GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESARROLLO DE UNA APLICACIÓN EN iOS Y ANDROID PARA LA REALIZACIÓN DE EXÁMENES TIPO TEST

Autor: Guillermo José Aldrey Pastor

Director: Víctor Láiz Pérez

Julio 2021, Madrid

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

Desarrollo de una Aplicación en iOS y Android

para la realización de exámenes tipo test

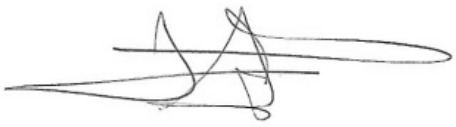
en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2020/21 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

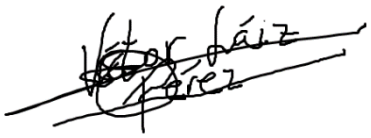
tomada de otros documentos está debidamente referenciada.



Fdo.: Guillermo José Aldrey Pastor Fecha: 10/07/2021

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO



Fdo.: Víctor Láiz Pérez Fecha: 10/07/2021

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESARROLLO DE UNA APLICACIÓN EN iOS Y ANDROID PARA LA REALIZACIÓN DE EXÁMENES TIPO TEST

Autor: Guillermo José Aldrey Pastor

Director: Víctor Láiz Pérez

Julio 2021, Madrid

Agradecimientos

En primer lugar, me gustaría agradecer a mi director de proyecto por haberme ofrecido su ayuda y los medios necesarios para poder desarrollar este proyecto.

En segundo lugar, me gustaría agradecer a mi familia, por haberme brindado la oportunidad de comenzar una carrera como es Ingeniería de Telecomunicaciones en esta universidad, a la que hoy pongo punto final con este proyecto, y a mis amigos, por haberme apoyado en los momentos más difíciles.

Por último, quiero agradecer a la Universidad y a todos mis profesores, por haberme dado los conocimientos, la fuerza y la confianza necesaria para afrontar una carrera y un trabajo como este.

Muchas gracias a todos, de corazón.

DESARROLLO DE UNA APLICACIÓN EN iOS Y ANDROID PARA LA REALIZACIÓN DE EXÁMENES TIPO TEST

Autor: Aldrey Pastor, Guillermo José.
Director: Láiz Pérez, Víctor.
Entidad Colaboradora: ICAI - Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

Palabras Clave: Nube, Serverless, AWS, Front-end, Back-end, API, Lambda.

Introducción

Esta memoria versa sobre el desarrollo de una aplicación móvil cuya misión es facilitar la práctica de exámenes para alumnos de titulaciones náuticas, tanto para el sistema operativo iOS, como para el sistema operativo Android.

La aplicación surge como solución a un reto cotidiano de la sociedad, como es la obtención de un permiso de circulación, en este caso náutico.

Hoy en día el mundo demanda la incursión de las personas en la constante transformación digital, y esto sumado a la reciente migración masiva hacia la nube, definen la necesidad de disponer de herramientas que hagan posible una nueva experiencia de usuario más accesible para todos.

Tradicionalmente en el desarrollo e implementación de aplicaciones web o móviles, se ha llevado a cabo una computación basada en servidores y muchas solicitudes HTTP hacia estos. Una aplicación se ejecuta en un servidor y el desarrollador es el encargado de administrar los recursos necesarios para ella.

Esto conlleva una serie de problemas como, por ejemplo, el precio que hay que pagar por mantener el servicio activo cuando no se está atendiendo ninguna solicitud, o la escalabilidad que requiere el servidor a medida que el uso de la aplicación aumenta, o por el contrario disminuye. Este tipo de inconvenientes puede frenar el verdadero objetivo que

tiene un desarrollador individual, que no es otro que construir y mantener la aplicación actual.

Por este motivo, y por muchos otros, el mundo de hoy en día se encuentra en una migración masiva a la nube. La introducción del Cloud Computing en nuestras vidas sirve como antesala a la transformación digital que ya experimentamos e irá creciendo exponencialmente en los próximos años. Los problemas que antes suponían un mayor coste y esfuerzo se solucionan con las nuevas e innovadoras tecnologías que surgen en todos los ámbitos de la vida, y especialmente en el sector de las telecomunicaciones.

Hasta el momento, se conocen varios modelos de C*loud Computing*. Por norma general, se distingue entre SaaS (Software as a Service o Software como Servicio), PaaS (Platform as a Service o Plataforma como Servicio) e IaaS (Infrastructure as a Service o Infraestructura como servicio).

En el mundo del desarrollo de aplicaciones también aparece la migración a la nube, y ante todos los inconvenientes que acarrea el uso de servidores de computación, aparece el concepto de computación sin servidor, o *serverless*.

En este modelo de ejecución, el proveedor de la capa de computación en la nube (AWS, Microsoft Azure...) es quien permite ejecutar durante un periodo de tiempo porciones de código denominadas “funciones”, evitando que sea el desarrollador quien se encargue de la infraestructura subyacente que se provisiona para dar servicio. El objetivo es convertir la computación en un servicio de fácil disposición para cualquier fin. El modelo de computación *serverless* ha adquirido el nombre de FaaS.

Este proyecto surge por lo tanto como una solución tecnológica a los problemas que suponen los modelos de computación con servidores en el ámbito de las aplicaciones móviles. Como objetivo general se tiene que apuntar la aplicación de la tecnología Cloud Computing en su modo sin servidores para facilitar la preparación de los exámenes para la obtención de titulaciones náuticas.

Metodología

Se ha dividido el desarrollo de la aplicación en dos fases. Desde septiembre hasta enero se lleva a cabo el desarrollo de la mayor parte del front end, y es a partir de enero cuando comienza el desarrollo de la parte lógica de la aplicación, o back end.

El diseño del front end se lleva a cabo utilizando el marco de programación de React Native, con objeto de conseguir utilizar código nativo para poder publicar la aplicación tanto en la plataforma iOS como en Android. Se utiliza el editor de texto Visual Studio. Esta fase se va a dividir a su vez en un primer periodo de investigación a través de la documentación ofrecida por React Native, para conocer el lenguaje y aprender a utilizar todo tipo de componentes, funciones y clases, seguido de un periodo de programación intenso donde se lleva a cabo todo el interfaz de usuario.

Se muestra a continuación un ejemplo del código nativo para ambas plataformas. Se trata del estilo que se aplica en un componente, en ese caso, un selector.

```
const pickerSelectStyle = StyleSheet.create({
   inputIOS: {
      height: 40,
      backgroundColor: 'white',
      borderColor: 'black',
      borderWidth: 0.5,
      marginHorizontal: 30,
      marginBottom: 10,
      borderRadius: 3
    },
    inputAndroid: {
      height: 40,
      backgroundColor: 'white',
      borderColor: 'black',
      borderWidth: 0.5,
      marginHorizontal: 30,
      borderRadius: 3
    },
 })
```

El diseño de la parte lógica o back-end de la aplicación se basa en los servicios que proporciona el proveedor de la nube Amazon Web Services (AWS). Cabe mencionar que el servicio de AWS donde reside la aplicación móvil es AWS Amplify, y desde esta consola se han incorporado todos los elementos que componen el back end de la misma (Bases de Datos, APIs, Autenticación de Usuarios, etc.)

En cuanto al plan de desarrollo de la aplicación, se han puesto en marcha técnicas "Agile" para su correcto desarrollo y seguimiento. Se van a llevar a cabo numerosos sprints de seguimiento del trabajo junto con el director del proyecto. En estas reuniones se decidirá cuáles son los siguientes pasos, de qué manera queremos afrontar lo que tenemos por delante, y se pondrá una fecha para la próxima reunión, dejando el suficiente tiempo para poder desarrollar y avanzar con los objetivos semanales del proyecto.

Descripción de la herramienta AWS

Amazon Web Services es uno de los proveedores de la nube más importantes en la actualidad. Ofrece una gran cantidad de servicios con funcionalidad completa, desde tecnologías de infraestructura como bases de datos hasta tecnologías emergentes como inteligencia artificial o el internet de las cosas (IOT).

AWS ofrece innumerables capacidades híbridas en almacenamiento, seguridad, herramientas de gestión, o implementación de aplicaciones. Además, cuenta con la comunidad de clientes más grande y dinámica, debido a su papel como el entorno más flexible y seguro en la actualidad. Esto, hace que sea la plataforma líder en la nube.

En este proyecto se va a desarrollar una aplicación móvil utilizando el conjunto de servicios y herramientas AWS Amplify, uno de los servicios de Amazon que permite conectar aplicaciones y configurar el back-end de las mismas en cuestión de minutos. Gracias a este servicio resulta sencillo añadir a la aplicación los distintos elementos que componen su parte lógica, bien a través de la propia consola de Amplify, o bien desde la terminal del editor Visual Studio Code, en el proyecto React Native creado.

Amplify admite el marco de aplicaciones móviles React Native de código abierto, el cual se usará para desarrollar y juntar el código front end de la aplicación en el momento de implementar la misma.

Los servicios AWS que componen la arquitectura de la aplicación son: Amazon Cognito para la administración de usuarios; AWS AppSync, para simplificar el acceso a los datos y el desarrollo de APIs; AWS Lambda, permite ejecutar código evitando administrar servidores; GraphQL, lenguaje basado en queries que actúa junto con AppSync y facilita la obtención de los datos desde AWS DynamoDB, el servicio de bases de datos que se utilizará.

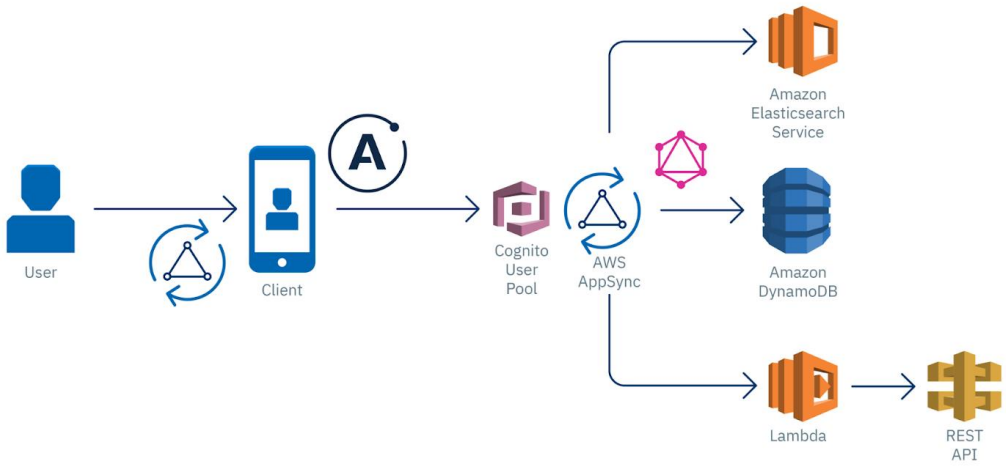


Figura 1 - Arquitectura AWS

Resultados

Tras las fases descritas en los puntos anteriores, se ha conseguido cumplir con los objetivos del proyecto.

Se dispone de una aplicación nativa para sendas plataformas iOS y Android, con la posibilidad de realizar todo tipo de exámenes tipo test para la obtención de un título náutico.

Desde el punto de vista del desarrollo del código para la interacción del usuario con la aplicación, se ha conseguido utilizar React Native para hacer funcionar los archivos JavaScript en ambas tecnologías, como se muestra a continuación:

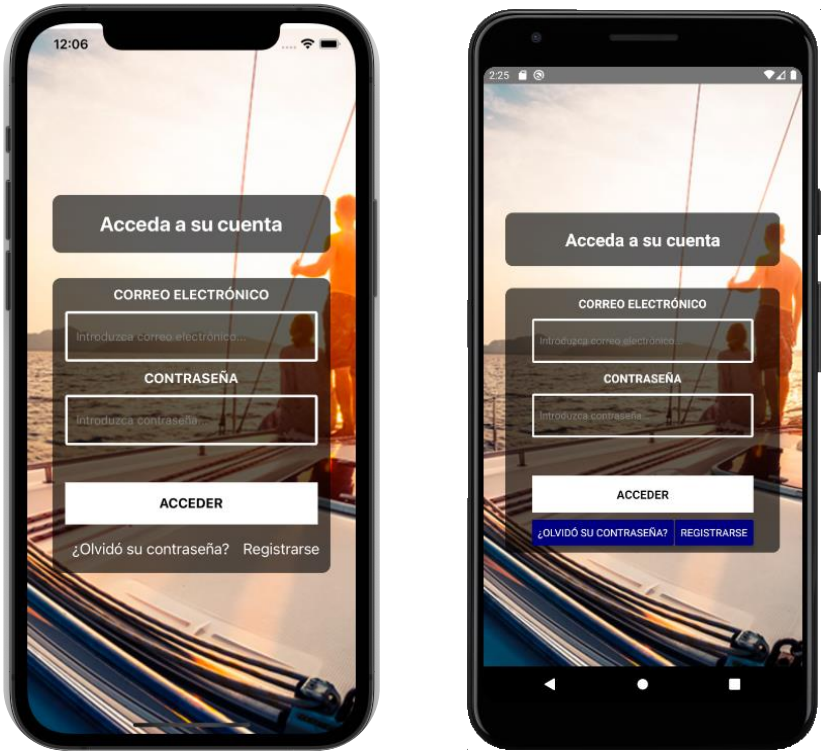


Figura 2 - Ejemplo Front End (izquierda: iOS, derecha: Android)

Desde el punto del back end, se ha conseguido implementar toda la lógica que requiere la aplicación, cumpliendo con los requisitos funciones especificados. De esta manera, se cuenta con la siguiente información en la nube de AWS.

Autenticación de Usuarios	**Nombre**
User Pool – Cognito	testpatron4dbb9557_userpool_d4bb9557_dev

Tabla 1 - Recursos AWS. Autenticación de Usuarios

API	Nombre
GraphQL API	grahqlApi-dev

Tabla 2 - Recursos AWS. API

Esta GraphQL API tiene una serie de funciones (queries y mutations) que se encarga de hacer las *requests* PUT y GET en las distintas bases de datos.

Función	Nombre
query para traer preguntas de bbdd (PY)	getPreguntasPy
query para traer preguntas de bbdd (PER)	getPreguntasPer
query para traer usuario de bbdd	UserByEmail
query para traer examen de bbdd	examenByUser
Mutation para insertar resultado de examen en la bbdd	createExamen
Mutation para insertar usuario en la bbdd	createUsuario
Mutation para actualizar preguntas falladas de un usuario en la bbdd	updateUsuario

Tabla 3 - Recursos AWS. Funciones

Base de Datos	Nombre	Clave de partición
Preguntas titulación PY (520 elementos)	PreguntasPy-tyfyaoesa5cmnlquo5gdwii6cq-dev	Id (cadena)
Preguntas titulaciones PER y PNB (780 elementos)	PreguntasPerPnb-tyfyaoesa5cmnlquo5gdwii6cq-dev	Id (cadena)
Exámenes realizados	Examen-tyfyaoesa5cmnlquo5gdwii6cq-dev	Id (cadena)
Usuarios registrados	Usuario-tyfyaoesa5cmnlquo5gdwii6cq-dev	Id (cadena)

Tabla 4 - Recursos AWS. Bases de Datos

Conclusiones

Una vez llegado al final del proyecto, y haber experimentado con el desarrollo las distintas fases de este, se obtienen una serie de conclusiones.

Se ha comprobado la mejoría que supone el hecho de no utilizar servidores en el desarrollo de una aplicación. Primero, por la velocidad a la que actúan las funciones Lambda dentro de un API, en este caso GraphQL. En cuestión de segundos la aplicación es capaz de obtener y trabajar con datos por sí misma. Además, supone un gran ahorro de dinero, pues si el desarrollador desea utilizar una instancia de EC2 (servicio de alquiler de servidores que proporciona AWS), debe mantener esa instancia corriendo todo el tiempo, lo que supone un mayor gasto. Nota: en EC2 la facturación se produce según el número de instancias activas y sus horas de funcionamiento mientras que en el caso de Lambda la facturación se realiza por llamadas al servidor.

También se llega a la conclusión de que cada consulta que se hace en la base de datos consume muy pocas unidades de capacidad de lectura. Se puede demostrar con el siguiente gráfico, que muestra el promedio de las unidades de capacidad de lectura consumidas por segundo a lo largo de un periodo de 1 minuto.

Figura 3 - Capacidad de lectura de la tabla "PreguntasPY"

El desarrollo del front end de la aplicación ha resultado más sencillo que el desarrollo del back end, y es que React Native es prácticamente idéntico al lenguaje JavaScript. Aun así, aparecen algunos impedimentos para los programadores, por ejemplo, con la multitud de librerías que necesita React Native a la hora de insertar componentes.

En cuanto a los programas Xcode y Android Studio, utilizados para la simulación e implementación de la aplicación en las respectivas plataformas iOS y Android, existen algunas diferencias. Xcode presenta una mayor complejidad a la hora de subir la aplicación al App Store, mientras que este proceso es relativamente sencillo para Android. Sin embargo, Android Studio implica fallos y complicaciones a la hora de instalar el programa y lanzar el proyecto React Native en el mismo.

Finalmente, considero de gran acierto mi decisión de afrontar un proyecto de este estilo debido al conocimiento que he podido adquirir sobre varias tecnologías y sobre la nube. Considero que la sociedad demanda millones de soluciones a los problemas cotidianos, y con este proyecto me he dado cuenta de la existente necesidad de profesionales capaces de ofrecer mejoras requeridas por toda la población. Esta aplicación puede suponer un paso en esa dirección.

Referencias

[1] *Amplify Framework Documentation*. (s.f.). Obtenido de https://docs.amplify.aws/

[2] Becerro, R. G. (2017). *Paradigma*. Obtenido de https://www.paradigmadigital.com/dev/aws-lambda-arquitectura-serverless-implementar-apis/

[3] Borillo, R. (30 de Marzo de 2019). *Genbeta*. Obtenido de https://www.genbeta.com/desarrollo/que-serverless-que-adoptarlo-desarrollo-tu-proxima-aplicación

[4] dfo. (7 de Septiembre de 2018). *Deusto Formation*. Obtenido de https://www.deustoformacion.com/blog/programacion-tic/para-que-sirve-conectar-aplicacion-movil-con-nube

[5] Diseño web illusion Studio. (12 de Febrero de 2020). *Illusionstudio*. Obtenido de https://www.illusionstudio.es/apps-moviles-importancia-actualidad

[6] Lamata, E. (2 de Marzo de 2018). *elEconomista*. Obtenido de https://www.eleconomista.es/emprendedores-innova/noticias/8976705/03/18/Aplicaciones-moviles-para-cumplir-propositos-un-negocio-en-auge.html

[7] Lozano, V. (17 de Octubre de 2019). *Qué es y hacia dónde va el cloud computing. FHIOS Consultoría Estratégica.* . Obtenido de https://www.fhios.es/cloud-computing-introduccion/

[8] Martín, Á. J. (18 de Junio de 2019). *OpenWebinars*. Obtenido de https://openwebinars.net/blog/react-native-que-es-para-que-sirve/

[9] Naciones Unidas. (s.f.). *Naciones Unidas*. Obtenido de La Agenda para el Desarrollo Sostenible: https://www.un.org/sustainabledevelopment/es/development-agenda/

DEVELOPMENT OF AN APPLICATION ON iOS AND ANDROID FOR TAKING MULTIPLE-CHOICE EXAMS

Author: Aldrey Pastor, Guillermo José.
Supervisor: Pérez Láiz, Víctor.
Collaborating Entity: ICAI - Universidad Pontificia Comillas.

ABSTRACT

Keywords: Cloud, Serverless, AWS, Front-end, Back-end, API, Lambda.

Introduction

This report deals with the development of a mobile application aimed at helping students on their preparation for their exams to obtain a nautical license, both for the iOS operating system and for the Android operating system.

The application arises as a solution to a daily challenge of society, such as obtaining a driving license, in this case nautical.

Today the world demands the incursion of people in the constant digital transformation, and this added to the recent massive migration to the cloud, define the need for tools that make possible a new user experience more complex and accessible to all.

Traditionally in the development and implementation of web or mobile applications, server-based computing and many HTTP requests to servers have been carried out. An application runs on a server and the developer is in charge of managing the resources needed for it.

This leads to a number of problems, such as the price to be paid for keeping the service active when no request is being served, or the scalability required by the server as the application's usage increases or decreases. These types of inconveniences can slow down the real objective that an individual developer has, which is none other than to build and maintain the actual application.

For this reason, and many others, the world today is in a massive migration to the cloud. The introduction of cloud computing into our lives serves as a prelude to the digital transformation we are already experiencing and will grow exponentially in the coming years. Problems that previously entailed greater cost and effort are being solved with new and innovative technologies that are emerging in all areas of life, and especially in the telecommunications sector.

Various Cloud Computing models are known to date. As a rule, a distinction is made between SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service).

In the world of application development, the migration to the cloud is also appearing, and in view of all the inconveniences associated with the use of computing servers, the concept of *serverless* computing has appeared.

In this execution model, the provider of the cloud computing layer (AWS, Microsoft Azure, etc.) allows portions of code called "functions" to be executed for a period of time, thus avoiding the developer being in charge of the underlying infrastructure to provide the service. The goal is to make the computing power a service at the disposal of any objective. The *serverless* computing model has acquired the name FaaS.

This project therefore arises as a technological solution to the problems posed by server-based computing models in the domain of mobile apps. As a general objective, the application of Cloud Computing technology in its serverless mode to help students practicing for their exams to obtain a nautical license.

Methodology

The development of the application has been divided into two phases. From September to January the development of most of the front end is carried out, and it is from January when the development of the logical part of the application, or back end, begins.

The design of the front end is carried out using the React Native programming framework, in order to use native code to be able to publish the application on both the iOS and Android platforms. The Visual Studio text editor is used. This phase will be divided in turn into a first period of research through the documentation provided by React Native, to learn the language and learn to use all kinds of components, functions and classes, followed by a period of intense programming where the entire user interface is carried out.

An example of the native code for both platforms is shown below. This is the style that is applied to a component, in this case, a selector.

```
const pickerSelectStyle = StyleSheet.create({
   inputIOS: {
      height: 40,
      backgroundColor: 'white',
      borderColor: 'black',
      borderWidth: 0.5,
      marginHorizontal: 30,
      marginBottom: 10,
      borderRadius: 3
    },
    inputAndroid: {
      height: 40,
      backgroundColor: 'white',
      borderColor: 'black',
      borderWidth: 0.5,
      marginHorizontal: 30,
      borderRadius: 3
    },
 })
```

The design of the logical or back-end part of the application is based on the services provided by the cloud provider Amazon Web Services (AWS). It is worth mentioning that the AWS service where the mobile application resides is AWS Amplify, and from this console all the elements that make up the back end of it (Databases, APIs, User Authentication, etc.) have been incorporated.

Regarding the application development plan, "Agile" techniques have been implemented for its correct development and monitoring. Numerous work follow-up sprints will be carried out together with the project manager. In these meetings we will decide what the next steps are, how we want to face what lies ahead, and we will set a date for the next meeting, leaving enough time to develop and advance with the weekly objectives of the project.

AWS Description

Amazon Web Services is one of the most important cloud providers today. It offers a wide range of full-featured services, from infrastructure technologies such as databases to emerging technologies such as artificial intelligence or the Internet of Things (IoT).

AWS offers countless hybrid capabilities in storage, security, management tools, or application deployment. In addition, it has the largest and most dynamic customer community, due to its role as the most flexible and secure environment today. This makes it the leading platform in the cloud.

In this project we are going to develop a mobile application using the AWS Amplify set of services and tools, one of Amazon's services that allows you to connect applications and configure their back end in a matter of minutes.

Thanks to this service, it is easy to add to the application the different elements that make up its logical part, either through the Amplify console itself, or from the Visual Studio Code editor terminal, in the React Native project created.

Amplify supports the open source React Native mobile application framework, which will be used to develop and assemble the front-end code of the application when deploying it.

The AWS services that make up the application architecture are: Amazon Cognito for user management; AWS AppSync, to simplify data access and API development; AWS Lambda, allows code execution while avoiding managing servers; GraphQL, a query-based language that acts in conjunction with AppSync and facilitates fetching data from AWS DynamoDB, the database service that will be used.

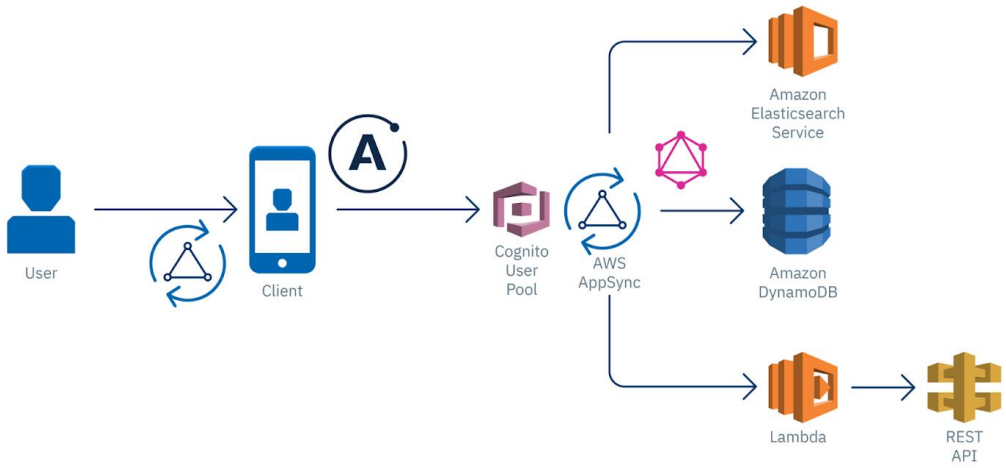


Figura 4 - AWS Architecture

Results

After the phases described in the previous points, the objectives of the project have been achieved.

A native application is available for both iOS and Android platforms, with the possibility of taking all kinds of tests for obtaining a nautical qualification.

From the point of view of code development for user interaction with the application, it has been possible to use React Native to make JavaScript files work in both technologies, as shown below:

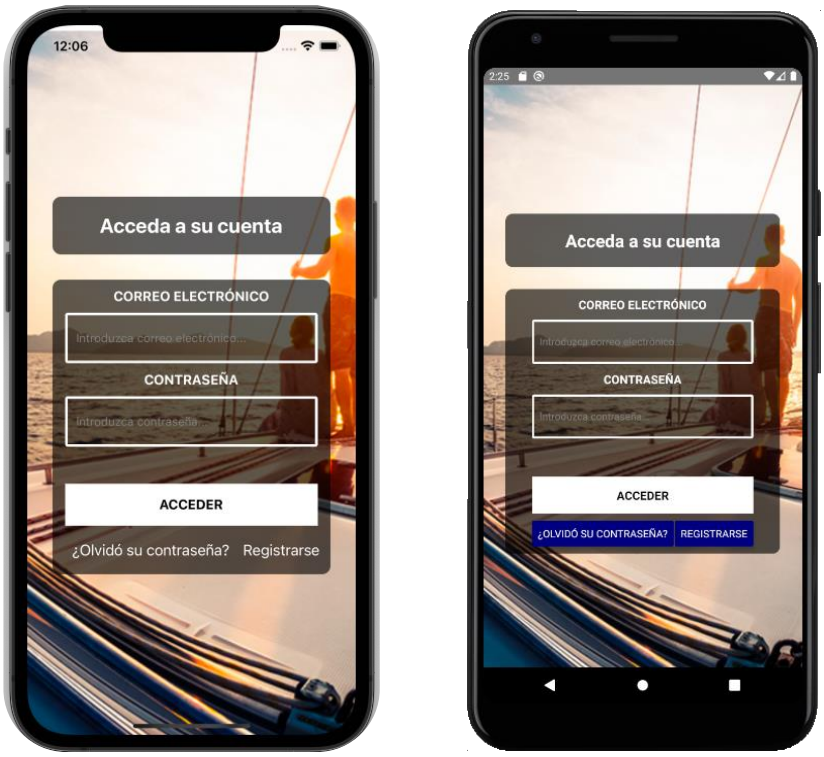


Figura 5 - Front End Example (left: iOS; right: Android)

From the back end point of view, it has been possible to implement all the logic required by the application, complying with the specified function requirements. In this way, the following information is available in the AWS cloud.

User Authentication	Name
User Pool – Cognito	testpatron4dbb9557_userpool_d4bb9557_dev

Tabla 5 - AWS Resources. User Authentication

API	Name
GraphQL API	grahqlApi-dev

Tabla 6 - AWS Resources. API

This GraphQL API has a series of functions (queries and mutations) that are in charge of making PUT and GET requests to the different databases.

Function	Name
query para traer preguntas de bbdd (PY)	getPreguntasPy
query para traer preguntas de bbdd (PER)	getPreguntasPer
query para traer usuario de bbdd	UserByEmail
query para traer examen de bbdd	examenByUser
Mutation para insertar resultado de examen en la bbdd	createExamen
Mutation para insertar usuario en la bbdd	createUsuario
Mutation para actualizar preguntas falladas de un usuario en la bbdd	updateUsuario

Tabla 7 - AWS Resources. Functions

Database	Name	Partition Key
Preguntas titulación PY (520 elementos)	PreguntasPy-tyfyaoesa5cmnlquo5gdwii6cq-dev	Id (cadena)
Preguntas titulaciones PER y PNB (780 elementos)	PreguntasPerPnb-tyfyaoesa5cmnlquo5gdwii6cq-dev	Id (cadena)
Examenes realizados	Examen-tyfyaoesa5cmnlquo5gdwii6cq-dev	Id (cadena)
Usuarios registrados	Usuario-tyfyaoesa5cmnlquo5gdwii6cq-dev	Id (cadena)

Tabla 8 - AWS Resources. Databases

Conclusions

Having reached the end of the project and having experimented with the development of the different phases of the project, a series of conclusions can be drawn.

The improvement of not using servers in the development of an application has been verified. First, because of the speed at which the lambda functions act within an API, in this case GraphQL. In a matter of seconds, the application is able to obtain and work with data by itself. It also saves a lot of money, because if the developer wants to use an EC2 instance (server rental service provided by AWS), he has to keep that instance running all the time, which is a higher expense. Note: On EC2, the billing is based on the number of active instances and the number of hours in use, while on Lambda, the billing is based on the number of requests.

It is also concluded that each query that is made on the database consumes very few units of read capacity. This can be demonstrated with the following graph, which shows the average number of read capacity units consumed per second over a period of 1 minute.

Figura 6 - Reading Capacity of the table "PreguntasPY"

The development of the front end of the application has been easier than the development of the back end, since React Native is practically identical to the JavaScript language. Even so, there are some impediments for programmers, for example, with the multitude of libraries that React Native needs when inserting components.

As for the programs Xcode and Android Studio, used for the simulation and implementation of the application on the respective iOS and Android platforms, there are some differences. Xcode presents a greater complexity when uploading the application to the App Store, while this process is relatively simple for Android. However, Android Studio involves bugs and complications when installing the program and launching the React Native project in it.

Finally, I consider my decision to take on a project of this style to be a great success due to the knowledge I have been able to acquire about various technologies and about the cloud. I believe that society demands millions of solutions to everyday problems, and with this project I have realized the existing need for professionals capable of offering improvements required by the entire population. This application can be a step in that direction.

Mentions

[1] *Amplify Framework Documentation*. (s.f.). Obtenido de https://docs.amplify.aws/

[2] Becerro, R. G. (2017). *Paradigma*. Obtenido de https://www.paradigmadigital.com/dev/aws-lambda-arquitectura-serverless-implementar-apis/

[3] Borillo, R. (30 de Marzo de 2019). *Genbeta*. Obtenido de https://www.genbeta.com/desarrollo/que-serverless-que-adoptarlo-desarrollo-tu-proxima-aplicación

[4] dfo. (7 de Septiembre de 2018). *Deusto Formation*. Obtenido de https://www.deustoformacion.com/blog/programacion-tic/para-que-sirve-conectar-aplicacion-movil-con-nube

[5] Diseño web illusion Studio. (12 de Febrero de 2020). *Illusionstudio*. Obtenido de https://www.illusionstudio.es/apps-moviles-importancia-actualidad

[6] Lamata, E. (2 de Marzo de 2018). *elEconomista*. Obtenido de https://www.eleconomista.es/emprendedores-innova/noticias/8976705/03/18/Aplicaciones-moviles-para-cumplir-propositos-un-negocio-en-auge.html

[7] Lozano, V. (17 de Octubre de 2019). *Qué es y hacia dónde va el cloud computing. FHIOS Consultoría Estratégica.* . Obtenido de https://www.fhios.es/cloud-computing-introduccion/

[8] Martín, Á. J. (18 de Junio de 2019). *OpenWebinars*. Obtenido de https://openwebinars.net/blog/react-native-que-es-para-que-sirve/

[9] Naciones Unidas. (s.f.). *Naciones Unidas*. Obtenido de La Agenda para el Desarrollo Sostenible: https://www.un.org/sustainabledevelopment/es/development-agenda/

Índice de la memoria

Índice de figuras

Índice de tablas

Notación

AWS	Amazon Web Services
front-end	front-end se refiere a la parte de la aplicación en la que se implementa la interacción del usuario con el software
back-end	back-end se refiere a la lógica de la aplicación, la parte pesada que conecta con la base de datos
Cloud Computing	Servicios de computación en la nube ofrecidos a través de una red
Serverless (FaaS)	Modelo de computación sin servidores
IaaS	Infrastructure as a Service - Infraestructura como servicio
PaaS	Platform as a Service - Plataforma como servicio
SaaS	Software as a Service - Software como servicio
FaaS	Function as a Service – Función como servicio
API	Application Programming Interface - Interfaz de programación de aplicaciones
CLI	Command Line Interface – Interfaz de línea de comandos
Wireframe	Plano de pantalla, guía visual
JSX	Extensión de JavaScript para el uso con su librería React
PER	Patrón de Embarcaciones de Recreo
PNB	Patrón de Navegación Básica
PY	Patrón de Yate
ODS	Objetivos de Desarrollo Sostenible
PNUD	Programa de las Naciones Unidas para el Desarrollo
BBDD	Database – Base de datos

Tabla 9 - Notación

Capítulo 1. INTRODUCCIÓN

Hoy en día el mundo sufre una constante transformación digital que hace imprescindible la adaptación de las personas a las nuevas tecnologías y a su forma de uso.

En este proyecto se lleva a cabo el desarrollo de una aplicación móvil nativa destinada a la realización de exámenes tipo test para preparación del título PER (Patrón de Embarcaciones de Recreo), PNB (Patrón para Navegación Básica) y PY (Patrón de Yate).

Cada vez son menos los alumnos que van a la autoescuela de forma presencial para sacarse el carné de coche, ya que existen múltiples plataformas online en las cuales estos se pueden preparar para el examen teórico, mediante la realización de pruebas idénticas a las que tendrán que responder el día del examen.

Con un título náutico pasa algo parecido. Cada vez son más los cursos online que se ofrecen para obtener este tipo de certificado. Por lo tanto, la idea de llevar a cabo esta aplicación puede ayudar a personas que busquen la comodidad de estudiar y practicar desde sus casas o durante sus rutinas diarias. Además, la crisis del Covid’19 ha hecho del estudio y el trabajo online un nuevo aliado en nuestras vidas, y la sociedad se ha acostumbrado a esta transformación.

Se va a desarrollar tanto el front-end como el back-end de la aplicación, y se va a sumergir en el mundo del Cloud Computing utilizando una de las plataformas más importantes y que más ha contribuido a la incursión de la computación en la nube en nuestras vidas, como es AWS (Amazon Web Services). También se va a utilizar la tecnología React Native, para poder realizar la aplicación móvil nativa tanto para dispositivos iOS como para dispositivos Android.

1.1 MOTIVACIÓN DEL PROYECTO

En los últimos años las posibilidades que ofrece la computación en la nube o *Cloud Computing* han crecido y variado en gran medida.

Esta transformación está cambiando la realidad en el mundo de los negocios, que se apoyan cada vez más en infraestructura IT. Resulta imposible pasar de largo por una tecnología que te brinda la capacidad de acceder a información desde cualquier sitio, en cualquier momento, y con cualquier dispositivo y que hace de la computación un servicio de fácil disposición para cualquier cometido.

Hoy en día, este paradigma ha incluido entre sus servicios el desarrollo de distintos tipos de aplicaciones y la gestión de infraestructura en la nube. Esto es, en gran medida, gracias a la aparición de plataformas como Amazon Web Services (AWS) o Microsoft Azure.

AWS comenzó a proporcionar servicios de informática en la nube en el año 2006. Basándose en un sistema de plataforma escalable y de bajo costo, ha conseguido que las empresas puedan obtener una gran cantidad de servidores en cuestión de minutos, sin tener que lidiar con la provisión de los equipos informáticos subyacentes y la consecuente inversión en capital, así como su correcto dimensionamiento y mantenimiento.

Este proyecto busca llevar esta tecnología a otros ámbitos, como es una escuela náutica. Se tiene como fin conseguir acceder a los diferentes tipos de exámenes almacenados mediante tecnología Cloud, e implementar un formato de examen destinado a la preparación del examen oficial. En concreto, se proporcionarán dos modalidades fundamentales: exámenes por temas (para practicar lo estudiado en una sesión) y exámenes completos, que simulan la composición del examen oficial al que se enfrentará el alumno. Además, se incluyen dos funcionalidades adicionales: “preguntas ilimitadas”, para práctica continua y “preguntas

falladas", destinada a que el alumno preste especial atención a aquellas preguntas que ha fallado anteriormente y pueda afianzar conceptos. Estas dos modalidades son una innovación específica de este proyecto, ya que no se han encontrado en las aplicaciones actualmente disponibles.

Con este proyecto también se busca ponerse a disposición de escuelas que no dispongan aún de ninguna aplicación móvil para sus alumnos y deseen añadir esta prestación a su oferta académica.

1.2 ESTRUCTURA DE LA MEMORIA

Resumen & Abstract

Se busca dar una visión general del proyecto para que alguien pueda hacerse una pequeña idea sobre el mismo.

Capítulo 1. Introducción

El primer capítulo versa sobre la problemática que se desea solucionar y cuáles son las razones que motivan a llevar a cabo el proyecto.

Capítulo 2. Descripción de las tecnologías

Este segundo capítulo describe todas las tecnologías utilizadas para llevar a cabo la aplicación.

Capítulo 3. Estado de la Cuestión

El Estado de la Cuestión trata de revisar soluciones existentes a el problema en cuestión e intenta mostrar en qué se diferencia este proyecto de aquellos mencionados.

Capítulo 4. Definición del trabajo

Se justifica la realización de la aplicación teniendo en cuenta lo comentado en el Estado de la Cuestión y se define como será el desarrollo del trabajo.

Capítulo 5. Sistema Desarrollado

En este capítulo se explican todos los detalles técnicos de la aplicación y se llevan a cabo una serie de análisis para facilitar el diseño y desarrollo de esta.

Capítulo 6. Análisis de Resultados

Se lleva a cabo el diseño y resultado de las pruebas.

Capítulo 7. Conclusiones

Las conclusiones sobre el proyecto se tratan en este capítulo.

Capítulo 8. Bibliografía

Anexo I. Alineación del proyecto con los ODS

Este Anexo es muy importante. Se hace una reflexión sobre la alineación del Proyecto con los Objetivos de Desarrollo Sostenible (ODS) de las Naciones Unidas.

Anexo II. Wireframes

Se muestran las capturas de las pantallas que componen la aplicación.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Dividiremos este capítulo en dos apartados, diferenciando las tecnologías utilizadas en el desarrollo del front-end, y aquellas utilizadas para el desarrollo del back-end.

2.1 TECNOLOGÍAS FRONT END

- **React Native.** Conviene comenzar describiendo la tecnología en la que se basa la aplicación. React Native es un *framework* de programación de aplicaciones nativas multiplataforma que está basado en JavaScript y ReactJS. Su funcionamiento es el siguiente:

  El compilador de React Native transforma nuestros documentos JSX en elementos nativos de la interfaz para iOS y Android. Es decir, se genera una interfaz nativa, y se consigue que las aplicaciones tengan un rendimiento y una experiencia de navegación y de usuario muy similar a las aplicaciones nativas. (Martín, 2019)

  Además, JavaScript se ejecuta nativamente, no se compila o transpila a Java o a ObjectiveC. Esto es así porque React Native está generando una especie de doble entre toda la parte que sigue ejecutando módulos nativos como la interfaz o cualquier librería que tengamos integrada ya existente con programación en Android en iOS, y una máquina virtual ejecutando JavaScript.

  Aparece el bridge de React Native. Es el puente que va a permitir la comunicación entre ambos threads para el paso de información o el acceso de cualquier componente del dispositivo.

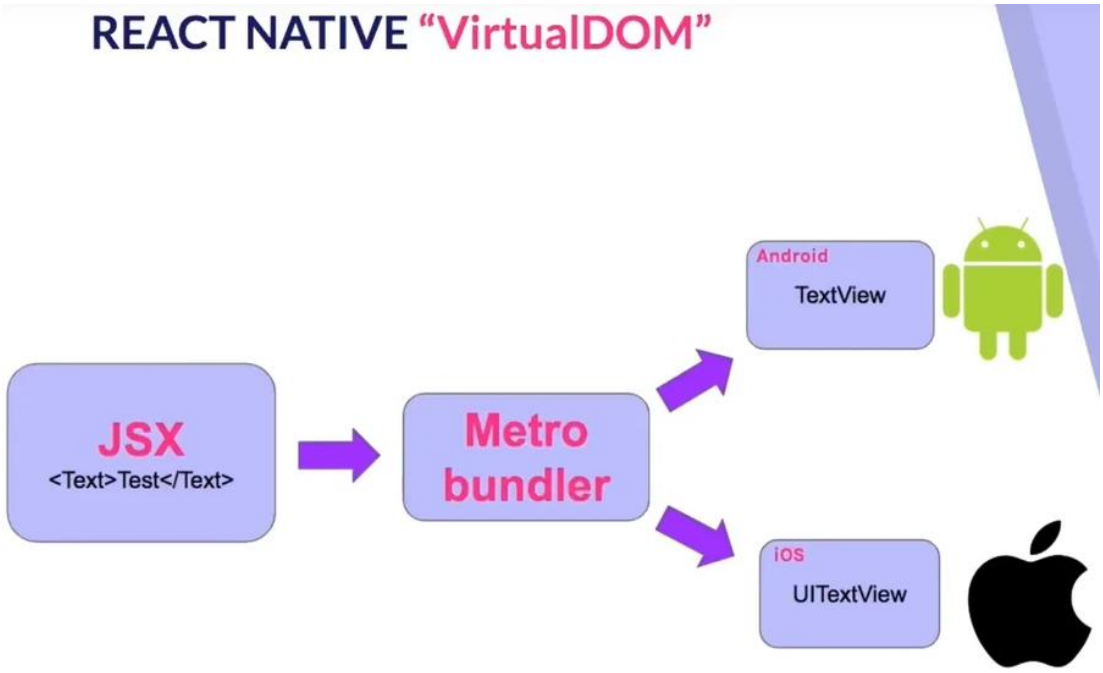


Figura 7 – Funcionamiento de React Native

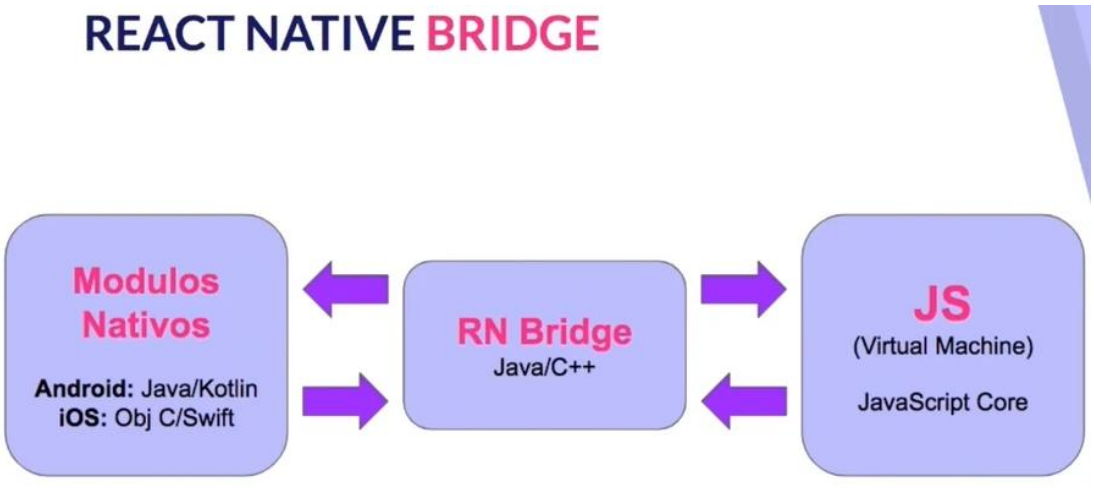


Figura 8 - RN Bridge

Una vez entendido el funcionamiento de React Native, se explican las tecnologías usadas para las dos diferentes opciones de destino.

Tecnologías para el desarrollo iOS:

- **Xcode** → Entorno de desarrollo integrado para macOS. Contiene un gran número de herramientas creadas por Apple destinadas al desarrollo de software para iOS.

  Será necesario instalar tanto las herramientas de línea de comando de Xcode, como un simulador con la versión correspondiente de iOS que se desea utilizar.

- **CocoaPods** → Se trata de un gestor de dependencias para Xcode, que permite agregar frameworks a los proyectos mediante un simple comando. En otras palabras, permite inyectar bibliotecas externas que dotarán de un nivel extra de calidad a la aplicación.

Tecnologías para el desarrollo Android:

- **Android Studio** → Entorno de desarrollo integrado para el desarrollo de aplicaciones para la plataforma Android. Está basado en IntelliJ IDEA.

- **JDK** → Kit de desarrollo de Java (Java Development Kit), es un software que provee herramientas de desarrollo para la creación de programas en Java. Será necesario instalarse una versión superior o igual a JDK8.

Tecnologías comunes:

- **Visual Studios Code** → Editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Desde el terminal de Visual Studios se desarrolla el back-end de la aplicación llamando a la consola de AWS.

- **Node.js** → Entorno de tiempo de ejecución de JavaScript. Incluye todo lo necesario para ejecutar un programa en JavaScript. Utiliza un modelo de solicitud y respuesta controlado por eventos que permite llevar a cabo cualquier tipo de operación requerida por la aplicación, desde leer archivos hasta realizar una solicitud HTTP.

- **Watchman** → Esta herramienta perteneciente a Facebook ayuda a observar cambios en el sistema de archivos. Se recomienda su uso para obtener un mejor rendimiento.

2.2 TECNOLOGÍAS BACK END

AWS (Amazon Web Services)

Todo el desarrollo de la lógica de la aplicación corre de la mano de este proveedor de servicios de computación Cloud. Se trata de la mayor plataforma de Cloud Computing del mundo, con más de 200 servicios integrales de centros de datos a nivel global.

Proporciona una gran variedad de servicios de infraestructura tales como almacenamiento, redes, bases de datos, servicios de aplicaciones, potencia de cómputo, mensajería, inteligencia artificial, servicios móviles, seguridad, identidad y conformidad, entre otros, los cuales permiten el crecimiento de las empresas.

Amazon Web Services permite desarrollar la lógica de una aplicación y adjuntar el código perteneciente al front-end de la misma en cuestión de minutos, y esa es una de las razones por las cuales se ha decidido utilizar el proveedor.

A continuación, se explicarán detalladamente qué servicios AWS componen la aplicación y cuál es la función de cada uno de ellos dentro de esta.

2.2.1. Servicios AWS

- **AWS Amplify:** Conjunto de herramientas y servicios que agilizan el desarrollo de la aplicación móvil. El Marco de Amplify permite crear el back-end de la aplicación e integrarlo con la aplicación React Native.

  La ventaja de utilizar la CLI de AWS Amplify es la rapidez con la que se puede configurar back-end escalables con autenticación, almacenamiento, datos, y otros casos de uso comunes.

  A continuación, se muestran dos imágenes en las que aparecen la consola de administración de un proyecto Amplify (en la cual se pueden juntar front y back de la aplicación), y el admin UI, donde trabaja con la parte lógica y se encentran almacenados el resto de los recursos de AWS utilizados por la aplicación.

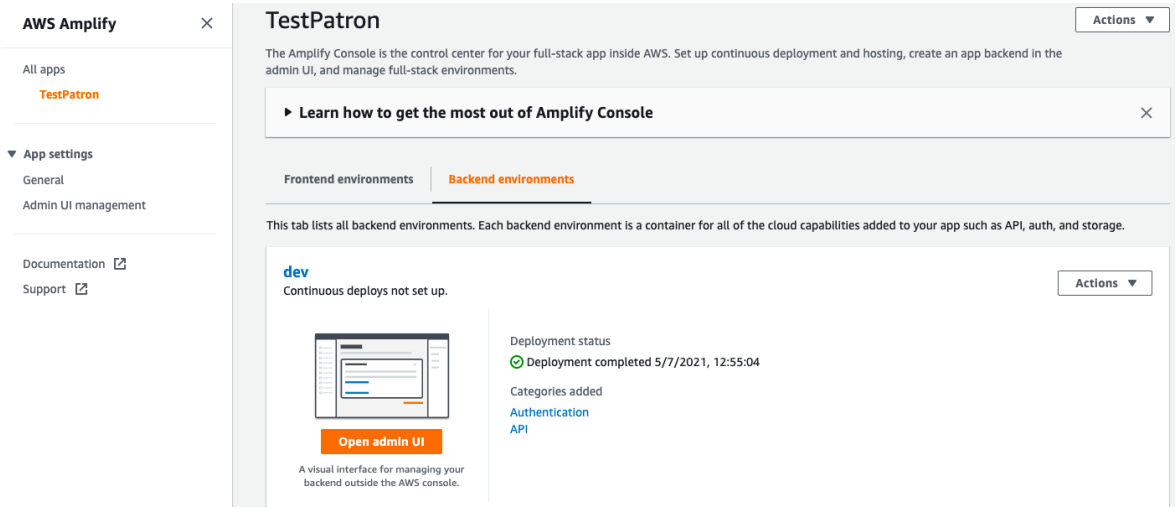


Figura 9 - Consola de AWS Amplify

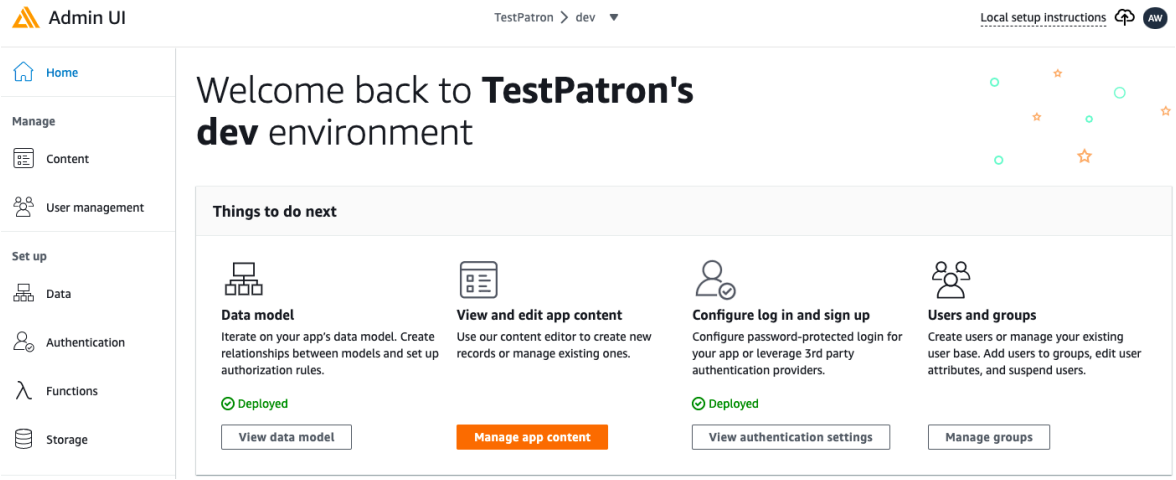


Figura 10 - Admin UI, TestPatron

Como se aprecia en las imágenes, resulta muy sencillo añadir los diferentes componentes de la aplicación al entorno de desarrollo desde la consola de Amplify.

- **AWS Cognito:** Proporciona autenticación, autorización y administración de usuarios para sus aplicaciones móviles y web.

- **Amazon API Gateway:** Se encarga de la creación, publicación, mantenimiento, monitoreo y protección de API REST, HTTP y WebSocket a cualquier escala.

- **AppSync:** Facilita el desarrollo de API de GraphQL ya que se encarga de la tarea de conectar de manera segura los orígenes de datos como AWS DynamoDB, Lambda y otros.

- **AWS Lambda:** Servicio que permite ejecutar código sin necesidad de administrar servidores. Son funciones *serverless* intermedias.

- **Amazon DynamoDB:** Servicio de base de datos no SQL que proporciona AWS, especialmente indicado para la interacción con Lambda.

2.2.2. Arquitectura del proceso de extracción e inserción de datos

Los tres últimos servicios descritos en el punto anterior permiten desarrollar la lógica de datos de la aplicación. A la hora de crear el back-end en Amplify, se han de añadir estos servicios al proyecto mediante el comando “amplify add api”.

De esta forma, implementaremos una GraphQL API que se encargará de generar código para llevar a cabo consultas e inserciones de datos a través de funciones Lambda y de AppSync.

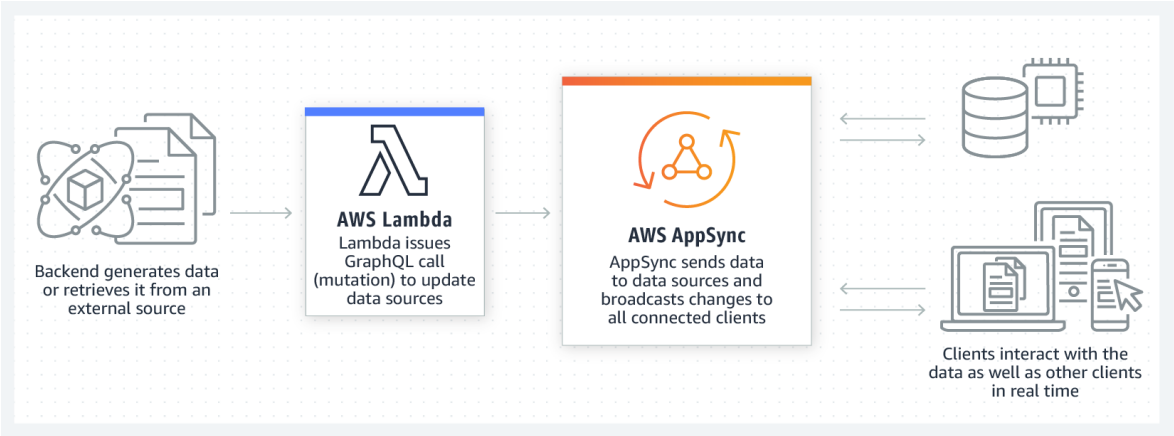


Figura 11 - Diagrama del funcionamiento de la lógica de datos

Capítulo 3. ESTADO DE LA CUESTIÓN

Este proyecto surge como una solución tecnológica a los problemas que suponen los modelos de computación con servidores en el ámbito de las aplicaciones móviles. Como objetivo general se tiene que apuntar la aplicación de la tecnología Cloud Computing en la creación de una aplicación para practicar exámenes de cara a la obtención de una titulación náutica de recreo.

Abordamos el estado de la cuestión en dos ángulos: por una parte, el panorama actual de las aplicaciones disponibles para la realización de exámenes preparatorios para la obtención de un título, licencia o certificación. Por otra parte, las tecnologías existentes en las que nos apoyaremos de cara al desarrollo de la aplicación objeto de este proyecto.

Aplicaciones de exámenes existentes:

Desde la aparición de los *Smartphones* en nuestra sociedad existe una creciente presencia de aplicaciones destinadas a resolver una gran variedad de problemas comunes. (Lamata, 2018). Existen aplicaciones de realización de exámenes para una amplia variedad de fines (carné de conducir, oposiciones, certificaciones profesionales) y con distintas modalidades (aplicaciones de ayuda a usuarios, aplicaciones administradas por centros de formación y aplicaciones que actúan como centro de formación *online* en sí mismas). En todo caso, apreciamos que tanto la oferta de aplicaciones como la demanda de estas es creciente y se aplica cada vez a más nichos concretos.

Tecnologías a utilizar:

El hecho de que haya cada vez más nichos de demanda que se deban cubrir con este tipo de aplicaciones (nuevos exámenes o certificaciones, necesidades de un centro de formación particular, etc.) será el que nos lleve a analizar las tecnologías existentes para desarrollar la

aplicación bajo criterios de eficiencia de prestaciones y economía de los servicios empleados.

Tradicionalmente en el desarrollo e implementación de aplicaciones web o móviles, se ha llevado a cabo una computación basada en servidores y muchas solicitudes HTTP hacia estos. Una aplicación se ejecuta en un servidor y el desarrollador es el encargado de administrar los recursos necesarios para ella (modelo tradicional "*On Premises*").

Esto conlleva una serie de problemas como, por ejemplo, el precio que hay que pagar por mantener el servicio activo cuando no se está atendiendo ninguna solicitud, o la escalabilidad que requiere el servidor a medida que el uso de la aplicación aumenta, o por el contrario disminuye. Este tipo de inconvenientes puede frenar el verdadero objetivo que tiene un desarrollador individual, que no es otro que construir y mantener la aplicación actual.

Por este motivo, y por muchos otros, el mundo de hoy en día se encuentra en una migración masiva a la nube. La introducción del Cloud Computing en nuestras vidas forma parte de la transformación digital que ya experimentamos e irá creciendo exponencialmente en los próximos años. Los problemas que antes suponían un mayor coste y esfuerzo se solucionan con las nuevas e innovadoras tecnologías que surgen en todos los ámbitos de la vida, y especialmente en el sector de las telecomunicaciones.

Hasta el momento, se conocen varios modelos de *Cloud Computing*. Por norma general, se distingue entre SaaS (Software as a Service o Software como Servicio), PaaS (Platform as a Service o Plataforma como Servicio) e IaaS (Infrastructure as a Service o Infraestructura como servicio).

En el mundo del desarrollo de aplicaciones también aparece la migración a la nube, y ante todos los inconvenientes que acarrea el uso de servidores de computación, aparece el concepto de computación sin servidor, o *serverless*.

En este modelo de ejecución, el proveedor de la capa de computación en la nube (AWS, Microsoft Azure...) es quien permite ejecutar durante un periodo de tiempo porciones de código denominadas "funciones", evitando que sea el desarrollador quien se encargue de la infraestructura subyacente que se provisiona para dar servicio. El objetivo es convertir la computación en algo transparente. El modelo de computación *serverless* ha adquirido el nombre de FaaS. (Borillo, 2019)

Como se va a trabajar con el proveedor de computación en la nube Amazon Web Services (AWS), es necesario introducirse en el mundo de las funciones Lambda.

Con estas funciones se puede ejecutar código para casi cualquier tipo de aplicación o servicio back-end sin tener que realizar tareas de administración. Simplemente requiere que el desarrollador cargue código y Lambda asigna de manera automática y precisa la potencia de ejecución de cómputo y ejecuta el código en función de la solicitud o el evento entrante para cualquier escala de tráfico. (Becerro, 2017)

Con el Marco de AWS Amplify y diversos servicios de la plataforma como API Gateway, se podrán poner en marcha estas funciones para que se encarguen de la lógica de la aplicación.

También se va a utilizar el servicio AppSync de AWS para facilitar el desarrollo de GraphQL APIs. El lenguaje GraphQL permite un rápido acceso a los datos a través de *queries* y *mutations.* Con este servicio se solucionan múltiples problemas que conlleva el uso de peticiones HTTP, y hace que el cliente pueda solicitar exactamente los datos que desea.

A la hora de iniciar el desarrollo de este proyecto, se ha investigado sobre otras soluciones al problema que se trata ya existentes en el mercado. Son algunas las aplicaciones basadas en exámenes tipo test que aparecen en el ámbito náutico, como ejemplo "Test Patrón". Se trata de una aplicación destinada a la preparación de alumnos para los diferentes títulos náuticos existentes y también a la gestión de las convocatorias de exámenes. El usuario debe seleccionar el lugar y la convocatoria que desea preparar.

Este proyecto se centra exclusivamente en la preparación de los exámenes. No hay ningún tipo de acuerdo con los examinadores para ofrecer la gestión de convocatorias. Sin embargo, existe la posibilidad de vender el producto a diferentes escuelas náuticas para mejorar su oferta académica. Aparece un apartado de estadísticas, que facilita al cliente la información necesaria para saber si está preparado para presentarse a una convocatoria de examen, o bien para conocer cuáles son los temas en los que debe focalizar un esfuerzo mayor.

Una ventaja con respecto a "Test Patrón" es la posibilidad de llevar a cabo exámenes con las preguntas ya falladas con el cliente, para poder repasar aquellos conceptos que todavía no han sido adquiridos a la perfección. También se incluye un nuevo concepto de prueba en el que aparecen preguntas ilimitadas de una titulación, y que sirven al usuario como práctica sin la presión de tener que obtener un número de preguntas acertadas o cumplir con tiempo especificado. En otras palabras, se busca que con este modelo de examen "ilimitado", el usuario pueda utilizar la aplicación de manera sencilla en el transcurso de sus rutinas diarias.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

En este apartado se va a tratar la justificación de este proyecto.

Hoy en día casi el 90% de las acciones que llevamos a cabo en el día están respaldadas por la tecnología. Con el paso de los años, se ha ido implantando en nuestra vida cotidiana hasta llegar a un punto en el que resulta complicado realizar algunas tareas o trabajos sin tecnología.

Además, los móviles han cobrado una gran importancia en nuestra vida y con ellos, han llegado las Apps móviles, que nos ayudan a tener cualquier servicio a mano sin necesidad de salir de casa. (Diseño web illusion Studio, 2020)

Es por ello por lo que tiene sentido introducirse en el mundo del desarrollo de aplicaciones móviles.

4.1.1 EL MERCADO NÁUTICO NO ESTÁ EXPLOTADO

Son pocas las aplicaciones móviles destinadas a la realización de exámenes náuticos tipo test, con el objetivo de preparar al alumno para obtener alguno de los títulos náuticos de recreo existentes. Si bien existen bastantes aplicaciones náuticas destinadas a la navegación, el número de aplicaciones didácticas es escaso.

Este proyecto busca ofrecer características distintas a su competencia de cara a explotar esta oportunidad de mercado. Ahí aparecen nuevas ideas como dotar de distintos nuevos

modelos de exámenes, presentar un informe estadístico al cliente sobre su desempeño, o dar la posibilidad de comprar el acceso Premium mediante Apple Store o Google Play.

4.1.2 COMPATIBILIDAD CON AMBAS PLATAFORMAS

Una de las ventajas que propone la aplicación a desarrollar es el hecho de desarrollar un servicio para todo tipo de alumnos. Ya que la oferta de aplicaciones para la preparación de títulos náuticos desde un dispositivo móvil es escasa, se entiende este proyecto como una necesidad actual de la sociedad, y se va a desarrollar con el objetivo de buscar un alcance de gran magnitud hacia dispositivos con ambos sistemas operativos, IOS y Android.

4.1.3 NECESIDAD DE MIGRAR APLICACIONES A LA NUBE

Existe una clara ventaja al desarrollar una aplicación móvil en la nube. Esta aplicación posee un rendimiento y una capacidad mucho mayor a otras aplicaciones que no se encuentran conectadas a servicios Cloud.

La conexión con la nube supone para la *aplicación* dotarse de un valor añadido muy importante y, de esta forma, conseguir que los usuarios se sientan más satisfechos y atraídos por la misma, aumentando su uso y multiplicándose las descargas desde las tiendas de aplicaciones (App Store, Google Play...). Además, la nube permite aprovecharse del extraordinario poder viral que ostentan las redes sociales hoy en día, lo que supone aumentar enormemente la visibilidad de la APP, haciéndola mucho más popular y conocida en el duro y competitivo sector de las aplicaciones móviles. (dfo, 2018)

4.2 OBJETIVOS

El objetivo final de este proyecto es poner en producción una aplicación para la realización de exámenes náuticos que funcione correctamente y sea capaz de cumplir con las expectativas de aquellos usuarios que busquen una buena preparación para conseguir la titulación en cuestión.

Así, los objetivos intermedios del proyecto que buscan el continuo progreso y mejora de la aplicación son los siguientes:

4.2.1 Desarrollo del *front end* de la aplicación, es decir, la parte del software en la que aparece la interacción de los usuarios en una interfaz gráfica. Se utilizará el editor de texto *Visual Studio Code* y la tecnología *React Native*. Dentro de esta, se utilizará *JavaScript* y *HTML*.

4.2.2 Creación de un proyecto AWS Amplify mediante la cadena de herramientas Amplify Command Line Interface (CLI) que permitirá administrar los servicios en la nube de AWS y crear el *back end*. Además, conectar el *front end* al proyecto de Amplify.

4.2.3 Llevar a cabo la autenticación de usuarios en nuestra aplicación mediante el servicio AWS Cognito. Implementar *Sign-Up* y *Sign-In*, los usuarios quedarán registrados en un pool de Cognito, que servirá de base de datos de usuarios.

4.2.4 Introducir un Interfaz de Programación de Aplicaciones (API). En especial se usará un *API GraphQL*, lenguaje que permitirá, junto con el servicio AppSync, poder consultar y manipular los datos referentes a los exámenes situados en una base de datos en DynamoDB. Se tomarán preguntas al azar de cada uno de los temas de la titulación específicos y se llevarán al *front end* para la realización de un examen.

4.2.5 Llevar a cabo un análisis de las *stats* de los usuarios registrados en la aplicación. Se estudiarán los porcentajes de los exámenes aprobados y suspendidos, así como el de las preguntas falladas.

4.3 METODOLOGÍA

Desde el momento de la asignación del proyecto, a principios de septiembre de 2020, se ha seguido un modo de desarrollo parecido a lo que sería una metodología Agile. Se han convocado reuniones con el director del proyecto para definir los próximos pasos a realizar, y se han convocado fechas (“sprints”) para entregar lo definido en las reuniones y comprobar errores tanto en el desarrollo como en el punto de vista al afrontar los diferentes desafíos de cada fase de la aplicación.

Este proyecto ha requerido un periodo de investigación y aprendizaje de la plataforma móvil para aplicaciones nativas *React Native*, así como para la plataforma de desarrollo en la nube AWS. Durante estos periodos, se ha llevado a cabo una búsqueda masiva de todo tipo de información, como tutoriales en YouTube o documentos de aprendizaje del propio entorno de Amazon Web Services.

Podemos dividir la aplicación móvil en dos partes: el desarrollo del *front end*, el cual ha supuesto la primera tarea llevada a cabo, entre octubre y diciembre de 2020, y la configuración del *back end***,** proceso que comienza a partir de enero de 2021.

Conectar ambas partes a la aplicación resulta sencillo gracias a AWS Amplify y el despliegue de la aplicación no supone un elevado periodo de tiempo.

En marzo de 2021, se comienza a escribir la Memoria del proyecto, cuya defensa ha de realizarse la segunda semana de julio de 2021.

A continuación, se muestra un cronograma en el que aparecen las distintas fases a realizar y su estimado período de desarrollo.

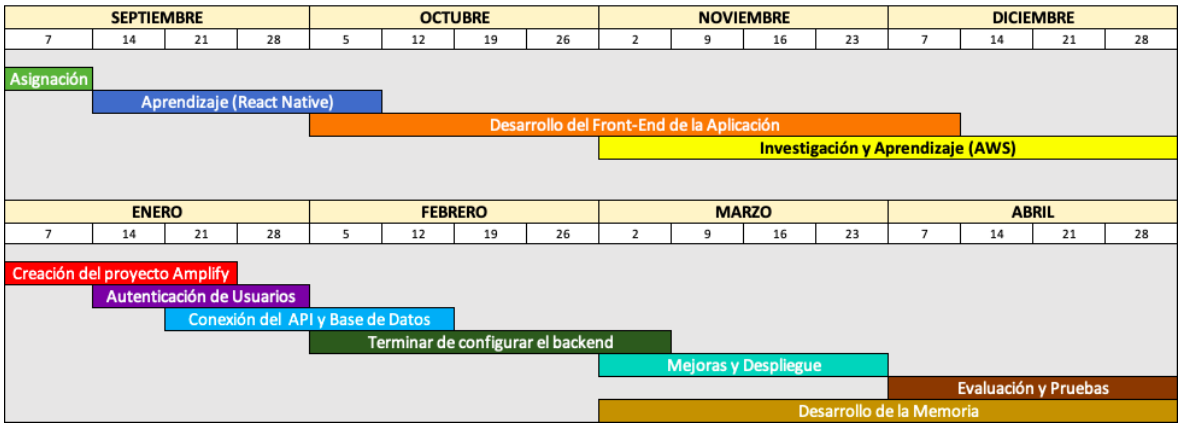


Figura 12 - Diagrama de Gantt

Como se puede comprobar, desde la asignación del proyecto se ha trabajado en el mismo de una manera constante, tanto desarrollando como investigando y aprendiendo a utilizar los recursos que se van a utilizar y que aparecen en el siguiente apartado de esta memoria.

Por último, como se ha mencionado antes y aunque no aparezca en el diagrama, durante los meses de mayo y junio, se va a seguir preparando la memoria del proyecto, y es en julio cuando se expondrá la defensa de este ante el tribunal.

Capítulo 5. SISTEMA DESARROLLADO

El desarrollo de esta aplicación está dividido en tres fases muy diferenciadas, como son la programación de la interfaz gráfica, la implementación de la parte lógica, y la subida de la aplicación a las tiendas en sus respectivas plataformas (iOS & Android).

5.1 ANÁLISIS DEL SISTEMA

Antes de comenzar a diseñar la parte lógica de la aplicación, es necesario llevar a cabo un análisis de las fronteras del sistema. En una temprana etapa de este análisis se puede ser bastante ambicioso, aunque es necesario delimitar bien estas fronteras para así poder cumplir con los objetivos mencionados en el apartado anterior.

Para cumplir con el análisis, se va a llevar a cabo un estudio sobre los modelos de contexto y los requisitos de la aplicación, así como el desarrollo de los casos de uso de esta.

- **Modelos de Contexto**

  Mediante el siguiente estudio se delimita el alcance del sistema.

  Modelo Arquitectónico

  Al tratarse de un sistema serverless, las funciones de este no recaen en un servidor que se comunica con el usuario, sino que es AWS quien con sus distintos servicios proporciona a la aplicación de todas sus funciones, que son las siguientes:

  - Gestión de Usuarios. Estos se guardan en un pool de Amazon Cognito, que proporciona autenticación, autorización y administración de usuarios para aplicaciones móviles y web.

Los usuarios pueden iniciar sesión directamente con una dirección de correo electrónico y una contraseña.

- Gestión de Exámenes. Creación de cualquier tipo de examen náutico de las titulaciones ofrecidas, además de su corrección y recogida en el apartado de estadísticas

- Gestión de Bases de Datos. Mediante AWS DynamoDB se almacena información sobre los usuarios y los exámenes.

- Gestión de Pagos. En caso de que un usuario desee obtener la subscripción Premium tendrá lugar un pago mediante tarjeta de crédito que convertirá la cuenta en cuestión en una cuenta sin ningún tipo de restricciones o bloqueos a la hora de realizar un examen.

Modelo de proceso empresarial

En este modelo se muestran las posibles relaciones del software con otras aplicaciones con el objetivo de optimizar el funcionamiento de este.

Se ha pensado en aplicaciones a largo plazo como puede ser Facebook**,** para una futura modificación a la hora de realizar el alta de los usuarios.

A muy corto plazo, se busca llegar a un acuerdo con diferentes autoescuelas náuticas para ofrecer este software como producto de enseñanza en su sistema. De esta forma, se puede optimizar la aplicación mediante el feedback que los propios alumnos que la usan, y podría llegar a existir la posibilidad de aumentar el número

de preguntas, o incluso las titulaciones ofrecidas, por ejemplo, incorporando al sistema el examen de moto náutica.

- **Requisitos**

Se desarrollan diferentes tablas de requisitos en función de los niveles en los que se divide la aplicación.

Requisito	Influencia
Compatibilidad de la aplicación con la plataforma iOS y la plataforma Android	React Native permite este modelo de aplicación, pudiendo volcar el mismo código nativo para todo tipo de dispositivos móviles

Tabla 10 - Influencia de los requisitos al nivel de presentación

Requisito	Influencia
Crear perfil de usuario con varias características generales	Amazon Cognito permite llevar a cabo el Sign-Up con todos los atributos necesarios
El Usuario normal tiene acceso a 2 exámenes diarios, mientras que el usuario Premium no tiene límites de exámenes	Existe una pantalla “Premium” para poder cambiar la subscripción
Tipos de Exámenes: Por temas, Completo, Examen de preguntas falladas, Examen de preguntas Ilimitadas	Se podrá acceder a cualquier tipo de examen desde la pantalla principal
Ver Estadísticas sobre los exámenes realizados	Se podrá acceder mediante la pantalla “Stats”, disponible en la pestaña inferior de la aplicación

Tabla 11 - Influencia de los requisitos al nivel de aplicación

Requisitos	Influencia
Traer preguntas de las respectivas bases de datos para mostrarlas en los exámenes	Las funciones Lambdas del API GraphQL lanzarán *queries*
Guardar los Usuarios con sus características	El pool de Cognito actúa como base de datos
Guardar los exámenes realizados	Se insertarán en una tabla DynamoDB

Tabla 12 - Influencia de los requisitos al nivel de base de datos

- **Diagrama Casos de Uso**

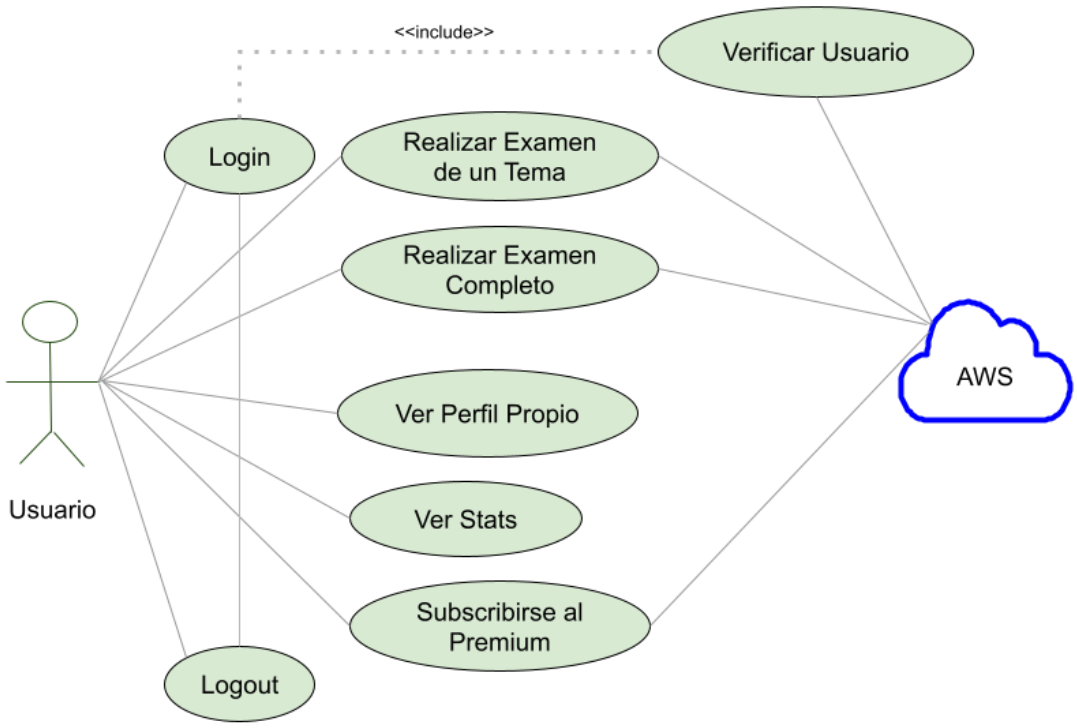


Figura 13 - Diagrama de casos de uso

5.2 DISEÑO DEL SISTEMA

Una vez realizado el análisis del punto anterior, es necesario analizar el diseño de la aplicación según las fases explicadas en la introducción de este apartado.

- **Diseño Arquitectura Front End**

  Como ya se ha explicado, la parte de la aplicación que compone el front end se trata del Interfaz de Usuario. Se diseña con el objetivo de cumplir con los requisitos y las restricciones del proyecto.

  Esta capa interacciona con el back end de la aplicación a través de los distintos servicios AWS. En el caso de la autenticación de los usuarios, se conecta con un pool de Cognito, y en el caso de la gestión de exámenes, se conecta con un API GraphQL.

  Se han utilizado las librerías que exige React Native, y se han añadido en forma de dependencias al proyecto. A continuación, se muestra el archivo "package.json", la parte donde se almacenan estas dependencias.

```
{
  "dependencies": {
    "@react-native-async-storage/async-storage": "^1.15.5",
    "@react-native-community/datetimepicker": "^3.5.2",
    "@react-native-community/netinfo": "^6.0.0",
    "@react-native-picker/picker": "^1.16.3",
    "@react-navigation/material-bottom-tabs": "^5.3.15",
    "@react-navigation/native": "^5.9.4",
    "@react-navigation/stack": "^5.14.5",
    "amazon-cognito-identity-js": "^5.0.3",
```

```
  "aws-amplify": "^4.1.2",
  "aws-amplify-react-native": "^5.0.2",
  "material-ui-community-icons": "^0.15.0",
  "radio-buttons-react-native": "^1.0.4",
  "react": "17.0.1",
  "react-devtools": "^4.13.5",
  "react-dom": "^17.0.2",
  "react-native": "0.64.2",
  "react-native-date-input": "^1.0.11",
  "react-native-date-picker": "^3.3.1",
  "react-native-datepicker": "^1.7.2",
  "react-native-elements": "^3.4.2",
  "react-native-gesture-handler": "^1.10.3",
  "react-native-paper": "^4.9.1",
  "react-native-payments": "^0.8.4",
  "react-native-picker-select": "^8.0.4",
  "react-native-safe-area-context": "^3.2.0",
  "react-native-screens": "^3.3.0",
  "react-native-vector-icons": "^8.1.0"
 },
}
```

Las vistas creadas mediante la programación en React Native son:

- **Main.** En esta vista se introduce un Stack Navigator de React Native, para poder navegar entre las diferentes pantallas de la aplicación.

- **Sign-In.** El usuario puede introducir su correo electrónico y contraseña para acceder a la aplicación.

- **Sign-Up.** El usuario puede crear una cuenta introduciendo una serie de datos personales.

- **Inicio.** Pantalla principal, donde se accede a cada uno de los exámenes ofrecidos. Además, aquí se encuentra un navegador en la parte inferior de la pantalla para navegar entre las vistas “Inicio”, “Stats”, “Perfil” y “Premium”.

- **Stats.** Aparecen las estadísticas de cada usuario en relación con sus exámenes llevados a cabo.

- **Perfil.** Aparecen los datos de cada usuario (correo, edad, créditos o tokens restantes, y tipo de subscripción).

- **Premium.** En esta vista se puede cambiar el tipo de subscripción de “normal” a “Premium”, llevando a cabo el pago correspondiente.

- **Examen por Temas.** Se accede a un examen de 10 preguntas con corrección.

- **Examen Completo.** Se accede a un examen oficial de la titulación seleccionada, con corrección.

- **Preguntas Ilimitadas.** Se accede a preguntas ilimitadas de la titulación seleccionada.

- **Preguntas Falladas.** Se accede a un examen compuesto por preguntas falladas por el usuario.

- **Diseño Arquitectura Back End**

En cuanto al diseño de la lógica, se crean las siguientes entidades a través del CLI de AWS Amplify:

- **AWS Cognito User Pool**, "testpatron4dbb9557_userpool_d4bb9557_dev", con verificación por correo electrónico.
- **API GraphQl,** "apigraphql", que contiene el siguiente esquema de datos:

Tipo 1 – Preguntas de la titulación PY

```
type PreguntasPy @model @key(name: "todosByTemaPy", fields: ["tema"],
    queryField: "todosByTemaPy")
@auth(rules: [{ allow: public, provider: iam }])
{
  id: ID!
  tema: String!
  question: String!
  ans1: String!
  ans2: String!
  ans3: String!
  ans4: String!
  correct: String!
  link: String!
}
```

Tipo 2 – Preguntas de la titulación Per y Pnb

```
type PreguntasPerPnb @model @key(name: "todosByTemaPer", fields:
    ["tema"], queryField: "todosByTemaPer")
@auth(rules: [{ allow: public, provider: iam }])
{
  id: ID!
  tema: String!
  question: String!
  ans1: String!
```

```
  ans2: String!
  ans3: String!
  ans4: String!
  correct: String!
}
```

Tipo 3 – Examen

```
type Examen @model @key(name: "examenByUser", fields: ["user"],
    queryField: "examenByUser")
@auth(rules: [{ allow: public, provider: iam }])
{
  id: String!
  preguntas: [String!]
  user: String!
  titulacion: String!
  status: String!
}
```

Tipo 4 – Usuario

```
type Usuario @model @key(name: "todosByEmail", fields: ["email"],
    queryField: "todosByEmail")
@auth(rules: [{ allow: public, provider: iam }])
{
  id: ID!
  email: String!
  subscripcion: String!
  token: Int!
  falladas: [String!]
}
```

Este esquema de datos hace que AWS cree automáticamente cuatro bases de datos en DynamoDB, y en cada una de ella se guardan como objetos los cuatro tipos definidos, con sus respectivos atributos. Por poner un ejemplo, se muestra un elemento de cada tabla.

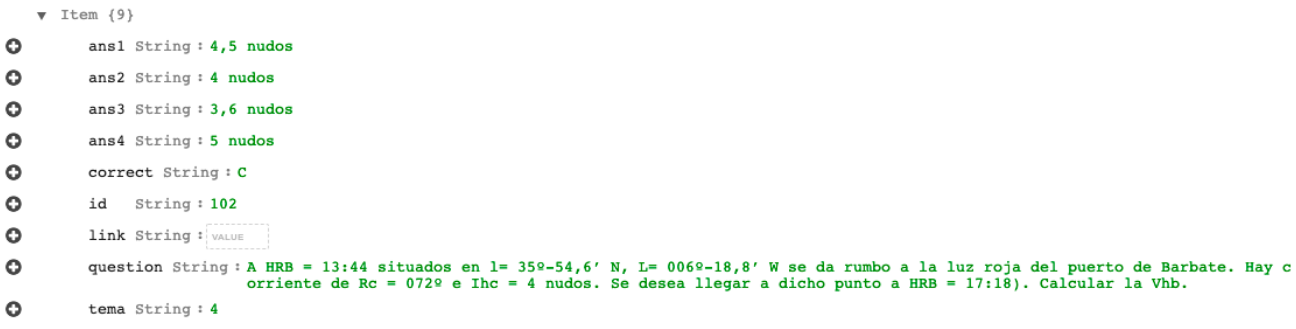


Figura 14 - Elemento de la tabla "PreguntasPy"

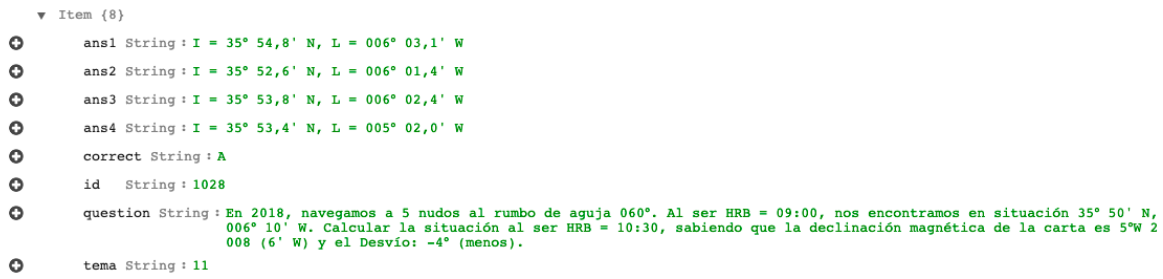


Figura 15 - Elemento de la tabla "PreguntasPerPnb"

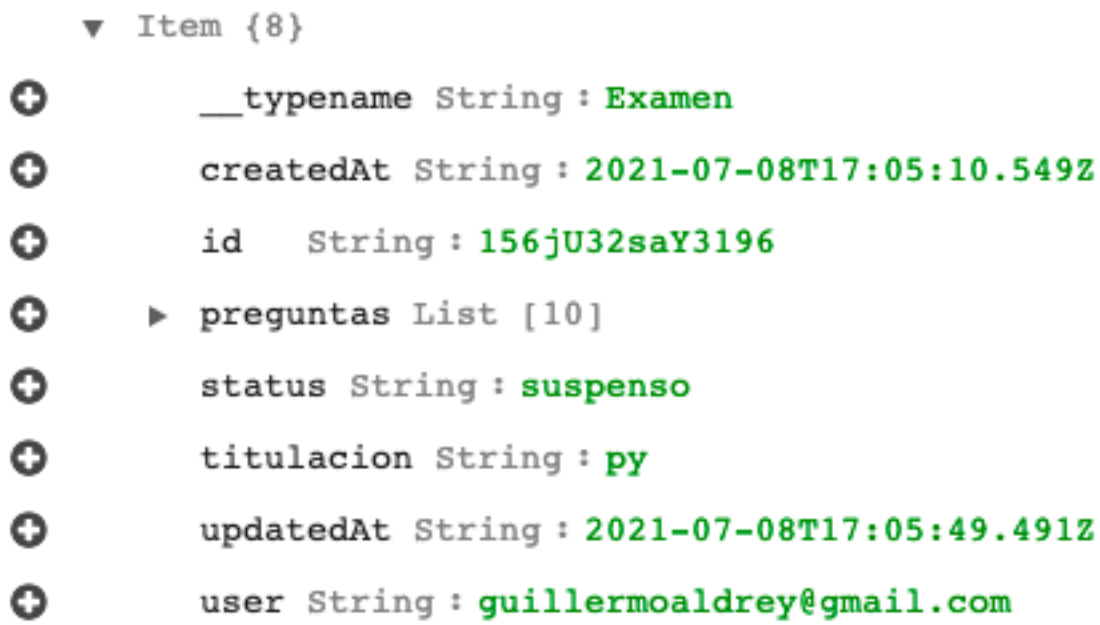


Figura 16 - Elemento de la tabla "Examen"

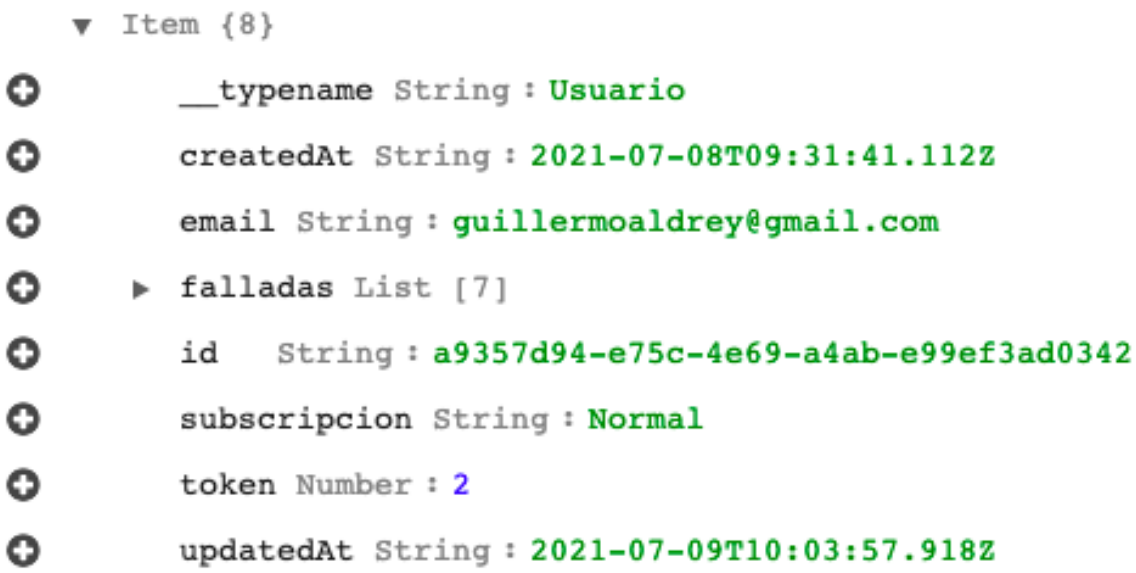


Figura 17 - Elemento de la tabla "Usuario"

Capítulo 6. ANÁLISIS DE RESULTADOS

En este capítulo se va a hacer hincapié en el resultado final de la aplicación, su funcionamiento y en si se han conseguido o no cumplir los objetivos propuestos en la etapa de análisis.

- Se ha conseguido desarrollar un interfaz de usuario que funciona para los dos sistemas operativos iOS y Android utilizando React Native con el editor de texto Visual Studios Code.

  Para ello, se han utilizado simuladores de sendas plataformas, con el objetivo de observar cómo quedan los componentes integrados al mismo tiempo que se programa.

  Para instalar Xcode y Android Studios, junto con sus simuladores, se han seguido los tutoriales que estos programas incluyen en sus respectivas páginas web.

  El resultado es la correcta construcción de la aplicación en los dos simuladores, con el resultado que se muestra a continuación:

```
(base) guillermoaldreypastor@Guillermos-Air TestPatron % npx react-native run-ios
info Found Xcode workspace "TestPatron.xcworkspace"
info Launching iPhone 12 (iOS 14.4)
info Building (using "xcodebuild -workspace TestPatron.xcworkspace -configuration Debug -scheme Tes
tPatron -destination id=69022C4A-7B33-459D-BB9D-07AAABAC57D9")
success Successfully built the app
info Installing "/Users/guillermoaldreypastor/Library/Developer/Xcode/DerivedData/TestPatron-eujmik
mxawlcpncdadhqduuuwtef/Build/Products/Debug-iphonesimulator/TestPatron.app"
info Launching "org.reactjs.native.example.TestPatron"
success Successfully launched the app on the simulator
```

Figura 18 - Simulación de la aplicación en iOS

```
(base) guillermoaldreypastor@Guillermos-Air TestPatron % npx react-native run-android
info Running jetifier to migrate libraries to AndroidX. You can disable it using "--no-jetifier
" flag.
Jetifier found 1232 file(s) to forward-jetify. Using 4 workers...
info JS server already running.
info Installing the app...

> Task :app:compileDebugJavaWithJavac

> Task :app:installDebug
Installing APK 'app-debug.apk' on 'Pixel_3a_XL_API_29(AVD) - 10' for app:debug
Installed on 1 device.

BUILD SUCCESSFUL in 1m 28s
233 actionable tasks: 14 executed, 219 up-to-date
info Connecting to the development server...
8081
info Starting the app on "emulator-5554"...
Starting: Intent { cmp=com.testpatron/.MainActivity }
```

Figura 19 - Simulación de la aplicación en Android

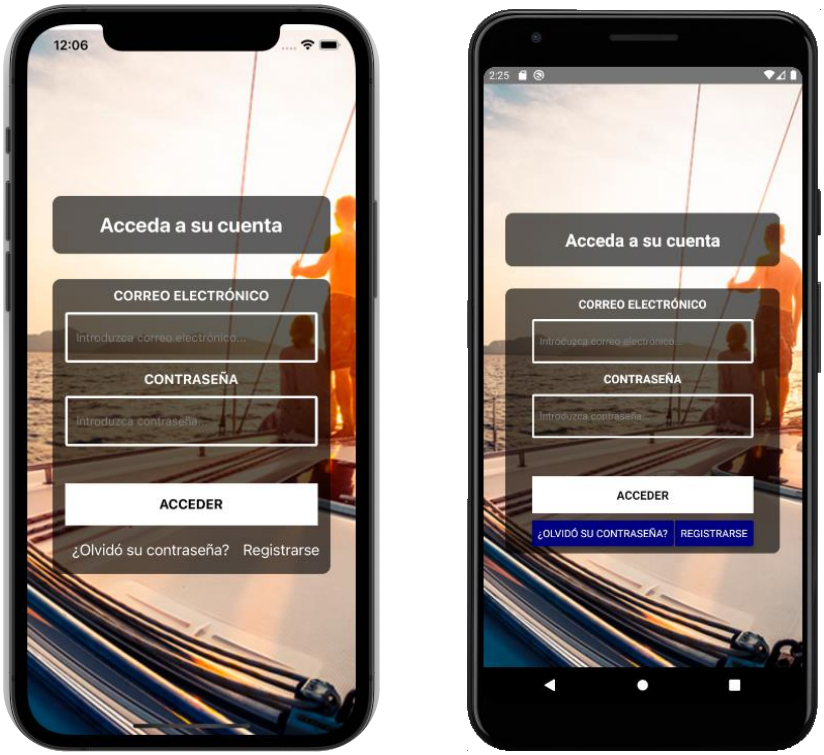


Figura 20 - Simuladores lanzando la aplicación (izquierda: iOS, derecha: Android)

- Se ha conseguido crear un proyecto en AWS Amplify donde alojar al código nativo y a su vez introducir recursos AWS para la lógica de la aplicación.

  La forma de trabajar con Amplify ha requerido el uso del Amplify CLI, utilizando el comando *amplify <comando>* para llevar a cabo las configuraciones en la nube.

  Por ejemplo, para añadir un API al proyecto Amplify, se introduce en el terminal de Visual Studio Code: **amplify add api**

  De esta manera se configuran todos los elementos pertenecientes al back end de una forma rápida y segura, ya que con el comando **amplify push** todo se guarda en la nube de AWS en cuestión de segundos

```
(base) guillermoaldreypastor@Guillermos-Air TestPatron % amplify help

amplify <command> <subcommand>

  init                Initializes a new project, sets up deployment resources in the cloud, and mak
es your project ready for Amplify.
  configure           Configures the attributes of your project for amplify-cli, such as switching
front-end framework and adding/removing cloud-provider plugins.
  push                Provisions cloud resources with the latest local developments.

  pull                Fetch upstream backend environment definition changes from the cloud and upda
tes the local environment to match that definition.
  publish             Executes amplify push, and then builds and publishes client-side application
for hosting.
  serve               Executes amplify push, and then executes the project's start command to test
run the client-side application locally.
  status              Shows the state of local resources not yet pushed to the cloud (Create/Update
/Delete).
  delete              Deletes all of the resources tied to the project from the cloud.

  <category> add      Adds a resource for an Amplify category in your local backend

  <category> update   Update resource for an Amplify category in your local backend.

  <category> push     Provisions all cloud resources in a category with the latest local developmen
ts.
  <category> remove   Removes a resource for an Amplify category in your local backend.

  <category>          Displays subcommands of the specified Amplify category.

  mock                Run mock server for testing categories locally.

  codegen             Generates GraphQL statements(queries, mutations and eventHandlers) and type a
nnotations.
  env                 Displays and manages environment related information for your Amplify project
.
  console             Opens the web console for the selected cloud resource.
```

Figura 21 - Comandos del Amplify CLI

- Se han cumplido a la perfección los objetivos relacionados con añadir los distintos recursos que necesita la aplicación para cumplir con su funcionalidad. Son los siguientes:

    - Autenticación de usuarios. Creación de Sign-Up y Sign-In y registro en un pool de Cognito

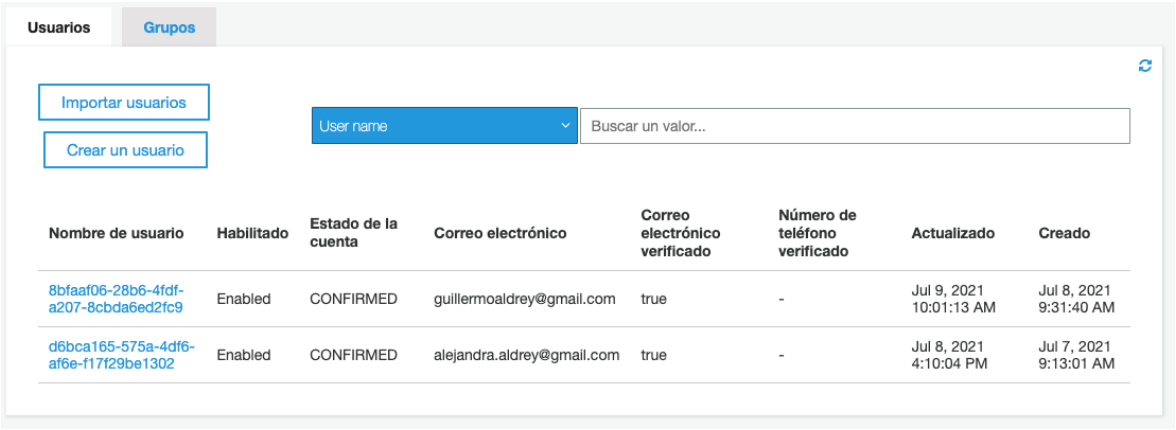


Figura 22 - User Pool de Cognito

    - Creación de las distintas bases de datos que requiere la lógica de la aplicación

Figura 23 - Bases de Datos utilizadas

- Creación de un API GraphQL para ejecutar llamadas a las distintas bases de datos (*queries* y *mutations)*

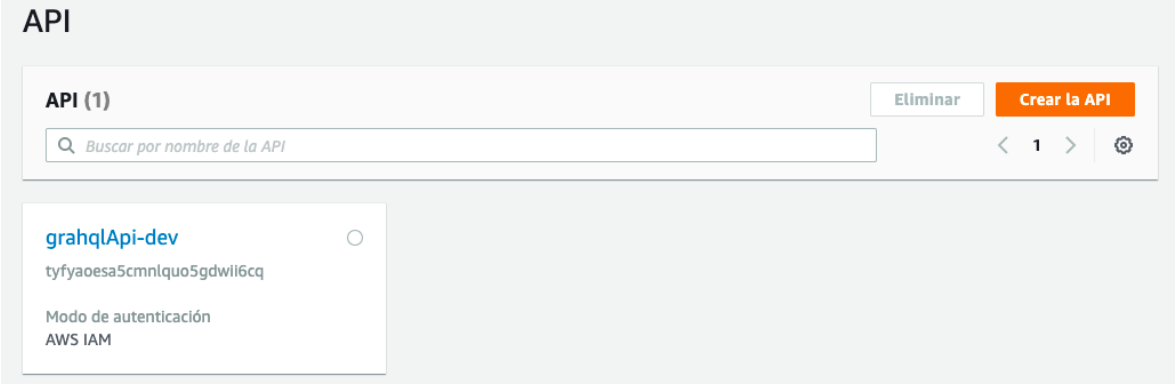


Figura 24 - API GraphQl almacenada en AppSync

Ahora gracias a la consola de AWS AppSync se puede comprobar que las llamadas a las bases de datos funcionan correctamente, y por lo tanto el API, y la lógica de la aplicación actúan como es debido.

Se van a probar dos consultas. La primera consulta hace referencia a la query “getPreguntasPy” y devuelve una pregunta con todos sus atributos de la tabla “PreguntasPY”, filtrando por su id.

La segunda consulta, hace referencia a la query “todosByTemaPer”, y devuelve 5 id de preguntas al azar de la tabla “PreguntasPer”

Figura 25 - Prueba de query "getPreguntasPy"

Figura 26 - Prueba de la query "todosByTemaPer"

- Por último, se ha conseguido llevar a cabo una recogida de las estadísticas de cada usuario en relación con los exámenes realizados, para mostrarlas en una screen "Stats". Se busca que el usuario obtenga información sobre su desempeño y su nivel de conocimientos actual.

  Se ha logrado ofrecer como información estadística:

  - Número de exámenes realizados por el usuario
  - Número de exámenes aprobados con su porcentaje
  - Número de exámenes suspendidos con su porcentaje
  - Número de exámenes realizados de cada titulación
  - Número de preguntas falladas
  - Número de preguntas falladas de cada titulación

Para comprobar el resultado no solo de las estadísticas mencionadas anteriormente, sino también del resto de objetivos conseguidos, se pueden observar los pantallazos de la aplicación en el Anexo II de esta memoria, en la página 83. Los pantallazos o wireframes se corresponden con las figuras desde la 24 hasta la 38.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1 TRABAJO

Durante el análisis del proyecto se decidió sumergirse en una aplicación *serverless* y dejar a un lado las instancias de EC2 que se pensaron en un primer momento. Todas las ventajas comentadas en esta memoria sobre FaaS condujeron a una rápida visión de lo que podría mejorar la aplicación utilizando funciones Lambda.

Asimismo, a la hora de llevar a cabo las llamadas a la base de datos, en una primera toma de contacto con AWS Lambda y AWS Api Gateway, se comenzó a desarrollar la lógica utilizando un API REST con métodos HTTP. Más tarde, se comprobó que utilizar API GraphQL simplificaba el trabajo de desarrollo y el acceso a los datos era mucho mas sencillo.

También se decidió en primer momento utilizar Stripe para los pagos de la subscripción Premium. Se trata de un software que permite a los usuarios de una aplicación realizar y recibir pagos por Internet. Una vez más, se cambió de opinión al respecto, pasando a usar directamente las pasarelas de pagos de Google Play y de App Store, y así trabajar directamente con las plataformas en las que se va a subir la aplicación.

Por último, la decisión de utilizar AWS como proveedor en la nube ha resultado ser todo un acierto. Configurar los servicios del back-end ha sido tarea sencilla comparado con otros proveedores, y, además, al tener la propiedad de ser auto-escalable, permite potenciar la aplicación conectándola con FaaS. No es lo mismo tener que lidiar con 100 usuarios, que cubrir el área geográfica de un propio país con millones de ellos.

7.2 PRÓXIMOS PASOS

Como trabajos fututos queda pendiente la subida de la aplicación a las plataformas de Google Play y de App Store. Se ha intentado llevar a cabo, pero no ha sido posible debido a varios problemas relacionados con la programación de las pasarelas de pagos necesarias para que los clientes puedan comprar la subscripción Premium.

El próximo paso sería configurar estas pasarelas y definir la subscripción que queremos (será de un periodo de un año). Cuando las pasarelas funcionen a la perfección, se podrá subir la aplicación nativa a las cuentas ya creadas de Google Play Console y Apple Developer. Con esto, se pondrá el proyecto en el mercado, y los usuarios tendrán vía libre para descargar y usar la aplicación

7.3 CONCLUSIÓN FINAL

Pasar a través de todas las fases de un desarrollo software, desde el análisis de requisitos y funcionalidad, hasta el futuro lanzamiento de la aplicación en las plataformas del mercado, es un proceso muy gratificante.

En cuanto a las plataformas Xcode y Android Studio, ambas tienen ciertas limitaciones y exponen al desarrollador a lidiar con algunos inconvenientes.

Xcode tiene una mejor experiencia de usuario, ya que, a diferencia de Android, tiene una completa descripción de las librerías necesarias.

Android, sin embargo, presenta facilidades a la hora de publicar la aplicación, un proceso de mayor dificultad con iOS.

Finalmente, considero de gran acierto mi decisión de afrontar un proyecto de este estilo debido al conocimiento que he podido adquirir sobre varias tecnologías y sobre la nube. Considero que la sociedad demanda millones de soluciones a los problemas cotidianos, y con este proyecto me he dado cuenta de la existente necesidad de profesionales capaces de ofrecer mejoras requeridas por toda la población. Esta aplicación puede suponer un paso en esa dirección.

Capítulo 8. BIBLIOGRAFÍA

[1] *Amplify Framework Documentation*. (s.f.). Obtenido de https://docs.amplify.aws/

[2] Becerro, R. G. (2017). *Paradigma*. Obtenido de https://www.paradigmadigital.com/dev/aws-lambda-arquitectura-serverless-implementar-apis/

[3] Borillo, R. (30 de Marzo de 2019). *Genbeta*. Obtenido de https://www.genbeta.com/desarrollo/que-serverless-que-adoptarlo-desarrollo-tu-proxima-aplicación

[4] dfo. (7 de Septiembre de 2018). *Deusto Formation*. Obtenido de https://www.deustoformacion.com/blog/programacion-tic/para-que-sirve-conectar-aplicacion-movil-con-nube

[5] Diseño web illusion Studio. (12 de Febrero de 2020). *Illusionstudio*. Obtenido de https://www.illusionstudio.es/apps-moviles-importancia-actualidad

[6] Lamata, E. (2 de Marzo de 2018). *elEconomista*. Obtenido de https://www.eleconomista.es/emprendedores-innova/noticias/8976705/03/18/Aplicaciones-moviles-para-cumplir-propositos-un-negocio-en-auge.html

[7] Lozano, V. (17 de Octubre de 2019). *Qué es y hacia dónde va el cloud computing. FHIOS Consultoría Estratégica.* . Obtenido de https://www.fhios.es/cloud-computing-introduccion/

[8] Martín, Á. J. (18 de Junio de 2019). *OpenWebinars*. Obtenido de https://openwebinars.net/blog/react-native-que-es-para-que-sirve/

[9] Naciones Unidas. (s.f.). *Naciones Unidas*. Obtenido de La Agenda para el Desarrollo Sostenible: https://www.un.org/sustainabledevelopment/es/development-agenda/

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

En este anexo de la memoria se presenta el alineamiento de este proyecto con los Objetivos de Desarrollo Sostenible (ODS).

Introducción

Los Objetivos de Desarrollo Sostenible (ODS) u Objetivos Mundiales, surgen en la Conferencia de las Naciones Unidas sobre el Desarrollo Sostenible que se celebró en el año 2012 en la ciudad de Río de Janeiro, Brasil.

Se pusieron en marcha en enero de 2016 y orientarán las políticas y la financiación del Programa de las Naciones Unidas para el Desarrollo (PNUD) durante los próximos 15 años.

Estos objetivos mundiales constituyen un llamamiento universal a la acción para poner fin a la pobreza, proteger el planeta y mejorar las vidas y las perspectivas de las personas en todo el mundo (Naciones Unidas, s.f.)

La colaboración entre Naciones Unidas y los gobiernos para conseguir la integración de los ODS en los planes y políticas nacionales de desarrollo da lugar a la creación de Agenda 30. Se trata de una herramienta dedicada al Desarrollo Sostenible en busca de promover la prosperidad y el bienestar de todas las personas, así como el medioambiente y la economía global.

Objetivos de Desarrollo Sostenible

Los Objetivos de desarrollo sostenible (ODS) de las Naciones Unidas son:

- **ODS 01**: Fin de la pobreza
- **ODS 02**: Hambre cero
- **ODS 03**: Salud y bienestar
- **ODS 04**: Educación de calidad
- **ODS 05**: Igualdad de genero
- **ODS 06**: Agua limpia y saneamiento
- **ODS 07**: Energía asequible y no contaminante
- **ODS 08**: Trabajo decente y crecimiento económico
- **ODS 09**: Industria, innovación infraestructura
- **ODS 10**: Reducción de desigualdades
- **ODS 11**: Ciudades y comunidades sostenibles
- **ODS 12**: Producción y consumo responsables
- **ODS 13**: Acción por el clima
- **ODS 14**: Vida submarina
- **ODS 15**: Vida de ecosistemas terrestres
- **ODS 16**: Paz, justicia e Instituciones sólidas
- **ODS 17**: Alianzas para lograr los objetivos

Figura 27 - Objetivos de Desarrollo Sostenible (ODS)

Alineación del Proyecto

Los Objetivos de Desarrollo Sostenible ODS establecen unas metas claras que alcanzar antes del 2030 a cumplir por parte de organizaciones públicas y privadas. El objetivo es claro, erradicar el hambre en el mundo a la vez que se alcanza la constitución de sociedades sostenibles.

Este Proyecto sigue la línea de la que habla las Naciones Unidas, pues cumple con algunos de los objetivos descritos.

Por una parte, todas aquellas personas que han obtenido un título náutico, o se han preparado para ello, conocen a la perfección una parte de la vida submarina, y como cuidar los mares para no perjudicar su vida e integridad. Es por ello por lo que este proyecto ofrece los conocimientos necesarios para cuidar el medio ambiente marino, ya que los usuarios responderán a multitud de preguntas sobre el cuidado y buenas prácticas en el mar. Esto tiene que ver con los principios 13 y 14.

El objetivo de conseguir una educación de calidad en todos los ámbitos permite garantizar el desarrollo sostenible. Una aplicación como la desarrollada, busca presentar al usuario la posibilidad de adquirir la mejor educación posible en el ámbito tratado, como es en este caso la obtención de un permiso náutico. Esto se corresponde con el principio 4.

Por último, se apoya a la innovación en la industria del desarrollo de software y de las aplicaciones móviles. Se intenta llevar a cabo buenas prácticas que faciliten el trabajo del desarrollador y que mejoren la experiencia de usuario, buscando el progreso y el bienestar de la sociedad. Esto cumple con el principio número 9.

ANEXO II: WIREFRAMES

Sign – Up

Figura 28 - Wireframe I. Sign-Up

Sign – In

Figura 29 - Wireframe II. Sign-In

Restaurar Contraseña

Figura 30 - Wireframe III. Restaurar Contraseña

Confirmar Registro

Figura 31 . Wireframe IV. Confirmar Registro

Pantalla Principal

Figura 32 - Wireframe V. Pantalla Principal

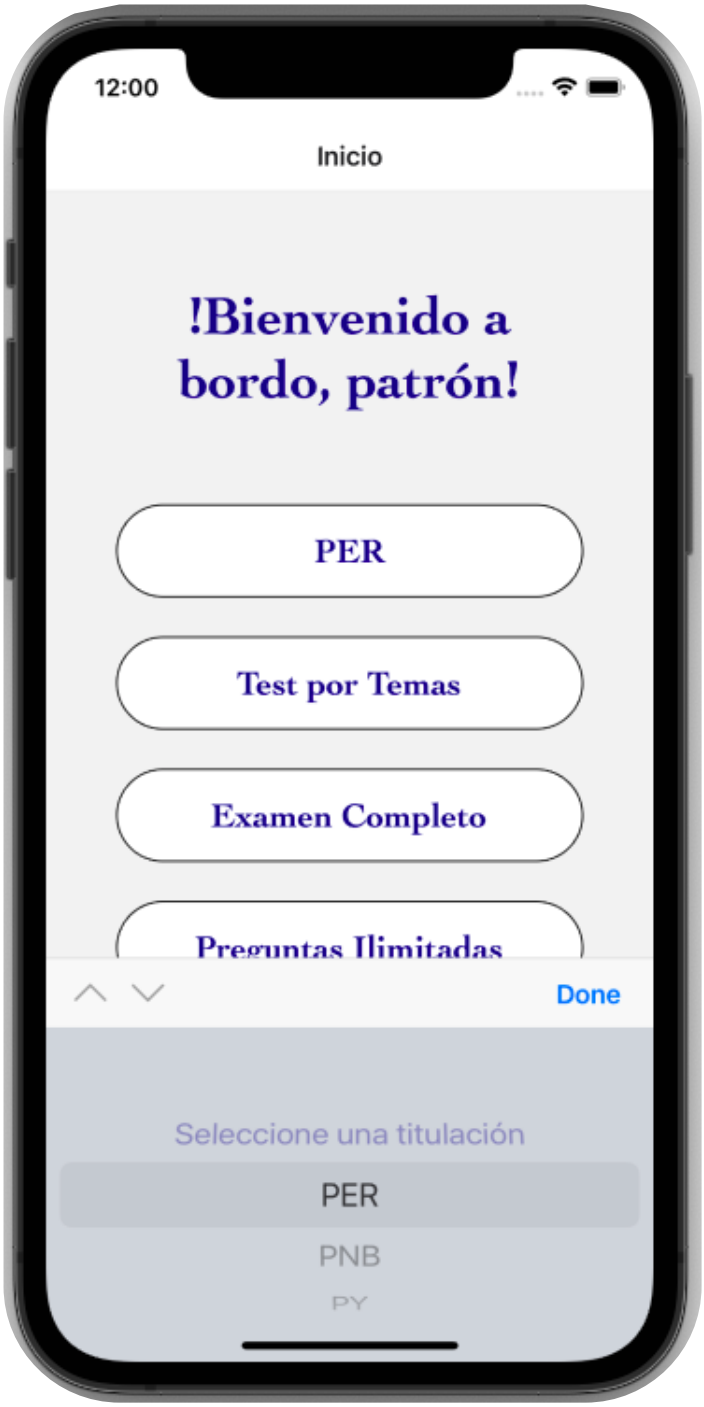


Figura 33 - Wireframe VI. Pantalla Principal, muestra del selector

Perfil

Figura 34 - Wireframe VII. Pantalla "Perfil"

Stats

Figura 35 – Wireframe VIII. Pantalla "Stats"

Premium

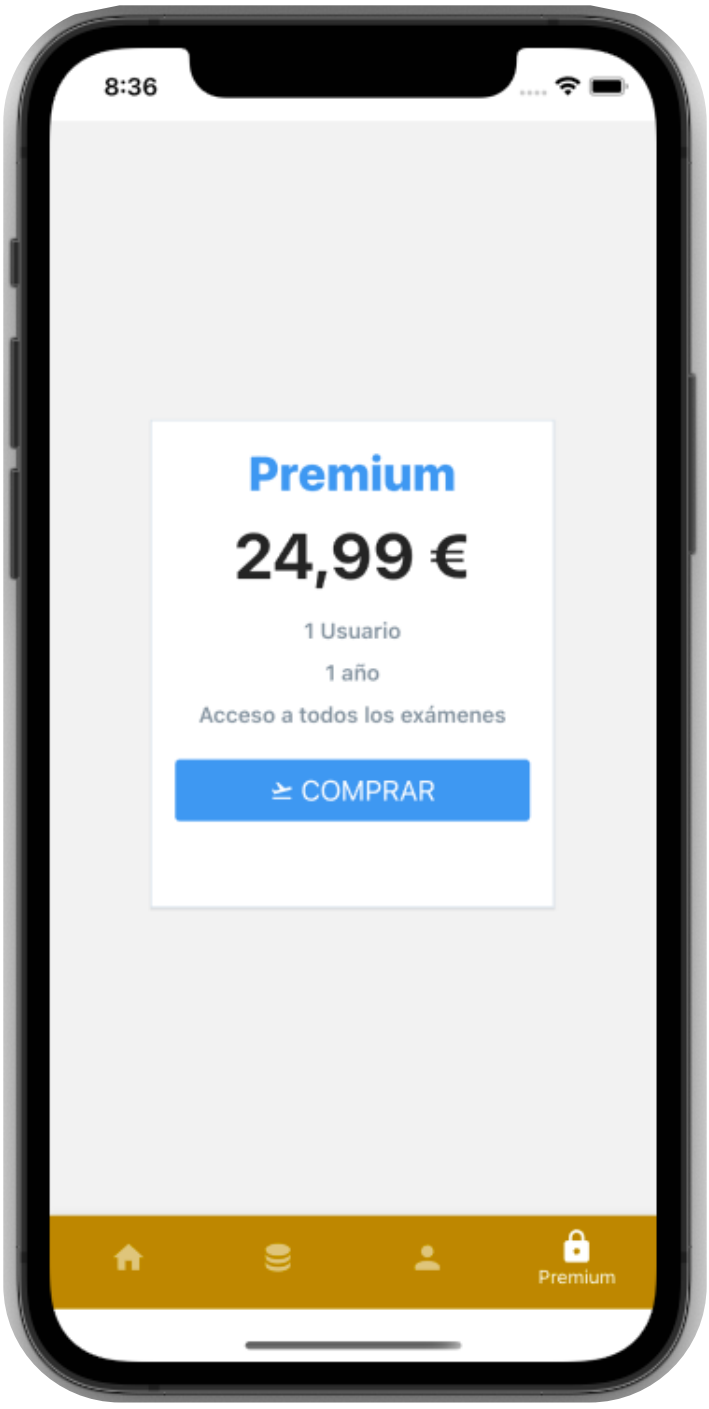


Figura 36 - Wireframe IX. Pantalla "Premium"

Examen por temas

Figura 37 - Wireframe X. Examen "Por Temas"

Figura 38 - Wireframe XI. Pantalla Examen, modo "Por Temas"

Figura 39 - Wireframe XI. Terminar examen "Por Temas" antes de tiempo

Figura 40 - Wireframe XII. Posibilidad de no corregir el examen

Pantalla de resultado

Figura 41 - Wireframe XIII. Resultado "Por Temas"

Examen de Preguntas Falladas

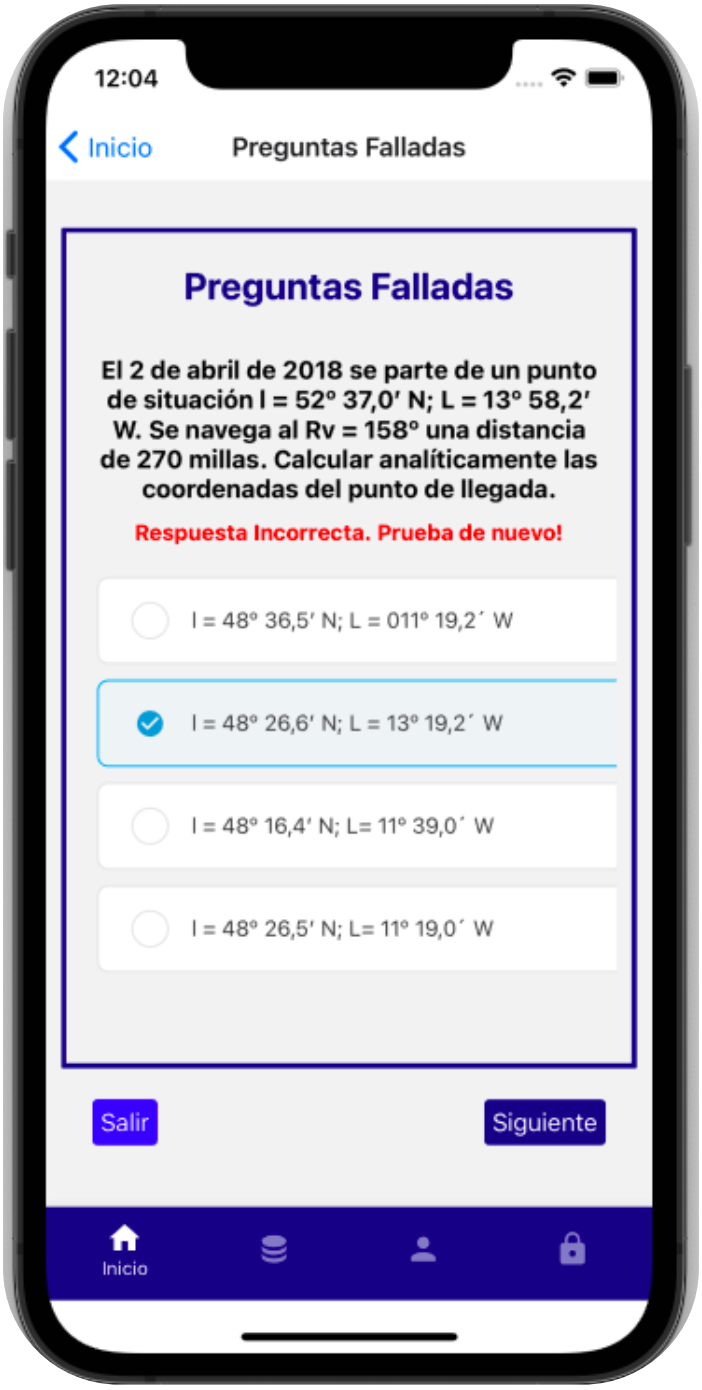


Figura 42 - Wireframe XIV. Examen de preguntas falladas

Lanzar Preguntas Ilimitadas

Figura 43 - Wireframe XV. Lanzar preguntas ilimitadas