de negocio. ¿Qué cambios en el comportamiento del cliente mostrarían que se ha resuelto un problema real de una manera que aporta valor al cliente?	• Aprendizaje sobre prácticas ciberseguras • Concienciación sobre la privacidad online • Distinción de peligros en el ciberespacio. • Colaboración con instituciones educativas y con instituciones especializadas en ciberseguridad.
Clientes y usuarios. ¿En qué usuarios y clientes debería el producto estar enfocado en un principio?	• Estudiantes y profesores. • Compañías y otras organizaciones. • Curiosos acerca de la ciberseguridad.
Beneficios de usuario. ¿Qué objetivos persigue el usuario? ¿Qué motiva al usuario a elegir el producto como su solución?	• Sentirse más seguro en el ciberespacio y aprender cómo implementar medidas en su vida online con ese objetivo. Contraseñas más seguras, uso de tarjetas virtuales, etc • Ser capaz de distinguir posibles peligros en Internet y no ser víctima de un posible ataque. • Minimizar la pérdida de información en caso de ser víctima de un ataque.
Ideas para la solución. Enumera ideas, características y funcionalidades del producto que ayuden a	• Explicaciones rápidas y preguntas cortas. • Incluir un sistema de puntuación basado en niveles. • Perfil modificable • Crear una comunidad. Enviar retos, ver puntuación de otros usuarios (ranking), compartir logros. • Sección de preguntas y respuestas.

solucionar el problema.	• Pruebas para subir de nivel • Retroalimentación con componentes gráficos. Vídeo y audio.

Tabla 2 Información incluida en el Lean UX Canvas

Tras llevar a cabo este análisis inicial de las necesidades del negocio y realizar una breve descripción de una posible solución y de lo que esta aportaría a los usuarios, se obtiene una visión general del producto. Se tiene una idea de qué solución se quiere desarrollar, qué características podría incluir y cómo aportaría valor al cliente.

5.1.1.3 Análisis de competidores

El siguiente de los pasos a completar en la fase de descubrimiento es un análisis de los competidores que existen en el mercado. No obstante, este análisis ya ha quedado reflejado en este informe en el apartado de Estado del Arte. Con el objetivo de no repetir información si el lector quiere encontrar información relativa a este análisis acuda a Capítulo 3.

5.1.1.4 Gráfica de posicionamiento de mercado

Tras llevar a cabo el análisis de los competidores, se ha creado una gráfica en la que se trata de situar a los mismos en función de las características que mejor los identifiquen. Para la creación de esta se han elegido dos medidas fundamentales. Lo innovador que el producto resulta y lo efectivo que este es. Una vez se han situado a todos los competidores en la gráfica se obtiene una idea general de cuál es la situación en el mercado.

En la Figura 20 se puede observar la gráfica resultante.

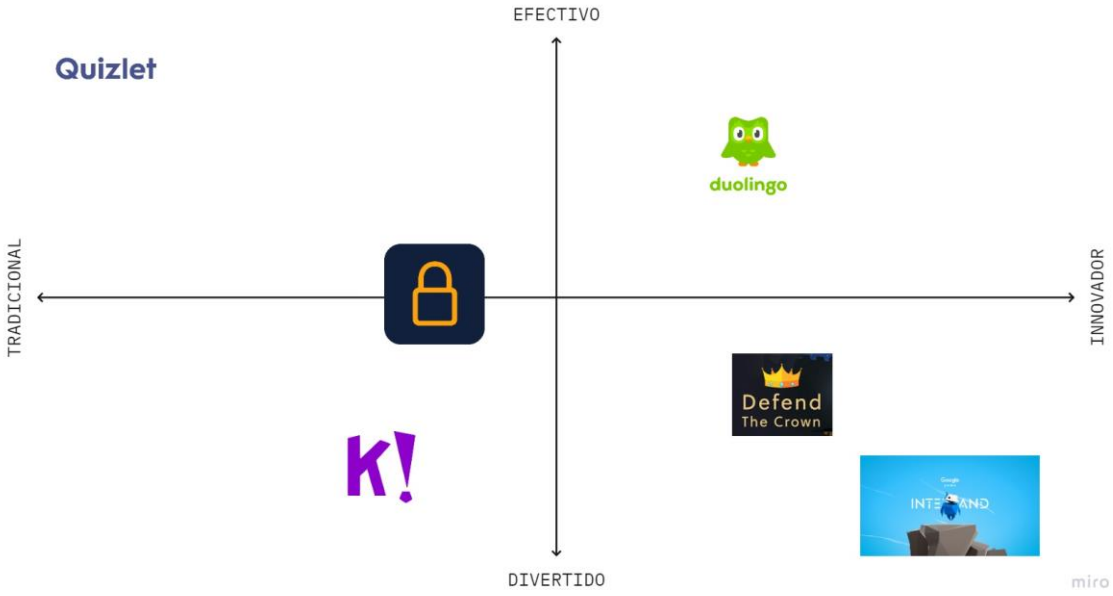


Figura 20 Gráfica de posicionamiento de mercado

Los juegos sobre ciberseguridad analizados: Defend The Crown e Interland ofrecen un producto divertido ya que estás inmerso en la dinámica del juego. No obstante, al tratarse de puramente juegos, el objetivo de enseñar puede verse afectado ya que el usuario puede estar más centrado en el juego que en el contenido de ciberseguridad al que se le está sometiendo. Esto sobre todo podría afectar a Interland que como se ha analizado con anterioridad está destinado a un público mayormente infantil. Por ejemplo, en el caso de un niño esté jugando a "Reality River", uno de los mundos con los que cuenta Interland en el cual el jugador controla una figura que cruza un rio y debe responder correctamente a las preguntas que aparecen en pantalla para no caer al agua, el niño puede estar más centrado en no caer al río que en aprender conceptos tales como crear una contraseña segura. Es por ello por lo que ambos juegos ocupan un puesto más innovador y divertido en la tabla. En cuanto a Defend The Crown, la explicación de la posición que ocupa en el gráfico es similar a la de Interland, pero se ha considerado que es más efectivo y tradicional que su competidor ya que está enfocado en un público más adulto.

A la hora de posicionar a Duolingo, se ha tenido en cuenta su efectividad la cual se ha demostrado en este informe con varias estadísticas que muestran resultados similares a clases

de idiomas de carácter universitario. Por otro lado, uno de los motivos del éxito de Duolingo fue la decisión de implementar técnicas de gamificación en la enseñanza de idiomas además de otras características únicas lo cual lo posiciona en el lado más innovador de la gráfica.

En cuanto a Quizlet, esta aplicación ocupa una posición caracterizada por la efectividad y la tradicionalidad. La efectividad de Quizlet viene dada por la implementación de técnicas de gamificación. A pesar de integrar estas técnicas no ocupa la zona innovadora de la gráfica como Duolingo ya que no cuenta con características como las animaciones e insignias. Se trata de una propuesta efectiva pero más formal, enfocada a un público más adulto tiene diseños y características más sencillas.

La principal característica de Kahoot es la diversión ya que cuenta con música, colores llamativos y dinamismo por las preguntas a contra reloj. No obstante, no deja de ser una aplicación con pruebas basadas principalmente en preguntas de respuesta múltiple a contra reloj. Técnica que se utiliza en numerosas aplicaciones de este estilo.

Una vez posicionados todos los competidores en la tabla, se puede observar que el nicho de mercado a cubrir debe de ser una propuesta a medio camino entre la efectividad y la diversión con un enfoque más tradicional. Es ahí donde se ha situado la plataforma desarrollada en este proyecto.

5.1.1.5 Encuesta en Google Forms

Como punto final de esta fase se ha llevado a cabo una encuesta en Google Forms con el objetivo de analizar el uso de aplicaciones móvil en la población para poder incorporar características en la aplicación móvil que se concluyan que son beneficiosas para la mayoría de los usuarios. Como se verá a continuación, las preguntas incluidas en la encuesta son cuantitativas y no cualitativas. A la hora de realizar investigación primaria para el proyecto, se evaluaron dos posibles métodos. Un método basado en entrevistas en el cual se entrevistara a personas de manera individual y se les realizaran preguntas cualitativas con una respuesta más abierta. Y un segundo método basado en una encuesta con preguntas más impersonales y cuantitativas. Debido a las restricciones temporales del proyecto se ha

decidido optar por la segunda opción ya que se considera que es un método con el que se puede llegar a un mayor número de personas de manera más rápida. No obstante, se entiende que, a la hora de llevar a cabo una tarea de diseño para la experiencia de usuario, es más ilustrativo el primer método.

Las preguntas con sus respectivas opciones que se han incluido en esta encuesta se recogen en ANEXO II: encuesta sobre el uso de aplicaciones móvil. En el anexo también se recogen las respuestas a las preguntas. La encuesta ha sido respondida por un total de 60 personas.

Cabe destacar que las tres primeras preguntas de la encuesta no hacen referencia al uso de aplicaciones si no que se trata de preguntas para caracterizar el sector demográfico que las ha respondido para poder analizar los resultados teniendo esto en cuenta.

5.1.2 FASE DE DEFINICIÓN

En esta fase se continua con el proceso de diseño tomando como punto de partida la encuesta de la fase anterior. En esta fase se analiza la encuesta realizada y se extraen conclusiones de esta.

5.1.2.1 Conclusiones de la encuesta

La primera de las tareas a completar en esta fase es extraer conclusiones generales acerca de la encuesta. Esto ayuda a completar la siguiente tarea de esta fase en la que se define al usuario.

Las principales conclusiones extraídas de la encuesta son las siguientes:

- La mayoría de los encuestados son varones menores a 30 años con titulación universitaria.
- Un porcentaje alto de los encuestados utiliza aplicaciones de enseñanza al menos una vez por semana siendo tan solo un 16.7% el porcentaje de encuestados que nunca las utiliza.
- Los usuarios dan importancia al tipo de preguntas en las aplicaciones con pruebas siendo la opción de rellenar los huecos la más popular.

- Los usuarios utilizan sus móviles principalmente para ver vídeos, escuchar música y utilizar las redes sociales. Además, identifican las características propias de las redes sociales de compartir contenido como una característica deseable en las aplicaciones.
- Los encuestados dan importancia a elementos relacionados con la interfaz de usuario y la experiencia de usuario.
- La realimentación es identificada como algo deseable.
- La posibilidad de customizar un perfil también es identificada como una funcionalidad deseable por los encuestados.

5.1.2.2 Perfil de consumidor: ventajas e inconvenientes

Una vez se han extraído las principales conclusiones de la encuesta, utilizando estos enunciados junto con otros resultados de la encuesta se ha tratado de crear un perfil de usuario. En este perfil se indican cuáles son las características que el usuario ha identificado como ventajosas y no ventajosas en las aplicaciones móviles además de las tareas que el usuario busca completar mediante el uso de esta aplicación. Para este último punto se han supuesto cuales son las tareas que el usuario buscaría en la aplicación ya que ninguna de las preguntas hace referencia de manera directa a esta plataforma de enseñanza. Para ello se han utilizado las otras respuestas como punto de partida para poder realizar suposiciones necesarias en la tarea de diseño.

A continuación, se trata de reflejar esta información en la Tabla 3 con el objetivo de que sea más ilustrativo para el lector.

Ventajas	*Inconvenientes*	*Tareas que completar*
• Opción de aprender más • Incluir recompensas y retos	• Pruebas largas • Obtener notificaciones para volver a la aplicación	• Aprender sobre un tema • Sentirse realizado • Sentirse motivado para volver a la aplicación

• Posibilidad de compartir contenido • Pruebas con barra de progreso y tipo de preguntas clásico • Posibilidad de tener un perfil personalizable • Contenido dinámico • Recibir realimentación corta y positiva	• Leer mucha información • Contenido estático • Tener una moneda para compras dentro de la aplicación • Indicar el progreso con tiempo restante • Tipos de preguntas que impliquen más tiempo de respuesta	• Sentirse más seguro en el ciberespacio • Estar entretenido • Compartir la experiencia con otros usuarios

Tabla 3 Perfil de usuario

5.1.2.3 User Persona

En este epígrafe se define el user persona. Este concepto hace referencia a un personaje ficticio que trata de reflejar la personalidad de los potenciales usuarios de la aplicación. Crear este personaje ayuda a reflejar las necesidades de un grupo más grande de usuarios y se utiliza para continuar el proceso de diseño y lograr obtener un producto que satisfaga esas necesidades.

Los datos demográficos del user persona son los siguientes. Su nombre es Cristina y se trata de una estudiante de una carrera relacionada con el mundo tecnológico como ingeniería o el análisis de negocio. Cristina vive en Nueva York y tiene 20 años.

Cristina se caracteriza por ser una persona extrovertida a la que le gusta salir con sus amigos. Pasa bastante tiempo con su móvil y lo usa para jugar a videojuegos, utilizar aplicaciones de streaming de vídeo y música y utilizar redes sociales. Además, es una persona que muestra interés por las nuevas tecnologías.

En cuanto a los principales rasgos de su personalidad, Cristina es una persona innovadora, ambiciosa e independiente. A Cristina le gusta aprender por su cuenta y busca saber más acerca de los temas que le interesan.

Una vez descrita la personalidad del user persona y su manera de actuar, se describen las razones por las que un usuario de estas características utilizaría el producto. Las posibles razones identificadas han sido:

- Quiere aprender sobre un tema de ciberseguridad en concreto
- No quiere tener malas experiencias en las redes sociales
- Quiere trabajar para una compañía tecnológica y busca ampliar sus conocimientos
- Tiene mucha información personal en Internet y desea saber cómo mantener dicha información protegida.

Una vez definidas estas razones, se indican cuáles son los motivos principales que harían que un usuario utilizara la aplicación y cuáles no. Estas características se indican en la Tabla 4.

Ventajas	*Inconvenientes*
• Aprender sobre un nuevo tema en poco tiempo (5-10 minutos) • Desarrollar contraseñas más seguras • Mantener privacidad en internet • Aceptar solo cookies necesarias • No compartir datos de localización si no son necesarios	• Poco tiempo para aprender • Experiencia no atractiva: información muy técnica y manera de aprender estática • No está dispuesto a cambiar sus hábitos online • No cree que haya un riesgo real en internet

Tabla 4 Clasificación de características de la aplicación según el punto de vista del usuario

5.1.2.4 Escenario actual

Una vez se ha definido el user persona, se lleva a cabo una tarea que consiste en definir cuál es el escenario actual en el ámbito del aprendizaje utilizando aplicaciones móviles para dicho usuario. El objetivo de esta tarea es observar cómo se comporta dicho usuario ante la situación actual para encontrar puntos a mejorar y puntos que ya están sirviendo al usuario y llevar a cabo una tarea de rediseño en el producto.

Para definir el escenario actual se definen tres preguntas. Qué está haciendo el usuario, qué piensa cuando lo hace y cómo se siente. Estas preguntas se aplican a las distintas secciones que componen las aplicaciones de enseñanza. A continuación, se indica esta información en la Tabla 5 y en la Tabla 6.

	Haciendo un test	*Eligiendo un tema*
¿Qué está haciendo?	• Leyendo pregunta • Respondiendo • Leyendo el feedback	• Elegir un tema en concreto • Navegar por las categorías • Elegir aleatoriamente
¿Qué está pensando?	• ¿Qué tal está siendo mi desempeño? • ¿Por qué la respuesta es incorrecta? • ¿Cuántas preguntas quedan por responder? • ¿Puedo completar el curso correctamente?	• ¿Cuántos puntos voy a ganar? • ¿Cuánto tiempo me va a llevar? • ¿Qué tipo de preguntas son? • ¿Cuánto voy a aprender? • ¿Sobre qué trata este curso?
¿Cómo se siente?	• Interesado • Aburrido • Motivado	• Curioso • Cansado • Motivado

	• Cansado	• Aburrido

Tabla 5 Actitud del usuario en el escenario actual de aplicaciones de enseñanza I

	Accede al perfil	*Comparte contenido*
¿Qué está haciendo?	• Comprobar los puntos de experiencia del nivel • Mirar logros obtenidos • Cambiar información de usuario • Personalizar el perfil	• Compartir un logro • Invitar a alguien a usar la aplicación • Ver los logros de otros
¿Qué está pensando?	• ¿Cuántos puntos me quedan para subir de nivel? • ¿Cómo puedo conseguir más accesorios para mi perfil? • ¿Qué cursos he guardado?	• ¿Cómo puedo hacerlo mejor? • ¿Por qué otro usuario tiene más logros?
¿Cómo se siente?	• Motivado • Ansioso	• Sobrepasado • Orgulloso

Tabla 6 Actitud del usuario en el escenario actual de aplicaciones de enseñanza II

5.1.2.5 CJM (Customer Journey Map)

Esta tarea consiste en realizar un CJM del inglés Customer Journey Map. El CJM es un gráfico donde se intenta reflejar cual es el proceso que tiene que seguir un usuario para conseguir un objetivo. Este mapa se ha realizado teniendo en cuenta las mismas secciones que en el epígrafe anterior. El objetivo de este punto es una vez ya se han descrito en detalle

las distintas partes que componen el proceso, identificar donde se encuentran los puntos a mejorar y cuales ya están funcionando para aportar valor al usuario.

En la Figura 21 se puede observar el diagrama creado en Miro Board donde se identifican las ventajas e inconvenientes.

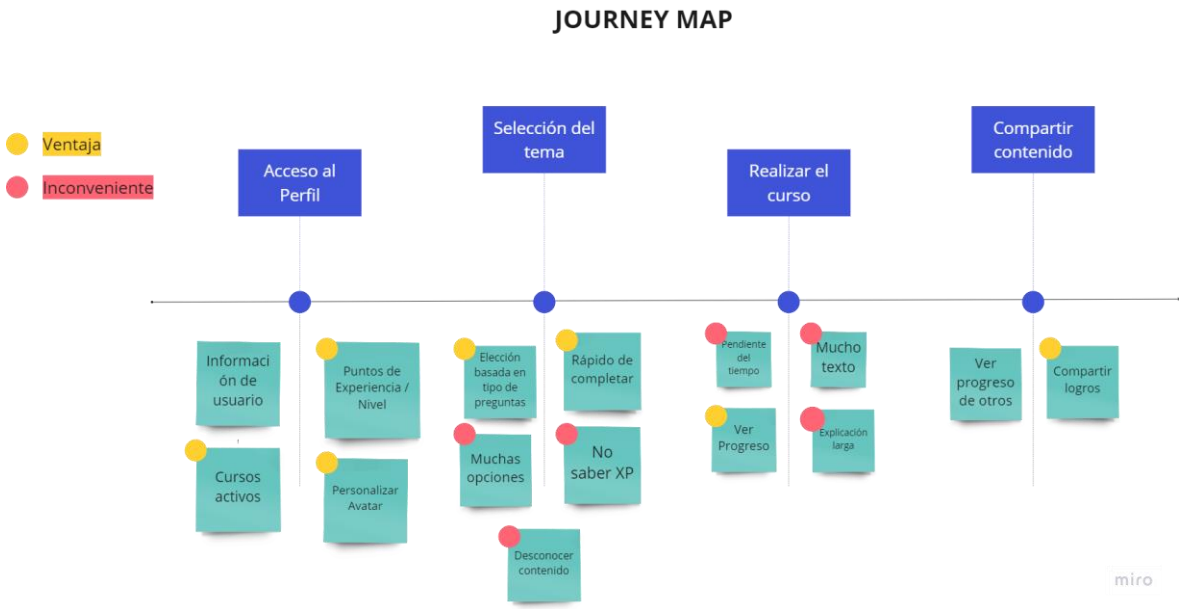


Figura 21 Diagrama del Journey Map

5.1.2.6 Conclusiones

En punto final de esta fase consiste en sacar conclusiones acerca de la experiencia actual que los usuarios, con características semejantes al user persona, tienen con las aplicaciones de enseñanza móviles. A través de estas conclusiones se identifican características o funcionalidades que deben incluirse en la aplicación para aportar valor al usuario.

Uno de los principales problemas es que los usuarios no tienen una manera fácil de acceder a información sobre ciberseguridad verificada en ninguna plataforma de enseñanza móvil. Este problema es abordado directamente con la idea general del proyecto

Los usuarios buscan motivación a la hora de completar los cursos y consideran como un punto ventajoso un sistema de niveles, logros y capacidad de conseguir artículos para

personalizar su perfil. Además, se ha identificado que dar la posibilidad al usuario de compartir sus logros y acceder a contenido compartido por otros usuarios es un punto que suma valor. Añadir este sistema de puntos, logros y una comunidad es una técnica para que el usuario no se sienta aburrido y quiera usar la aplicación periódicamente.

A la hora de acceder al perfil el usuario debe de poder encontrar información personalizada como los cursos que ha guardado para más adelante.

En cuanto a la dinámica de los cursos, los usuarios buscan ser capaces de completar los cursos en un tiempo reducido. Además, mientras se completan los cursos se ve que es preferible la ausencia de textos largos y realimentación muy detallada que pueden aburrir al usuario y no animarle a continuar. Debe recompensarse al usuario por completar el curso con algún logro. También se debe indicar al usuario cuál es su progreso en el curso. Respecto a la hora de elegir el curso a realizar, los usuarios deben de conocer sobre que trata el curso y poder elegir rápidamente. Por el contrario, el usuario puede verse ofrecido por una gran variedad de cursos de los cuales desconoce el tema y no sentirse motivado a completarlos.

5.1.3 FASE DE DESARROLLO

Esta fase consiste en dar forma a las conclusiones extraídas durante las dos fases anteriores del diseño. Durante la fase de descubrimiento y definición se lleva a cabo una tarea principalmente de investigación. En ellas se han definido los competidores de la aplicación, así como el tipo de cliente objetivo al que se dirige esta aplicación y las características que los consumidores buscan en aplicaciones de enseñanza. En esta etapa se trata de definir que funcionalidades y características debe de incluir la aplicación para resultar exitosa conociendo ya el público al que se dirige y la situación del mercado.

5.1.3.1 Lienzo de proposición de valor

Esta tarea trata de indicar las funcionalidades que debe de incluir la aplicación para resolver las necesidades descritas en el perfil de consumidor de la fase anterior. Estas pueden encontrarse en el Capítulo 5. en la sección 5.1.2 en el epígrafe Perfil de consumidor: ventajas e inconvenientes. Las funcionalidades que se han descrito en esta tarea son aquellas que

incorporan características que el usuario ha identificado como ventajosas y funcionalidades para aliviar las características no ventajosas de las aplicaciones móviles de enseñanza.

Las funcionalidades incorporadas para fomentar características ventajosas son:

- Avatar personalizable
- Niveles e insignias
- Opción de aprender más
- Opción de compartir contenido

Las funcionalidades que tienen como objetivo aliviar características no deseables en la aplicación son:

- Contenido dinámico y diferentes categorías sobre las que aprender.
- Pruebas con pocas preguntas que se puedan terminar rápidamente.
- Preguntas y respuestas con poco texto.

5.1.3.2 Tabla MoSCoW ("Must-Should-Could-Won´t")

Esta tarea consiste en la construcción de una tabla en la que se indican aquellas funcionalidades que la aplicación tiene que incluir ("Must"), debería incluir ("Should"), podría incluir ("Could") y que no va a incluir ("Won´t"). Es una manera de definir qué es y qué no es la aplicación. El marcar estos límites permite completar la siguiente tarea con mayor facilidad (definición del Producto Mínimo Viable) y saber qué características sí se podrían llegar a incluir sin salir de los límites definidos que cambiarían la esencia de la aplicación.

En la Tabla 7 pueden verse las funcionalidades descritas.

"Must"	**"Should"**
• Aplicación basada en pruebas • Contenido de ciberseguridad	• Incluir distintas temáticas • Incluir opción de compartir contenido

- Incluir niveles e insignias
- Incluir personalización de perfil
- Preguntas y respuestas cortas
- Pruebas cortas
- Opción de aprender más
- Diferentes tipos de ejercicios
- Premios por uso continuado
- Ser escalable a otro contenido

"Could"

- Tener monedas y tienda dentro de la aplicación
- Incluir sección de preguntas y respuestas
- Realimentación en formato vídeo
- Puntos de energía que limiten el uso de la aplicación por tiempo

"Won´t"

- No debe de tratarse de una aplicación de lecciones magistrales
- No debe de ser un juego
- No debe incluir largos textos
- No debe de estar basado en sesiones privadas entre un usuario y un profesor
- Los test no ponen a prueba un contenido sobre el que se ha aprendido anteriormente. Se debe de aprender como consecuencia de completar los cursos.

Tabla 7 Tabla MoSCoW

5.1.3.3 MVP (Mínimo producto viable)

Esta tarea consiste en la definición del mínimo producto viable. Se trata de una definición del producto que incluye unas características mínimas con las que satisfacer a los clientes principales y recibir realimentación.

En Mínimo producto viable es una aplicación de enseñanza móvil para Android e iOS que incluye:

- Información sobre ciberseguridad verificada

- Pruebas cortas
- Aprender realizando las pruebas
- Indicadores de progreso personal

5.1.3.4 Flujo de Usuario

Esta tarea consiste en diseñar cual será el flujo principal del usuario dentro de la aplicación. Ya se han descrito las principales funcionalidades de la aplicación y conocidas estas se puede indicar el flujo de acciones dentro de la misma.

En la Figura 22 y en la Figura 23 se puede observar el flujo comentado. Existen tres tipos de bloques explicados en la leyenda de la figura. Como puede observarse, el gráfico se ha dividido en dos figuras para poder mostrar toda la información claramente.

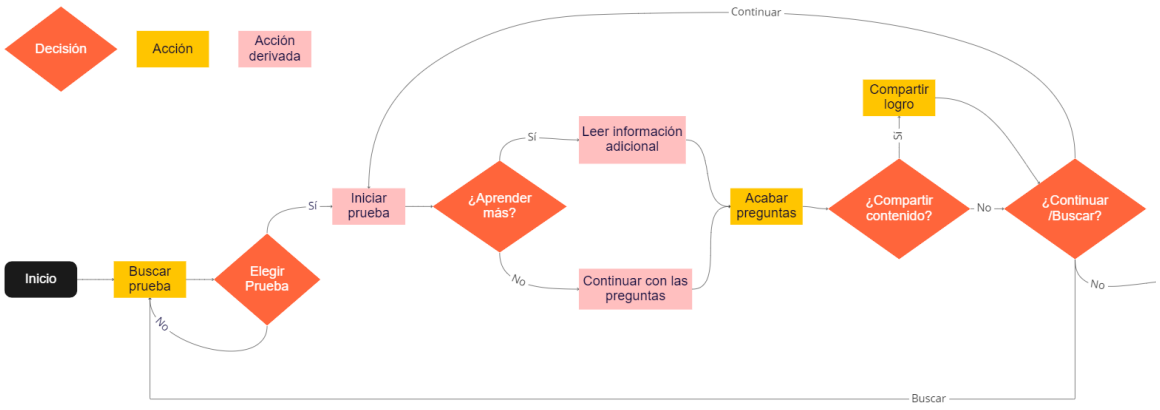


Figura 22 Flujo de usuario I

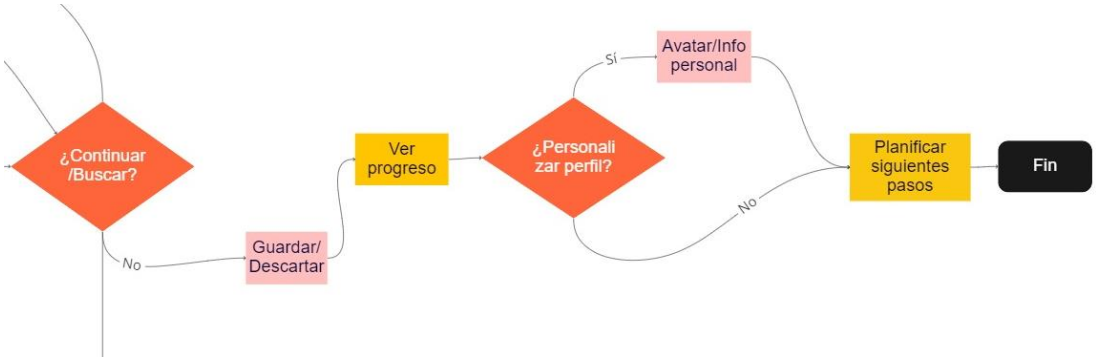


Figura 23 Flujo de usuario II

5.1.3.5 Arquitectura de la información

La última de las partes de esta fase consiste en describir como se organiza la información dentro de la aplicación desde el punto de vista del usuario. En este diagrama se indican las cuatro principales secciones que el usuario encontrará en la aplicación. Una sección dedicada al perfil donde el usuario será capaz de acceder a sus cursos guardados, modificar su avatar, modificar otros detalles relativos a la cuenta como la contraseña o el nombre de usuario y, por último, verificar su nivel y puntos de experiencia.

Otra de las secciones será la sección de cursos. En ella el usuario podrá observar cual es el curso recomendado, así como acceder a las diferentes categorías de los cursos. Dentro de las 4 principales categorías que ofrece la aplicación el usuario podrá navegar para encontrar la prueba que desee. Además, dentro de esta sección en caso de que el usuario haya dejado un curso a medias también podrá ver información relativa al mismo y acceder de nuevo para completarlo.

La tercera de las secciones es la sección del curso recomendado. En ella se muestra la descripción de un curso en concreto el cual aportará un mayor nivel de puntos de experiencia al usuario y podrá ser modificado periódicamente con el objetivo de que el usuario se sienta motivado a utilizar la aplicación regularmente.

La última de las secciones hace referencia a la opción de compartir contenido. No se comentará de manera específica esta sección ya que no forma parte del desarrollo que conforma este proyecto.

En la Figura 24 se puede observar la información descrita en formato de diagrama.

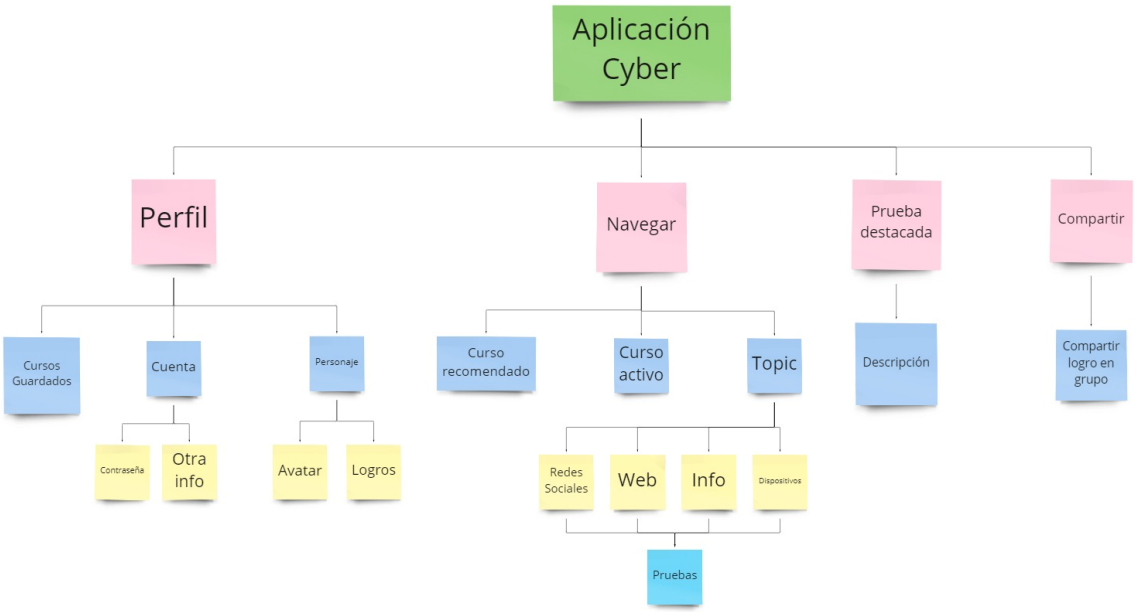


Figura 24 Diagrama de información de la aplicación

5.1.4 FASE DE ENTREGA

Esta fase consiste en dar forma a las funcionalidades y características descritas en la fase anterior. Para ello se crea un prototipo en el que se refleja principalmente la arquitectura de la información y el flujo de usuario. Como se ha podido observar en la descripción de las diferentes fases de diseño, el proceso es gradual y las tareas completadas sirven de punto de partida para las siguientes. A la hora de crear la arquitectura de la información, el flujo de usuario y la tabla MoSCoW es necesario haber realizado toda la tarea previa de investigación sobre la situación del mercado y la actitud de los usuarios.

El prototipo creado tiene tres fases. Cada fase es un perfeccionamiento de la anterior. En los siguientes puntos se muestran los prototipos creados y se describen sus características

principales. **Se debe de tener en cuenta que los prototipos no representan el estado y las características finales de la aplicación.**

5.1.4.1 Prototipo de fidelidad baja

El prototipo de fidelidad baja debe de reflejar las primeras impresiones que se han obtenido del diseño sin entrar en un nivel de detalle profundo ya que para ello están las otras dos versiones del prototipo. Para este prototipo no se ha utilizado la herramienta Figma si no que se ha llevado a cabo en papel.

En la Figura 25 se puede observar el diseño realizado para uno de los pilares de la aplicación que es el flujo de los cursos. En esta figura se ha descrito la página de acceso a la aplicación en la que se debe de indicar el nombre de usuario y la contraseña. Tras iniciar sesión en la aplicación el usuario sería redirigido a una página en la que se le mostrarían las 4 categorías sobre las que podría aprender: Redes Sociales, Web, Dispositivos e Info. También tendría la opción en esta pantalla de pulsar el botón navegar o de ver el curso recomendado y acceder directamente a él. Cuando el usuario pulse un botón de las categorías o el botón de navegar se le redirigiría a otra página con una lista de los cursos. Si ha pulsado en una categoría en concreto la lista estaría filtrada. Una vez en la lista el usuario tiene la opción de pulsar la tarjeta del curso que desee completar, navegando de esta manera a una descripción más detallada del curso.

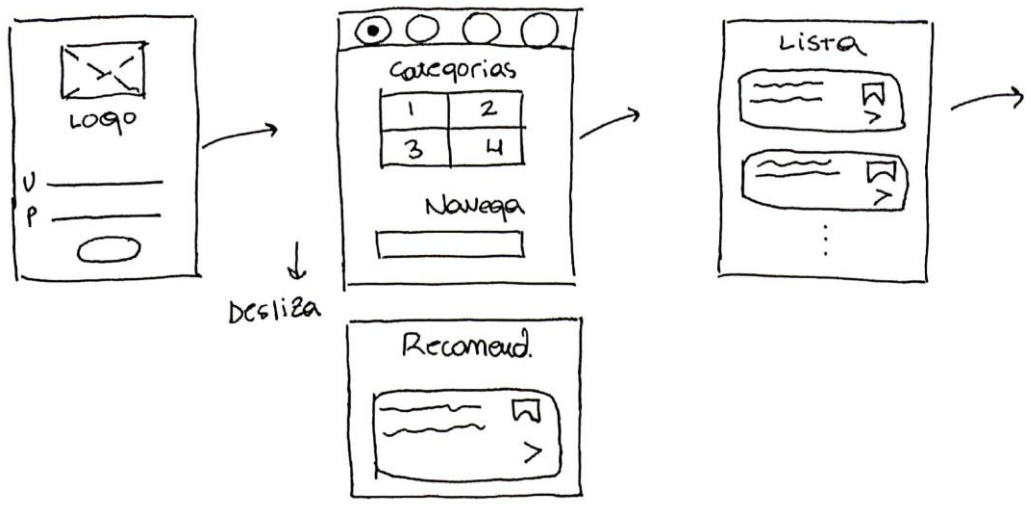


Figura 25 Pantallas del prototipo de fidelidad baja I

En la Figura 26 se muestra la continuación del flujo de los cursos. La primera de las ventanas corresponde a la descripción detallada del curso desde donde se puede comenzar a completarlo. En la siguiente página se muestra como serían las pantallas correspondientes a las diferentes preguntas que compondrían el curso. Una vez respondida la pregunta, aparecería en la pantalla una ventana de diálogo mostrando la realimentación y dando la opción al usuario de continuar o aprender más acerca de la pregunta. Los puntos suspensivos tratan de indicar que esta situación se repetiría hasta terminar el curso.

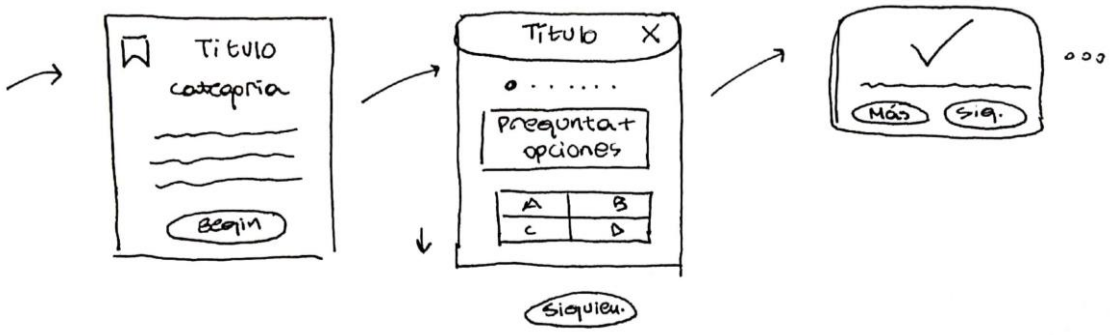


Figura 26 Pantallas del prototipo de fidelidad baja II

Una vez terminado el curso, se accedería a las dos páginas mostradas en la Figura 27. Si el usuario consigue la insignia asociada al curso, se le muestra y se le da la enhorabuena. Una vez el usuario pase a la siguiente página se le mostraría una tarjeta con el siguiente curso a completar y se le daría la opción de volver a la ventana de navegar o compartir su logro.

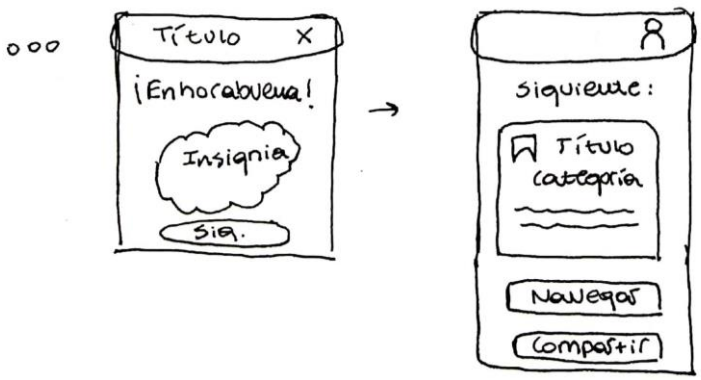


Figura 27 Pantallas del prototipo de fidelidad baja III

En la Figura 28 se pueden ver las ventanas correspondientes a dos de las otras secciones de la aplicación. La primera página muestra la sección del curso destacado, en ella se mostraría la insignia asociada a dicho curso y la posibilidad de ir al curso. La segunda y tercera página muestran la sección de los grupos. En ellas el usuario podría ver una lista con los grupos a los que pertenece y una vez pulsara en las tarjetas que se muestran sería redirigido a una lista con las notificaciones relativas al grupo en las que se mostrarían qué usuarios han completado determinados cursos.

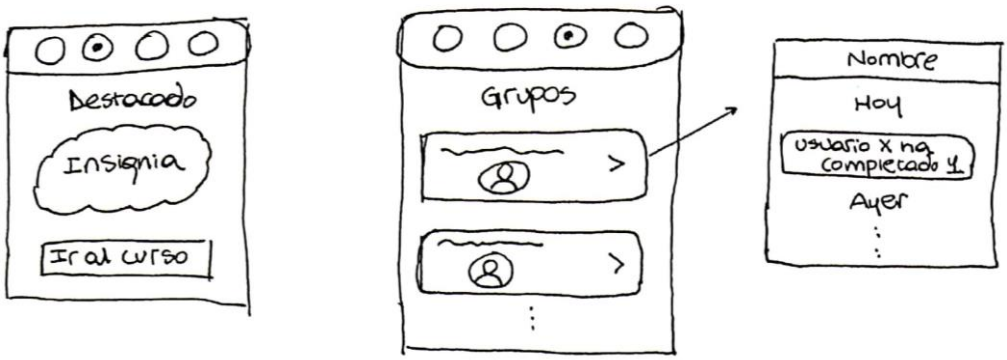


Figura 28 Pantallas del prototipo de fidelidad baja IV

En la Figura 29 se puede observar las páginas relativas a la última de las secciones, la sección del perfil de usuario. Se puede ver hay una página principal a la que accedería el usuario mediante el menú de navegación mostrado en la parte superior de la pantalla. En esta página principal el usuario vería información relativa a su perfil como su avatar, su nivel y su nombre de usuario. El botón de logros redirigía al usuario a una pantalla en la que aparecerían sus insignias. El usuario tendría la posibilidad de modificar su avatar desde la ventana mostrada encima de las insignias. A ella accedería pulsando el icono de al lado del avatar. Por último, también se puede observar la presencia de un botón en la parte superior izquierda de la ventana el cual mostraría un menú desde el que el usuario podría ver sus cursos guardados, modificar su información, cerrar sesión o acceder a la ayuda de la aplicación.

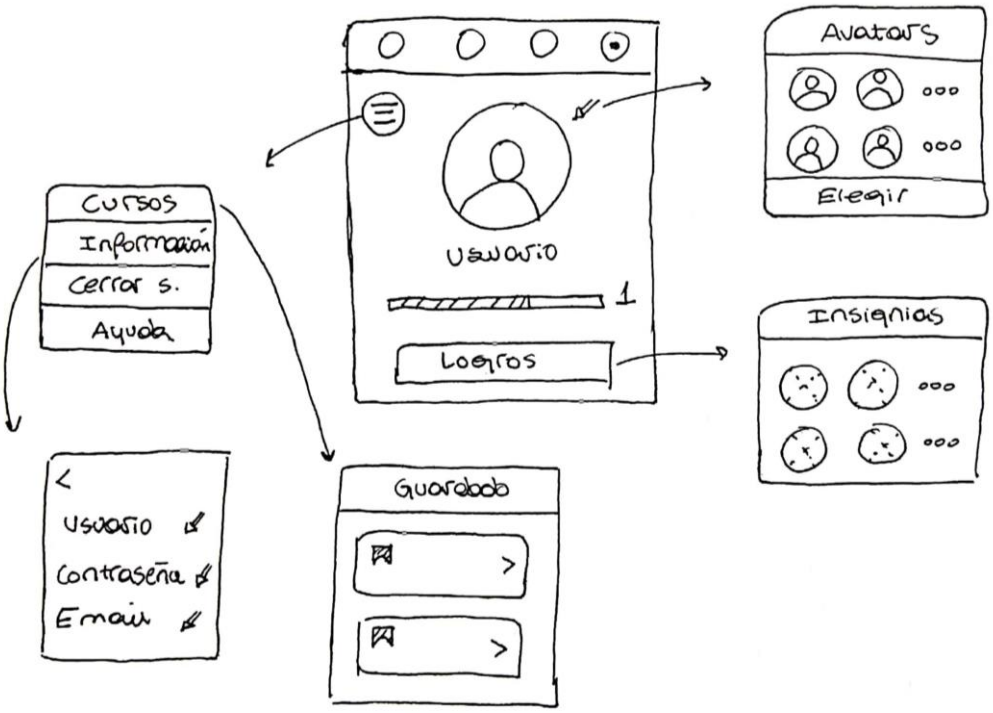


Figura 29 Pantallas del prototipo de fidelidad baja V

5.1.4.2 Prototipo de fidelidad media

Este prototipo trata de perfeccionar el prototipo de fidelidad baja y para ellos se ha llevado a cabo en Figma. A continuación, se mostrarán las pantallas creadas para las diferentes secciones de la aplicación que ya se han mencionado en el prototipo de fidelidad baja. En ellas se podrá observar cómo se ha dado forma a las ideas reflejadas anteriormente y las diferencias implementadas.

Primero se va a analizar la sección relativa a los cursos. En la Figura 30 se pueden observar 3 pantallas que muestran en qué consistiría la experiencia de elegir un curso. En la primera pantalla se puede observar una diferencia respecto al prototipo de fidelidad baja y es que aparece una tarjeta de un curso. Esta tarjeta representa el curso que el usuario ha podido dejar a medias y podrá volver a él desde esta pantalla qué es el panel de usuario. Las otras pantallas muestran la lista de los cursos y como se mostraría la descripción de un curso. Las

pantallas dan forma a los bloques reflejados en el prototipo de fidelidad baja con un mayor nivel de precisión y realismo respecto a lo que será la aplicación final.

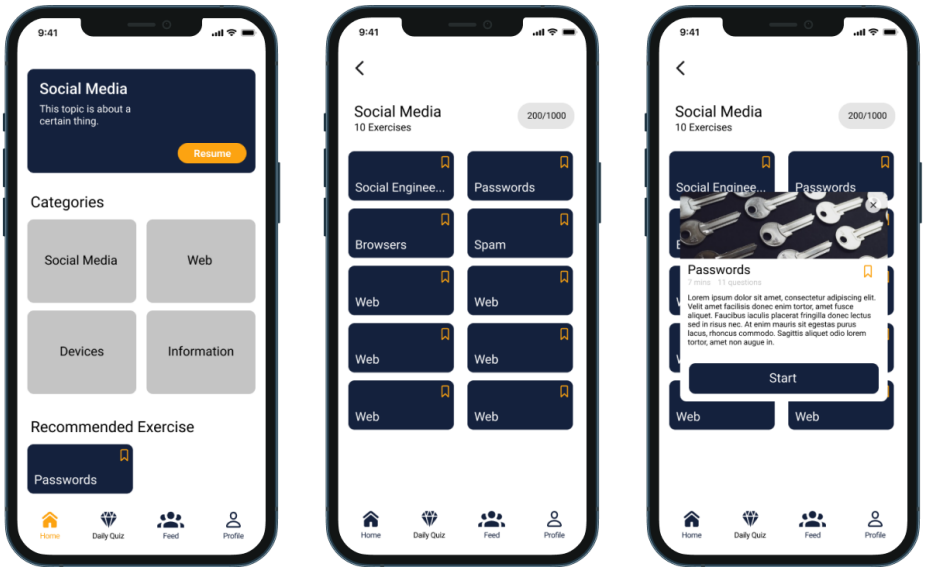


Figura 30 Pantallas del prototipo de fidelidad media I

En la Figura 31 se pueden observar las pantallas que conforman la experiencia de completar un curso. Las dos primeras pantallas muestran dos tipos de preguntas dentro de una prueba: una pregunta con respuesta múltiple y una pregunta de tipo rellenar los huecos. Las otras dos pantallas muestran cómo se vería la obtención de una insignia y el progreso dentro de una categoría. Cabe destacar que esta última pantalla difiere de la idea presentada en el epígrafe anterior ya que en el prototipo de fidelidad baja se incluía una pantalla que ofrecía al usuario la posibilidad de ir al siguiente curso mientras que en este prototipo se muestra el progreso de la categoría y no se da la opción de continuar al siguiente curso sino de finalizar el curso activo lo cual redirigiría al usuario al panel inicial.

Por otro lado, en la Figura 32 se puede ver como se vería la realimentación relativa a cada pregunta. En ella se muestran dos pantallas para mostrar también la opción de aprender más que muestra una explicación más detallada acerca de la respuesta.

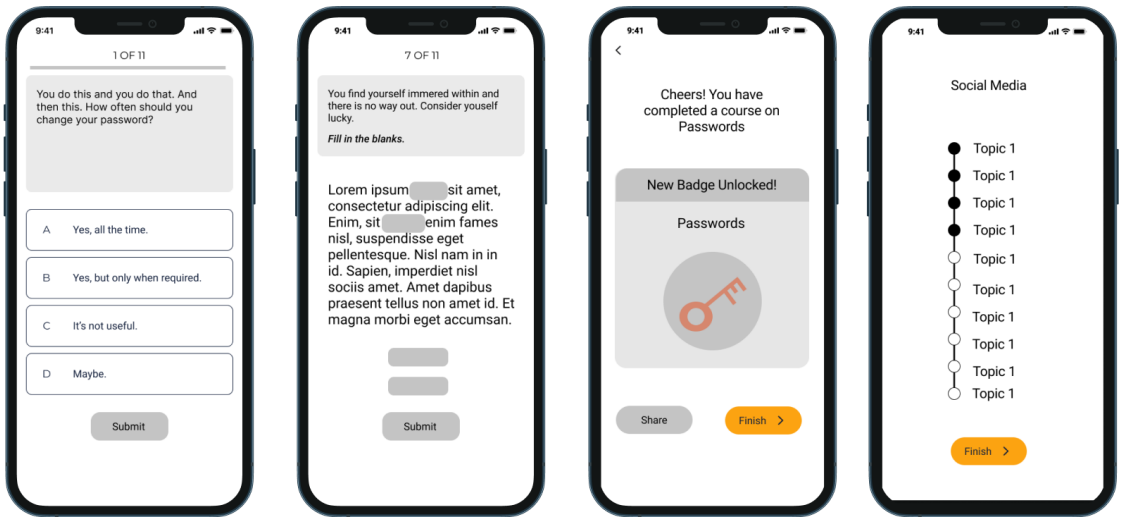


Figura 31 Pantallas del prototipo de fidelidad media II

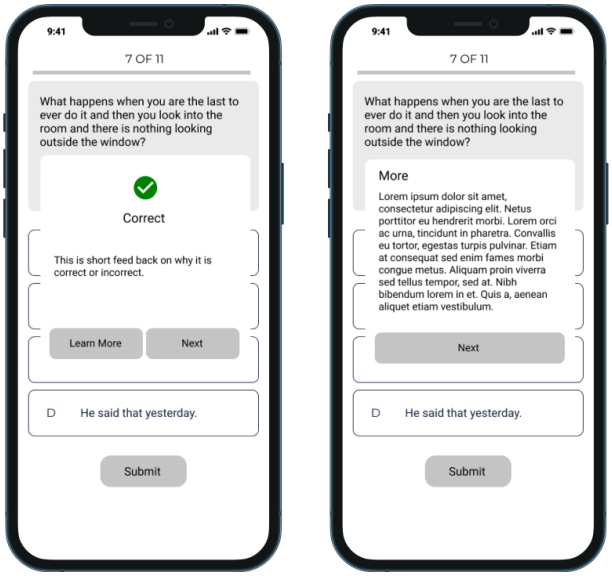


Figura 32 Pantallas del prototipo de fidelidad media III

En la Figura 33 se muestran las secciones del curso destacado y de los grupos. En cuanto a la sección de los grupos se ha entrado en un mayor nivel de detalle respecto a la lógica de los grupos. Es posible que el usuario no pertenezca a ningún grupo y quiera unirse a uno. En este caso la principal pantalla que se le mostrará será la segunda de la figura. Si el usuario pulsa en el icono verá la tercera pantalla y tendrá la posibilidad de unirse a un nuevo grupo mediante un código. En caso contrario, es decir, que el usuario pertenezca a grupos, sí

aparecerían los mismos en la pantalla. Esta situación no se ha reflejado. En la última de las pantallas también se han incluido varias características no presentes anteriormente. Se incluye un icono desde el cual el usuario puede abandonar el grupo y se han indicado distintos tipos de notificaciones. En cada tipo de notificación se puede llevar a cabo una acción distinta al apretar el botón asociado. Se han reflejado todos los tipos de notificación ya que en este punto del diseño todavía no se había decidido cual era la funcionalidad que se deseaba implementar.

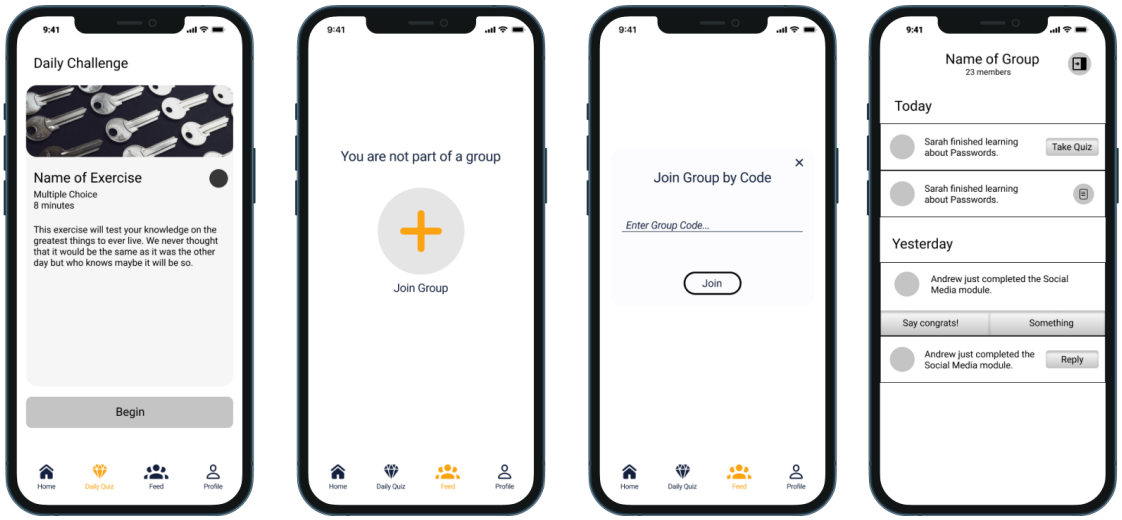


Figura 33 Pantallas del prototipo de fidelidad media IV

En la Figura 34 se muestran tres de las pantallas relativas a la sección de perfil. En la primera de las pantallas se puede observar la página inicial del perfil de usuario la cual implementa varios cambios respecto al prototipo de baja fidelidad. Como se puede observar, no se ha incluido el botón para mostrar el menú de opciones si no que estas opciones ahora aparecen en forma de botones en la parte inferior de la pantalla. El usuario puede modificar su avatar pulsando en el botón “Avatars” el cual lo redirige a la segunda de las pantallas.

Figura 34 Pantallas del prototipo de fidelidad media V

En la Figura 35 se muestran las pantallas restantes relativas a esta sección. Si el usuario pulsa el botón “My Courses” accede a la tercera de las ventanas donde podrá observar qué cursos ha guardado organizados por categorías. Por el contrario, si el usuario pulsa el botón “My badges” accede a la segunda de las pantallas donde se muestran las insignias desbloqueadas. Para acceder a la primera pantalla y modificar información relativa a la cuenta, el usuario debe de presionar el icono de la esquina superior derecha presente en la pantalla principal mostrada en la figura anterior.

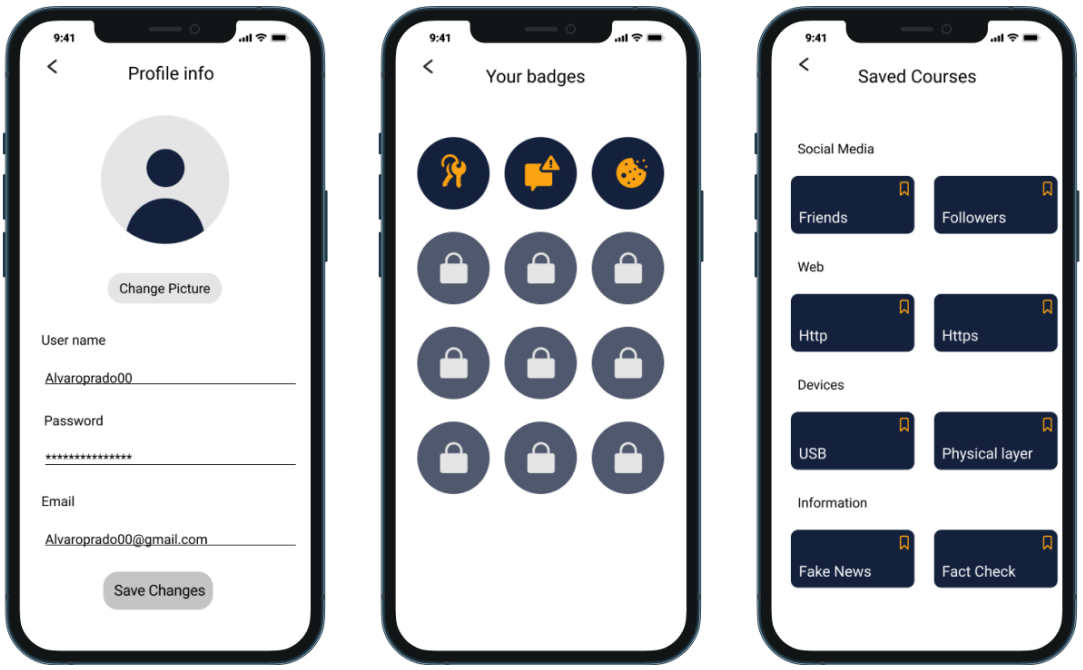


Figura 35 Pantallas del prototipo de fidelidad media VI

En la Figura 36 se pueden observar las últimas pantallas diseñadas en el prototipo de fidelidad media. Estas pantallas no se han especificado en el prototipo de fidelidad baja y muestran el flujo que el usuario seguiría para autenticarse y acceder al panel principal o para darse de alta en la aplicación creando una nueva cuenta. Puede observarse que se ha creado una pantalla que permite al usuario unirse a un grupo directamente antes de crear su cuenta en caso de que cuente con el código que identifica dicho grupo.

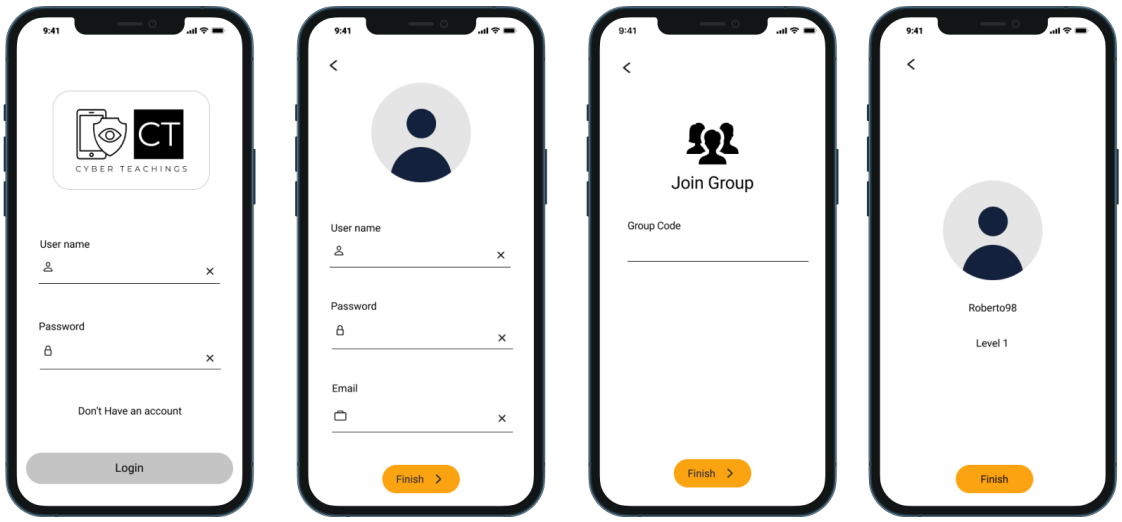


Figura 36 Pantallas del prototipo de fidelidad media VII

5.1.4.3 Prototipo de fidelidad alta

En este apartado no se van a adjuntar las figuras del diseño que se ha llevado a cabo en Figma. Esto se debe a que en la fase de desarrollo sí se adjuntarán las pantallas ya que son necesarias para explicar el proceso y las pantallas de la aplicación corresponden con las del prototipo de fidelidad alta.

5.1.4.4 Aclaraciones

Es importante recalcar que la sección de los grupos no forma parte del trabajo realizado en este proyecto en cuanto al desarrollo del código y de la interfaz de usuario. No obstante, sí forma parte del proceso de diseño.

5.2 DIAGRAMAS UML

En este apartado se van a presentar distintos diagramas UML que se llevaron a cabo antes de comenzar con el desarrollo de la aplicación en Android Studio. Estos diagramas han tenido como objetivo facilitar la tarea de programación. Se presentan en el informe con el objetivo de que el lector se familiarice más con los datos y el funcionamiento de la aplicación y entienda con mayor facilidad el apartado de desarrollo.

5.2.1 DIAGRAMA DE CASOS DE USO

El diagrama de casos de uso identifica las acciones que los distintos actores pueden llevar a cabo en la aplicación. En esta aplicación hay dos autores principales, el usuario y el administrador. El administrador puede realizar todas las acciones que el usuario normal puede realizar más otras acciones exclusivas que son otorgadas por su nivel de autorización. En la Figura 37 se muestran los casos de uso relativos al usuario.

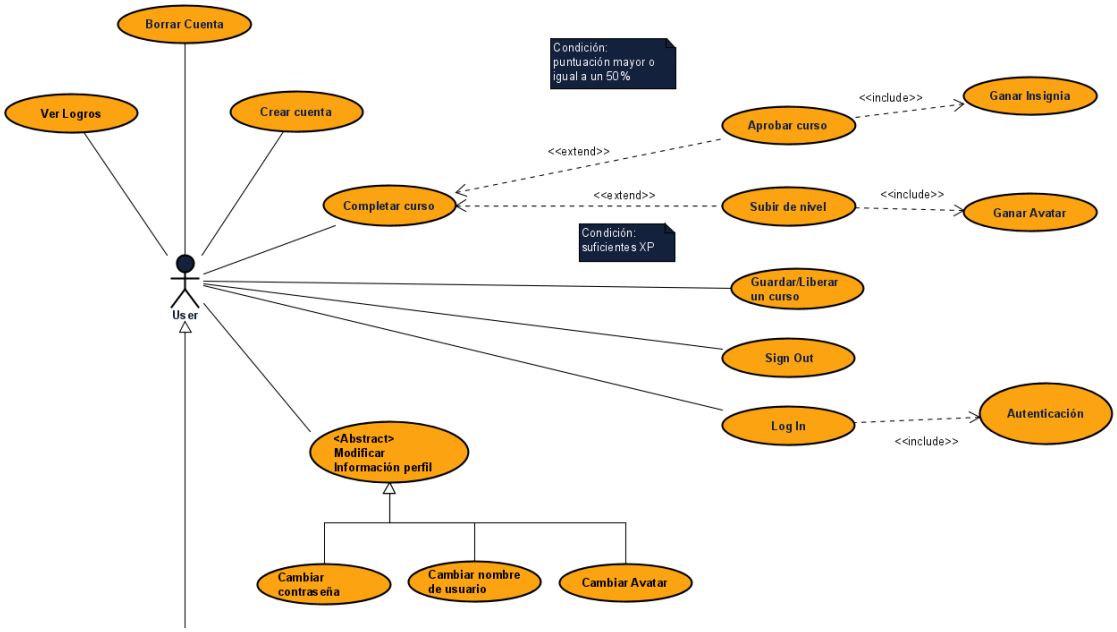


Figura 37 Diagrama de casos de uso I

En la Figura 38 se muestran los casos de uso relativos a los usuarios con rol de administrador de la aplicación. Como puede observarse el administrador está conectado con el actor usuario mediante una flecha. Esta flecha indica herencia de casos de uso como se ha explicado anteriormente.

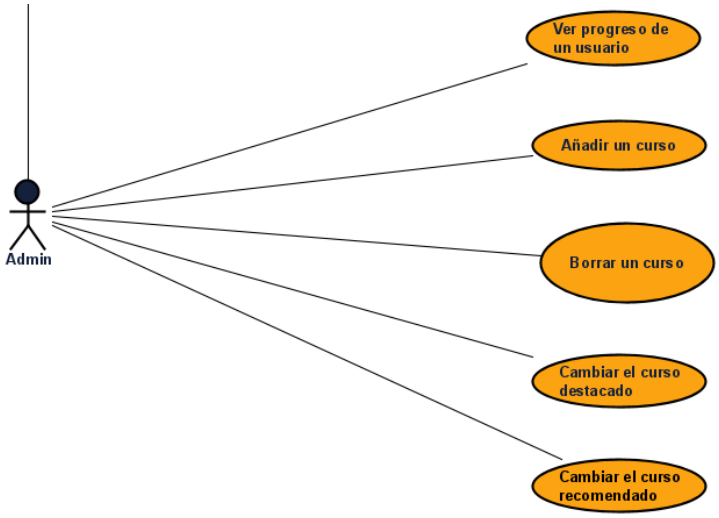


Figura 38 Diagrama de casos de uso II

5.2.2 DIAGRAMA DE CLASES

En este apartado se presenta el diagrama de clases de la aplicación. Se explicarán ciertas propiedades de las clases con el objetivo de reducir las explicaciones en el apartado de desarrollo.

En la figura se puede observar la primera parte del diagrama de clases de la aplicación. En la figura se adjunta la segunda parte de este. A continuación, se explicarán las clases de manera individual.

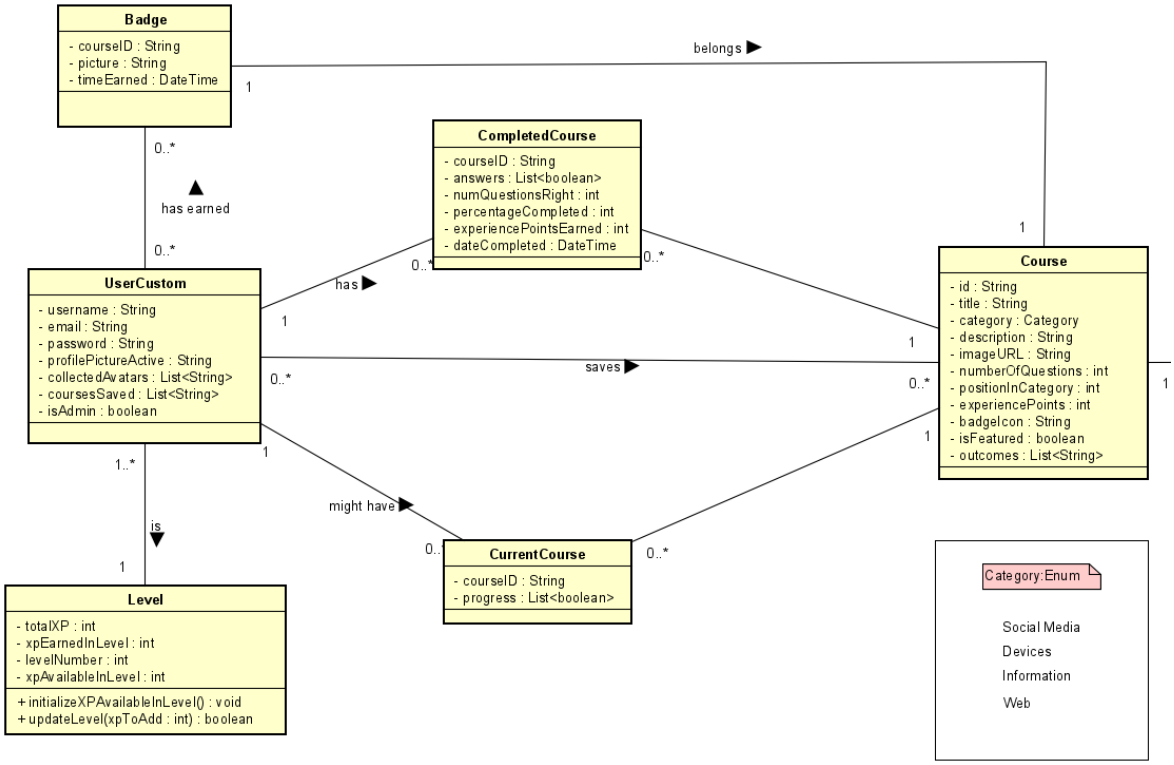


Figura 39 Diagrama de clases parte I

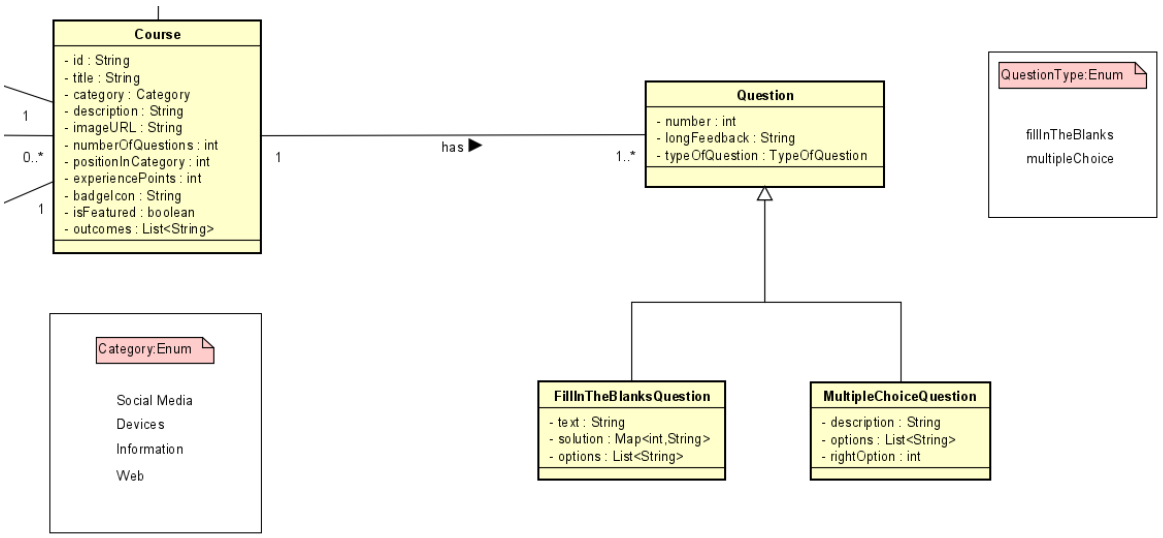


Figura 40 Diagrama de clases parte II

5.2.2.1 Clase UserCustom

Esta clase se utiliza para identificar a los usuarios de la aplicación. Los atributos "email" y "password" son utilizados para identificar al usuario unívocamente en la aplicación y permitirle el acceso mediante el sistema de autenticación ofrecido por Firebase Authentication. El usuario también cuenta con un nombre de usuario ("username") que también es único. A continuación, se enumeran y explican el resto de los atributos:

- "profilePictureActive": este atributo es un String que identifica el avatar del usuario. Se trata de un String debido a que con este atributo se realiza una llamada HTTP a la API de Robohash con la que se construye la imagen del Avatar. Robohash es una aplicación web que devuelve imágenes ante llamadas de tipo GET. Para cada String asociado al final de la URL, Robohash devuelve una imagen única. Flutter ofrece una clase específica con la que se puede construir una imagen a través de una URL. La URL con la que se realiza la llamada es: "https://robohash.org/valor-atributo" En caso de que el usuario cambie su avatar, se cambia el valor de este atributo por otro String y se realiza la llamada con el nuevo valor.
- "collectedAvatars": este atributo es una lista de String con todos los String que se corresponden con los avatares que el usuario ha ganado. Este atributo determina que avatares disponibles tiene el usuario.
- "coursesSaved": este atributo es una lista de String formada por todos los identificadores de los cursos que el usuario ha guardado. Cuando un usuario guarda un curso para completarlo posteriormente se añade un String con el identificador del curso en la Database. Es importante resaltar que se ha establecido una relación entre el usuario y los cursos en el diagrama, pero el usuario no tiene como atributo instancias de tipo Course sino que la relación se establece como se ha explicado mediante el identificador del curso. Esta decisión de diseño se ha llevado a cabo para que, si el curso es modificado de alguna manera en la base de datos, los cambios se vean reflejados en los cursos guardados. Por el contrario, si el usuario guardara instancias de los cursos estos cambios no se verían reflejados.

- "isAdmin": este booleano sirve para diferenciar el rol del usuario en la aplicación. En caso de que el booleano este a "true", el usuario será identificado como administrador y tendrá acceso a las funcionalidades descritas en el diagrama de casos de uso.

En cuanto a las relaciones con otras clases, el usuario también cuenta con un atributo de tipo Level que indica su nivel y una lista de objetos tipo Badge con todas las insignias que ha ganado dentro de la aplicación. No se han incluido estos atributos en la propia clase UserCustom ya que se refleja mediante las relaciones entre clases.

El usuario también cuenta con una lista llamada "coursesCompleted". Esta lista está formada por objetos de tipo CourseCompleted donde se almacenan aquellos cursos completados con el usuario.

Por último, el usuario tiene un atributo llamado "currentCourse" que puede tener valor null o ser una instancia de un objeto tipo CurrentCourse. Este atributo identifica el curso que el usuario ha podido dejar a medias y en él se registra en qué estado lo ha dejado.

5.2.2.2 Clase Badge

Esta clase identifica las insignias de los cursos. Tiene 3 atributos:

- "courseID": este atributo identifica el curso al que le corresponde la insignia. De nuevo, como se ha explicado anteriormente en el caso del atributo "coursesSaved", la relación con el curso se establece mediante este atributo donde se guarda el identificador de la base de datos del curso.
- "picture": este atributo de tipo String sirve para obtener la imagen de la insignia. El método para obtener esta imagen merece una explicación detallada. La imagen de las insignias se obtiene de una página web de iconos llamada FontsAwesome. Para obtener esta imagen se ha utilizado una librería externa en la aplicación que permite obtener cualquier icono de la web mediante una clase llamada FontsAwesome en la que están descritos como atributos estáticos de clase todos los valores con los que se puede realizar una llamada a la página web. La imposibilidad de almacenar este valor

en la base de datos por no ser un tipo de dato básico ha exigido crear un mapa en la aplicación en el que se identificaban los valores descritos en la clase FontsAwesome con Strings. Siendo estos Strings los valores que se guardan en el atributo.

- "timeEarned": es un atributo de tipo DateTime en el que se especifica la fecha en la que se ha ganado la insignia para mostrar dicha información al usuario.

5.2.2.3 Clase Level

El usuario tiene asociado un nivel, ese nivel se guarda en la clase usuario en un atributo llamado "level" de tipo Level. Esta clase cuenta con los siguientes atributos:

- "totalXP": en este atributo se almacena el número total de puntos de experiencia que el usuario ha acumulado.
- "xpEarnedInLevel": en este atributo se indica cuantos puntos de experiencia tiene el usuario dentro del nivel en el que se encuentra. Es decir, si el usuario es nivel 3 y ese nivel cuenta con 1700XP, este atributo indicaría en qué punto se encuentra entre 0 y 1700.
- "levelNumber": atributo de tipo int que indica el número del nivel.
- "xpAvailableInLevel": este atributo indica el número de puntos de experiencia con los que cuenta el nivel. Cada nivel tiene un número de puntos de experiencia distinto. Los puntos de experiencia quedan limitados por la siguiente operación: $(levelNumber - 1) * levelScale + baseLevel$. Los valores "baseLevel" y "levelScale" son dos constantes descritas en el código cuyo valor es 200 y 1000, respectivamente. Esto quiere decir que todos los niveles cuentan con una base de 1000 puntos de experiencia y cada nivel cuenta con 200 puntos de experiencia más que el nivel anterior.

En cuanto a los métodos descritos en esta clase, son los siguientes:

- "initializeXPAvailableInLevel()": este método se utiliza en el constructor y sirve para inicializar el campo "xpAvailableInLevel" cuando se crea el nivel por primera vez.

- "bool updateLevel(int xpToAdd)": este método se utiliza para actualizar el nivel cuando un usuario completa un curso y gana puntos de experiencia. Se indica por parámetro el número de puntos de experiencia que el usuario ha ganado y mediante la lógica pertinente se actualiza el campo "xpEarnedInLevel" y "totalXP". En caso de que el usuario suba de nivel también se actualiza el campo "levelNumber" y solo en este caso se devuelve true como valor para saber que el usuario ha subido de nivel y actuar consecuentemente.

5.2.2.4 Clase CompletedCourse

Esta clase se utiliza para almacenar en el usuario en forma de lista todos los cursos que ha completado. Los atributos con los que cuenta la clase CompletedCourse son los siguientes:

- "courseID": atributo de tipo String con el que se identifica este objeto con el curso que le corresponde.
- "answers": este atributo es una lista de boolean en donde se almacenan las respuestas del usuario en el curso. En Dart las listas son ordenadas, por tanto, se sabe cuál es el resultado para cada una de las preguntas que componen el curso. Es decir, si el usuario ha respondido correctamente la 3ª pregunta del curso, este atributo contendrá el valor "true" en la tercera posición de la lista.
- "numQuestionsRight": para que sea más sencillo saber cuántas preguntas ha acertado el usuario en un curso sin tener que recorrer la lista definida en "answers", se guarda este número en este atributo.
- "percetangeCompleted": con este atributo se identifica el porcentaje de preguntas que el usuario ha respondido correctamente. Este valor se utiliza para determinar si el usuario obtiene la insignia asociada al curso cuando es mayor o igual que 50% como se describe en el diagrama de casos de uso.
- "experiencePointsEarned": indica el número de puntos de experiencia del total disponible en el curso que el usuario ha obtenido.
- "dateCompleted": indica la fecha en la que el usuario ha completado el curso.

5.2.2.5 Clase CurrentCourse

Esta clase se utiliza para almacenar una referencia al curso que el usuario ha guardado para completar más adelante además de su progreso en el mismo. Sus atributos son:

- "courseID": referencia al identificador del curso que le corresponde.
- "progress": una lista ordenada de booleanos que indica los resultados que el usuario ha obtenido en el curso hasta el momento.

5.2.2.6 Clase Course

Una de las clases principales de la aplicación. Esta clase se utiliza para modelar los cursos. A continuación, se indican sus atributos:

- "id": guarda el valor del identificador del curso asignado en la base de datos. Se ha decidido almacenar este valor en la aplicación para facilitar ciertas operaciones en el desarrollo como, por ejemplo, crear una instancia de "CurrentCourse" indicando el identificador sin tener que acceder a la base de datos para obtenerlo.
- "title": String que indica el título del curso.
- "category": este atributo es de tipo Category, una enumeración definida en el diagrama de clases. Esta enumeración cuenta con cuatro valores que son las cuatro categorías de cursos sobre los que se puede aprender en la aplicación: Dispositivos, Web, Redes Sociales e Info.
- "description": String que indica una descripción detallada del contenido del curso.
- "imageURL": String que almacena el valor de la URL de la imagen asociada al curso. Como se ha indicado anteriormente Flutter cuenta con una clase que permite la creación de imágenes mediante la especificación de una URL. Los cursos cuentan con una imagen que se muestra en la pantalla de descripción del curso.
- "numberOfQuestions": atributo para guardar el número de preguntas que tiene un curso.
- "positionInCategory": indica la posición que ocupa el curso dentro de su categoría.

- "experiencePointsAvailable": indica el límite de puntos que se pueden obtener al completar el curso.
- "badgeIcon": en este atributo se almacena el String que identifica el icono de la insignia asociada al curso. Con este String se obtiene el icono mediante el mapa mencionado anteriormente en la expliación de la clase "Badge". Cabe destacar que el curso no cuenta con un atributo tipo "Badge" sino que únicamente almacena este valor ya que no es necesario guardar más información acerca de la insignia, basta con su imagen para mostrarla al usuario en caso de que la obtenga. Para el resto de información relativa al logro se crea una instancia de tipo "Badge" que se asocia al usuario.
- "isFeatured": este atributo indica si el curso es el curso destacado. En ese caso el número de puntos de experiencia que el usuario puede obtener del curso se multiplica por dos.
- "outcomes": una lista de Strings en la que se identifican los resultados de aprendizaje esperados del curso. Estos se muestran en la pantalla de descripción del curso.

5.2.2.7 Clase Question

Esta clase define las preguntas que están asociadas a los cursos. Cada curso cuenta con un atributo llamado "questions" que es una lista de objetos de tipo Question. Se trata de una clase donde se definen ciertos atributos comunes a todos los tipos de preguntas. Sus atributos se explican en la siguiente enumeración:

- "number": se trata de un atributo de tipo int que indica el número de pregunta del curso. Es decir, si su valor es 3, la pregunta será la 3ª pregunta del curso.
- "longFeedback": atributo de tipo string donde se almacena la explicación detallada de la pregunta. Este texto es el que se muestra al usuario cuando este pulsa la opción de aprender más acerca de una pregunta cuando se le ofrece la realimentación.
- "typeOfQuestion": este atributo es de tipo TypeOfQuestion, una enumeración con dos posibles valores: multipleChoice y fillInTheBlanks. Esta enumeración determina

los dos tipos de preguntas que se han incluido en los cursos de la aplicación que son preguntas de opción múltiple y de rellenar los huecos.

5.2.2.8 Clase FillInTheBlanksQuestion

En esta clase se definen aquellos atributos únicos a este tipo de pregunta los cuales se explican a continuación:

- "text": este atributo de tipo String almacena el texto que se muestra con los huecos en la pregunta. En las posiciones donde debe ir un hueco, el texto presenta el siguiente caracter: "X". De esta manera cuando se está manipulando la pregunta se pueden identificar los huecos de manera más sencilla buscando este caracter.
- "solution": se trata de la solución de la pregunta almacenada en un mapa con entradas de tipo <int, String> donde el entero indica el hueco y el String la solución. Es decir, si una de las entradas del mapa es <1, password> significa que en el primer hueco la solución es el String "password".
- "option": es una lista de String donde se especifican las opciones entre las cuales el usuario puede elegir. De esta manera el usuario no tiene que escribir la solución y se presenta una manera de resolver la pregunta más dinámica.

5.2.2.9 Clase MultipleChoiceQuestion

En esta clase se definen los atributos únicos del otro tipo de pregunta que son los siguientes:

- "description": un String en el que se almacena la descripción de la propia pregunta.
- "options": una lista de String donde se presenta al usuario con las 4 opciones disponibles a elegir.
- "rightOption": se trata de un entero en el que se indica que String de la lista de opciones es la correcta.

5.3 FASE DE DESARROLLO

En este apartado se describirán cada una de las secciones principales de la aplicación explicando sus características relativas al desarrollo. Para ello se presentarán las pantallas que componen cada una de las secciones y se describirá la lógica más relevante que se lleva a cabo en cada pantalla adjuntando para ello el código y los diagramas pertinentes. Al tratarse de una aplicación con dimensiones grandes no se podrá llevar a cabo un análisis exhaustivo del código ya que esto escaparía el objetivo del informe

5.3.1 AUTENTICACIÓN

En la Figura 41 se pueden observar las dos primeras pantallas que el usuario observa al acceder a la aplicación.

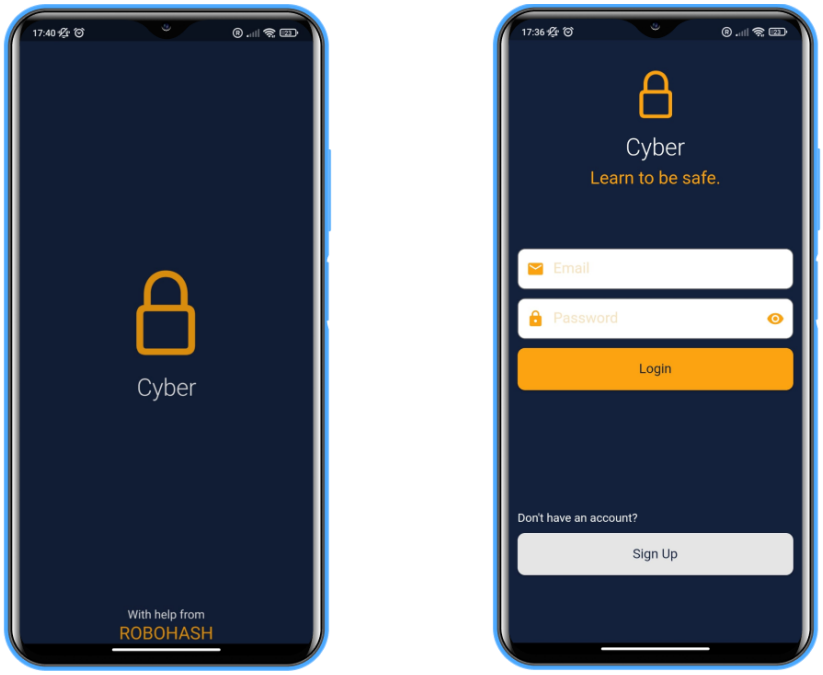


Figura 41 Splashscreen y login

La primera de las pantallas es lo que se conoce como SplashScreen y es la pantalla que se despliega mientras la aplicación se carga. Flutter de manera predeterminada ofrece una pantalla en blanco como SplashScreen. No obstante, se ha utilizado una librería llamada

"flutter_native_splash" la cual permite al desarrollador determinar el look and feel de la SplashScreen en el archivo llamado "pubspec.yaml". Este archivo lo incluyen todos los proyectos de tipo Flutter y en él se indican las librerías de terceros que el proyecto incluye para que de manera automática se incluyan en el proyecto. Además, en este archivo en ocasiones se indican atributos de las librerías como en ese caso. En el código adjuntado a continuación se observan los atributos especificados:

```
flutter_native_splash:

  color: "#14213D"
  image: assets/images/logo_big.png
  branding: assets/images/welcomeMessage_big.png
  branding_mode: bottom
  flutter_lints: ^1.0.0.
```

Como puede verse en el código se ha especificado que color debe utilizarse para el fondo, así como dos imágenes para utilizar que se encuentran en el directorio "assets" del proyecto. También se ha especificado la posición de la segunda de las imágenes.

En cuanto a la segunda de las pantallas de la figura esta muestra el formulario que el usuario debe de rellenar para identificarse y ganar acceso a la aplicación. En caso de que el usuario no tenga una cuenta tiene la posibilidad de pulsar el botón "Sign Up" que le redirigirá al flujo de pantallas para crearse una cuenta.

A la hora recoger información del usuario se han utilizado formularios. Cuando el usuario pulsa en el botón "Login", la siguiente función se ejecuta:

```
void Function() login = () {
  if (_formKey.currentState!.validate()) {

    String email =
        _controllerEmail.text.trim(); //trim() deletes blank spaces
    String password = _controllerPassword.text;

    UserController.loginWithEmailAndPassword(
            email: email, password: password)
        .then((value) {
      if (value is UserCredential) {
        //If the user logs In then we again navigate to the home Page since it
would have been notified
```

```
        //of the changes and now will display the dashboard
        Navigator.of(context).pushNamedAndRemoveUntil(
            HomePage.routeName, (Route<dynamic> route) => false);
      } else {
        SnackBar snBar = SnackBar(
          content: Text(
            value,
            style: getNormalTextStyleBlue(),
          ),
          backgroundColor: secondaryColor,
        );
        ScaffoldMessenger.of(context).showSnackBar(snBar);
      }
    });
  }
};
```

Lo primero que se realiza es validar el formulario. Para ello se utilizan unas funciones definidas en los textFormFields que componen el formulario llamadas validators en las que se define como debe ser el texto introducido para considerarse valido. En el caso del email se ha definido que debe tener formato de correo electrónico y en el caso de la contraseña que debe de ser por lo menos 6 caracteres ya que es una condición impuesta por Firebase.

Una vez se ha validado el formulario se obtiene la información de los campos del formulario y se ejecuta la función definida en el controlador userController para hacer login con las credenciales indicadas por parámetro. En caso de que haya un fallo se le muestra al usuario un SnackBar indicándole el motivo del fallo. Un SnackBar no es más que un cuadro de diálogo que aparece por la parte inferior de la pantalla.

En el controlador userController se han definido todas las funciones que se deben ejecutar para acceder a la capa de persistencia relativa al usuario de Firebase. Con el objetivo de que sea más fácil para el lector se ha adjuntado el código del controlador "UserController" en ANEXO III: Código del controlador "UserController". En este caso se utiliza la función "loginWithEmailAndPassword".

Como puede verse en la función descrita en el anexo, se utiliza una clase llamada FirebaseAuth. Esta clase se puede utilizar ya que para poder utilizar las funcionalidades de Firebase se han tenido que incluir previamente tres librerías. Las tres librerías que se han

incluido han sido: "firebase_auth", "cloud_firestore" y "firebase_core". La tercera de las librerías incluye funcionalidades básicas de Firebase mientras que la primera permite utilizar los servicios de autenticación de Firebase Authentication y la segunda, las funcionalidades relativas a las bases de datos de Cloud Firestore.

Utilizando esta clase se ejecuta la función "signInWithEmailAndPassword" para verificar si las credenciales aportadas son correctas. En caso de serlo se devuelve un objeto de tipo UserCredential, en caso contrario, se devuelve un String explicando el motivo del fallo.

Con dicha información en el front se actúa en consecuencia, dando acceso a las siguientes pantallas al usuario o explicándole el motivo del fallo en caso de que las credenciales sean incorrectas.

5.3.2 REGISTRO

En la Figura 42 se pueden observar las tres pantallas relativas al registro de una nueva cuenta. Para poder crear una nueva cuenta se necesitan especificar 3 valores: un correo electrónico válido, una contraseña y un nombre de usuario.

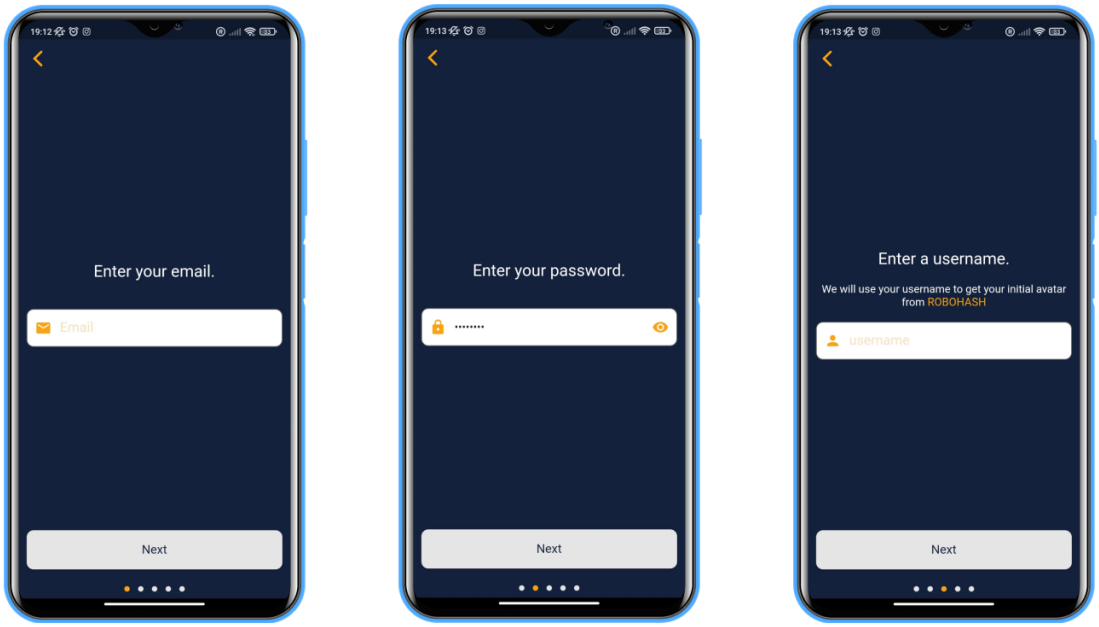


Figura 42 Páginas para el registro de un nuevo usuario

Cada vez que el usuario pulsa el botón de "Next" se ejecuta una función en la que se valida el formulario observando que la información introducida es correcta. Para ello se comprueba que el campo no está vacío y que cumple con las exigencias. Estas varían en función del campo. En el caso del correo, este debe de tener el formato "correo@dominiovalido.com", en el caso de la contraseña esta debe de tener al menos 6 caracteres y en el caso de nombre de usuario no existe ninguna limitación.

Si la información indicada es correcta, se crea la siguiente de las pantallas y se recupera la información especificada en la pantalla anterior mediante el siguiente código:

```
class SignUpPasswordPage extends StatelessWidget {
  const SignUpPasswordPage({Key? key}) : super(key: key);

  static final routeName = '/SignUpPassword';

  @override
  Widget build(BuildContext context) {
    //I get the email from the previous page
    final email = ModalRoute.of(context)!.settings.arguments as String;
.
.
.
```

El código anterior hace referencia a la página donde se indica la contraseña y se puede observar que cuando se construye el widget mediante el método "build", la primera línea de código recupera el email especificado en la pantalla anterior. Para ello es necesario pasar este valor a la hora de navegar a la nueva página como argumento. En el código adjunto a continuación se indica la función ejecutada con dicho objetivo.

```
void Function() goToNextSignUpPage = () {
  if (_formKey.currentState!.validate()) {
    Navigator.pushNamed(
      context,
      PasswordForm.routeName,
      arguments: _controllerEmail.text.trim(),
    );
  }
```

Esta función se ejecuta en la página donde se indica el correo electrónico. Se realiza una validación del formulario y se navega a la siguiente página indicando el nombre de la ruta y qué contenido se pasa como argumento que en este caso es el correo.

Una vez se ha llegado a la última de las páginas, cuando se pulsa el botón "Next", se crea una nueva instancia de "UserCustom" y se ejecuta una función del "UserController" para crear un nuevo usuario. La función que se ejecuta es "addUserToAuthAndFirestore". El código se puede ver en ANEXO III: Código del controlador "UserController".

En esta función lo primero que se verifica es que no existe ningún usuario en la base de datos con el mismo nombre de usuario. En caso de que sea así, se trata de añadir al usuario a la base de datos de Autenticación. Si esto falla por algún motivo como por ejemplo que el correo electrónico ya está en uso o la contraseña especificada es muy débil, se deja de ejecutar la función y se le indica el fallo al usuario. Si se logra crear una instancia del usuario en la base de datos de Firebase Authentication entonces se crea otra instancia en la base de datos creada en Cloud Firestore con el objetivo de guardar la información detallada del usuario.

Para crear una entrada en la base de datos de Cloud Firestore para el nuevo usuario creado se ha definido la función "addUserToFireStore" también presente en el controlador.

En esta función, primero se crea una referencia a la base de datos de Cloud Firestore. Una vez creada la referencia se crea un nuevo documento cuyo identificador es el mismo que se ha asignado de manera automática en la base de datos de autenticación. Para crear el nuevo documento se convierte la instancia del objeto a formato JSON mediante el método "toJson()". Este método se ha creado de manera automática utilizando una librería llamada "json_serializable".

La librería "json_serializable" permite crear para las clases del modelo que se quieran dos métodos. Uno para pasar la información de instancia de un objeto a formato JSON, este se llama "toJson()" y otro para pasar la información de formato JSON a una instancia del objeto

definido en el modelo que se llama "fromJson()". Estos métodos se han descrito en todas las clases del modelo y en la mayoría de los casos se ha utilizado para ello esta librería.

En caso de que todos los métodos descritos tengan un resultado favorable, se crean dos instancias del usuario. Una en la base de datos de autenticación donde se guarda la contraseña y el correo electrónico del usuario y una en la base de datos de Cloud Firestore donde queda registrada toda la información detallada del usuario como sus cursos guardados o su nivel.

En la Figura 43 se puede observar cómo es el esquema de la base de datos de autenticación.

Figura 43 Base de datos de autenticación

En la Figura 44 se puede observar cómo es el esquema de la base de datos personalizada de Cloud Firestore. En esta imagen se puede ver la colección de usuarios que está formada por diversos documentos, donde cada documento es un usuario con sus datos almacenados en forma de atributos. En la imagen se muestra como quedaría un usuario recién creado donde todos los atributos están inicializados a valores nulos o vacíos excepto aquellos que ha especificado.

Cuando se crea un nuevo usuario se inicializa su nivel al nivel 1. Además, se utiliza su nombre de usuario como la String necesaria para otorgarle un Avatar, por tanto, se modifican los campos "profilePictureActive" y "collectedAvatars" para incluir este valor. Por defecto el campo "isAdmin" es inicializado a "false".

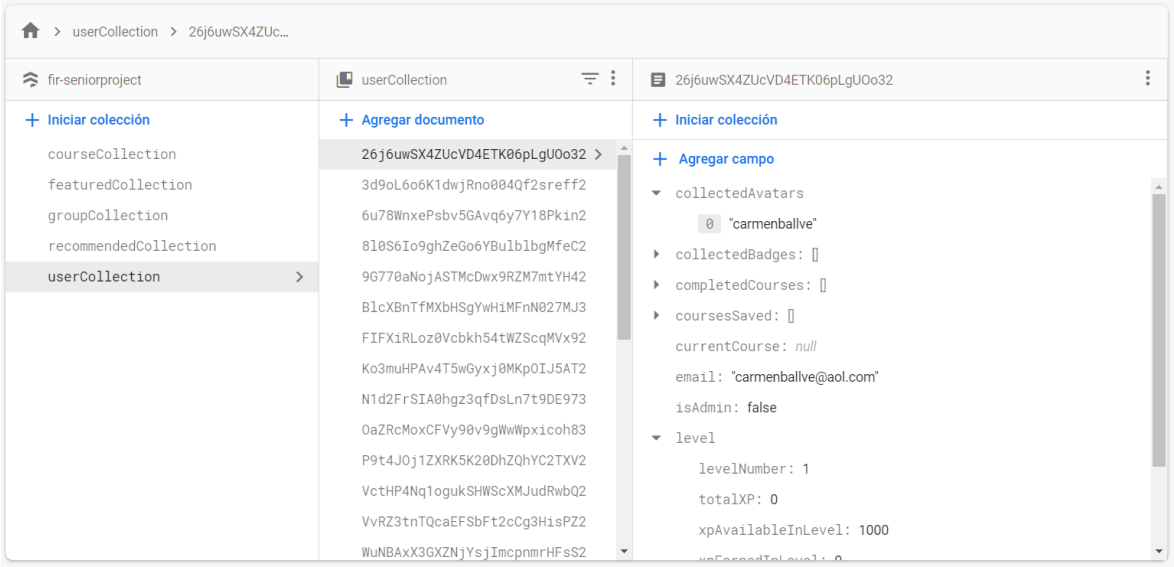


Figura 44 Colección de usuarios de Cloud Firestore

Una vez ya se ha registrado el nuevo usuario en la aplicación, el usuario termina el flujo de registro navegando por las 3 pantallas que se pueden observar en la Figura 45.

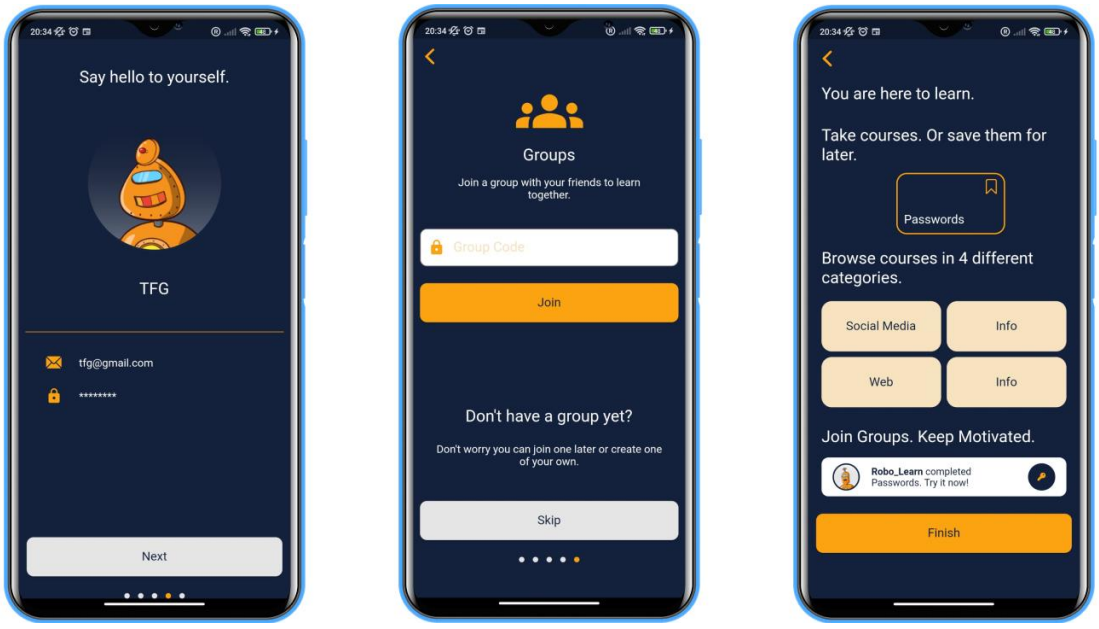


Figura 45 Fin del registro

En la primera de las pantallas se muestra al usuario un resumen del perfil creado. En este resumen se muestra su correo electrónico, su nombre de usuario y el avatar que se le ha

asignado. Para mostrar esta información se pasa como argumento el usuario creado a la página de resumen de la misma manera que se ha explicado en páginas anteriores. Para crear la imagen del Avatar se utiliza una clase llamada Avatar. En el siguiente código se muestra la clase:

```
class Avatar extends StatelessWidget {
  Avatar({required this.nameOfAvatar, required this.size});
  final String nameOfAvatar;
  final double size;
  late final String url;

  Future<Uint8List> fetchAvatar() async {
    http.Response response = await http.get(Uri.parse(url));
    return response.bodyBytes;
  }

  Widget loadingWidget() {
    return new FutureBuilder<Uint8List>(
      future: fetchAvatar(),
      builder: (context, snapshot) {
        if (snapshot.hasData) {
          return new Image.memory(snapshot.data!);
        } else if (snapshot.hasError) {
          print('${snapshot.error}');
          return new Center(
              child: new Text('❌', style: TextStyle(fontSize: 72.0)));
        } else {
          return new Container(
            padding: EdgeInsets.all((size) / 2.0), //-50
            child: new CircularProgressIndicator(),
          );
        }
      },
    );
  }

  @override
  Widget build(BuildContext context) {
    url = 'http://robohash.org/${nameOfAvatar}';
    return new Container(
      width: size,
      height: size,
      decoration: new BoxDecoration(
          shape: BoxShape.circle,
          border: new Border.all(
            color: primaryColor,
            width: 2.0,
          ),
          gradient: LinearGradient(
              begin: Alignment.topCenter,
```

```
                end: Alignment.bottomCenter,
                colors: [
                  Colors.transparent,
                  Colors.white38,
                ])),
      child: new ClipOval(
        child: loadingWidget(),
      ),
    );
  }
}
```

Como puede observarse en el código, a la hora de crear esta clase se debe de especificar el nombre del avatar, así como el tamaño. Cuando se crea el widget, se ejecuta el método "build" y en él se indica que el contenedor tiene como widget hijo otro widget llamado "loadingWidget". Este último widget es del tipo "FutureBuilder" lo cual significa que se ejecuta una función asíncrona, en este caso la función ejecutada es "fetchAvatar() que realiza una llamada GET a la página web de Robohash. En función del estado de la función, el FutureBuilder devuelve un widget u otro cuando se ejecuta su método "build".

Las funciones asíncronas en Flutter devuelven un objeto de tipo Future. Estos objetos tienen diferentes estados en función de en qué punto se encuentra la función. El Future puede estar completándose o haberse completado. Una vez el Future se ha completado este lo puede haber hecho con resultado exitoso o con un resultado desfavorable. Los tres estados descritos del future se tienen en cuenta en el FutureBuilder. De tal manera que, si el Future está en proceso de completarse, se devuelve un widget llamado "CircularProgressIndicator" que muestra que hay un proceso cargándose, en caso de que el Future se haya completado de manera no satisfactoria se devuelve un icono en forma de X y en caso de que el future se haya completado satisfactoriamente, se devuelve una imagen construida a partir del resultado de la llamada GET a la página web de Robohash.

Una vez el usuario pulsa el botón "next", se le redirige a la segunda de las pantallas mostradas en la figura. En esta pantalla el usuario tiene la posibilidad de unirse a un grupo en caso de que cuente con el código que lo identifica. Si el usuario no cuenta con el código puede pulsar el botón "skip" lo cual resultará en una navegación a la última de las pantallas.

Si el usuario inserta un código para unirse a un grupo en esta página, entonces se modifica la instancia del usuario y se añade el identificador de grupo. No obstante, como esta lógica es relativa al apartado de grupos no se comentará en detalle ya que forma parte de la única sección de la aplicación que no forma parte del código desarrollado en este proyecto.

La última de las pantallas es un resumen de lo que el usuario va a encontrar en la aplicación. En ella se informa al usuario que este será capaz de aprender acerca de cuatro categorías relacionadas con la ciberseguridad. Estas categorías son: Redes Sociales, Información, Web y Dispositivos. En esta pantalla también se muestra como son las tarjetas que el usuario encontrará para acceder a un curso determinado. En este caso se muestra la tarjeta relativa al curso de contraseñas y se le indica al usuario que tendrá la posibilidad de guardar los cursos para completarlos más adelante. En cuanto a los widgets mostrados, no se trata de imágenes estáticas, sino que son tarjetas que se pueden pulsar. El resultado de pulsar las tarjetas no es el que el usuario encontrará en la aplicación, pero se ha realizado así para que el usuario pueda ver cómo es la dinámica de la aplicación, por ejemplo, como se comportará la tarjeta de los cursos al pulsar el icono de guardar el curso.

5.3.3 FLUJO DE LOS CURSOS

5.3.3.1 Dashboard

La primera de las páginas del flujo de cursos que se va a explicar es la principal llamada "Dashboard", se trata del panel desde el cual el usuario accede a la información principal en la aplicación. Esta información es el curso recomendado, las cuatro categorías y el curso que el usuario ha dejado a medias en caso de que exista. El contenido del Dashboard depende del usuario que esté utilizando la aplicación ya que en él se utiliza información relativa al usuario como su nombre y el curso pendiente.

Antes de explicar el contenido del Dashboard se va a explicar cuál es el flujo de inicio en la aplicación y como se obtiene la información relativa al usuario en esta pantalla. Uno de los primeros widgets en ser creados en la aplicación es el widget "HomePage". A continuación, se puede observar el código relativo a esta clase:

```
class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);

  static final routeName = '/HomePage';

  @override
  Widget build(BuildContext context) {
    //In this page I get the information of the screen and I initialize it in
variables stored in other file
    widthOfScreen = MediaQuery.of(context).size.width;
    heightOfScreen = MediaQuery.of(context).size.height;
    var padding = MediaQuery.of(context).padding;

    //I update the height by subtracting the status bar height
    heightOfScreen = heightOfScreen - padding.top;

    //The builder from the consumer will be called everytime the Application
state notifies its listeners

    return Consumer<ApplicationState>(builder: (context, appState, _) {
      switch (appState._loginState) {
        case ApplicationLoginState.loggedIn:
          {
            return PageViewScreen();
          }
        case ApplicationLoginState.loggedOut:
          {
            return LogInPage();
          }
        default:
          {
            return CircularProgressIndicator();
          }
      }
    });
  }
}
```

En el método “build” de este widget, se obtienen las medidas del dispositivo en el que se está ejecutando la aplicación y se guardan en unas variables globales ya que se dimensionarán el resto de los componentes de la aplicación utilizándolas. Después, de esta operación se devuelve un objeto de tipo “Consumer” de la clase “ApplicationState”. La clase “ApplicationState” es una clase que se crea siempre que se inicia la aplicación y escucha cambios en la base de datos de Autenticación. En el siguiente código se puede ver cómo funciona:

```
enum ApplicationLoginState { loggedIn, loggedOut }

class ApplicationState extends ChangeNotifier {
  //We need to initialize this variable so we suppose the user is loggedOut
  ApplicationLoginState _loginState = ApplicationLoginState.loggedOut;

  ApplicationState() {
    init();
  }

  Future<void> init() async {
    await Firebase.initializeApp();

    //Now we create a listener of the user state and after this we know the state
of the user
    FirebaseAuth.instance.userChanges().listen((user) {
      if (user == null) {
        _loginState = ApplicationLoginState.loggedOut;
      } else {
        _loginState = ApplicationLoginState.loggedIn;
      }
      notifyListeners();
    });
  }
}
```

El "Consumer" devuelto por la clase "HomePage" devuelve a su vez otro widget. El contenido devuelto depende del estado del usuario en la aplicación. En caso de que el usuario ya se haya identificado, se devuelve una página llamada "PageViewScreen" y en caso de que no, se redirige al usuario a la página de log in ya presentada en el informe. La página "PageViewScreen" es una página que contiene instancias de las cuatro pantallas principales en la aplicación y que permite la navegación entre ellas por medio de la barra de navegación que se presenta en la parte inferior de la pantalla y que el lector podrá observar en la figura donde se presenta el "Dashboard".

Es en esta página "PageViewScreen" donde se inicializa el controlador "ActiveUserController". Este controlador contiene una instancia del usuario activo y en él se definen todos los métodos para modificar dicha instancia y persistir los cambios en la base de datos mediante la comunicación con el controlador "UserController". A continuación, se adjunta el código donde se inicializa el controlador:

```
Future initializeUser() async {
  try {
    //I get the active user
    UserCustom uc = await UserController.getActiveUser();

    //I initialize the global variable activeUser that is used by the GetX
    //controller to initialize the info
    activeUser = uc;

    if (Get.isRegistered<ActiveUserController>()) {
      Get.delete<ActiveUserController>();
    }
    Get.put(ActiveUserController());

    return Future.value('Done');
  } catch (error) {
    throw Exception('Error initializing the user');
  }
}
```

Como puede verse en el código y se explica en los comentarios, se inicializa "activeUser" que es una variable global con la instancia del usuario activo. Posteriormente se verifica si existe una instancia del controlador "ActiveUserController". En caso de que exista se borra y posteriormente se crea una nueva instancia. Cuando se crea la nueva instancia, el controlador utiliza en el constructor el valor de la variable global descrita.

El controlador "ActiveUserController" es una pieza clave del desarrollo de la aplicación ya que se utiliza continuamente para acceder a la información del usuario y cambiar esta. Es por ello por lo que se ha decidido adjuntar el código de la clase entera en ANEXO IV: Código del controlador "ActiveUserController".

Una vez descrito el flujo de código que se ejecuta al iniciar la aplicación y como se obtiene la información relativa al usuario activo, se procede a explicar la página "Dashboard". En la Figura 46 puede observarse como se ve esta pantalla.

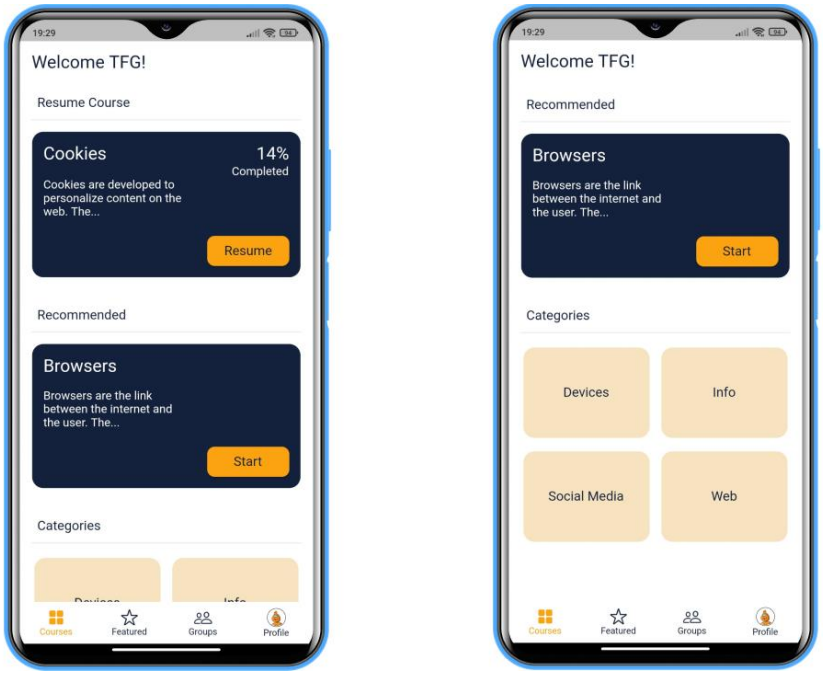


Figura 46 Dashboard

En la imagen se puede observar dos versiones del "Dashboard". La primera de las pantallas muestra cómo se vería en caso de que el usuario tenga el atributo "currentCourse" a un valor distinto de "null". La segunda muestra cómo se vería en el otro caso.

A la hora de crear esta pantalla, se accede a la información del usuario activo mediante el controlador "ActiveUserController". En caso de que la variable "currentCourse" no sea nula, se crea una tarjeta con la información relativa al curso que el usuario ha dejado pendiente. Para ello, se utiliza el atributo "courseID" del objeto de tipo "CurrentCourse" para obtener de la base de datos el curso que tiene como ID esa String. La función ejecutada con tal objetivo es "getCourseByID". La función puede observarse en ANEXO V: Código del controlador "CourseController".

En cuanto, como se almacenan los cursos en la base de datos, en la Figura 47 puede obsevarse como se organiza la información y como se identifican los cursos mediante el String mencionado. A su vez puede observarse como cada documento asociado a un curso tiene una colección asociada llamada "questions" en donde se encuentran las preguntas relativas al curso.

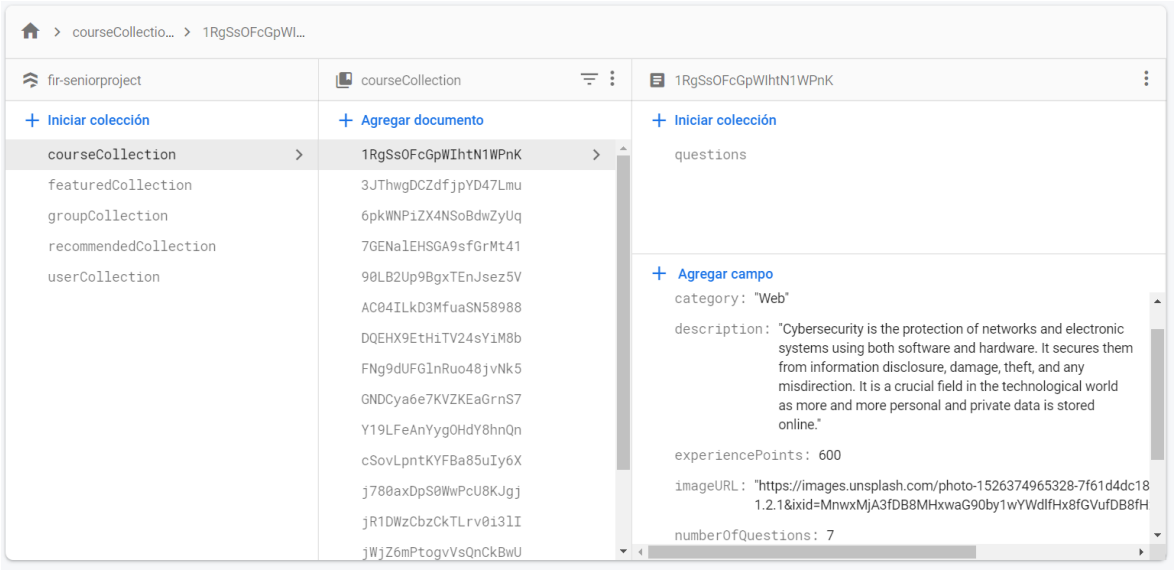


Figura 47 Colección de cursos de Cloud Firestore

Para mostrar la tarjeta del curso recomendado, se ejecuta la función "getRecommendedCourse" del mismo controlador. En esta función se busca el identificador del curso recomendado en la colección "RecommendedCollection" de la base de datos. Una vez se ha obtenido el String que lo identifica, se utiliza la función "getCourseByID" mencionada anteriormente y se construye la tarjeta con la información necesaria. En la Figura 48 se puede observar cómo es la colección "RecommendedCollection" de la base de datos.

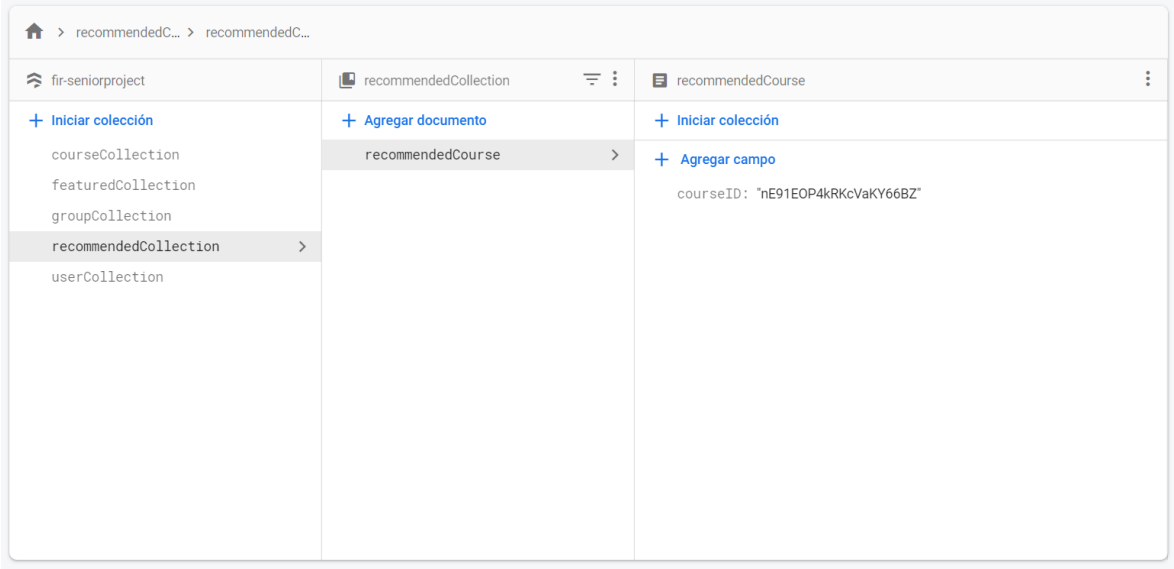


Figura 48 Colección del curso recomendado de Cloud Firestore

La última de las secciones del "Dashboard" es la relativa a las categorías. En esta pantalla se pueden ver 4 botones cada uno de los cuales se identifica con una categoría. Al pulsar uno de los botones, el usuario se ve redirigido a una pantalla con los cursos relativos a dicha categoría. Esta pantalla se describe en el siguiente epígrafe.

5.3.3.2 CategoryPage

Esta página se construye recibiendo como argumento una categoría en concreto. El valor de esta variable depende del botón que se haya pulsado en el "Dashboard" para acceder a esta pantalla. Una vez se tiene este valor se ejecuta la función "getCourseNamesFromCategory" del controlador "CourseController" especificando la categoría de la que se quieren obtener los cursos.

La Figura 49 muestra como se ve la pantalla en caso de presionar la categoría "Info". Al ejecutar la función indicada se obtiene un mapa con entradas del formato <courseID, title>. Con este mapa se construyen las tarjetas de los cursos. De tal manera que cuando el usuario pulsa en una de las tarjetas, se navega a la página "CourseDescription" en la cual se debe de especificar el ID del cuso en el constructor. Esto se puede hacer ya que en la tarjeta se ha guardado el ID del curso.

En la Figura 49 se puede observar también que hay información relativa al usuario. Esta información es el número de cursos que el usuario ha completado en la categoría, el número de puntos de experiencia que ha obtenido y los cursos que ha guardado o completado. Para acceder a esta información se utiliza el controlador "ActiveUserController". Más concretamente se utilizan los métodos: "getCompletedCoursesInCategory" y "getXPInCategory". Para observar qué cursos ha guardado o completado el usuario, cuando se obtiene el mapa mencionado de cursos en la categoría, se recorren las entradas de dicho mapa y se comprueba si los identificadores de los cursos se encuentran en la lista "coursesSaved" o "coursesCompleted" del usuario activo mediante el controlador "ActiveUserController".

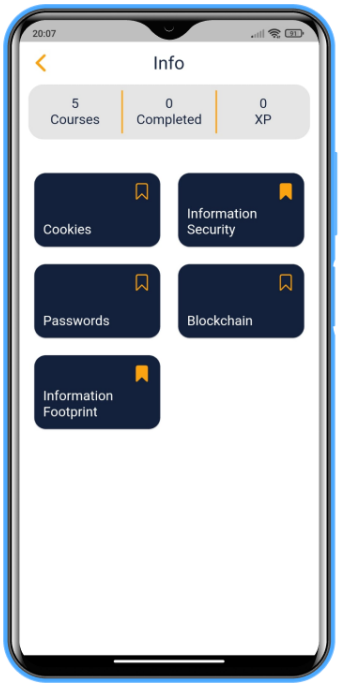


Figura 49 Cursos dentro de una categoría

5.3.3.3 CourseDescription

Cuando el usuario accede a un curso pulsando una de las tarjetas de cursos presente en la aplicación se muestra una descripción del curso. Las tarjetas de los cursos están presentes en la página "CategoryPage" y en la sección del perfil donde el usuario puede ver sus cursos guardados y completados. Como se ha explicado anteriormente las tarjetas se crean

especificando el título del curso y el identificador de este. Una vez se pulsa la tarjeta se navega a la página "CourseDescriptionPage" pasando como argumento el identificador del curso. En el método "build" de esta página se crea un "FutureBuilder" en el cual se ejecuta la función "getCourseByID" del controlador "CourseContoller" para obtener de la base de datos el curso y poder mostrar la información relativa al mismo. Una descripción más detallada de la función se puede encontrar en ANEXO V: Código del controlador "CourseController".

En la figura puede observarse como se ve esta página en caso de pulsar en el curso "Blockchain". Se pueden observar dos versiones de la misma pantalla. La primera de las capturas de la figura muestra cómo se vería la pantalla en caso de que el usuario no hubiera completado ni guardado el curso previamente. En la segunda de las capturas se puede observar cómo se vería la misma pantalla en caso de que el usuario ya hubiera realizado un intento del curso y lo hubiera guardado en su biblioteca.

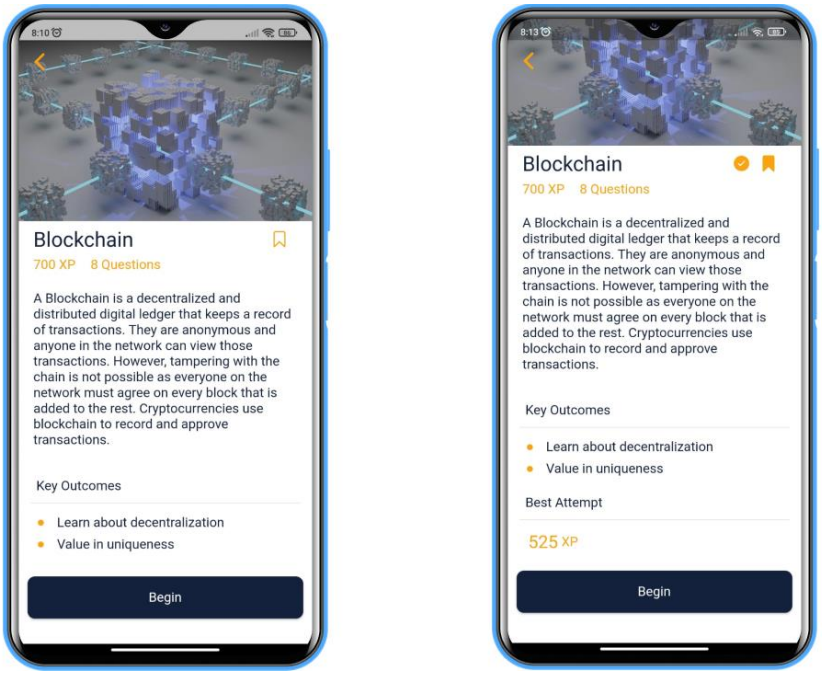


Figura 50 Descripción del curso

A la hora de crear esta pantalla se realizan dos funciones principales. Primero, como se ha indicado previamente, se obtiene el curso de la base de datos y se muestra la información

del curso descrita en el diagrama de clases. Entre otros atributos se muestra la imagen del curso, su descripción, el número de puntos de experiencia que aporta y el número de preguntas que tiene.

Por otro lado, se muestra información relativa al usuario activo. Para ello se utiliza el controlador "ActiveUserController". Se comprueba si el "courseID" se encuentra en la lista de cursos guardados "coursesSaved" del usuario para mostrar el botón de guardar como pulsado. También se observa si el curso ya ha sido completado mediante la función "isCompleted" definida en el controlador. En caso de que se haya completado el curso, se muestra un icono y el número de puntos de experiencia que se obtuvo en dicho curso. También se comprueba si el curso es el "currentCourse" del usuario mediante la función "isCurrentCourse". En caso de que sea así se muestra una sección similar a la de "Best Attempt" en la que se indica que porcentaje del curso lleva el usuario completado y se cambia el texto mostrado en el botón por "Resume".

Una vez se accede a esta página, el siguiente fragmento de código se ejecuta:

```
//Once I get the new-course I assign its value to a global variable
// so I can access the info from other pages easily
globals.activeCourse = snapshot.data;
globals.activeQuestionNum = 1;
globals.userProgress = [];

if (controller.isCurrentCourse(courseID: courseID)) {
  globals.activeQuestionNum =
      globals.activeUser!.currentCourse!.progress.length + 1;
  globals.userProgress = globals.activeUser!.currentCourse!.progress;
}
```

Lo que se hace es inicializar una serie de variables globales para que si el usuario decide comenzar con el curso saber de qué curso se trata e ir guardando el progreso del usuario en el mismo. En caso de que el curso ya estuviera iniciado, las variables se inicializan al valor ya existente.

5.3.3.4 Questions

Una vez el usuario inicia un curso, este va completando las preguntas relativas al mismo. Las preguntas son de dos tipos: opción múltiple y de tipo rellenar los huecos. A la hora de crear estas páginas se ejecuta la siguiente función:

```
Function nextQuestion = (BuildContext context) async {
  if (globals.activeCourse!.numberOfQuestions >= globals.activeQuestionNum!) {
    //I get the question in the new-course
    Question q =
        globals.activeCourse!.questions[globals.activeQuestionNum! - 1];

    //Once I have the first question I check what type of question it is
    //to navigate to the appropriate page

    if (q.typeOfQuestion == TypeOfQuestion.multipleChoice) {
      Navigator.pushNamedAndRemoveUntil(
          context, MultipleChoiceQuestionPage.routeName, (r) => false,
          arguments: q);
    } else {
      Navigator.pushNamedAndRemoveUntil(
          context, FillInTheBlanksQuestionPage.routeName, (r) => false,
          arguments: q);
    }
  } else {
    ActiveUserController activeUserController =
        Get.find<ActiveUserController>();
    final SaveCompletedCourseArgs args =
        await activeUserController.saveCompletedCourse();
    Navigator.pushNamedAndRemoveUntil(context, Overview.routeName, (r) => false,
        arguments: args);
  }
};
```

En la función se accede al contenido de las variables globales presentadas anteriormente. Se observa qué tipo de pregunta es la siguiente en el curso y de acuerdo con el valor de este atributo se crea una página u otra pasando como argumento la pregunta. En la figura se puede observar cómo se ven las pantallas para cada uno de los tipos de pregunta:

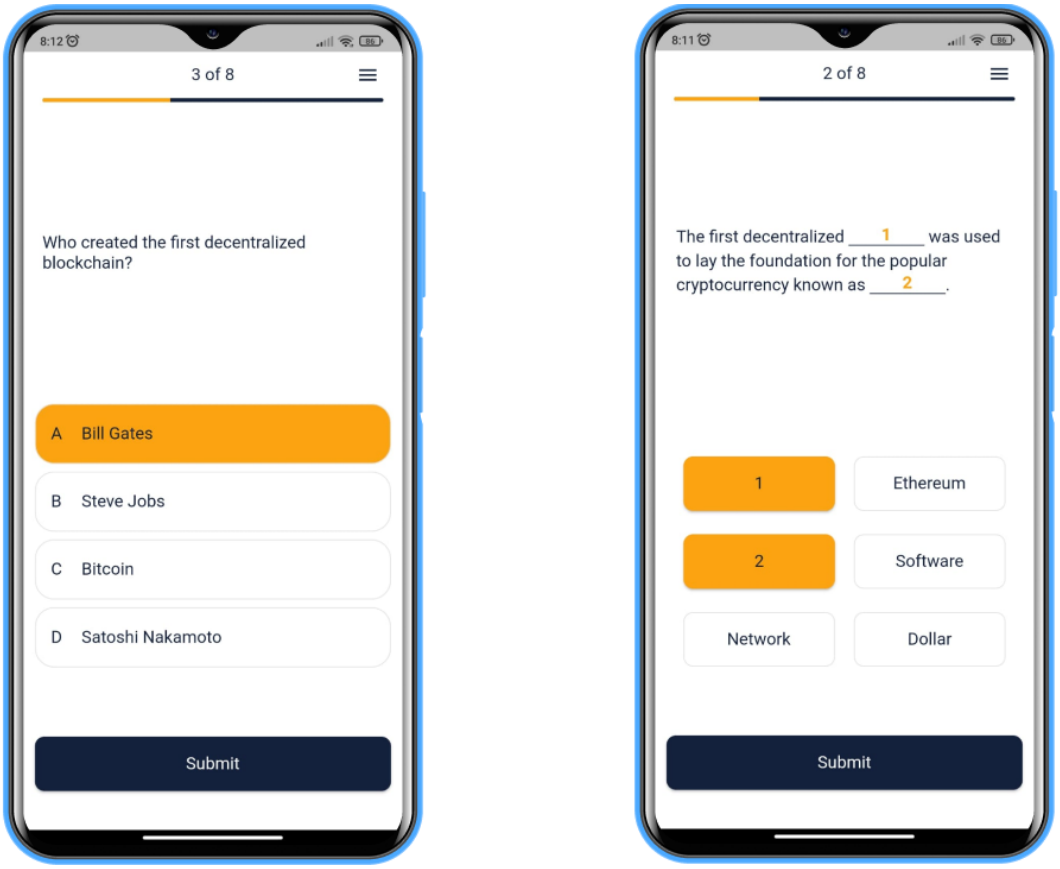


Figura 51 Tipos de pregunta en un curso

La primera de las pantallas presentadas es la página "MultipleChoiceQuestionPage". En esta página lo primero que se muestra es el atributo "description" de la pregunta. Después de esto se construye un widget llamado "ToggleButtons" especificando las opciones de la pregunta. Este widget te permite crear un conjunto de botones en el que solo puede estar uno pulsado. El botón pulsado se puede determinar mediante la variable "isSelectedNew" de dicho widget. Una vez el usuario pulsa el botón para entregar la respuesta, se ejecuta la siguiente función:

```
void Function() submitMultipleChoiceFunction = () {
  int i = 0;
  int optSelected = 5;
  //With this loop I get what option the user has selected
  for (bool selected in ToggleButtonOptions.isSelectedNew) {
    if (selected) {
      optSelected = i;
    }
```

```
    i++;
  }
  bool isRight = false;
  if (optSelected == question.rightOption) {
    isRight = true;
  }

  //I update the global variables once answer submitted
  globals.userProgress.add(isRight);
  globals.activeQuestionNum = globals.activeQuestionNum! + 1;

  showDialog(
      context: context,
      barrierDismissible: false,
      builder: (_) {
        return QuestionFeedback(
            args: FeedbackArguments(isRight, question.longFeedback,
                question.getSolutionAsString()));
      });
  optSelected = 5;
};
```

Como se puede observar en la función, se comprueba si la respuesta es correcta y se guarda el progreso del usuario en la variable global antes de mostrar el cuadro de dialogo con la realimentación.

La segunda de las pantallas de la figura muestra como se ve la página "FillInTheBlanksQuestionPage". En esta página se muestra el atributo "text" de la pregunta. En aquellos puntos donde el texto incluye el carácter X, se muestra un hueco con el número que corresponda. Después de mostrar el texto, se recorre la lista "options" de la pregunta y se crea un botón por cada una de las opciones que se ofrecen en la pregunta. Una vez el usuario se dispone a responder la pregunta, se crea un mapa con las opciones que el usuario elige y el orden en el que lo hace. Es decir, si el usuario pulsa "Ethereum" y luego "Software" se crea un mapa con entradas <1, Ethereum>, <2, Software>. A la hora de crear este mapa se tiene en cuenta si el usuario ya había pulsado esa opción. En caso de que sea así, la opción se elimina del mapa. Una vez se pulsa una opción, se cambia el estado del botón como se muestra en la imagen.

Cuando el usuario entrega la respuesta, se comprueba si el mapa construido es igual al mapa solución descrito en el atributo "solution". Después de esto, se actualizan las variables globales acordemente. Esta comprobación se puede observar en el siguiente fragmento de código:

```
void Function() fillInTheBlanksFunction = () {
  //In case the user has not picked 3 options, dont continue
  if (proposedSolution.length < question.solution.length) {
    SnackBar snBar = SnackBar(
      content: Text(
        'Pick more options',
        style: getNormalTextStyleBlue(),
      ),
      backgroundColor: secondaryColor,
    );
    ScaffoldMessenger.of(context).showSnackBar(snBar);
    return;
  }

  int numberOfMatches = 0;
  bool isRight = false;
  for (int i = 1; i <= question.solution.length; i++) {
    if (question.solution[i] == proposedSolution[i]) {
      numberOfMatches++;
    }
  }

  if (numberOfMatches == question.solution.length) {
    isRight = true;
  }

  //I update the global variables once answer submitted
  globals.userProgress.add(isRight);
  globals.activeQuestionNum = globals.activeQuestionNum! + 1;

  // Before navigating to the next page we have to
  // reset all the variables that we have used so far

  blankCounter = 1;
  proposedSolution = {};

  showDialog(
      context: context,
      barrierDismissible: false,
      builder: (_) {
        return QuestionFeedback(
            args: FeedbackArguments(isRight, question.longFeedback,
                question.getSolutionAsString()));
      });
};
```

5.3.3.5 Feedback

Como se ha podido observar en el código adjuntado para los dos tipos de preguntas, después de responder a las preguntas se muestra un cuadro de diálogo con la realimentación. En esta página descrita en "QuestionFeedback" se especifica como argumento un objeto de tipo "FeedbackArguments" que cuenta con atributos que describen si la respuesta es correcta, cuál es la solución y cuál es la explicación detallada.

En la figura se puede observar los dos estados del widget QuestionFeedback. Este widget es de tipo "stateful" lo cual quiere decir que su estado cambia. La condición para que su estado cambie es que el usuario haya pulsado el botón de "More". En caso de que el usuario lo haga, en lugar de mostrarse el progreso en el curso y la solución de la pregunta, se muestra una explicación más detallada. El estado de este widget se puede volver a cambiar pulsando en "back".

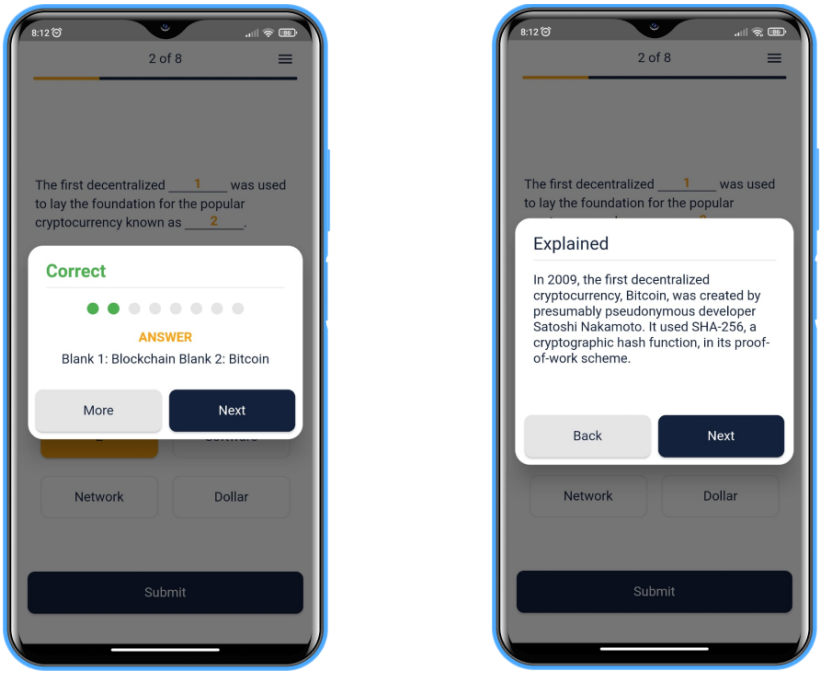


Figura 52 Realimentación de una pregunta

5.3.3.6 Options Menu

Dentro de un curso el usuario tiene acceso a un menú de opciones al cual se accede pulsando en el icono de la esquina superior derecha que aparece en todas las preguntas. Una vez el usuario pulsa este icono puede observar la pantalla de la figura.

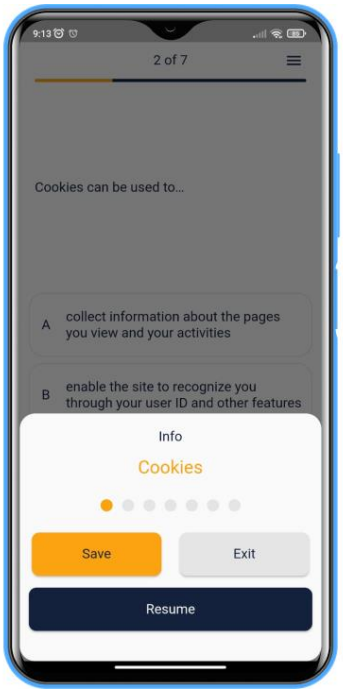


Figura 53 Menú de opciones dentro del curso

Desde este menú de opciones el usuario puede ver su progreso en el curso. El usuario puede volver al curso pulsando fuera del cuadro o pulsando el botón de "Resume". También se da la opción al usuario de abandonar el curso o de dejarlo para más adelante. En caso de que el usuario pulse "Save", se ejecutará la función "updateCurrentCourse" definida en el controlador "ActiveUserController". Con esta función se cambia el valor del atributo "currentCourse" del usuario activo y se establece el presente curso como el curso pendiente.

5.3.3.7 Final Course Pages

Una vez el usuario ha completado el curso, este observará la primera de las pantallas de la Figura 54: "Overview". En ella se muestra el nivel del usuario y cuál es su progreso en el mismo después de haber añadido el número de puntos de experiencia relativos al curso que

acaba de completar. Para ello se ejecuta la función "saveCompletedCourse" del controlador "ActiveUserController". En esta función se llevan a cabos varios procesos. Se añade el curso a la lista de cursos completados. Esto solo se hace si es la primera vez que el usuario completa el curso o si lo ha completado con una mejor puntuación que la vez anterior. Se suman los puntos de experiencia que se han obtenido como consecuencia de haber completado el curso. Cabe destacar que solo se añaden los puntos de experiencia que no se hayan obtenido en intentos previos. En caso de que el usuario haya subido de nivel debido a los puntos de experiencia ganados, se le recompensa otorgándole un nuevo avatar. Esto se pone en práctica añadiendo un string aleatorio a su atributo "collectedAvatars". Se comprueba si el usuario ha respondido un 50% o más de las preguntas del curso bien. En caso de que sea así y no haya obtenido previamente la insignia, se crea una instancia de la clase "Badge" y se añade la insignia a su lista.

Esta función devuelve una serie de booleanos para saber si el usuario ha obtenido la insignia del curso o si ha subido de nivel para actuar en consecuencia en la página. En caso de que el usuario haya subido de nivel, se muestra directamente un cuadro de diálogo con el avatar ganado. En caso de que el usuario haya ganado la insignia, se crea un texto y un botón en la página. Si el usuario pulsa en el botón, se abre el cuadro de diálogo mostrado en la segunda de las pantallas de la figura.

También se muestra el número de puntos de experiencia que el usuario ha ganado completando el curso y cuantas preguntas ha acertado. Además, se puede ver un resumen del progreso de las preguntas.

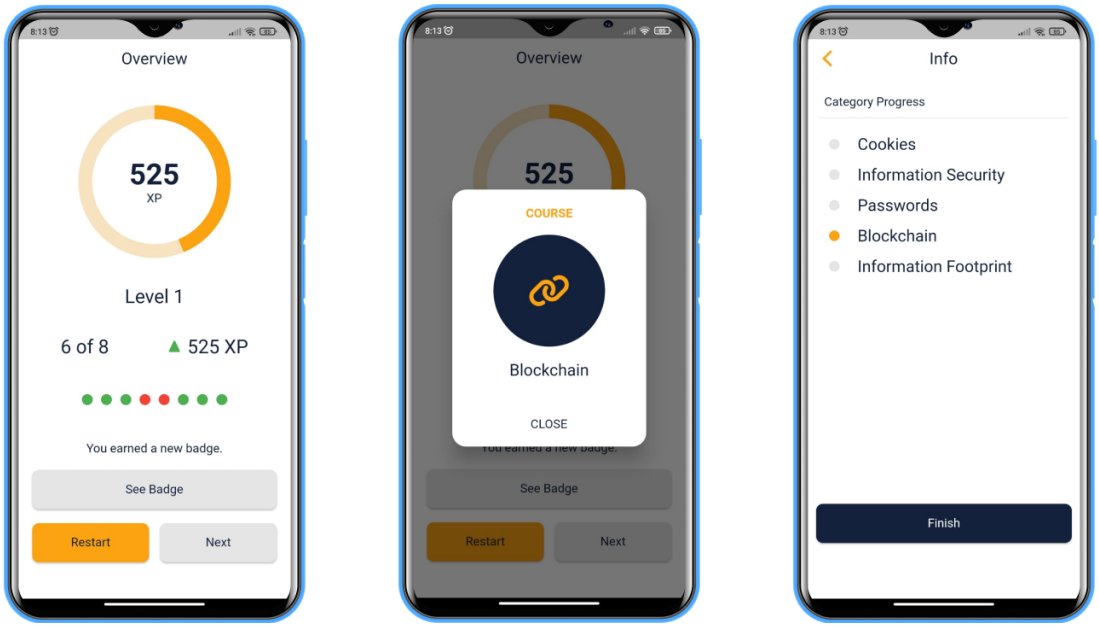


Figura 54 Resumen del desempeño en el curso y progreso en la categoría

En la última de las páginas se ve el contenido de la página "CategoryProgress". En esta página se ejecuta la función "getCourseNamesFromCategory" para construir una lista con todos los cursos de la categoría. Aquellos cursos que se han completado por el usuario aparecen acompañados con un círculo relleno. Esto se puede verificar en la lista "coursesCompleted" del usuario a la cual se accede mediante el controlador "ActiveUserController".

5.3.4 CURSO DESTACADO

Esta es otra de las secciones principales de la aplicación a través de la cual se puede acceder mediante la barra de navegación. En esta sección se muestra el curso destacado. El curso destacado es determinado por el administrador de la aplicación. Para que un curso sea el curso destacado el atributo "isFeatured" del mismo debe establecerse a "true". Para saber que curso es el destacado, se debe acceder a la colección "FeaturedCollection" donde se define el identificador del curso destacado en el documento "FeaturedCourse". El esquema de la colección descrito se puede observar en la Figura 55.

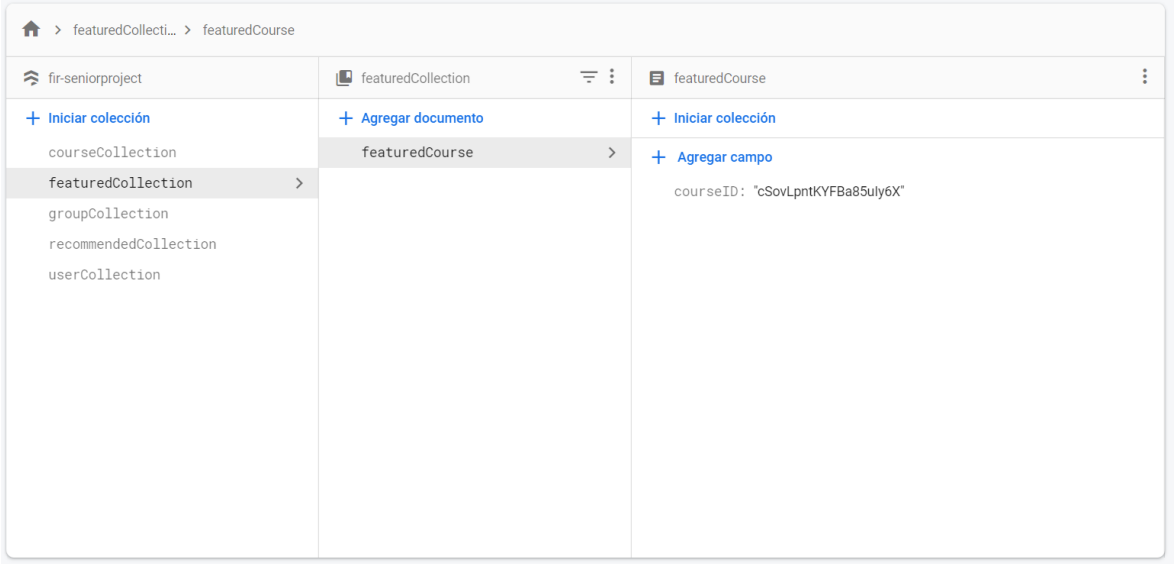


Figura 55 Colección para el curso destacado de Cloud Firestore

La página que contiene el curso destacado se llama “FeaturedCoursePage” y se trata de una página en la cual se ejecuta un “FutureBuilder” en su método “build” en el cual la función que se ejecuta es “getFeaturedCourse” definida en el controlador “CourseController” cuyo código puede consultarse en ANEXO V: Código del controlador “CourseController”. Dicha función devuelve el identificador del curso destacado mediante la consulta a la colección descrita. Una vez se ha obtenido dicho código se devuelve una página “courseDescription” en la cual se pasa por argumento el “courseID”. En la pantalla “courseDescription” aparte de las funcionalidades ya descritas en el epígrafe que se le ha dedicado en este informe, también se comprueba si el curso que se está mostrando es el destacado mediante el atributo “isFeatured”. En caso de que sea así se muestra un botón flotante. El aspecto del botón puede verse en la Figura 56.

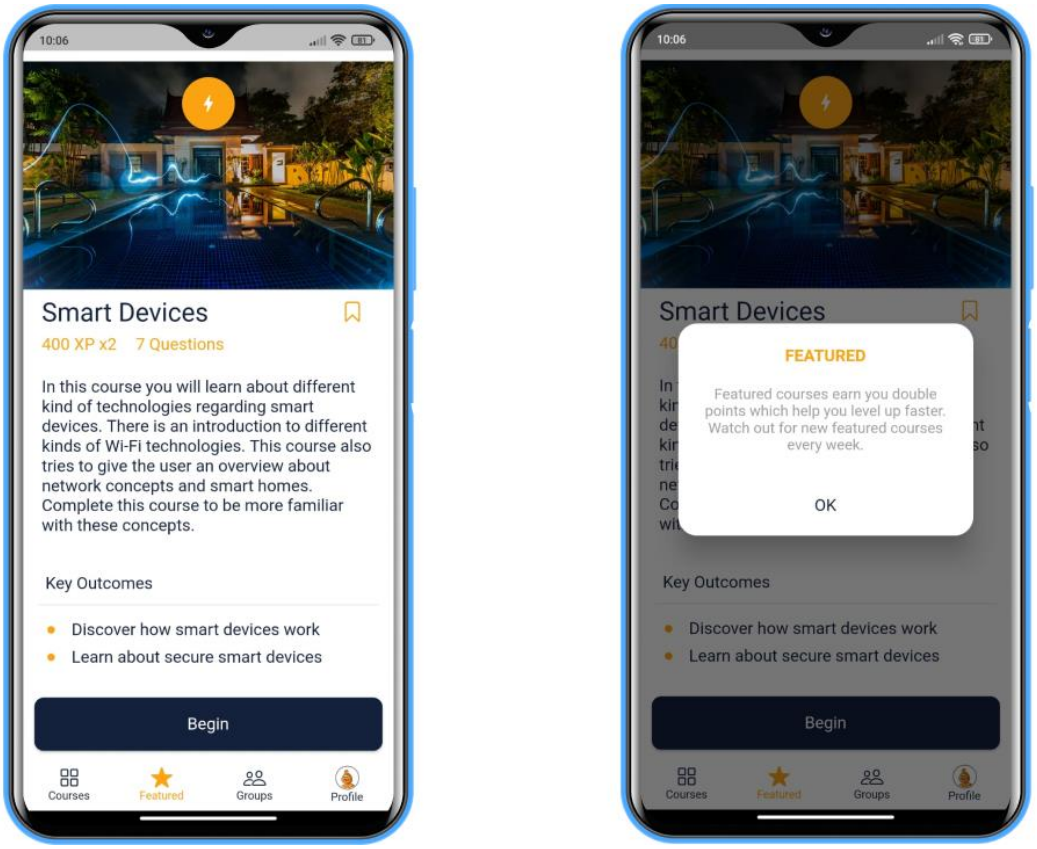


Figura 56 Curso destacado

Cuando el usuario pulsa el botón, se le notifica mediante un cuadro de diálogo que el curso en el que se encuentra es el destacado y que el número de puntos de experiencia está multiplicado por dos.

5.3.5 PERFIL

Otra de las secciones principales de la aplicación es la relativa al perfil del usuario. En esta sección se puede consultar toda la información relativa al usuario activo y para ello se utiliza el controlador "ActiveUserController" de donde se puede obtener la información del usuario y modificarla.

5.3.5.1 ProfilePage

Esta es la página principal del perfil a la cual se accede mediante la barra de navegación. En la Figura 57 se pueden observar dos pantallas cada una de las cuales muestra una sección de la página.

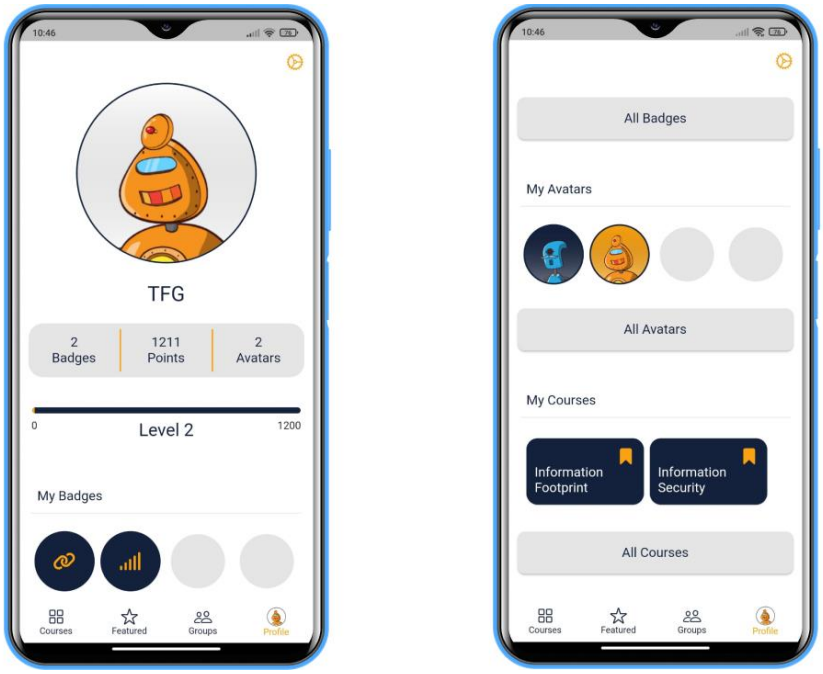


Figura 57 Página principal del perfil

Lo primero que se muestra es el Avatar activo del usuario y su nombre de usuario. Para ello se utiliza la clase Avatar explicada anteriormente en el informe y se especifica el valor del String que lo identifica mediante el atributo "profilePictureActive" del controlador "ActiveUserController". En cuanto al nombre de usuario, este se obtiene mediante el atributo "username" del controlador. Después, se puede observar un cuadro gris que muestra un resumen del progreso del usuario. En él se indica el número de insignias y de avatares que el usuario ha obtenido, así como el número total de puntos de experiencia que ha ganado. Para ello se ejecutan las funciones "getNumBadges", "getTotalPoints" y "getNumAvatars" definidas en el controlador. Debajo de este cuadro gris se puede ver un indicador de progreso lineal en el que se muestra la información del nivel. Para ello se accede al atributo "level" de tipo Level del controlador y se obtiene la información sobre el número de puntos de

experiencia que se pueden obtener en ese nivel para indicar el límite superior, así como el progreso dentro del nivel para obtener una proporción con la que rellenar el indicador de progreso. También se indica el número del nivel en el texto de debajo.

En las pantallas se puede observar que existen tres secciones. La primera de las secciones es relativa a las insignias que el usuario ha ganado. En esta sección, se accede a la lista "collectedBadges" del usuario activo y por cada una de las instancias de tipo "Badge" de la lista se crea una insignia como la mostrada en la figura mediante el código siguiente:

```
/**
 * Function to get a container with a badge from
 * the name of the badge stored in the user
 */
getContainerForBadge({required String nameOfIcon, required double size}) {
  return Container(
    width: size,
    height: size,
    decoration: BoxDecoration(
      shape: BoxShape.circle,
      color: primaryColor,
    ),
    child: Icon(
      FontAwesomeIconsMap[nameOfIcon],
      color: secondaryColor,
      size: 0.3 * size,
    ),
  );
}
```

Como puede observarse se ve cual es la entrada que le corresponde en el mapa "FontAwesomeIconsMap" donde se han definido los valores de tipo "IconData" con los cuales se puede construir un icono.

En caso de que el usuario cuente con menos de 4 insignias, se mostrarán círculos grises para completar la fila.

La segunda de las secciones hace referencia a los avatares que el usuario ha conseguido. Para ello se recorre de manera similar la lista definida en el atributo "collectedAvatars". Para cada Avatar se construye un widget para mostrarlo mediante el siguiente código:

```
class AvatarContainer extends GetView<ActiveUserController> {
  const AvatarContainer(
      {Key? key, required this.avatarName, required this.size})
      : super(key: key);

  final String avatarName;
  final double size;

  @override
  Widget build(BuildContext context) {
    return Obx(() => Container(
        child: Avatar(nameOfAvatar: avatarName, size: size),
        decoration: BoxDecoration(
          shape: BoxShape.circle,
          color: controller.profilePictureActive.value == avatarName
              ? secondaryColor
              : primaryColor,
        )));
  }
}
```

En caso de que el avatar sea el que el usuario tiene seleccionado como activo, se muestra con un fondo amarillo. Al igual que en la sección de las insignias si el usuario tiene menos de 4 avatares ganados se completa la fila con círculos grises.

La última de las secciones muestra los cursos que el usuario ha guardado. Si la lista "coursesSaved" contiene un único elemento, se muestra un cuadro gris donde iría el último curso. En caso de que la lista sea mayor de dos elementos se crea un widget desplazable donde se muestran todos los cursos guardados. Las tarjetas que se muestran en este caso son iguales que las que se han presentado en el epígrafe CategoryPage. Es decir, para construir las tarjetas se necesita el ID del curso y su título. Es por ello, que antes de crear el contenido de esta página se ejecuta un "FutureBuilder" donde la función que se ejecuta es "getCourseNamesByIDs". Con esta función definida en el "CourseController" se obtienen un mapa con entradas del estilo <courseID, title> con todos los cursos guardados por el usuario. A continuación, se puede observar el código que se ejecuta para crear una tarjeta de un curso:

```
Card getCardForCourse(
    {String courseID = '',
    bool isSaved = false,
```

```
  bool isCompleted = false,
  required BuildContext context,
  required String title,
  required double widthOfCard,
  required double heightOfCard,
  required bool isTemplate}) {
void Function() navigateToCourse = () {
  Navigator.pushNamed(context, CourseDescription.routeName,
      arguments: courseID);
};

return Card(
  color: primaryColor,
  borderOnForeground: true,
  shape: RoundedRectangleBorder(
      side: BorderSide(
          color: isTemplate ? secondaryColor : primaryColor, width: 2.0),
      borderRadius: BorderRadius.circular(15.0)),
  child: InkWell(
    splashColor: secondaryColor,
    borderRadius: BorderRadius.circular(15.0),
    onTap: isTemplate
        ? () {
            print('This is a template');
          }
        : navigateToCourse,
    child: SizedBox(
      height: heightOfCard,
      width: widthOfCard,
      child: Stack(children: [
        Row(
          mainAxisAlignment: MainAxisAlignment.end,
          children: [
            isCompleted
                ? Icon(
                    CupertinoIcons.checkmark_alt_circle_fill,
                    color: secondaryColor,
                  )
                : SizedBox(
                    width: 0,
                  ),
            isTemplate
                ? Padding(
                    padding: const EdgeInsets.all(8.0),
                    child:
                        Icon(CupertinoIcons.bookmark, color: secondaryColor),
                  )
                : SaveButton(isFilled: isSaved, courseID: courseID),
          ],
        ),
        Align(
          alignment: Alignment.bottomLeft,
          child: Padding(
```

```
                padding: EdgeInsets.only(
                  left: 0.07 * widthOfCard,
                  bottom: heightOfCard * 0.12,
                  right: 0.07 * widthOfCard,
                ),
                child: Text('$title', style: getNormalTextStyleWhite()),
                // Align(
                //     alignment: Alignment.center,
                //     child: Text('$title', style: getNormalTextStyleWhite())),
              ),
            ),
          ]),
        ),
      ),
    );
}
```

En todas las secciones también se presenta un botón para que el usuario acceda a una página con mayor detalle acerca de dicha sección. Las páginas se describen a continuación.

5.3.5.2 Badges

En la Figura 58 se pueden observar las páginas a las que se accede desde el botón "All Badges" de la página principal del perfil. En la primera de las páginas llamada "AllBadgesPage" se presentan cuatro secciones, una por cada categoría. En cada una de ellas se ejecuta un "FutureBuilder" a la hora de construirlas en las que se ejecuta la función "getCourseNamesFromCategory" para obtener un mapa con entradas del estilo <courseID, title> con todos los cursos pertenecientes a la categoría. Esta función está descrita en ANEXO V: Código del controlador "CourseController". Una vez se tiene este mapa se obtienen las tres primeras insignias que el usuario ha logrado dentro de esa categoría. Para ello se recorre la lista "collectedBadges" del usuario definida en el controlador "ActiveUserController" y se verifica qué insignias tienen un "courseID" contenido en el mapa con todos los cursos de la categoría.

En caso de que el usuario tenga un número menor de 3 insignias, se mostrarán las que tiene y círculos grises. En caso de que el usuario no tenga insignias se muestra un texto. Ejemplos de ambos casos aparecen en la primera página de la Figura 58. A la hora de crear el widget mostrado se ejecuta el código descrito a continuación:

```
Widget getIconButtonForBadge(
    {required Badge badge,
    required String nameOfCourse,
    required double size,
    required BuildContext context}) {
  return Container(
    decoration: BoxDecoration(
      color: primaryColor,
      shape: BoxShape.circle,
    ),
    width: size,
    height: size,
    child: IconButton(
      splashColor: secondaryColor,
      onPressed: () {
        showDialog(
            context: context,
            barrierDismissible: true,
            builder: (_) {
              return BadgeDialog(badge: badge, nameOfCourse: nameOfCourse);
            });
      },
      icon: Icon(
        FontAwesomeIconsMap[badge.picture],
        color: secondaryColor,
        size: 24,
      ),
    ),
  );
}
```

A diferencia del widget que se ha presentado en la página principal del perfil, este widget es un botón que cuando se pulsa muestra información relativa a la insignia como se aprecia en la tercera de las pantallas. El último de los widgets que se muestra en la fila es otro botón mediante el cual se navega a la página que ocupa la segunda posición en la figura. Esta página se llama “CategoryBadges” y al navegar a ella se especifica como argumento una instancia de un objeto llamado “CategoryBadgesArgs” formado por el mapa obtenido con los cursos de la categoría y la lista de insignias del usuario activo. Esta página existe para mostrar todas las insignias sin el límite de 3 insignias de la primera de las páginas.

Figura 58 Insignias del usuario

5.3.5.3 Courses

En la Figura 59 se muestran las dos páginas a las que el usuario puede acceder si pulsa "All Courses" en la página principal del perfil. El comportamiento de estas páginas es muy similar al descrito para las insignias en el epígrafe Badges. Por tanto, solo se comentarán las diferencias. En este caso, se construye como widgets tarjetas de cursos cuyo código se ha presentado anteriormente en el informe. Se construye una tarjeta por cada curso en la categoría que haya sido guardado, completado o ambas.

Además, en este caso también se presenta un widget al inicio de la primera página llamada "AllBadgesPage" en la cual se indica cuántos cursos el usuario ha completado, cuántos ha guardado y el número de puntos de experiencia que ha obtenido. Para obtener el número de cursos que el usuario ha completado y que ha guardado simplemente se obtiene la longitud de las listas "savedCourses" y "completedCourses" del usuario activo definidas en el controlador "ActiveUserController". Para obtener los puntos se ejecuta la función "getTotalPoints".

Siempre que se pulse cualquiera de las tarjetas, el usuario se verá redirigido a la página de descripción del curso.

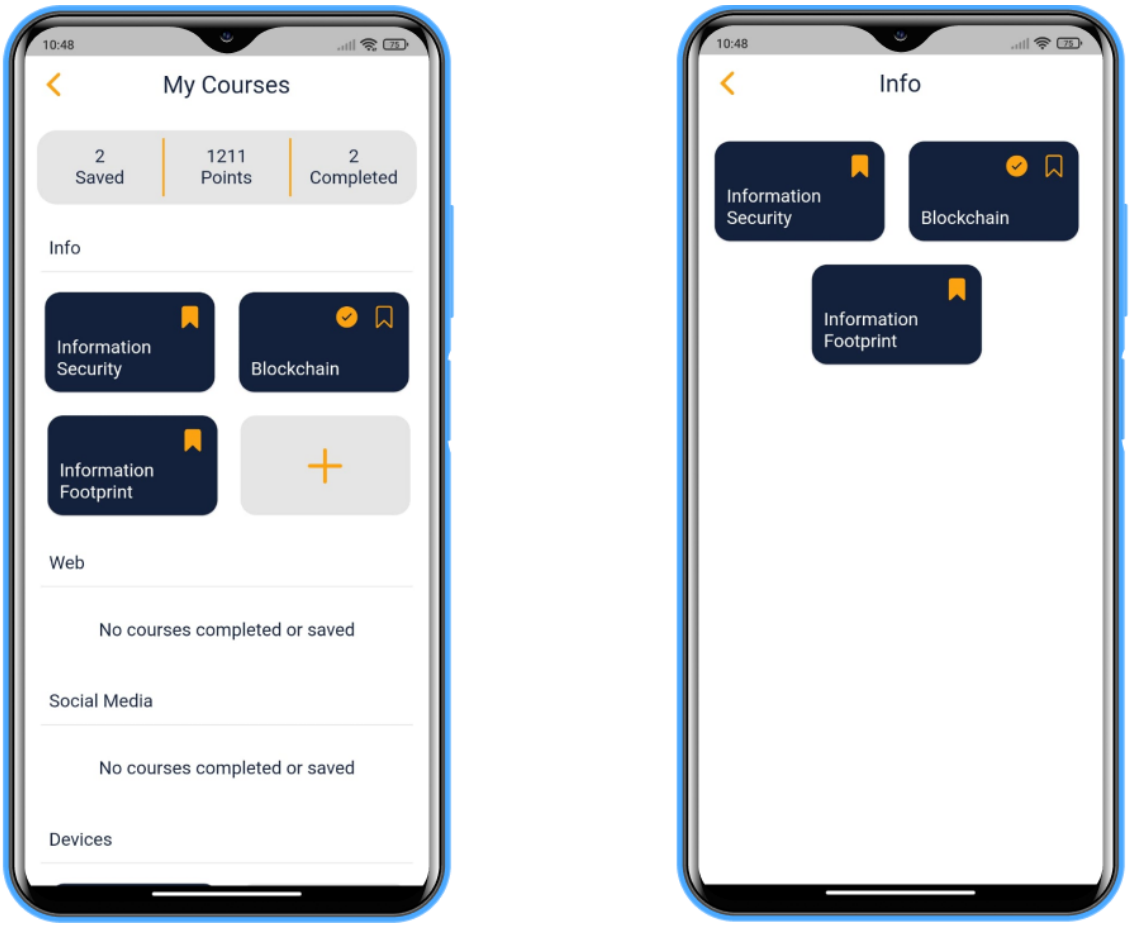


Figura 59 Cursos del usuario

5.3.5.4 Avatars

En la Figura 60 se puede observar la página a la que el usuario accede si pulsa el botón "All Avatars" presente en la página principal del perfil. Esta página se llama "AllAvatarsPage". En ella no se realiza ninguna llamada a la base de datos ya que se utiliza directamente información relativa al usuario activo que se obtiene del controlador "ActiveUserController".

A la hora de crear esta página se recorre la lista "collectedAvatars" del usuario activo y para cada uno de los avatares se crea un botón con el código adjuntado a continuación. Cabe

destacar que si el botón es para el avatar que el usuario tiene como avatar activo se crea con un color de fondo amarillo para diferenciarlo.

```
class AvatarButton extends StatelessWidget {
  AvatarButton(
      {Key? key, required String this.avatarName, required double this.size})
      : super(key: key);

  final String avatarName;
  final double size;

  ActiveUserController activeUserController = Get.find<ActiveUserController>();

  @override
  Widget build(BuildContext context) {
    return Obx(() => ElevatedButton(
          style: ElevatedButton.styleFrom(
              shape: const CircleBorder(),
              primary: activeUserController.profilePictureActive == avatarName
                  ? secondaryColor
                  : primaryColor),
          onPressed: () {
            showDialog(
                context: context,
                barrierDismissible: true,
                builder: (_) {
                  return AvatarDialog(
                    avatarName: avatarName,
                  );
                });
          },
          child: Avatar(nameOfAvatar: avatarName, size: size),
        ));
  }
}
```

Cuando el usuario pulsa el botón del avatar se abre un cuadro de diálogo en el que se muestra la imagen del avatar, su identificador y se da la posibilidad al usuario de establecerlo como su avatar activo. Esto se muestra en la segunda de las pantallas de la figura. En caso de que el usuario pulse el botón aceptar, se cambia su atributo “profilePictureActive” y se establece dicho avatar como su avatar activo. Para ello se ejecuta la función definida en “ActiveUserController”:“updateProfilePictureActive”. Esta función se puede observar con detalle en ANEXO IV: Código del controlador “ActiveUserController”. No solo se modifica el valor del atributo del controlador si no que se persiste dicho cambio en la base de datos.

Esto siempre ocurre cuando se modifica cualquiera de los atributos del usuario ya que si no se desincronizaría la información de la aplicación con la de la base de datos.

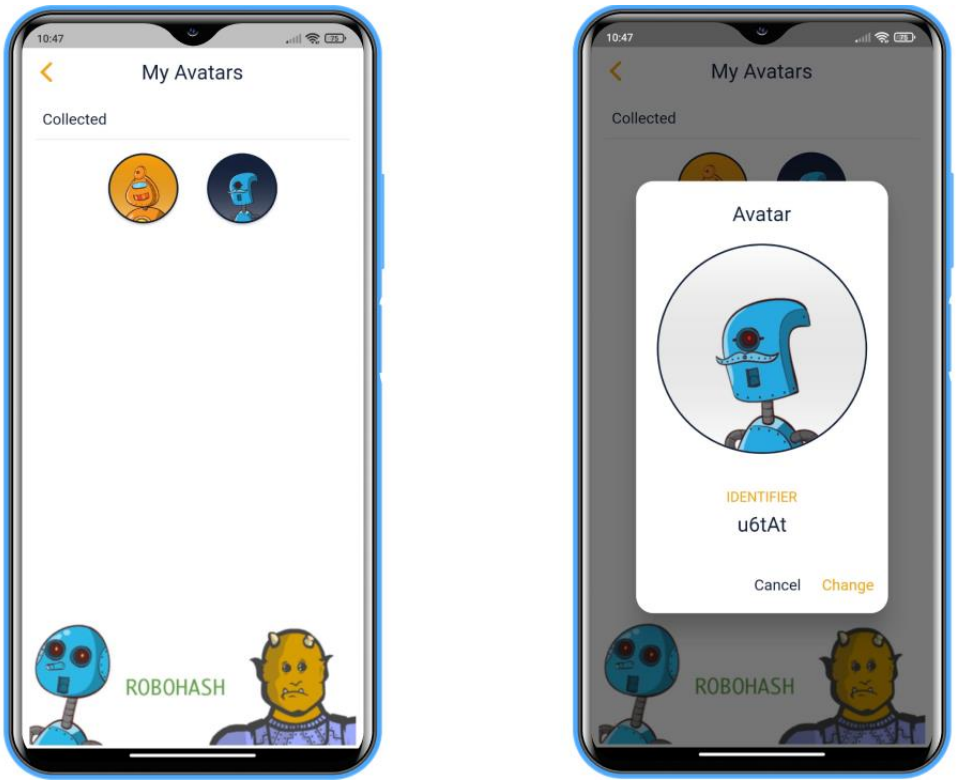


Figura 60 Avatares del usuario

5.3.5.5 Edit Profile Info

La última de las subsecciones dentro del perfil son las pantallas relativas a la modificación de la información del usuario y a acciones más relacionadas con la configuración. En la Figura 61 se pueden observar las pantallas restantes de la sección del perfil a las cuales se puede acceder pulsando en el icono presente en la esquina superior derecha de la página principal del perfil.

Las primeras dos pantallas muestran dos secciones de la misma pantalla llamada "EditProfilePage". En ella lo primero que se muestra es el avatar activo del usuario y un botón desde el cual el usuario puede acceder a la página "AllAvatarsPage" para modificarlo. Debajo de este botón se ofrece la posibilidad al usuario de cambiar su atributo "username".

Para ello el usuario debe introducir un nuevo texto en el campo y al pulsar el botón se abre un cuadro de diálogo preguntando al usuario de si está seguro de que quiere llevar a cabo dicha acción. En caso afirmativo la función "changeUsername" definida en "ActiveUserController" se ejecuta. Esta función llama a su vez a la función "updateUsername" definida en "UserController" que verifica si ya existe el nombre de usuario escogido y notifica al usuario en ese caso.

En esta página el usuario también tiene la posibilidad de navegar a la página presentada en la tercera captura de la Figura 61. En esta página el usuario puede cambiar su contraseña por una nueva. Para ello debe de especificar de nuevo su contraseña actual por motivos de seguridad. Una vez presiona el botón se le muestra un cuadro de diálogo para confirmar la acción. En cuanto el usuario la confirma se ejecuta la función del "UserController" llamada "updatePassword". Cabe destacar que en este caso el flujo no es primero "ActiveUserController" y luego "UserController" ya que la contraseña no se almacena en la información del usuario activo. En la función indicada, se vuelve a autenticar al usuario con su contraseña actual ya que es necesario para llevar a cabo operaciones como esta. Una vez el usuario se ha vuelto a autenticar, se procede a cambiar la contraseña y cualquier error en el proceso se le notifica al usuario. Para ver de manera detallada el código de la función "updatePassword" el usuario puede ir a ANEXO III: Código del controlador "UserController".

Por último, se presenta al usuario la posibilidad de salir de la cuenta o borrar la cuenta mediante los botones del final de la página. A la hora de pulsar cualquiera de los botones se abre un cuadro de diálogo para confirmar la acción. Si el usuario continua, se ejecuta la función asociada a la acción. En el caso de que el usuario pulse el botón "Sign Out" se ejecuta la función del "UserController" llamada "signOutUser". Si, por el contrario, el usuario pulsa el botón "Delete Account" se ejecuta la función "deleteActiveUser". En esta función también se llama a la función "deleteUserFromFirestore" ya que no tiene sentido borrar la cuenta de la base de datos de autenticación mediante la función que Firebase Authentication ofrece para ello y dejar en la colección de usuarios de Cloud Firestore una entrada con toda la información relativa al usuario. Para que la función de borrar la cuenta

de la base de datos de Firebase Authentication funcione se requiere que el usuario se haya identificado recientemente al igual que cuando se cambia la contraseña por ser una operación delicada. A diferencia del cambio de contraseña, en este caso no se realiza esta acción por el usuario si no que se le notifica de la necesidad de que esté recientemente autenticado en caso de que la función falle.

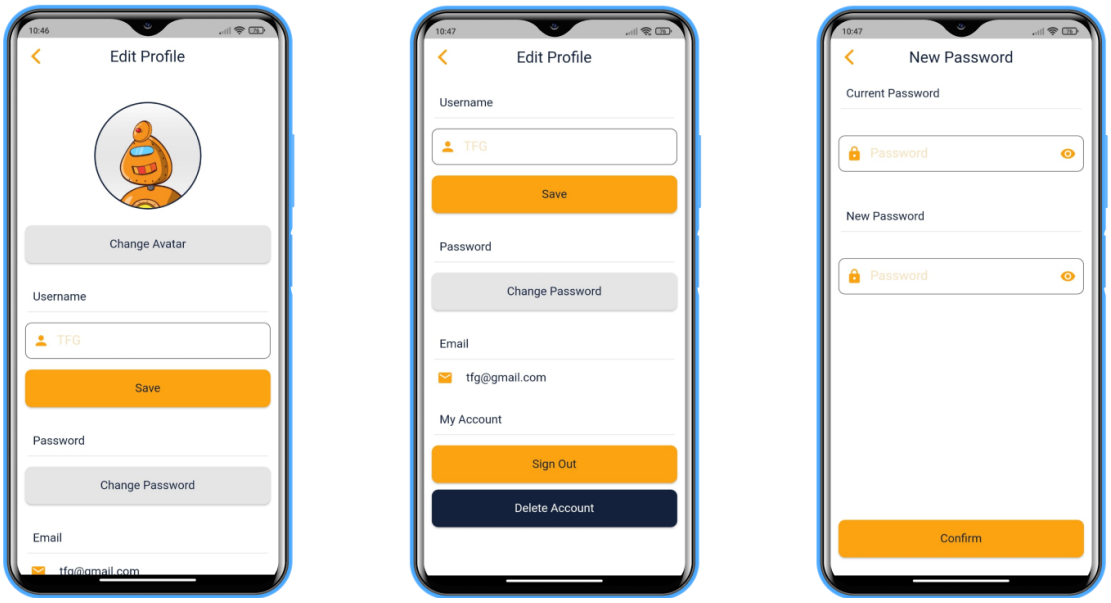


Figura 61 Configuración del usuario

5.3.6 ADMINISTRADOR

La última de las secciones principales de la aplicación es la relativa al administrador. Cuando el usuario se autentica en la aplicación es redirigido a el "Dashboard" como ya se ha discutido previamente. A la hora de construir esta página mediante el método "build" del widget, se comprueba el valor del atributo "isAdmin" del usuario activo mediante el controlador "ActiveUserController". En caso de que este valor este a "true", el usuario tiene derechos de administrador y se muestra un botón flotante para que pueda acceder desde el "Dashboard" al panel de administración.

En la Figura 62 se puede observar cómo se ve la página "Dashboard" cuando un administrador accede a la aplicación. En la segunda de las pantallas se puede ver el panel de

administrador, esta página se llama "AdminDashboardPage". En ella se aprecian las distintas acciones que el administrador puede llevar a cabo. Estas se describen en mayor detalle a continuación.

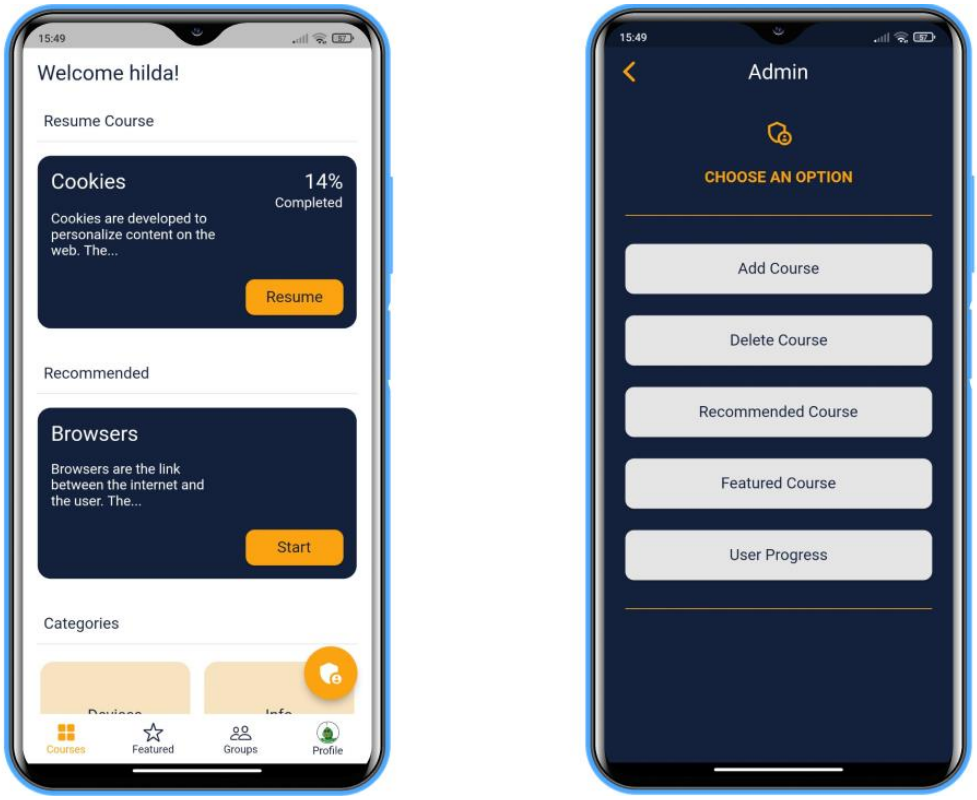


Figura 62 Panel de administrador

5.3.6.1 User Progress

El administrador tiene la posibilidad de comprobar el progreso de un usuario en la aplicación. Para ello, debe de introducir el nombre de usuario en el buscador y presionar el icono de la lupa. En la Figura 63 se pueden ver tres pantallas. La primera de ellas muestra como se ve la página "UserProgressPage" antes de realizar ninguna búsqueda. La segunda muestra como se ve la pantalla cuando se introduce un nombre de usuario existente y la tercera muestra el mensaje de error que se muestra cuando no hay ningún usuario con dicho nombre de usuario.

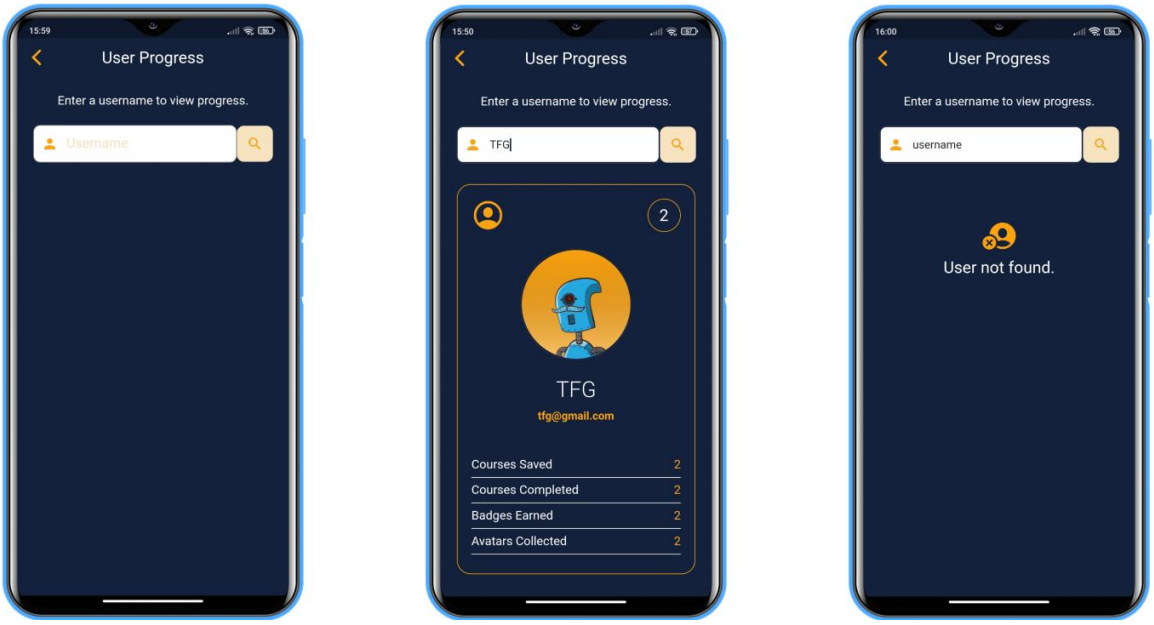


Figura 63 Comprobación de progreso de usuario

A la hora de construir esta página, se carga previamente la información de todos los usuarios de la aplicación con la función "getAllUsers" definida en el controlador "UserController". Esta información se guarda en una lista de objetos tipo "UserCustom". Esta función se ejecuta en un "FutureBuilder" al crear la página. En este caso, la página es un widget "stateful" esto quiere decir que su estado es cambiante. El estado de la página varía cuando se pulsa la lupa y se ejecuta la siguiente función:

```
void Function() formFunction = () {
  if (_formKey.currentState!.validate()) {
    setState(() {
      _userSearched = true;
      _username = _controllerUsername.text.trim();
      if (checkIfUsernameExists(username: _username)) {
        _userValid = true;
      } else {
        _userValid = false;
      }
    });
  }
};
.
.
.
bool checkIfUsernameExists({required String username}) {
  for (UserCustom u in widget.users) {
```

```
    if (u.username.toLowerCase() == username.toLowerCase()) {
      return true;
    }
  }
  return false;
}
```

Al ejecutar esta función se busca en la lista de usuarios un usuario con el mismo username que el texto que se ha introducido en el campo. De acuerdo con el resultado de esta busca se cambian los valores de las variables "_userSearched" y "_userValid" que determinan el contenido de la página.

5.3.6.2 Change Featured/Recommended

El administrador también tiene la posibilidad de modificar el curso recomendado que aparece en el "Dashboard" y de cambiar el curso destacado que ya se ha comentado. A continuación, se describe el proceso para cambiar el curso recomendado únicamente ya que es igual que para cambiar el curso destacado.

Como se puede ver en la primera captura de la Figura 64, el administrador debe de elegir primero una de las categorías donde se encuentra el curso que quiere convertir en recomendado. Una vez el administrador pulsa una de las categorías, se le redirige a la pantalla mostrada en la segunda de las capturas que se llama "PickACoursePage". Esta página tiene un "FutureBuilder" en su método "build" y la función que se ejecuta es "getCourseNamesFromCategory" definida en el controlador "CourseController" cuyo código puede consultarse en ANEXO V: Código del controlador "CourseController". Con la función se obtienen los cursos relativos a la categoría que el administrador ha pulsado en la página anterior. Estos cursos no se obtienen completamente, sino que se obtiene un mapa del estilo <courseID, title> para formar la lista que se puede observar en la captura.

Cabe destacar que en la página anterior llamada "UpdateRecommendedPage" en el caso del curso recomendado y "UpdateFeaturedPage" en el caso del curso destacado se ha obtenido el identificador del curso recomendado o destacado, respectivamente. Este identificador se pasa como argumento a la siguiente página con el objetivo de marcar el curso recomendado

o destacado al construir la lista, en caso de que se encuentre en la misma. También se utiliza el valor para no ejecutar ninguna función en caso de que se quiera convertir en curso destacado aquel que ya lo es.

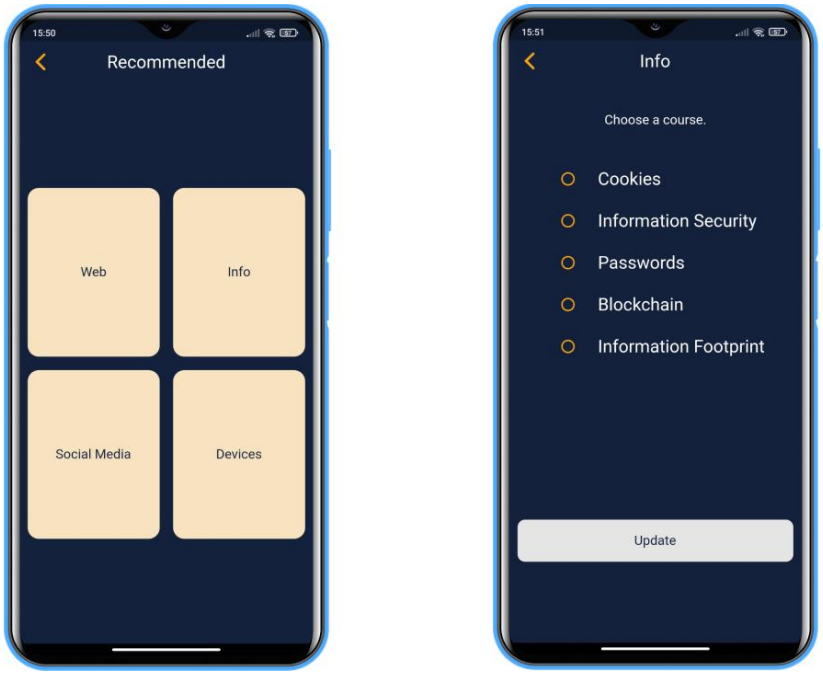


Figura 64 Cambiar curso recomendado

Una vez el administrador ha decidido el nuevo curso y presiona el botón “Update” se ejecuta la siguiente función:

```
void Function() updateFunction = () async {
  String message = '';
  if (widget.args.idOfCourse != _selectedCourseID) {
    final Future Function({required String courseID}) futureToExecute =
        widget.args.isFeatured
            ? cc.updateFeaturedCourseByID
            : cc.updateRecommendedCourseByID;

    await futureToExecute(courseID: _selectedCourseID!).then((val) {
      message =
          'Update was successful. Course:
${widget.coursesMap[_selectedCourseID]!}';
    }).catchError((error) {
      message = 'Update was unsuccessful.';
    });
  } else {
    message = 'This course is already selected.';
```

```
  }

  var snackBar = SnackBar(
    backgroundColor: secondaryColor,
    content: Text(
      message,
      style: getNormalTextStyleWhite(),
    ),
  );
  ScaffoldMessenger.of(context).showSnackBar(snackBar);
  Navigator.pushNamedAndRemoveUntil(
      context, HomePage.routeName, (r) => false);
};
```

Como puede observarse, dentro de esta función se ejecuta otra función que se ha denominado "futureToExecute". El valor de esta variable depende de si se ha accedido a la página "PickACoursePage" desde la opción de cambiar el curso recomendado o el destacado. Esto se sabe ya que se indica como argumento al navegar a la página. Ambas funciones son similares y el código se puede consultar en ANEXO V: Código del controlador "CourseController". En estas funciones se cambia el identificador del curso recomendado o destacado en la base de datos por el identificador del curso que se haya elegido.

5.3.6.3 Delete Course

El administrador tiene la posibilidad de eliminar un curso ya creado. Para ello, debe de presionar el botón "Delete Course" en la página principal y se ve redirigido a la página "DeleteCoursePage" que se muestra en la Figura 65.

A la hora de construir esta página se obtienen todos los cursos de la aplicación mediante la función "getCourseNames" con el mismo formato que se ha descrito en el epígrafe anterior. La función es ejecutada en un "FutureBuilder". Una vez se ha obtenido el mapa con los identificadores de los cursos y sus títulos, se muestra una lista con todos los títulos ordenados alfabéticamente para que el administrador pueda encontrar fácilmente el curso que desea eliminar.

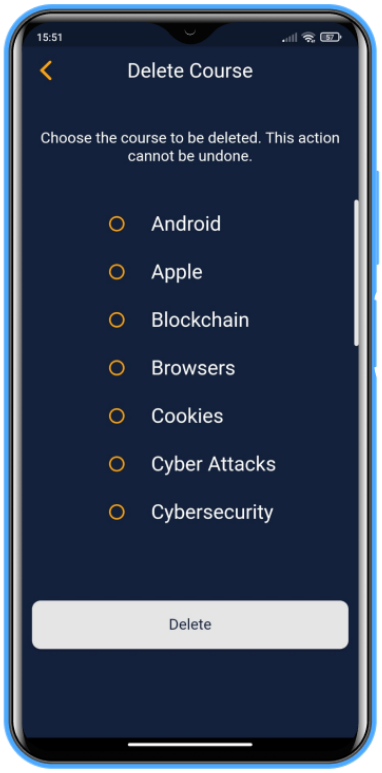


Figura 65 Borrar un curso

Una vez el usuario ha elegido el curso que desea eliminar y presiona el botón, se ejecuta la siguiente función:

```
void Function() deleteFunction = () async {
  String message = '';

  await cc
      .deleteCourseByTitle(widget.courses[_selectedCourseID!]!)
      .then((value) {
    message = 'Deleted course with ID: ${value}';
  }).catchError((onError) {
    message = 'Error deleting course';
  });
```

Como puede observarse esta función llama a la función “deleteCourseByTitle” definida en el controlador “CourseController” cuyo código puede consultarse en ANEXO V: Código del controlador “CourseController”. Cabe destacar de esta última función que para eliminar el documento relativo al curso de la colección de cursos de Cloud Firestore hace falta eliminar

la colección asociada llamada “questions” donde se guardan las preguntas del curso. Si este paso no se realiza manualmente, la colección no se eliminará de manera automática, aunque se elimine el documento.

5.3.6.4 Add Course

La última de las funcionalidades que tiene el administrador es la de añadir un nuevo curso a la base de datos de Cloud Firestore. Para ello el administrador debe de rellenar una serie de campos que se le ofrecen en forma de formulario con valores correctos.

En la Figura 66 se pueden observar la primera de las pantallas en la que se indica información relativa al curso. En esta página se debe de indicar el título del curso, la categoría a la que pertenece, el número de puntos de experiencia que aporta y la URL de la imagen del curso. En cuanto a la categoría, se ofrece al administrador una serie de opciones fijas formadas por las cuatro categorías de la aplicación. Para especificar la URL de la imagen, el administrador debe de pulsar el botón “Choose” lo cual despliega un cuadro de diálogo. En el cuadro de diálogo que es un widget de tipo “stateful” cuando el administrador especifica una URL y pulsa el botón “Submit” se cambia el estado de dicho widget para mostrar la imagen en caso de que la URL especificada sea válida. En caso de que esta no lo sea, se notifica al usuario con un texto.

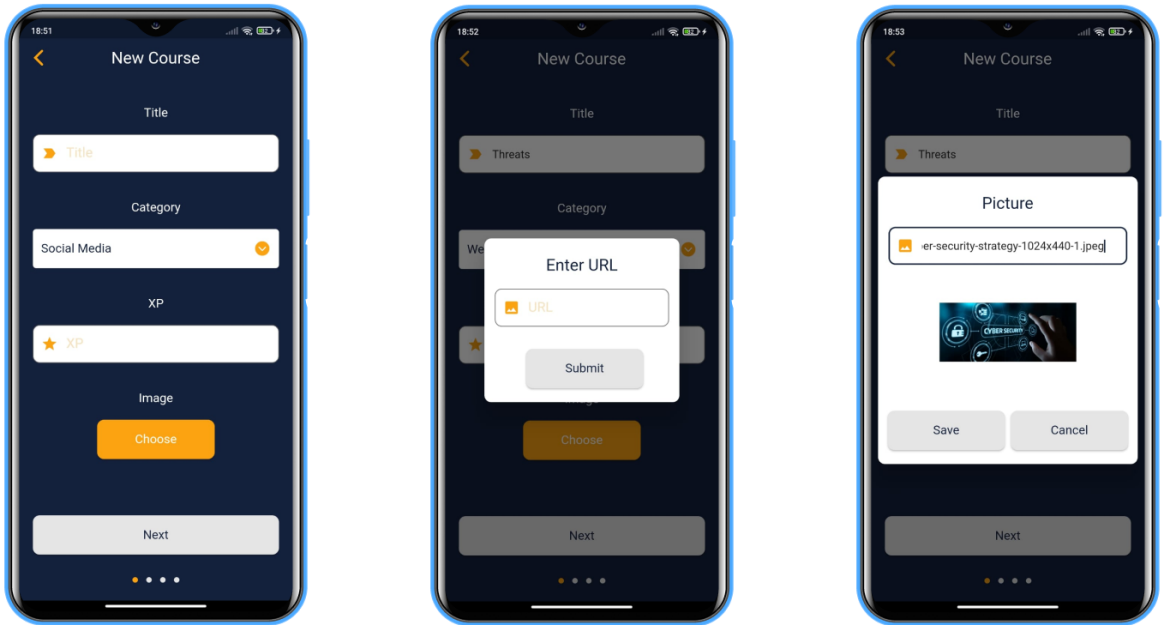


Figura 66 Añadir un curso parte I

Una vez el usuario ha rellenado todos los campos y pulsa el botón "Next" se ejecuta el siguiente código:

```
void Function() updateCourseFields = () async {
  bool imageIsValid = await checkImageSF();

  if (!imageIsValid) {
    var snackBar = SnackBar(
      backgroundColor: secondaryColor,
      content: Text(
        'Valid image not found',
        style: getNormalTextStyleWhite(),
      ),
    );
    ScaffoldMessenger.of(context).showSnackBar(snackBar);
  }

  //Returns true if the form is valid
  if (_formKey.currentState!.validate() && activeButton && imageIsValid) {
    //I create a new-course with dummy values
    activeButton = false;

    Course newCourse = Course(
        imageURL: '',
        title: '',
        category: k_values.Category.info,
        positionInCategory: 1,
        numberOfQuestions: 1,
```

```
        experiencePoints: 1,
        description: '',
        badgeIcon: '',
        outcomes: [],
        questions: []);

    //I update the fields
    newCourse.title = _controllerTitle.text.trim();
    newCourse.experiencePoints = int.parse(_controllerXP.text.trim());
    newCourse.category = value;
    newCourse.imageURL = await getImageUrlSF();

    Navigator.pushNamed(context, NewCourseOutcomesPage.routeName,
        arguments: newCourse);
  }
};
```

Como puede verse en el código, si la imagen especificada es válida y el formulario está correctamente rellenado, se crea una instancia del objeto Course en la cual se inicializan todos sus atributos especificando los valores obtenidos en esta pantalla. Aquellos atributos que no se han especificado todavía se inicializan a valores nulos o aleatorios. Posteriormente se navega a la siguiente página y se pasa el curso como argumento.

En la Figura 67 se puede ver las siguientes páginas en las que se especifican más atributos relativos al curso. En la primera de las páginas se especifican los puntos de aprendizaje del curso, en la segunda se especifica la descripción del curso y la posición que ocupa en la categoría. Por último, se indica cual será el icono que se utilizará en la insignia del curso. Para ello, el administrador debe de seguir los pasos descritos en la pantalla. Principalmente consiste en buscar el nombre del icono en la página web “Font Awesome” de donde se han obtenido los iconos. Cabe destacar que cuando el usuario especifica un String, no se busca una coincidencia en la página web sino en el mapa que se ha definido como constante en el proyecto. Como ya se ha comentado en otras ocasiones en este informe, se ha hecho necesario definir un mapa con entradas del estilo <String, IconData> en la aplicación, para a partir de un atributo de tipo String utilizado tanto en el curso como en la propia clase Badge, construir un icono con su equivalente IconData asociado del mapa. Esto se ha hecho ya que la base de datos no permitía guardar un tipo de dato de este estilo.

Esto se ha comentado ya que el mapa de la aplicación no representa la versión más actualizada de los iconos de la página web "Font Awesome". Por tanto, en ocasiones se mostrará un error a pesar de que el icono si existe en la web ya que no existe en la versión local del mapa.

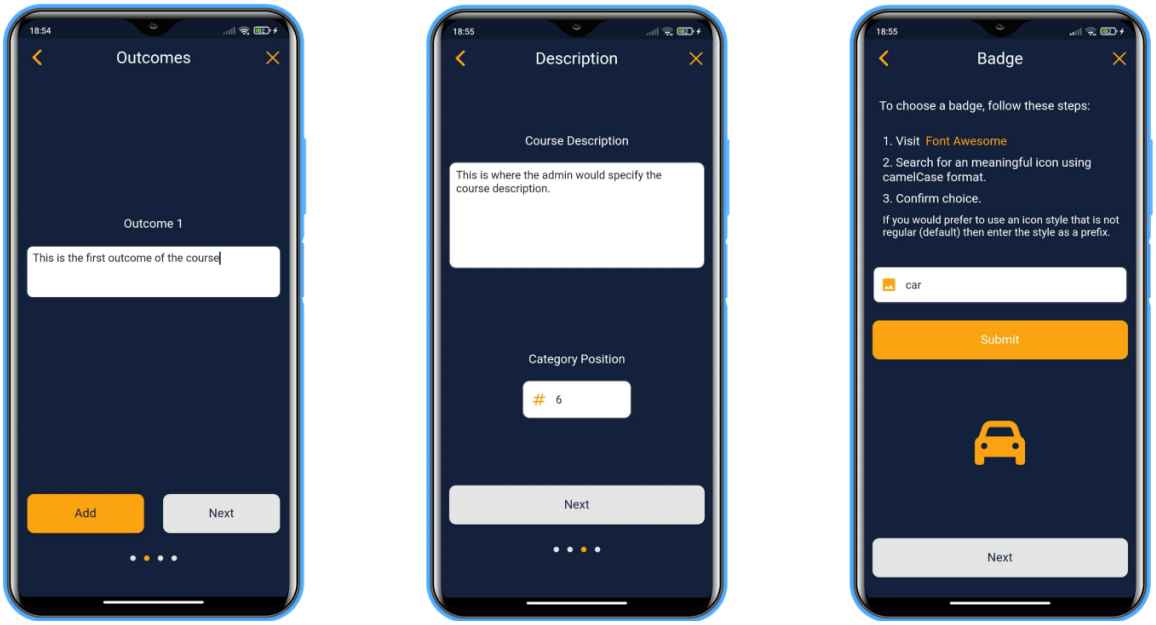


Figura 67 Añadir un curso parte II

Una vez se han pasado estas tres páginas y se ha modificado la información de la instancia del objeto Course con la información de cada pantalla, se procede a crear las preguntas.

En la Figura 68 se ve la pantalla a la que se accede después de tener toda la información relativa al curso menos las preguntas.

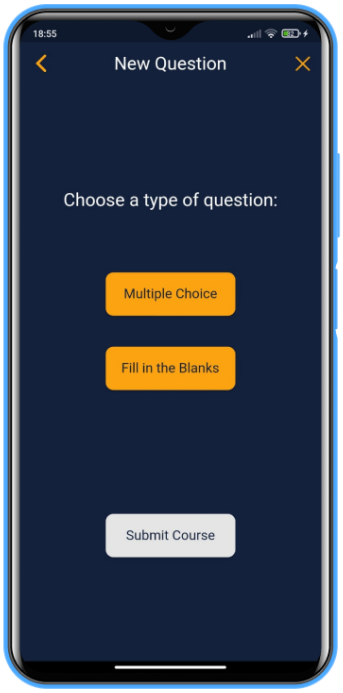


Figura 68 Añadir una pregunta

Una vez en esta pantalla se puede crear una nueva pregunta de tipo opción múltiple o de tipo rellenar los huecos. También existe la opción de subir el curso. No obstante, esto dará un error en caso de que no se hayan añadido al menos 5 preguntas al curso.

En la Figura 69 se puede ver el flujo de pantallas para crear una nueva pregunta de tipo opción múltiple. En la primera de las pantallas se especifica la pregunta como tal. Una vez se pulsa el botón "Next", se ejecuta la siguiente función:

```
void Function() setQuestionDescription = () async {
  //I validate the form

  if (_formKey.currentState!.validate()) {
    //I create a Multiple Choice Question
    Question newQuestion = MultipleChoiceQuestion(
        number: newQuestionNum!,
        description: _controllerDescription.text.trim(),
        typeOfQuestion: TypeOfQuestion.multipleChoice,
        longFeedback: '',
        options: [],
        rightOption: 1);

    //Some fields are initialized to random variables
    Navigator.pushNamed(context, MultipleChoiceOptionsPage.routeName,
```

```
        arguments: newQuestion);
  }
};
```

En esta función se crea una instancia de un objeto tipo "MultipleChoiceQuestion" en el cual se especifica el atributo "description". Acto seguido se navega a la segunda de las pantallas pasando el curso como argumento. En la segunda y tercera de las pantallas se terminan de especificar los atributos de la pregunta que son las opciones y cuál es la opción correcta. Respecto al atributo que faltaría que es "longFeedback" este se especifica en una pantalla común a los dos tipos de pregunta que no se muestra en esta figura ya que se muestra en la siguiente figura cuando se habla de la pregunta de tipo rellenar los huecos.

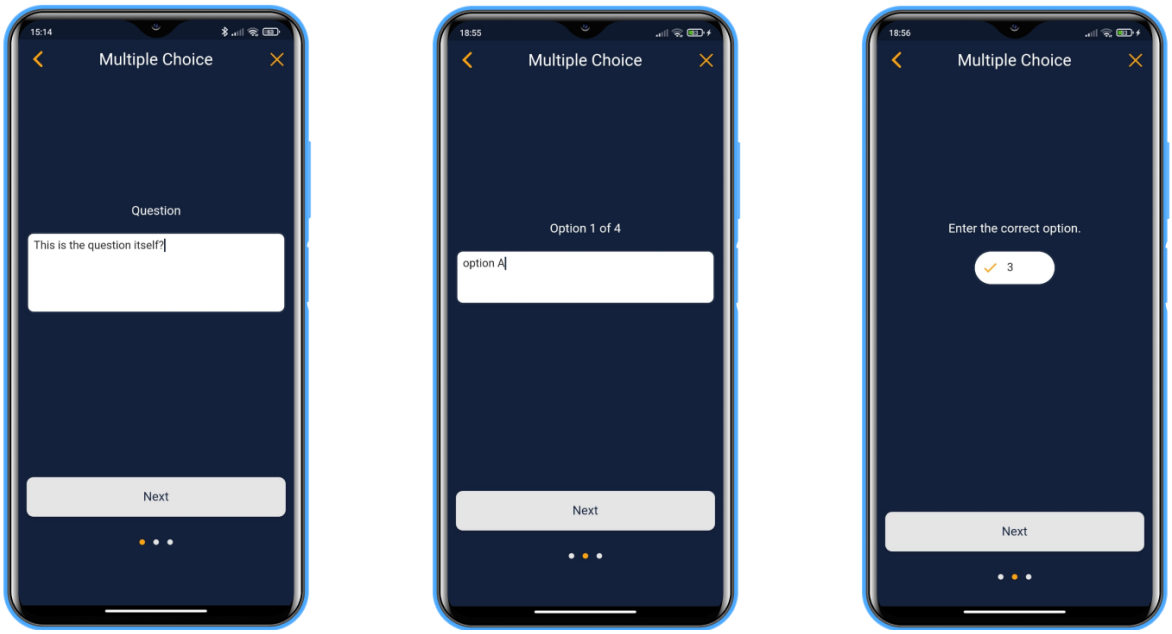


Figura 69 Añadir pregunta tipo opción múltiple

En la Figura 70 se pueden ver las pantallas relativas a la creación de una pregunta del tipo rellenar los huecos. Como puede verse en la primera de las pantallas se especifica el atributo "text" situando una X donde se quiere que se coloquen los huecos. En la segunda de las pantallas se crea el atributo "solution" y el atributo "options". Se van indicando Strings en el campo de texto que se ofrece y el administrador elige si quiere que el texto introducido sea una opción o una solución. Cabe destacar que, si el administrador añade un texto como

solución, este es automáticamente añadido a la lista de opciones como es lógico. Si el usuario intenta avanzar sin especificar un número correcto de soluciones se le avisara que no puede continuar y el motivo de ello. La última de las pantallas es la que es común a ambos tipos de pregunta y es en la que se especifica la explicación detallada de la respuesta a la pregunta.

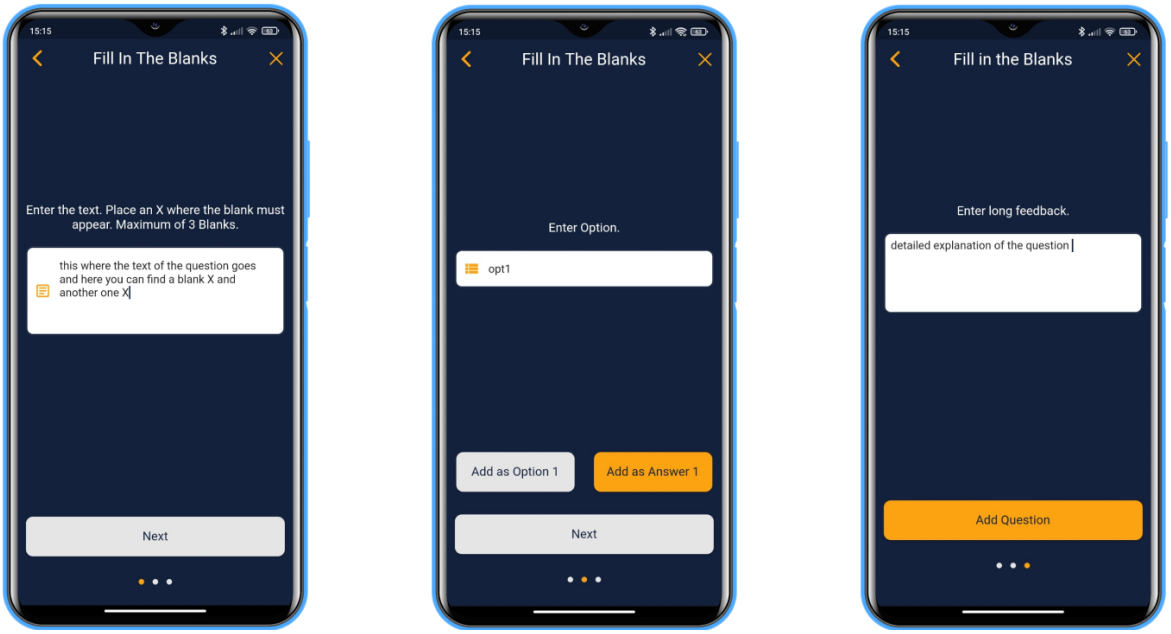


Figura 70 Añadir pregunta tipo rellenar los huecos

En la primera de las pantallas, el código ejecutado al presionar el botón es similar al presentado para el otro tipo de pregunta. En él se crea una instancia de un objeto tipo "FillInTheBlanksQuestion" y se navega a la siguiente página pasando dicha instancia como argumento para posteriormente poder especificar el resto de los atributos.

Cuando el usuario pulsa el botón "Add Question" en la última de las pantallas, se ejecuta el siguiente código:

```
void Function() addQuestionToCourse = () {
  if (_formKey.currentState!.validate()) {
    widget.question.longFeedback = _controllerFeedback.text;

    //I have the question completely built so I add it to the new-course
    //Also I need to update the question number

    newCourse!.questions.add(widget.question);
    newQuestionNum = newQuestionNum! + 1;
```

```
    //I show a message of success
    ScaffoldMessenger.of(context).showSnackBar(snBar);

    //I wait before navigating
    Future.delayed(Duration(seconds: 3));

    Navigator.pushNamed(
      context,
      NewQuestionPage.routeName,
    );
  }
};
```

Como puede verse se añade la pregunta al curso. Para ello, se accede al curso mediante una variable global que se ha creado para facilitar el acceso al curso y que se inicializa una vez se han especificado todos los atributos relativos al curso menos las preguntas y se accede por primera vez a la página llamada "NewQuestionPage" en donde se empiezan a crear dichas preguntas. Después de añadir la pregunta al curso se vuelve a la página "NewQuestionPage".

Una vez ya se han añadido todas las preguntas que se quieren al curso y se pulsa el botón "Submit Course" se ejecuta la siguiente función:

```
void Function() addCourse = () async {
  if (newCourse!.questions.length < 5) {
    showDialog<void>(
      context: context,
      barrierDismissible: true,
      builder: (BuildContext context) {
        return AlertDialog(
          shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(15)),
          title: Text(
            'Questions Required',
            style: getSubheadingStyleBlue(),
            textAlign: TextAlign.center,
          ),
          content: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            mainAxisSize: MainAxisSize.min,
            children: [
              Icon(
                Icons.clear_rounded,
                size: 0.3 * widthOfScreen,
```

```
                color: Colors.red,
              ),
              SizedBox(
                height: 0.05 * heightOfScreen,
              ),
              Text('Add at least 5 questions to add a new course.',
                  style: getNormalTextStyleBlue(),
                  textAlign: TextAlign.center),
            ],
          ),
        );
      },
    );
  } else {
    final CourseController courseController = CourseController();

    String message='';

    //Before uploading the new-course we have to set the number of questions

    newCourse!.numberOfQuestions = newCourse!.questions.length;

    //After adding the new-course we show a message of success
   await courseController
        .addCourseToFirebase(courseToAdd: newCourse!)
        .then((value) {
        message='Course added';
    }).catchError((error){
        message='Course not added';
    });

      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
      content: Text(
      message,
      style: getNormalTextStyleBlue(),
      ),
      backgroundColor: secondaryColor,));

      //After that I navigate
    Navigator.pushNamedAndRemoveUntil(context, HomePage.routeName, (r) => false);

  }
};
```

Esta función como puede observarse, a su vez llama a la función “addCourseToFirebase” del controlador “CourseController” cuyo código se puede consultar en ANEXO V: Código del controlador “CourseController”. En el anexo se puede observar que dicha función crea

un documento en la base de datos, más concretamente en la colección de los cursos y que una vez ha creado el curso, crea la subcolección asociada al documento para las preguntas.

Capítulo 6. ANÁLISIS DE RESULTADOS

En este capítulo se indican los resultados que se han obtenido en el proyecto y como se han comprobado los mismos.

6.1 CAPA DE VISTA

Una de las primeras comprobaciones que se ha llevado a cabo en durante la realización del proyecto es la validación de la capa de vista a medida que se programaban las distintas pantallas diseñadas en Figma. Android Studio ofrece la posibilidad de utilizar un emulador de dispositivos móviles. En este emulador se iba comprobando que las pantallas diseñadas tenían el aspecto que se esperaba y que el tamaño de los espacios y de los widgets era el deseado ya que se utilizaba como medida una proporción del ancho y alto de la pantalla del dispositivo que se obtiene en una de las primeras líneas de código que se ejecutan al correr la aplicación.

El emulador también ha permitido comprobar el funcionamiento general de la aplicación en cuanto a la navegación entre pantallas y a al comportamiento esperado al pulsar los botones y realizar diferentes funciones en la aplicación.

En la Figura 71 se observa cómo se ve este emulador de dispositivos y como permite observar el aspecto de las pantallas en un dispositivo móvil. Además, cabe destacar que al usar este emulador se corrigieron problemas en la pantalla relacionados con el desbordamiento de pixeles. También, se utilizó la función de “Hot Reload” que ofrece Flutter con la cual a la hora de programar se pueden ver cambios realizados en la vista de manera inmediata sin tener que volver a compilar el código. Una funcionalidad que ahorra tiempo a la hora de programar.

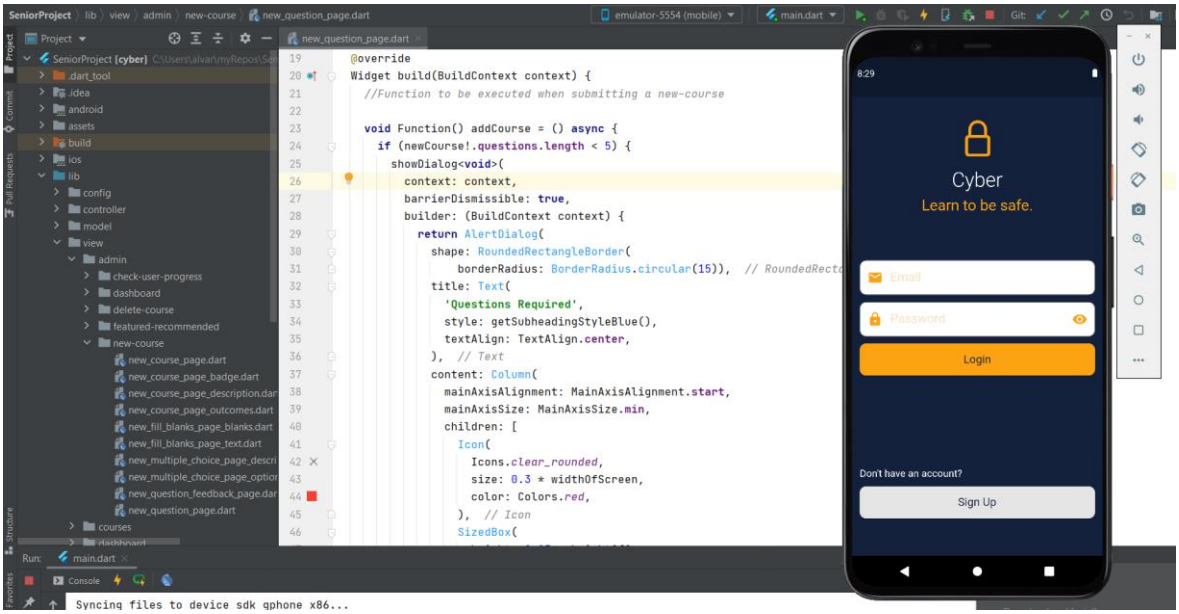


Figura 71 Emulador de dispositivos junto al IDE

6.2 PERSISTENCIA

Uno de los aspectos clave del proyecto es los servicios utilizados de Firebase aportados por Google. De estos servicios se ha utilizado el servicio de autenticación llamado Firebase Authentication y la base de datos no relacional Cloud Firestore.

A medida que las pantallas se iban diseñando y se iba incorporando lógica relacionada con la capa de persistencia y con la autenticación de usuarios se iba comprobando si estas funcionalidades estaban operando correctamente mediante la ejecución de funciones y consulta de las bases de datos.

A continuación, se enumeran las distintas operaciones que se realizan en las bases de datos y cuyo funcionamiento ha sido comprobado mediante la consulta de estas:

- Creación de un usuario nuevo con su correo electrónico y contraseña en la base de datos de autenticación.
- Creación de un usuario nuevo con toda su información asociada en forma de un documento en la colección llamada “userCollection” de Cloud Firestore.

- Añadir, eliminar y modificar los atributos relativos a un usuario en la base de datos.
- Eliminar la información relativa a un usuario de la base de datos de autenticación.
- Eliminar el documento relativo a un usuario de la colección "userCollection" de la base de datos de Cloud Firestore.
- Modificación del atributo "courseID" en el documento "featuredCourse" de la colección "featuredCollection" donde se almacena el identificador del curso destacado.
- Modificación del atributo "courseID" en el documento "recommendedCourse" de la colección "recommendedCollection" donde se almacena el identificador del curso recomendado
- Creación de un curso nuevo mediante la creación de un documento en la colección "courseCollection" donde se especifican todos sus atributos.
- Creación de una subcolección asociada al documento de un curso para poder guardar las preguntas asociadas al curso. Cada una de ellas se guarda como un documento con los atributos que definen la pregunta.
- Eliminación de un curso mediante la eliminación de su documento asociado y de la subcolección donde se almacenan sus preguntas en la colección "courseCollection".

A la hora de realizar cualquiera de estas operaciones se ha podido comprobar su funcionamiento gracias a los cambios reflejados en la capa de vista, es decir, las pantallas de la aplicación y gracias a la consola de Firebase donde se puede consultar el estado de las bases de datos.

6.3 DISPOSITIVOS REALES

También se ha podido comprobar el funcionamiento de la aplicación en dispositivos reales. Este proceso ha seguido dos pasos diferenciados.

6.3.1 FASE DE DESARROLLO

En los estados avanzados del desarrollo de la aplicación cuando casi toda la funcionalidad se había implementado y la mayoría de las pantallas estaban funcionando se utilizó un dispositivo Android para comprobar que la aplicación funcionaba en un dispositivo real a parte de en el simulador. Para ello se utilizó un Xiaomi Redmi Note 8. Es en este dispositivo donde se llevaron a cabo las últimas etapas del desarrollo.

Es también en este dispositivo donde se realizó una etapa de pruebas. Esta etapa consistió en usar de manera exhaustiva la aplicación para comprobar que todos los casos de usos funcionaban correctamente. A continuación, se enumeran algunas de las funcionalidades que cuyo funcionamiento se comprobó y, en ocasiones, se corrigió:

- Se puede completar un curso correctamente.
- Al acabar un curso se añaden los puntos de experiencia al usuario y dicha información se ve reflejada en el perfil.
- Al acabar un curso con más de un 50% de las preguntas bien contestadas se logra una insignia.
- En la página de progreso de categoría, la información sobre qué cursos ha completado el usuario se corresponde con la realidad.
- Al ganar los puntos suficientes para subir de nivel, se muestra el cuadro de diálogo correspondiente mostrando el avatar que se ha ganado y dicha información se ve reflejada en el perfil.
- El usuario es capaz de guardar y liberar cursos.
- La sección de cursos del perfil muestra tanto los cursos guardados como los completados.
- Todas las insignias que el usuario ha ganado aparecen en su información y la información relativa a las mismas como el icono y la fecha en la que se ha ganado es correcta.
- El usuario puede cerrar sesión y eliminar su cuenta exitosamente.
- El usuario puede modificar su nombre de usuario y su contraseña.

- El usuario puede modificar su avatar y esa información se ve reflejada.
- Los widgets donde se muestra la información relativa al usuario como el número de cursos completados o guardados y los puntos de experiencia ganados en una categoría son correctos.
- El usuario puede dejar un curso a medias y este aparece en su panel o "Dashboard".
- El usuario puede salir de un curso sin guardar su progreso.

6.3.2 FASE DE ENTREGA

Una vez la versión alfa de la aplicación estaba desarrollada y se había incluido toda la funcionalidad deseada, se procedió subir dicha versión de prueba a TestFlight y a una prueba interna de Play Console. Esto consiste en dos plataformas ofrecidas por App Store y Google Play, respectivamente, para subir versiones de prueba de una aplicación.

Esta versión de prueba se ha dado a probar a varios usuarios de los cuales se ha recibido realimentación que ha servido para terminar de descubrir errores en la aplicación y corregirlos, así como para implementar cambios en el diseño para mejor la experiencia de usuario.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

En este apartado se comentan cuáles son los objetivos del proyecto que se han completado y de cuáles son los posibles próximos pasos para completar.

En cuanto a los objetivos del proyecto definidos en se han completado la mayoría de ellos:

- Se ha conseguido desplegar la plataforma en forma de una aplicación desarrollada en Flutter válida para dispositivos Android e iOS.
- La aplicación ofrece contenido dinámico ya que un usuario con derechos de administrador puede añadir y eliminar cursos además de cambiar el curso recomendado y el curso destacado. El administrador también tiene la posibilidad de ver el progreso de otros usuarios.
- Se ha creado una base de datos funcional en Cloud Firestore.
- Se ha implementado un método de autenticación mediante Firebase Authentication y un método de autorización mediante el atributo “isAdmin”.
- Se han insertado técnicas de gamificación en la aplicación ya que se recompensa a los usuarios mediante avatares, puntos de experiencia e insignias.
- Se ofrece el contenido en forma de cursos con pocas preguntas (5-10 preguntas) y son rápidos de completar.

No obstante, el objetivo secundario en el que se indica que se quería desplegar una versión beta disponible en Google Play y en App Store no se ha logrado. Por el contrario, a lo que se ha llegado es a una versión al desplegada en TestFlight y en una prueba interna de Google Console.

En cuanto al trabajo futuro relativo al proyecto, existen diversas funcionalidades que se podrían implementar en la aplicación y que aportarían valor al usuario. Algunos de los siguientes pasos se especifican a continuación:

- Incluir la lógica necesaria para que la sección de administración sea relativa y única para una institución o compañía. Es decir, que, si la plataforma se vendiera a una institución, los cambios realizados por los administradores solo se vieran reflejados para los usuarios pertenecientes a dicha institución.
- Incluir la opción de que el administrador modifique un atributo o pregunta específicos de un curso.
- Establecer un sistema de autenticación más sofisticado que no dependa del valor de un atributo de tipo booleano.
- Incorporar más contenido en forma de cursos.
- Establecer más tipos de preguntas.
- Incluir realimentación en formato de vídeo y audio.
- Ofrecer un contenido más personalizado al usuario. Para ello se podrían estudiar sus tendencias y modificar la aplicación en función de ellas.
- Incluir repetición de las preguntas mal contestadas para mejorar la curva de aprendizaje.

Capítulo 8. BIBLIOGRAFÍA

[1] Kaspersky, "What is a digital footprint ? And How to protect from Hackers", [En línea]. Disponible: https://www.kaspersky.com/resource-center/definitions/what-is-a-digital-footprint.

[2] Galeano, S. "El número de usuarios de internet crece un 4% y roza los 5000 millones (2022)", [En línea]. Disponible: https://marketing4ecommerce.net/usuarios-de-internet-mundo/.

[3] Accenture. "Estado de resiliencia en ciberseguridad 2021", [En línea]. Disponible: https://www.accenture.com/es-es/insights/security/invest-cyber-resilience.

[4] Marín, I. y Hierro, E. "Gamificación. El poder del juego en la gestión empresarial y la conexión con los clientes".

[5] Infinitia Industrial Consulting."Design thinking y el método del Doble Diamante para procesos de diseño e innovación", [En línea]. Disponible: https://www.infinitiaresearch.com/noticias/design-thinking-y-metodo-de-doble-diamante-para-procesos-de-diseno-e-innovacion/

[6] Cirujano, A. "Qué es Figma", [En línea]. Disponible: https://3ymedia.school/que-es-figma/

[7] Thomas, G. "What is Flutter and Why You Should Learn it in 2020", [En línea]. Disponible: https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/

[8] López, S. "Firebase: qué es, para qué sirve, funcionalidades y ventajas", [En línea]. Disponible: https://www.digital55.com/desarrollo-tecnologia/que-es-firebase-funcionalidades-ventajas-conclusiones/

[9] Aguilar, JM. "Qué es el patrón MVC en programación y por qué es útil", [En línea]. Disponible: https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx

[10] Duolingo, "Eficacia en Duolingo", [En línea]. Disponible: https://es.duolingo.com/efficacy

[11] Edwards, L. "What is Quizlet and How Can I Teach With It?", [En línea]. Disponible: https://www.techlearning.com/how-to/what-is-quizlet-and-how-can-i-teach-with-it

[12] Waluyo, B & Bucol J.L., "The Impact Of Gamified Vocabulary Learning Using Quizlet on Low-Proficiency Students" [En línea]. Disponible: http://callej.org/journal/22-1/Waluyo-Bucol2021.pdf

[13] Cornerstone. "5 Companies Using Gamification to Boost Business Results", [En línea]. Disponible: https://www.cornerstoneondemand.com/resources/article/5-companies-using-gamification-boost-business-results/

[14] Playmotiv. "Gamificación: cifras y casos de éxito", [En línea]. Disponible: https://playmotiv.com/nada-mejor-que-una-dosis-de-datos-para-comprender-mejor-la-repercusion-de-la-gamificacion-en-los-negocios/

[15] González, R. "Los ciberataques en España crecen un 125%. La pyme la gran perjudicada", [En línea]. Disponible: https://cincodias.elpais.com/cincodias/2021/03/25/pyme/1616706362_846686.html

[16] Computer World. "El 55% de las empresas no se defiende eficazmente de los ciberataques", [En línea]. Disponible: https://cso.computerworld.es/tendencias/el-55-de-las-empresas-no-se-defiende-eficazmente-de-los-ciberataques

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

En este anexo se evalúa la alineación de este proyecto con los Objetivos de Desarrollo Sostenible. En 2015 la ONU aprobó la Agenda 2030 sobre el Desarrollo sostenible y en ella se definieron 17 objetivos de desarrollo sostenible con el objetivo de mejorar la vida de todos.

De los objetivos definidos en la Agenda 2030, este proyecto está más ligado con el cumplimiento del objetivo 4: Educación de Calidad. Según la ONU este objetivo consiste en garantizar una educación inclusiva, equitativa y de calidad y promover oportunidades de aprendizaje durante toda la vida para todos.

Este proyecto da forma a una plataforma de enseñanza sobre ciberseguridad basada en la gamificación mediante una aplicación. Por tanto, con este proyecto se ofrece la posibilidad de que cualquiera que disponga de un teléfono móvil con conexión a internet, independientemente de si este es Android o iOS, pueda descargarse la aplicación y acceder a los conocimientos que esta ofrece. No se ha llevado a cabo ningún tipo de discriminación a la hora de ofrecer estos servicios ya que no existen distinciones en quién puede o no utilizar la aplicación. Además, en caso de cumplir con los siguientes pasos descritos, la aplicación estaría disponible en App Store y Google Play de manera gratuita sin cobrar por su uso, ofreciendo de esta manera información de calidad gratis.

El contenido de los cursos que se puede encontrar en la aplicación ha sido entregado por expertos del departamento de ciberseguridad de la University Of Miami. Esto quiere decir que el contenido cuenta con un estándar de calidad.

Otro de los objetivos relacionados con los que el proyecto se relaciona es el objetivo 10: reducción de las desigualdades. De acuerdo con la descripción de la ONU, este objetivo intenta reducir las desigualdades entre distintos países y en un propio país.

Con la aplicación desarrollada en el proyecto se trata de que los usuarios adquieran unos conocimientos básicos de ciberseguridad y que, mediante las técnicas de gamificación, terminen por desarrollar hábitos higiénicos, basados en esos conocimientos adquiridos, para estar más seguros en el ciberespacio. Estos conocimientos no siempre son compartidos por la gran mayoría de la gente y con este proyecto se podría reducir el nivel de analfabetismo relacionado con la ciberseguridad combatiendo de esta manera la desigualdad a nivel educativo.

Con estos conocimientos aportados en la aplicación los usuarios podrían navegar en internet sintiéndose más seguros y se crearía un mayor nivel de igualdad ya que más gente sabría como tener contraseñas seguras, qué implica aceptar las cookies o cómo evitar que su información personal sea robada por un atacante.

El último de los objetivos con los que el proyecto se relaciona es el objetivo 8: trabajo decente y crecimiento económico. Según la descripción de la ONU este objetivo consiste en promover el crecimiento económico inclusivo y sostenible, el empleo y el trabajo decente para todos.

Al ofrecer un mayor nivel de formación en el ámbito de la ciberseguridad con el contenido ofrecido en la aplicación, las personas que decidan utilizarla podrán demostrar esos conocimientos en el ámbito laboral generando de esta manera un mayor interés por parte de las empresas para ofrecerles un trabajo. Además, sabiendo que un gran porcentaje de los ataques que sufren las empresas se debe a los errores humanos por desconocimiento de su vulnerabilidad ante el robo de su información personal e información de la empresa, con la aplicación se podría reducir la tasa de éxito de estos ataques generando de esta manera un crecimiento económico. Este crecimiento seria consecuencia de recortar en las pérdidas generadas por estos ataques.

ANEXO II: ENCUESTA SOBRE EL USO DE APLICACIONES MÓVIL

En este anexo se recogen las preguntas que se han llevado a cabo en la encuesta de Google Forms así como las respuestas recogidas sobre el uso de las aplicaciones móvil.

Las preguntas que se han realizado con sus respectivas opciones son las siguientes:

1. Indica tu grupo de edad.
    a. Menor 18
    b. 18 – 22
    c. 23 – 29
    d. 30 – 39
    e. 40+
2. Indica tu género.
    a. Masculino
    b. Femenino
    c. Prefiero no responder
3. ¿Cuál es el título académico de mayor nivel que has obtenido o que estas en proceso de obtener?
    a. Bachillerato
    b. Algún curso de la universidad
    c. Ciclo Formativo de Grado Medio
    d. Grado
    e. Máster
    f. Doctorado
4. ¿Para qué actividades utilizas tu móvil? Indica las tres más importantes.
    a. Jugar

   b. Redes Sociales
   c. Ver vídeos
   d. Escuchar música
   e. Aprender
   f. Leer
   g. Navegar
   h. Fotografía
   i. Otro

5. ¿Cada cuánto utilizar aplicaciones de enseñanza en tu móvil?
   a. Diariamente
   b. Semanalmente
   c. Mensualmente
   d. Anualmente
   e. Nunca
6. ¿Del 1 al 4 (siendo 1 apenas afecta y 4 imprescindible) cómo influyen las siguientes características en tu interés a la hora de utilizar aplicaciones de enseñanza?
   a. Música de fondo
   b. Diseño gráfico
   c. Tema
   d. Tipo de pregunta
   e. Animaciones
   f. Simplicidad
7. ¿Del 1 al 4 (siendo 1 apenas afecta y 4 imprescindible) cómo influyen las siguientes características a la hora de mantenerte motivado para volver a una aplicación que usas recurrentemente?
   a. Gráficos
   b. Historia
   c. Multijugador
   d. Modo offline
   e. Mundo privado

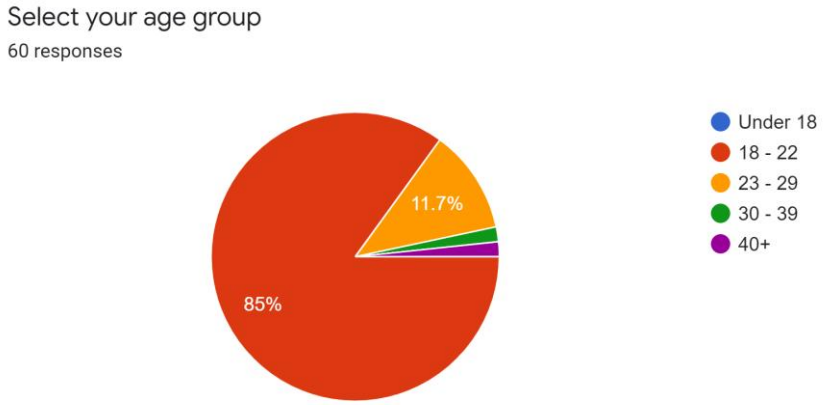
f. Mundo público
g. Música
h. Construcción de un personaje
i. Habilidades

8. Para juegos de móvil elije las tres características que consideres más importantes
    a. Barra de Progreso
    b. Respuestas Competadas / Respuestas Restantes
    c. Tiempo restante
    d. Hitos / Logros
    e. Vidas / Intentos
9. ¿Cuándo estas completando una prueba cual es tu método favorito para medir tu progreso?
    a. Completar los huecos
    b. Verdadero o Falso
    c. Relacionar
    d. Escribir la respuesta
    e. Opción múltiple (1 respuesta correcta)
    f. Respuesta múltiple (1 o varias respuestas correctas)
10. ¿Cómo te gusta ser recompensado cuando utilizas aplicaciones de aprendicaje? Ordena por orden de preferencia.
    a. Puntos
    b. Insignias
    c. XP
    d. Ranking
    e. Vidas / Intentos
    f. Monedas
11. Del 1-4 (siendo 1 irrelevante y 4 imprescindible) indica cómo valoras recibir realimentación POSITIVA después de responder a una pregunta.
12. Del 1-4 (siendo 1 irrelevante y 4 imprescindible) indica cómo valoras recibir realimentación NEGATIVA después de responder a una pregunta.

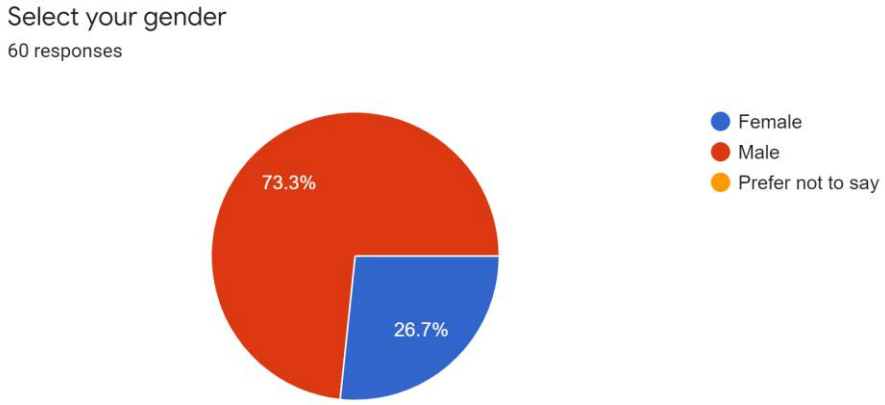
13. ¿Cómo prefieres recibir esa realimentación?
    a. Explicación corta
    b. Explicación opcional
    c. Explicación corta con opción de ver más
    d. Explicación formato video
    e. No recibir realimentación
14. ¿Cómo prefieres interactuar con otros en una aplicación? Ordena por orden de preferencia.
    a. Compartir mi progreso
    b. Retar a otros usuarios
    c. Preguntas y respuestas
    d. Mensajes directos
15. ¿Te gusta que una aplicación te recuerde que llevas mucho sin utilizarla? Si es así, ¿cómo te gusta que te lo recuerde?
    a. Mail
    b. Mensaje de Texto
    c. Notificación
    d. No, no me gusta que me lo recuerden
    e. Otro:

A continuación, se adjuntan los resultados de la encuesta en formato de gráficas:

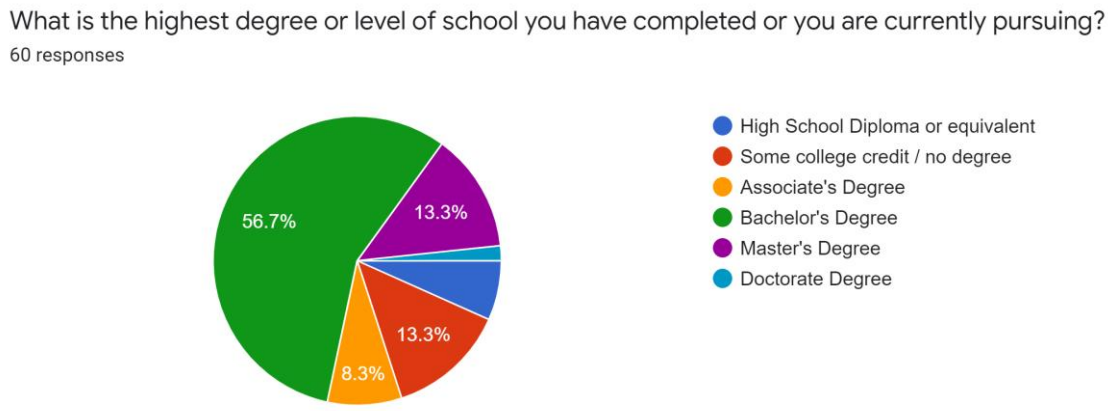
Pregunta 1.



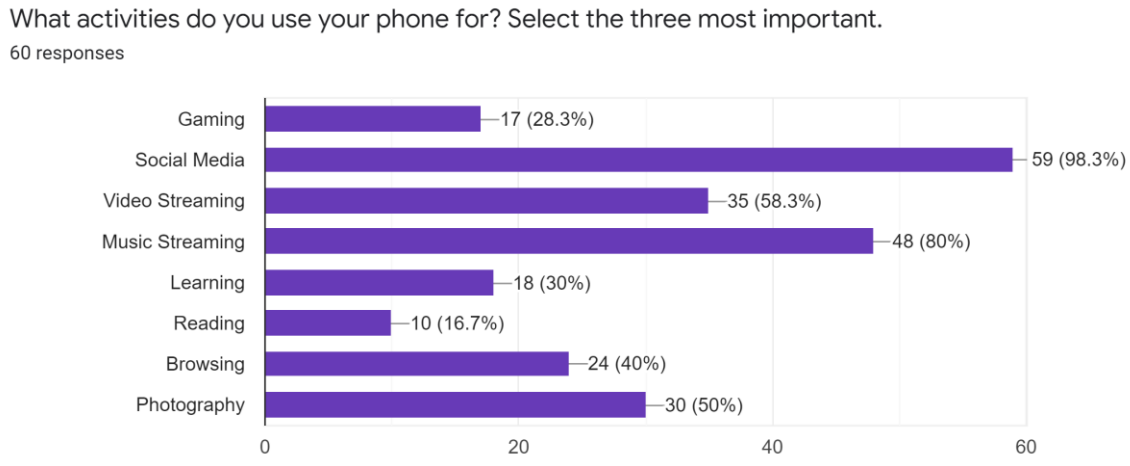
Pregunta 2.



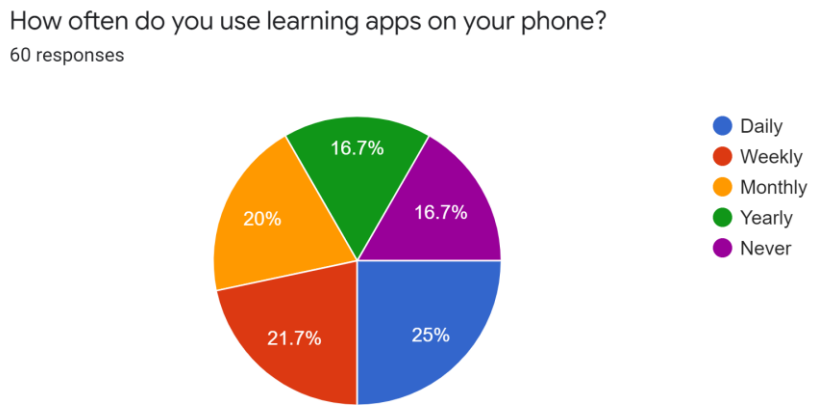
Pregunta 3.



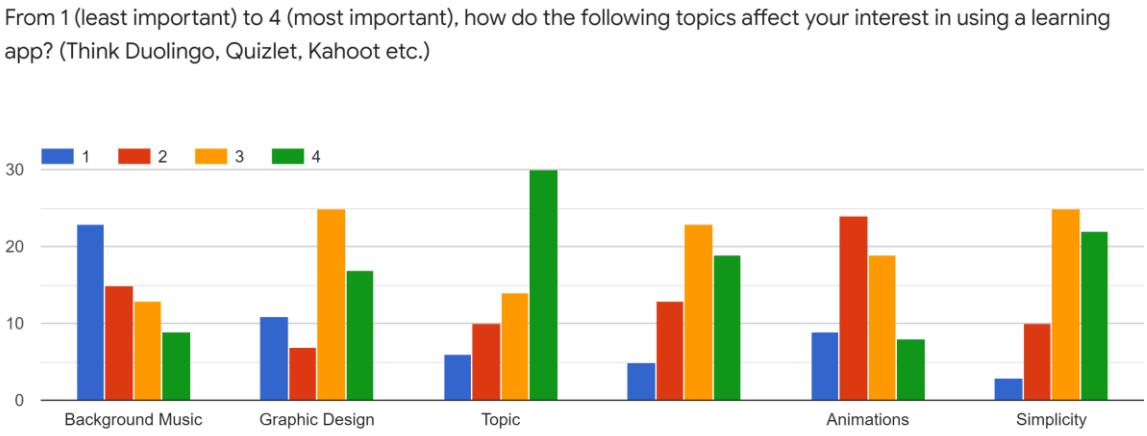
Pregunta 4.



Pregunta 5.

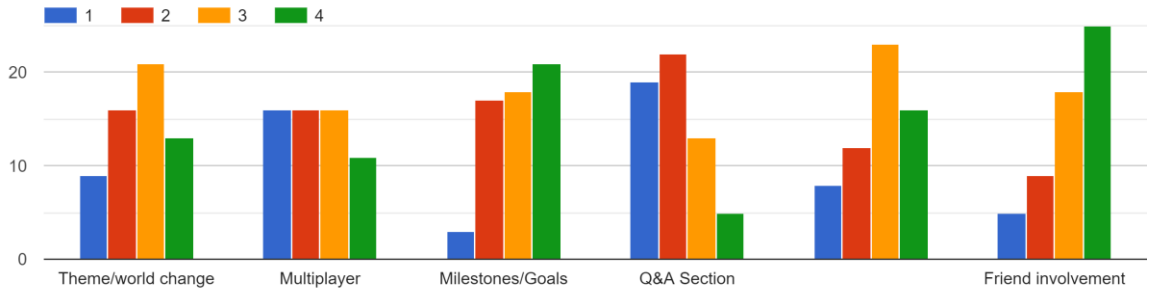


Pregunta 6.



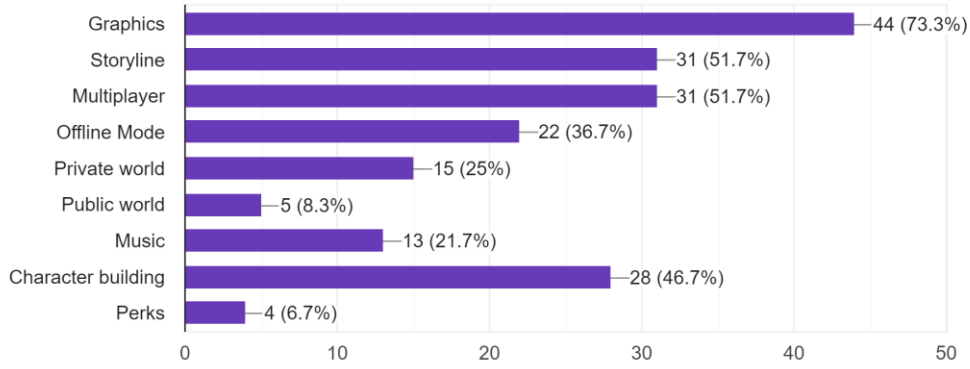
Pregunta 7.

From 1 (least important) to 4 (most important), what makes you motivated to keep returning to an app that you regularly use?

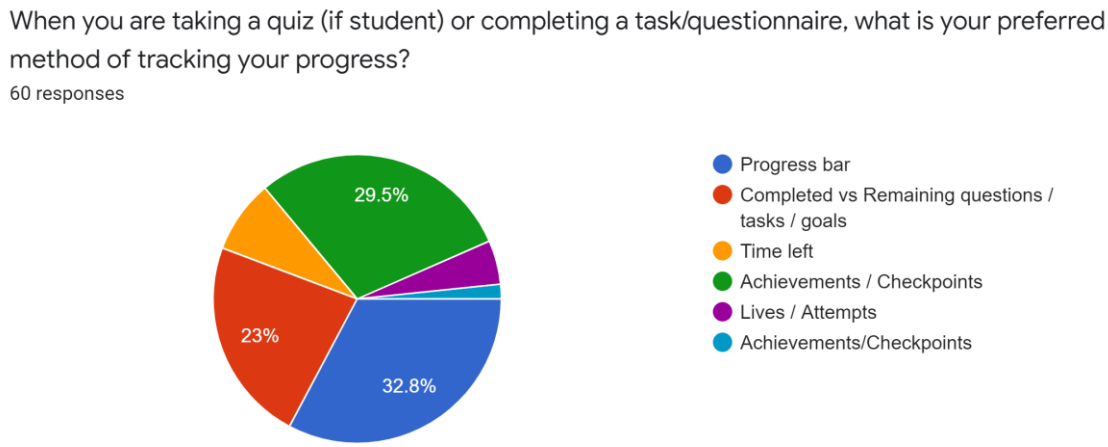


Pregunta 8.

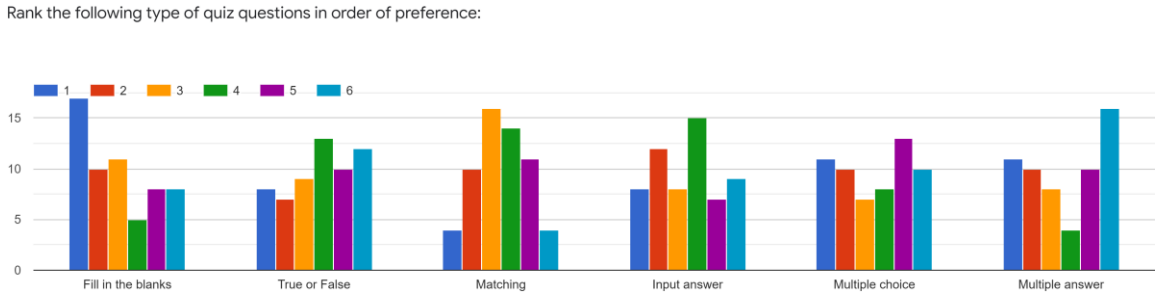
For mobile games, choose the three most important to you from the following:
60 responses



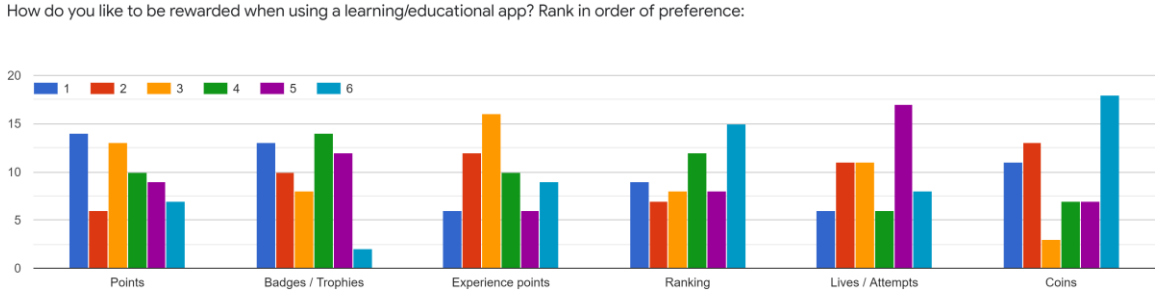
Pregunta 9.



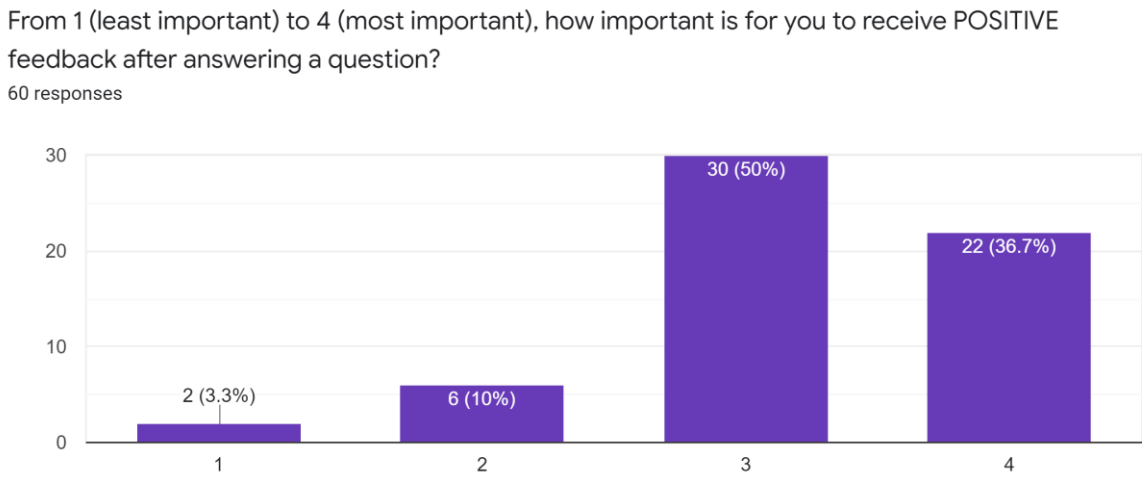
Pregunta 10.



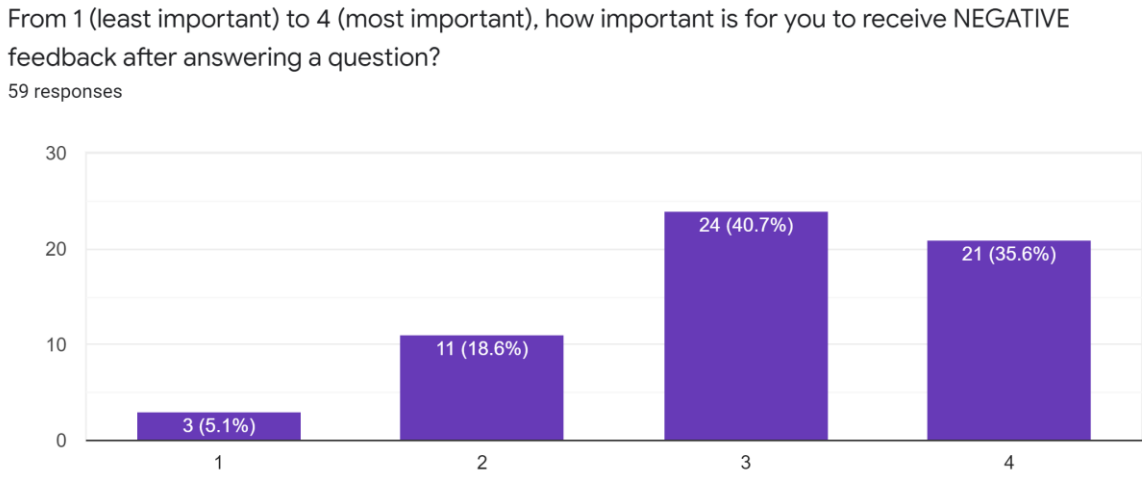
Pregunta 11.



Pregunta 12.

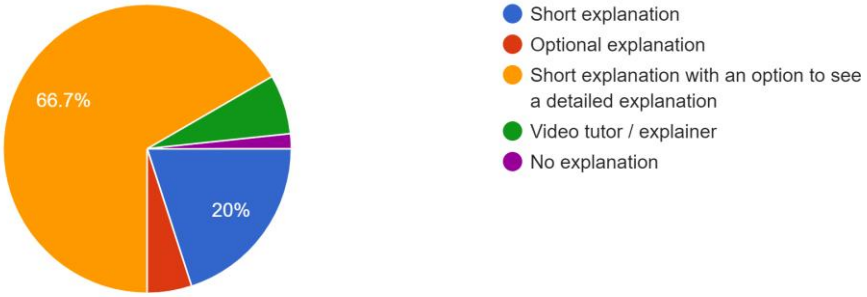


Pregunta 13.



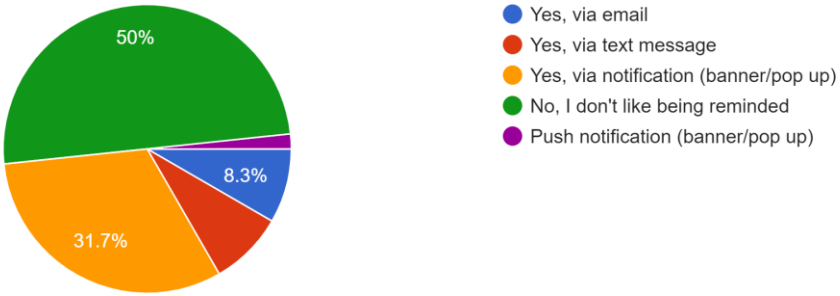
Pregunta 14.

How do you prefer to receive such feedback?
60 responses



Pregunta 15.

When you have not used an app in a while, do you like being reminded to return? If so, how do you like being reminded?
60 responses



ANEXO III: CÓDIGO DEL CONTROLADOR "USERCONTROLLER"

En este anexo se adjunta el código de la clase "UserController". En esta clase se definen todos los métodos para obtener y persistir información en la base de datos de Autenticación y en la colección de usuarios definida en Cloud Firestore.

```
class UserController {
  /**
   * Method to login into Auth DB provided an email and a Password
   * If the credentials are OK it returns the user credentials and changes
   * the state of the user to loggedIn. In case the credentials are KO returns
   * a String with the reason of the error
   */
  static Future loginWithEmailAndPassword(
      {required String email, required String password}) async {
    try {
      UserCredential userCredential =
          await FirebaseAuth.instance.signInWithEmailAndPassword(
        email: email,
        password: password,
      );

      return userCredential;
    } on FirebaseAuthException catch (e) {
      if (e.code == 'user-not-found') {
        return 'No user found with that email.';
      } else if (e.code == 'wrong-password') {
        return 'Wrong password provided.';
      } else if (e.code == 'invalid-email') {
        return 'Wrong email format.';
      } else {
        return e.code.toString();
      }
    }
  }

  /**
   * Method to add a User to Auth DB. If it suceeds then we call
   * a method to also create a user in the Firestore collection
   */
  static Future addUserToAuthAndFirestore(
      {required UserCustom u, required String password}) async {
    //Before creating the user in any DB we check if the userName is in use
```

```
    //Because if you dont check that you will end up creating a user in
    //Auth with no corresponding user in Firestore

    bool usernameInUse = await userNameExists(username: u.username);
    if (usernameInUse) {
      return 'Username already in use';
    }

    //First we upload the user in Auth DB
    try {
      UserCredential userCredential =
          await FirebaseAuth.instance.createUserWithEmailAndPassword(
        email: u.email,
        password: password,
      );

      //If we succeed in creating the user in the Auth DB, then we create the
entry for the other info in the separate collection
      // That separate collection is were we will have our custom info of the
user

      return addUserToFireStore(userCustom: u, userId: userCredential.user!.uid)
          .then((value) {
        return value;
      }).catchError((error) {
        print('Adding user to Firestore error');
        throw Exception('Adding user to Firestore error');
      });
    } on FirebaseAuthException catch (e) {
      if (e.code == 'weak-password') {
        return 'The password provided is too weak.';
      } else if (e.code == 'email-already-in-use') {
        return 'An account already exists for that email.';
      } else if (e.code == 'invalid-email') {
        return 'Invalid email format.';
      }
    } catch (e) {
      print(e);
    }
  }

  /**
   * Method to add a user to the userCollection in Firestore.
   * The document created will be named with the userId from the AuthDB
   */
  static Future<dynamic> addUserToFireStore(
      {required UserCustom userCustom, required String userId}) async {
    // Call the user's CollectionReference
    CollectionReference users =
        FirebaseFirestore.instance.collection(userCollectionName);

    //Access specific entry and set info
    return users.doc(userId).set(userCustom.toJson()).then((value) {
```

```
    print("User created");
    return true;
  }).catchError((error) {
    print("ERROR when persisting user in collection");
    return false;
  });
}

/**
 * Method to check if a username provided as param exists in the collection
 */
static Future userNameExists({required String username}) {
  CollectionReference usersRef =
      FirebaseFirestore.instance.collection(userCollectionName);

  return usersRef.get().then((querySnapshot) {
    for (int i = 0; i < querySnapshot.size; i++) {
      //For each document in the questions collection I create a UserCustom
      Map<String, dynamic> json =
          querySnapshot.docs[i].data() as Map<String, dynamic>;

      if (json['username'].toLowerCase() == username.toLowerCase()) {
        return true;
      }
    }
    return false;
  }).catchError((error) {
    print('Error checking if username exists');
    throw Exception('Error checking if username exists.');
  });
}

static Future getActiveUser() {
  CollectionReference users =
      FirebaseFirestore.instance.collection(userCollectionName);

  //I can get the uid of the active user with the following method

  String uidOfActiveUser = FirebaseAuth.instance.currentUser!.uid;
  return users.doc(uidOfActiveUser).get().then((snapshot) {
    Map<String, dynamic> json = snapshot.data() as Map<String, dynamic>;

    return UserCustom.fromJson(json);
  }).catchError((error) {
    print('Failed to get User: ${error.toString()}');
    throw Exception('Error getting active user.');
  });
}

static Future getUsername() async {
  try {
    UserCustom u = await getActiveUser();
    return u.username;
```

```
    } catch (error) {
      throw Exception('');
    }
  }

  static Future updateActiveUser() {
    //We get the id of the active user
    String uidOfActiveUser = FirebaseAuth.instance.currentUser!.uid;

    //Now we update it in the Firestore DB

    return addUserToFireStore(userCustom: activeUser!, userId: uidOfActiveUser);
  }

  /**
   * Method to get a List with all the users in the Database
   */
  static Future getAllUsers() {
    CollectionReference usersRef =
        FirebaseFirestore.instance.collection(userCollectionName);

    List<UserCustom> users = [];

    return usersRef.get().then((querySnapshot) {
      for (int i = 0; i < querySnapshot.size; i++) {
        //For each document in the questions collection I create a UserCustom
        Map<String, dynamic> json =
            querySnapshot.docs[i].data() as Map<String, dynamic>;
        users.add(UserCustom.fromJson(json));
      }
      return users;
    }).catchError((error) {
      print('Failed to get users');
      throw Exception('Error getting list of users.');
    });
  }

  /**
   * Method to update the password of the user in
   * FireBase Auth. It tries not to make the user identify
   * again
   */
  static Future updatePassword(
      {required String currentPassword, required String newPassword}) async {
    final user = await FirebaseAuth.instance.currentUser;
    final cred = EmailAuthProvider.credential(
        email: user!.email!, password: currentPassword);

    return user.reauthenticateWithCredential(cred).then((value) {
      return user.updatePassword(newPassword).then((_) {
        print('Password changed for active user');
        return true;
      }).catchError((error) {
```

```
        //Since I have just reauthenticated the user the
        //only possible error ir weak password
        print('Error when updating password');
        return false;
      });
    }).catchError((err) {
      print('Wrong password provided for update');
      throw Exception('Wrong password provided for update');
    });
  }

  /**
   * Method to sign out the active user
   */
  static Future signOutUser() async {
    try {
      await FirebaseAuth.instance.signOut();
      return 'Signed out';
    } on FirebaseAuthException catch (e) {
      print('Error in the sign out');
      return 'Error when signing out.';
    }
  }

/**
 * Method to delete the account of the active user
 */
  static Future deleteActiveUser() async {
    try {
      await deleteUserFromFirestore(
          userID: FirebaseAuth.instance.currentUser!.uid);
      await FirebaseAuth.instance.currentUser!.delete();
      return 'Success';
    } on FirebaseAuthException catch (e) {
      if (e.code == 'requires-recent-login') {
        print(
            'The user must reauthenticate before this operation can be
executed.');
        return 'User must reauthenticate before';
      }
    } on Exception catch (e) {
      print('Error deleting account');
      return 'Error deleting the account';
    }
  }

  /**
   * Method to delete user from the user collection
   */
  static Future deleteUserFromFirestore({required String userID}) {
    CollectionReference users =
        FirebaseFirestore.instance.collection(userCollectionName);
    return users.doc(userID).delete();
```

```
}

/**
 * Method to update any field in the user
 */
static Future updateSimpleUserField(
    {required String nameOfField, required dynamic field}) {
  CollectionReference users =
      FirebaseFirestore.instance.collection(userCollectionName);

  //I get the id of the active user
  String uidOfActiveUser = FirebaseAuth.instance.currentUser!.uid;

  //Access specific entry and set info
  return users.doc(uidOfActiveUser).update({
    nameOfField: field,
  }).then((value) {
    print("Updated field ${nameOfField}");
    return true;
  }).catchError((error) {
    print("Error updating field ${nameOfField}");
    return false;
  });
}

/**
 * Method to update the username. Since we have to check if it already
 * exists, I made a separate method
 */
static Future updateUsername({required String newUsername}) async {
  CollectionReference users =
      FirebaseFirestore.instance.collection(userCollectionName);

  bool exists = await userNameExists(username: newUsername);

  if (exists) {
    return 'Username already exists';
  }

  //I get the id of the active user
  String uidOfActiveUser = FirebaseAuth.instance.currentUser!.uid;

  //Access specific entry and set info
  return users.doc(uidOfActiveUser).update({
    'username': newUsername,
  }).then((value) {
    print("Updated username");
    return true;
  }).catchError((error) {
    print("Error updating username");
    return false;
  });
}
```

```
/**
 * Method to upload a list of complex fields in the json of the
 * user in the Firebase DB. For More info see method
 * updateComplexUserField
 */
static Future updateComplexListUserField(
    {required String nameOfField, required dynamic field}) {
  CollectionReference users =
      FirebaseFirestore.instance.collection(userCollectionName);

  //I get the id of the active user
  String uidOfActiveUser = FirebaseAuth.instance.currentUser!.uid;

  //Access specific entry and set info
  return users.doc(uidOfActiveUser).update({
    nameOfField: field.map((e) => e.toJson()).toList(),
  }).then((value) {
    print("Updated field ${nameOfField}");
    return true;
  }).catchError((error) {
    print("Error updating field ${nameOfField}");
    return false;
  });
}

static Future updateComplexUserField(
    {required String nameOfField, required dynamic field}) {
  CollectionReference users =
      FirebaseFirestore.instance.collection(userCollectionName);

  //I get the id of the active user
  String uidOfActiveUser = FirebaseAuth.instance.currentUser!.uid;

  //Access specific entry and set info
  return users.doc(uidOfActiveUser).update({
    nameOfField: field.toJson(),
  }).then((value) {
    print("Updated field ${nameOfField}");
    return true;
  }).catchError((error) {
    print("Error updating field ${nameOfField}");
    return false;
  });
}

static Future addGroupCodeToUser({required List<String> groupCode}) async {
  CollectionReference users =
      await FirebaseFirestore.instance.collection(userCollectionName);

  String uidOfActiveUser = FirebaseAuth.instance.currentUser!.uid;

  return users
```

```
        .doc(uidOfActiveUser)
        .update({"userGroups": FieldValue.arrayUnion(groupCode)});
  }

  static Future<UserCustom> getUserByUserName({required String userName}) {
    var user = FirebaseFirestore.instance
        .collection(userCollectionName)
        .where("username", isEqualTo: userName)
        .get()
        .then((value) {
      Map<String, dynamic> json = value.docs.single.data();

      return UserCustom.fromJson(json);
    }).catchError((error) {
      print('Failed to get User: ${error.toString()}');
      throw Exception('Error getting active user');
    });
    return user;
  }
}
```

ANEXO IV: CÓDIGO DEL CONTROLADOR "ACTIVEUSERCONTROLLER"

En este anexo se adjunta el código del controlador "ActiveUserController". Esta clase guarda toda la información relativa al usuario activo que está usando la aplicación. En ella definen los métodos para cambiar la información relativa a la instancia de ese usuario activo así como para cambiar la información en la base de datos utilizando el controlador "userController".

```
class ActiveUserController extends GetxController {
  final username = 'username'.obs;
  final profilePictureActive = 'profilePictureActive'.obs;
  final level = Level(xpEarnedInLevel: 0, levelNumber: 0, totalXP: 0).obs;
  final email = 'email@gmail.com'.obs;
  final isAdmin = true.obs;
  final userGroups = <String>[].obs;
  final coursesSaved = <String>[].obs;
  final completedCourses = <CompletedCourse>[].obs;
  final collectedBadges = <Badge>[].obs;
  final collectedAvatars = <String>[].obs;
  final currentCourse = Rxn<CurrentCourse?>();

  @override
  void onInit() {
    super.onInit();

    username.value = activeUser!.username;
    profilePictureActive.value = activeUser!.profilePictureActive;
    level.value = activeUser!.level;
    email.value = activeUser!.email;
    isAdmin.value = activeUser!.isAdmin;
    userGroups.value = activeUser!.userGroups;
    coursesSaved.value = activeUser!.coursesSaved;
    currentCourse.value = activeUser!.currentCourse;
    completedCourses.value = activeUser!.completedCourses;
    collectedBadges.value = activeUser!.collectedBadges;
    collectedAvatars.value = activeUser!.collectedAvatars;

  }

  getNumBadges() {
    return this.collectedBadges.length;
```

```
}

getNumAvatars() {
  return this.collectedAvatars.length;
}

getTotalPoints() {
  return this.level.value.totalXP;
}

isCompleted({required String courseID}) {
  for (CompletedCourse cc in this.completedCourses.value) {
    if (cc.courseID == courseID) {
      return true;
    }
  }

  return false;
}

isSaved({required String courseID}) {
  return this.coursesSaved.contains(courseID);
}

bool isCurrentCourse({required String courseID}) {
  if (this.currentCourse.value != null &&
      this.currentCourse.value!.courseID == courseID) {
    return true;
  } else {
    return false;
  }
}

int getCompletedCoursesInCategory({required List<String> courseIDs}) {
  int i = 0;

  for (CompletedCourse completed in this.completedCourses) {
    if (courseIDs.contains(completed.courseID)) {
      i++;
    }
  }

  return i;
}

int getXPInCategory({required List<String> courseIDs}) {
  int i = 0;

  for (CompletedCourse completed in this.completedCourses) {
    if (courseIDs.contains(completed.courseID)) {
      i = i + completed.experiencePointsEarned;
    }
  }
```

```
  return i;
}

unsaveCourse({required String courseID}) {
  this.coursesSaved.remove(courseID);
  UserController.updateSimpleUserField(
      nameOfField: 'coursesSaved', field: this.coursesSaved);
}

saveCourse({required String courseID}) {
  this.coursesSaved.add(courseID);
  UserController.updateSimpleUserField(
      nameOfField: 'coursesSaved', field: this.coursesSaved);
}

Future<void> updateCurrentCourse() async {
  CurrentCourse cc =
      CurrentCourse(courseID: activeCourse!.id!, progress: userProgress);
  this.currentCourse.value = cc;
  await UserController.updateComplexUserField(
      nameOfField: 'currentCourse', field: cc);
}

Future updateProfilePictureActive({required String newPicture}) {
  this.profilePictureActive.value = newPicture;
  return UserController.updateSimpleUserField(
      nameOfField: 'profilePictureActive', field: newPicture);
}

Future signOut() {
  //We need to dispose the active user controller since
  //there is no longer an active user

  this.dispose();
  return UserController.signOutUser();
}

Future delete() {
  //No active user, so we delete this controller
  this.dispose();
  return UserController.deleteActiveUser();
}

Future saveCompletedCourse() async {
  bool levelUp = false;
  bool earnedBadge = false;

  //This is to check that the user has completed the active new-course
  //If the method is called by error does nothing
  if (activeCourse == null) {
    return false;
  } else if (activeCourse!.numberOfQuestions != userProgress.length) {
    return false;
```

```
    }

    //Calculate number of questions right
    int questionsRight = 0;
    for (bool answer in userProgress) {
      if (answer) {
        questionsRight++;
      }
    }

    //Calculate xpEarned and percentage
    int xpEarned = (((activeCourse!.experiencePoints.toDouble()) /
                activeCourse!.numberOfQuestions) *
            questionsRight)
        .round();

    if (activeCourse!.isFfeatured!) {
      xpEarned = xpEarned * 2;
    }

    int percentageCompleted =
        ((questionsRight.toDouble() / activeCourse!.numberOfQuestions) * 100)
            .round();

    //Before creating the completed new-course, we check if the user has already
done
    //the new-course and has a better rate of correct answers

    int xpEarnedLastTime = 0;
    int xpBalance = 0;
    bool earnedBadgeLastTime = false;

    int positionOfCourseToDelete = 0;
    bool deleteCourse = false;
    for (CompletedCourse cCourse in this.completedCourses.value) {
      if (cCourse.courseID == activeCourse!.id) {
        xpEarnedLastTime = cCourse.experiencePointsEarned;

        if (xpEarnedLastTime >= xpEarned) {
          //Exit the function because nothing is going to change
          return SaveCompletedCourseArgs(
              levelUp: false, earnedBadge: false, balanceXP: 0);
        } else {
          //In this case the user has already done the new-course but worse

          //We save the xpPoints he earned and check if he earned the badge
          xpEarnedLastTime = cCourse.experiencePointsEarned;
          deleteCourse = true;
          if (cCourse.percentageCompleted >= 50) {
            earnedBadgeLastTime = true;
          }
        }
      }
```

```
      if (!deleteCourse) {
        positionOfCourseToDelete++;
      }
    }

    if (deleteCourse) {
      this.completedCourses.removeAt(positionOfCourseToDelete);
    }

    xpBalance = xpEarned - xpEarnedLastTime;

    //We add the new-course to the list
    CompletedCourse completedCourse = CompletedCourse(
        answers: userProgress,
        numQuestionsRight: questionsRight,
        percentageCompleted: percentageCompleted,
        experiencePointsEarned: xpEarned,
        dateCompleted: DateTime.now(),
        courseID: activeCourse!.id!);
    this.completedCourses.add(completedCourse);

    //In case the user didn't earn the badge and now the percentage is over 50%
we add the badge
    if (completedCourse.percentageCompleted >= percentageToGetBadge &&
        !earnedBadgeLastTime) {
      await this.addNewBadge();
      earnedBadge = true;
    }

    //If the user levels up we need to add a new profile pic
    if (this.level.value.updateLevel(xpToAdd: xpBalance)) {
      this.level.refresh();
      await this.addNewAvatar();
      levelUp = true;
    }
    ;

    await this.updateLevel();

    //If the user had as a current new-course this one, we need to set that
    //attribute to null
    if (this.currentCourse.value != null &&
        this.currentCourse.value!.courseID == completedCourse.courseID) {
      await this.removeCurrentCourse();
    }

    //After all of this we need to update the user in the DB
    try {
      await UserController.updateComplexListUserField(
          nameOfField: 'completedCourses', field: this.completedCourses.value);
      return SaveCompletedCourseArgs(
          levelUp: levelUp, earnedBadge: earnedBadge, balanceXP: xpBalance);
    } catch (error) {
```

```
      throw Exception(error.toString());
    }
  }

  Future<void> updateLevel() async {
    await UserController.updateComplexUserField(
        nameOfField: 'level', field: this.level.value);
  }

  /**
   * Function to add a new avatar to the user
   */
  Future<void> addNewAvatar() async {
    this.collectedAvatars.add(getRandomString(5));
    await UserController.updateSimpleUserField(
        nameOfField: 'collectedAvatars', field: this.collectedAvatars.value);
  }

  /**
   * Function to add a new badge to the user
   */
  Future<void> addNewBadge() async {
    Badge newBadge = Badge(
        courseID: activeCourse!.id!,
        picture: activeCourse!.badgeIcon,
        timeEarned: DateTime.now());
    this.collectedBadges.add(newBadge);
    await UserController.updateComplexListUserField(
        nameOfField: 'collectedBadges', field: this.collectedBadges.value);
  }

  /**
   * Function to set the currentCourse to null
   */
  Future<void> removeCurrentCourse() async {
    this.currentCourse.value = null;
    await UserController.updateSimpleUserField(
        nameOfField: 'currentCourse', field: null);
  }

  /**
   * Function to change the username of the user
   */
  Future changeUsername({required String newUsername}) async {

    return UserController.updateUsername(newUsername: newUsername).then((value){

      if(value is bool){
        if(value){
          this.username.value = newUsername;
          return 'Username updated';
        }
        return 'Error updating username';
```

```
      }
      return value;
    }).catchError((error) {
      print('An error occurred when updating the username');
      return 'Error occurred';
    });
  }


  Future<void> updateUserGroups({required String groupCode}) {
    this.userGroups.value.add(groupCode);
    var res = UserController.addGroupCodeToUser(groupCode: [groupCode]);
    update();
    return res;
  }

  removeUserFromGroup({required String groupCode}) {
    this.userGroups.value.remove(groupCode);
    update();
    userGroups.refresh();
  }

  @override
  void dispose() {
    super.dispose();
  }

}
```

ANEXO V: CÓDIGO DEL CONTROLADOR "COURSECONTROLLER"

Para facilitar la narrativa del informe, se ha decidido adjuntar el código de la clase "UserController" en este anexo. En esta clase se definen todas las funciones descritas para obtener información y depositar información en la capa de persistencia de la aplicación. Más concretamente en la colección de cursos de la base de datos de Cloud Firestore.

```
class CourseController {
  //Reference to the collection of courses
  CollectionReference coursesRef =
      FirebaseFirestore.instance.collection(courseCollectionName);

  //Reference to the collection of recommended
  CollectionReference recommendedCollectionRef =
      FirebaseFirestore.instance.collection(recommendedCollectionName);

  //Reference to the collection featured
  CollectionReference featuredCollectionRef =
      FirebaseFirestore.instance.collection(featuredCollectionName);

  /**
   * Function to get a Course from the database when the title is specified.
   * First the course is gotten from the DB, after that, the ID is initialized
   * Then using another method the questions are initialized
   * Before the course is returned I set the attribute featuredCourse
   */
  Future getCourseByTitle({required String title}) async {
    //Before doing anything I format the title

    //I eliminate any blank space
    title = title.trim();

    //I set the first letter as capital and the rest as normal
    title = title[0].toUpperCase() + title.substring(1).toLowerCase();

    return coursesRef
        .where('title', isEqualTo: title)
        .get()
        .then((snapshot) async {
      //We get the first new-course from the list
      Map<String, dynamic> json =
          snapshot.docs[0].data() as Map<String, dynamic>;
```

```
      //Here I build the new-course from the json however the questions are not
initialized
      Course course = Course.fromJson(json);
      course.id = snapshot.docs[0].id;

      //To add the questions to the new-course I need to call to another future

      Course courseFilled = await getQuestionsForCourse(
          course: course, courseId: snapshot.docs[0].id);

      //Then we also need to set the attribute isFeatured
      String featuredCourseID = await getFeaturedCourseID();

      courseFilled.isFeatured = false;
      if (featuredCourseID == course.id) {
        courseFilled.isFeatured = true;
      }
      return courseFilled;
    }).catchError((error) {
      print('Error when getting course by title');
      throw Exception('Error when getting course by title');
    });
  }

  /**
   * Method to get a course from the DB by specifying its ID
   * To see how this method works refer to getCourseByTitle
   */
  Future getCourseByID({required String id}) {
    return coursesRef.doc(id).get().then((snapshot) async {
      Map<String, dynamic> json = snapshot.data() as Map<String, dynamic>;
      Course course = Course.fromJson(json);

      //I set the Id
      course.id = snapshot.id;

      //I get the questions for the course

      Course courseFilled =
          await getQuestionsForCourse(course: course, courseId: id);

      //After that I check if it is the featured course

      String featuredCourseID = await getFeaturedCourseID();

      courseFilled.isFeatured = false;
      if (featuredCourseID == course.id) {
        courseFilled.isFeatured = true;
      }
      return courseFilled;
    }).catchError((error) {
      print('Error getting the course by its id');
```

```
      throw Exception('Error getting the course by its id');
    });
  }

  /**
   * This function is used by the function getCourseByID and getCourseByTitle to
be able to get
   * the subcollection of questions of the new-course and add them to the
instance
   */
  Future getQuestionsForCourse(
      {required Course course, required String courseId}) async {
    //I create a reference to the subcollection of questions
    CollectionReference questions =
        coursesRef.doc(courseId).collection(questionCollectionName);

    Question q;
    return questions.get().then((snapshot) {
      for (int i = 0; i < snapshot.size; i++) {
        //For each document in the questions collection I create a json

        Map<String, dynamic> json =
            snapshot.docs[i].data() as Map<String, dynamic>;

        //Then I check what kind of question it is so I can create an object from
it

        if (typeOfQuestionFromString[json['typeOfQuestion']] ==
            TypeOfQuestion.multipleChoice) {
          q = MultipleChoiceQuestion.fromJson(json);
        } else {
          q = FillInTheBlanksQuestion.fromJson(json);
        }

        //Finally I can add the question to the new-course
        course.questions.add(q);
      }

      //Here I order the questions according to their number
      course.questions.sort((a, b) => a.number.compareTo(b.number));

      return course;
    }).catchError((error) {
      print('Could not get the questions for the course specified');
      throw Exception('Error getting questions for specified course');
    });
  }

  /**
   * This method is used to get the recommended course from the collection
   * created in Firestore for that purpose
   */
  Future getRecommendedCourse() async {
```

```
  try {
    final String recommendedCourseID = await this.getRecommendedCourseID();
    final Course course = await this.getCourseByID(id: recommendedCourseID);

    return course;
  } catch (error) {
    throw Exception(error.toString());
  }
}

Future getRecommendedCourseID() {
  return recommendedCollectionRef
      .doc(recommendedDocName)
      .get()
      .then((snapshot) async {
    Map<String, dynamic> json = snapshot.data() as Map<String, dynamic>;

    return json['courseID'] as String;
  }).catchError((error) {
    print('Error when getting recommended course');
    throw Exception('Not recommended found');
  });
}

/**
 * Function to get all the courses in the DB in a map
 * with the format Map<courseID, title>.
 */
Future getCourseNames() {
  Map<String, String> courses = {};

  return coursesRef.get().then((snapshot) {
    snapshot.docs.forEach((doc) {
      courses[doc.id] = doc['title'];
    });
    return courses;
  }).catchError((error) {
    print('Error when looking for courses');

    throw Exception('No courses found');
  });
}

/**
 * Function that returns a Map with the following map entries
 * <courseID, title> for all the courses in the category specified.
 * If not any then the map will be empty.
 */
Future getCourseNamesFromCategory({required Category category}) {
  Map<String, String> coursesInCategory = {};

  return coursesRef
      .where('category', isEqualTo: categoryToString[category]!)
```

```
      .get()
      .then((snapshot) {
    snapshot.docs.forEach((doc) {
      coursesInCategory[doc.id] = doc['title'];
    });


    return coursesInCategory;
  }).catchError((error) {
    print('Error when looking for a category');

    throw Exception('No courses in ${categoryToString[category]!}');
  });
}

/**
 * This function is used to get a map with entries
 * <courseID, title> when provided just with the ids
 */
Future getCourseNamesByIDs({required List<String> ids}) async {
  Map<String, String> courses = {};

  try {
    for (String id in ids) {
      Course newCourse = await getCourseByID(id: id);
      courses[id] = newCourse.title;
    }
  } catch (err) {
    throw Exception('Error when getting the courses');
  }

  return courses;
}

/**
 * Method to search the FeaturedCollection looking for the ID
 * of the featuredCourse
 */
Future getFeaturedCourseID() {
  return featuredCollectionRef
      .doc(featuredDocName)
      .get()
      .then((snapshot) async {
    Map<String, dynamic> json = snapshot.data() as Map<String, dynamic>;

    return json['courseID'] as String;
  }).catchError((error) {
    throw Exception('Could not get the Featured Course\'s ID');
  });
}

/**
 * Method to get the Featured Course complete by using the method
```

```
 * getFeaturedCourseID and getCourseByID
 */
Future getFeaturedCourse() async {
  try {
    final String featuredCourseID = await this.getFeaturedCourseID();
    final Course course = await this.getCourseByID(id: featuredCourseID);

    return course;
  } catch (error) {
    throw Exception(error.toString());
  }
}

/**
 * Function to check if a course exists on the courses collection
 */
Future courseExists({required String courseID}) {
  return coursesRef.doc(courseID).get().then((docSnapshot) {
    if (docSnapshot.exists) {
      return true;
    } else {
      return false;
    }
  }).catchError((onError) {
    print('Failed to access the collection recommended');
    throw Exception('Failed to access the collection recommended');
  });
}

/**
 * Function to add the new-course that has been filled in the admin pages
 * to firebase. The new-course to be added is a global variable
 */
Future addCourseToFirebase({required Course courseToAdd}) {
  return coursesRef.add(courseToAdd.toJson()).then((value) async {
    print('Course added');
    await addQuestionsToFirebase(courseId: value.id);
    print('Course created');
  }).catchError((error) {
    print('Course NOT created');
    throw Exception(error.toString());
  });
}

/**
 * Function used by addCourseToFirebase to create a subcollection of questions
 * in the document of the new-course
 */
Future addQuestionsToFirebase({required String courseId}) {
  CollectionReference questions =
      coursesRef.doc(courseId).collection(questionCollectionName);

  try {
```

```
    for (Question q in newCourse!.questions) {
      if (q is MultipleChoiceQuestion) {
        questions.add(q.toJson());
      } else if (q is FillInTheBlanksQuestion) {
        questions.add(q.toJson());
      }
    }
  } catch (error) {
    print('Error when adding questions to course');
    throw Exception('Error when adding questions to course');
  }
  print('Questions added');
  return Future<String>.value('Complete new-course added');
}

/**
 * Function to update the course Id of the document in the
 * recommended collection
 */
Future updateRecommendedCourseByID({required String courseID}) async {

  if (await courseExists(courseID: courseID)) {
    recommendedCollectionRef
        .doc(recommendedDocName)
        .set({'courseID': courseID});
  } else {
    throw Exception('Course does not exist');
  }
}

/**
 * This Method updates the recommended course when provided with
 * the name of the new course. To do so, I use the method getCourseByTitle
 * and then I update the recommended course with updateRecommendedCourseWithID
 *
 * If completed correctly it returns the id of the course
 */
Future updateRecommendedCourseByTitle(String title) async {
  try {
    Course newRecommendedCourse = await getCourseByTitle(title: title);
    await updateRecommendedCourseByID(courseID: newRecommendedCourse.id!);
    return newRecommendedCourse.id;
  } catch (error) {
    print('Error when updating course');
    throw Exception('Error updating course');
  }
}

/**
 * Method to update the featured Course using the course's title
 * To do so, I get the Course from the DB so i can get the ID and
 * I can call updateFeaturedCourseByID
 */
```

```
  Future updateFeaturedCourseByTitle(String title) async {
    try {
      Course newFeatured = await getCourseByTitle(title: title);
      await updateFeaturedCourseByID(courseID: newFeatured.id!);
      return newFeatured.id;
    } catch (error) {
      print('Error when updating featured course');
      throw Exception('Error updating featured course');
    }
  }

  /**
   * Method to set the field courseID of the
   * featuredCourse to the one specified in the param
   */
  Future updateFeaturedCourseByID({required String courseID}) async {
    try {
      if (await courseExists(courseID: courseID)) {
        featuredCollectionRef.doc(featuredDocName).set({'courseID': courseID});
      } else {
        throw Exception('Course does not exist');
      }
    } catch (error) {
      print('Error setting field courseID in featured course');
      throw Exception('Error when setting new id');
    }
  }

  Future deleteCourseByTitle(String title) async {
    try {
      Course courseToDelete = await getCourseByTitle(title: title);

      //Before deleting the document of the course, you have to delete the
subcollection questions
      await
deleteQuestionsFromCourse(cr:coursesRef.doc(courseToDelete.id).collection(questio
nCollectionName));
      await coursesRef.doc(courseToDelete.id!).delete();
      return courseToDelete.id;
    } catch (error) {
      print('Error deleting a course');
      throw Exception('Error deleting the course');
    }
  }

  Future deleteQuestionsFromCourse({required CollectionReference cr}){

    //Get the documents in the collection

    return cr.get().then((QdSnapshot) {
      for (int i = 0; i < QdSnapshot.size; i++) {
        //I delete each document
        QdSnapshot.docs[i].reference.delete();
```

```
      }
    }).catchError((error) {
      print('Error deleting questions from course');
      throw Exception('Error deleting questions from course');
    });
  }
}
```