Universidad de Alcalá

Escuela Politécnica Superior

GRADO EN INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Trabajo de Fin de Grado

Desarrollo de una aplicación multiplataforma para la gestión de un club de pádel

ESCUELA POLITÉCNICA SUPERIOR

Autor: Miguel Vicente Arará Luján

Tutor/es: Iván González Diego

2021/2022

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGNIERÍA EN SISTEMAS DE INFORMACIÓN

Trabajo Fin de Grado

DESARROLLO DE UNA APLICACIÓN MULTIPLATAFORMA PARA LA GESTIÓN DE UN CLUB DE PÁDEL

Autor: Miguel Vicente Arará Luján

Tutor/es: Iván González Diego

TRIBUNAL:

Presidente:

Vocal 1º:

Vocal 2º:

FECHA:

ÍNDICE

1. Introducción 1
2. Resumen 3
3. Abstract 4
4. Planteamiento del proyecto 5
4.1 Planteamiento del trabajo 5
4.2 Objetivos por conseguir 5
4.3 Requisitos de la aplicación 6
5. Descripción del trabajo 9
5.1 Medios 9
5.1.1 Lenguajes de programación 9
5.1.2 Lenguaje de peticiones y consultas 9
5.1.3 Herramientas 9
5.1.4 Dispositivos Físicos 9
5.1.5 Versionado. 9
5.2 Proceso 10
5.2.1 Requisitos 10
5.2.2 Diseño y Construcción 11
5.3 Node.js 11
5.3.1 ¿Qué es Node.js? 11
5.3.2 ¿Cómo utilizar Node.js? 11
5.4 React Native 12
5.4.1 ¿Qué es React Native? 12

5.4.2 Funcionalidades de React Native 13
5.4.3 Diferencias con Flutter y elección del framework 14
5.4.4 ¿Cómo utilizar React Native? 14
5.5 Expo 15
5.5.1 ¿Qué es Expo? 15
5.5.2 ¿Cómo utilizar Expo? 15
5.6 AWS 15
5.6.1 Servicios de AWS 16
5.6.1.1 Casos de éxito 22
5.6.2 Consola de AWS 23
5.6.3 Instalación y uso de AWS en el proyecto 24
5.6.3.1 Backend con AWS Amplify en el Cloud de Amazon 27
5.7 GraphQL 32
5.7.1 ¿Qué es GraphQL y por qué se ha elegido este lenguaje? 32
5.7.2 ¿Cómo utilizar GraphQL? 32
5.7.2.1 Queries 33
5.7.2.2 Mutations 33
5.8 Estructura 34
5.8.1 Organización de carpetas 34
5.9 Desarrollo 35
5.9.1 Inicio del proyecto 35
5.9.2 Diagrama de Navegación 38
5.9.2.1 Diagrama de navegación del Entrenador 39
5.9.2.2 Diagrama de navegación del Jugador 40
5.9.3 Funcionamiento 40
5.9.3.1 DataFlow del Registro de un usuario 41

5.9.3.2 Sistema de consultas y peticiones GraphQL 42
5.9.4 Manual de Usuario 46
5.9.4.1 Inicio de sesión o Login 47
5.9.4.2 Registro 48
5.9.4.3 Menú de navegación del jugador 49
5.9.4.4 Pantallas 49
6. Presupuesto 56
7. Conclusiones 58
Trabajo futuro 58
8. Bibliografía 60

1. Introducción

En estos últimos años se ha producido un crecimiento exponencial en la práctica del Pádel y países como Suecia, Italia, Francia, Bélgica y Países Bajos están sufriendo la "fiebre del Pádel". "En 2014, el pádel se practicaba en Europa en 14 países, donde había 1.894 clubes, de acuerdo con los datos recopilados por La Guía europea de clubes de pádel, una cifra que en 2017 ascendió a 22 países y 2.800 clubes y que no han dejado de crecer hasta ahora [1]. En España, entre tres y cuatro millones de personas practican Pádel, en las más de 20.000 pistas que existen en los distintos clubs del país. La federación de Pádel española contaba con 75.000 licencias federativas en 2020, cifra que ascendió hasta 96.872 el año pasado, convirtiéndose la federación española en referente y modelo para las federaciones europeas que quieren promocionar este deporte en sus países (datos de la FEP) [2].

El pádel es un juego entre dos parejas , muy parecido al tenis, pero se juega entre cuatro paredes o cristales de metacrilato y en el que la pelota se golpea con una pala de mango corto [3].

Hasta el momento, el pádel avanzaba a un ritmo relativamente normal, pero a partir del año 2020, las redes sociales, la pandemia y sus consecuentes restricciones en la reunión de cierto número de personas han hecho que los deportes colectivos pasen a un segundo plano y el pádel se alce como deporte rey en España. Nuevos inversores de todo el mundo están apostando por este deporte, y en consecuencia se ha incrementado el uso de aplicaciones para facilitar, por ejemplo, la reserva y el pago de una pista de pádel [4]. La app para móviles y web *PlayTomic*, en este momento es la aplicación más utilizada en la Comunidad de Madrid para la reserva y pago de pistas, e incluso organización de partidos. También es utilizada en otras comunidades autónomas como Andalucía, Cataluña y Comunidad Valenciana, en Europa y en EE.UU. Estos últimos dos años ha incluido un conjunto de funcionalidades, tales como gamificación[1] por niveles, subida de resultados de partidos, feedback para contrastar el nivel que se asignan los propios jugadores, y patrocinio de marcas, que le han dado un valor diferencial a la app respecto a sus competidoras, por ejemplo, *Pista3*.

Debido al gran interés en aprender a controlar nuevas tecnologías, me dispongo a desarrollar el proyecto de final de carrera en el mundo del desarrollo de aplicaciones multiplataforma. Para ello, en este trabajo se comenzará realizando una investigación para encontrar qué frameworks son los mejores y más eficientes para desarrollar una aplicación de estas características. Con más de doce años de experiencia como jugador y entrenador de este deporte, he podido aprender el funcionamiento de clubs de pádel y las necesidades que requieren los entrenadores y los jugadores. Habiendo cursado el Grado en Ingeniería en Sistemas de Información y teniendo dicha experiencia

[1] Gamificación: es una adaptación del inglés **gamification**. El sinónimo que más se acerca en nuestro lenguaje a este término es ***ludificación***: Aplicar técnicas o dinámicas propias del juego a actividades o entornos no recreativos para potenciar la motivación y la participación, o facilitar el aprendizaje y la consecución de objetivos (Diccionario de la Real Academia Española)

en el deporte, se ha decidido desarrollar una aplicación capaz de realizar la gestión de un club de Pádel, pero tomando un camino distinto a las aplicaciones existentes, ofreciendo *feedback* de los entrenamientos a los jugadores y por supuesto, permitiendo reservar pista a éstos.

2. Resumen

PaddelApp permite la gestión integral de un club de pádel y ofrece un servicio a sus usuarios para mejorar las actividades. El entrenador puede crear sesiones de entrenamiento con distintos jugadores de la base de datos y registrar sus estadísticas permitiendo dar feedback a los jugadores con los errores y aciertos cometidos. El jugador podrá visualizar en el apartado “Estadísticas” dichos resultados y podrá realizar la reserva de pista desde la pantalla “Reserva pista”, entre otras funciones.

Es una aplicación multiplataforma Node.js escrita en TypeScript bajo el framework de React Native utilizando los servicios de Amazon Web Services (en adelante AWS), con una base de datos NoSQL DynamoDB alojada en el Cloud de Amazon, y estando el *backend* en una máquina virtual *t2.small.elasticsearch* construida a través de AWS Amplify.

Palabras clave: Pádel, React Native, AWS, Aplicación multiplataforma, TypeScript, GraphQL.

3. Abstract

PaddelApp allows the integral management of a paddle club and offer a service to its users to improve the activities. The coach can create training sessions with different players from the database and record their statistics allowing to give feedback to the players with the mistakes and successes made. The player will be able to visualize the results in the Statistics section and will be able to make a court reservation from the Court Reservation section, among other functions.

It is a multiplatform Node.js application written in TypeScript under the React Native framework using Amazon Web Services, with a DynamoDB NoSQL database hosted in the Amazon Cloud, and being the *backend* in a virtual machine *t2.small.elasticsearch* built through AWS Amplify.

Keywords: Paddle, React Native, AWS, Cross-platform aplication, TypeScript, GraphQL.

4. Planteamiento del proyecto

4.1 Planteamiento del trabajo

Se ha decidido, de momento, no desarrollar una aplicación web ya que las necesidades de los usuarios son cubiertas con una aplicación móvil. La aplicación será accesible, por tanto, en dispositivos Android y iOS. A diferencia de las aplicaciones nativas, que necesitan implementaciones distintas para cada dispositivo, esta es creada bajo una única base de código, y sólo son necesarios unos cambios mínimos para su completa adaptación a cada SO. En la siguiente ilustración se ve la diferencia entre el desarrollo de apps con lenguajes nativos o con lenguajes de desarrollo multiplataforma:

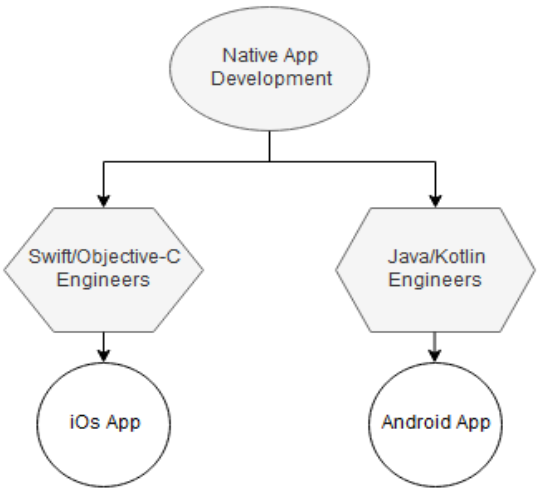

Ilustración 1 - Diagrama de desarrollo de Apps Nativas

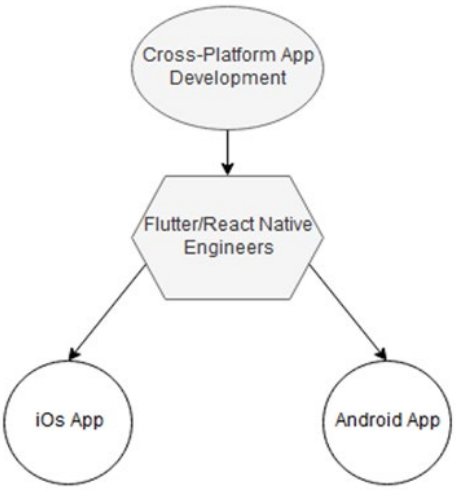

Ilustración 2 - Diagrama de desarrollo de Apps Multiplataforma

La meta es realizar una aplicación de principio a fin con los recursos y conocimientos que se han adquirido hasta el momento, y que seguirán ampliándose a medida que avance el trabajo.

4.2 Objetivos por conseguir

A continuación, se indican los objetivos que serán completados a medida que avance el proyecto, hasta el fin de este:

- Realizar una aplicación capaz de englobar toda **la gestión de un club sin que sea necesario externalizar ningún tipo de servicio**.
- Uso de una **tecnología relativamente nueva**.
- Investigación y elección de qué framework de **desarrollo multiplataforma** se ajusta mejor para la aplicación.
- Desarrollo de la app en un lenguaje de programación **sin ninguna experiencia** previa (p.ej. TypeScript).

- Descubrir las **ventajas y desventajas del uso del framework** de desarrollo multiplataforma.
- Estudio y uso de los servicios que ofrece **AWS** dentro **de sus límites de tarifa gratis**, así como la evaluación de la complejidad de la implementación de estos.
- Desarrollo de una **aplicación totalmente funcional** y que pueda ser lanzada a producción una vez finalizado el proyecto.

4.3 Requisitos de la aplicación

La idea de la aplicación, siendo una idea propia, tomó forma manteniendo conversaciones con expertos del Pádel. Coordinadores de clubs y entrenadores profesionales del WPT (World Padel Tour), Ángel González Martínez y Jesús Cañal, fueron entrevistados con el objetivo de recopilar información para especificar cuáles eran las necesidades de éstos.

Siendo requisitos iniciales y esperando que la aplicación escale todo lo posible, en las siguientes tablas se van a mostrar los requisitos funcionales y no funcionales que fueron recopilados:

REQUISITOS FUNCIONALES	
Código del requisito	**Descripción del requisito**
RF-001	El usuario podrá registrarse e iniciar sesión en la aplicación, guardando sus datos en algún sistema de base de datos.
RF- 002	El usuario tendrá un nombre de usuario y una contraseña, pudiendo iniciar sesión con estas credenciales.
RF-003	Tiene que existir un tipo de usuario que sea capaz de realizar entrenamientos.
RF-004	Tiene que existir un tipo de usuario que sea capaz de visualizar las estadísticas de los entrenamientos.
RF-005	Se debe mantener un histórico de los entrenamientos que se han realizado, quién ha sido el entrenador y los/as jugadores/as.
RF-006	Se debe mantener un histórico de las estadísticas que los/as jugadores/as van realizando en los entrenamientos.

RF-007	Se podrán realizar cuentas rápidamente de la puntuación en un *TieBreak*[2].
RF-008	Un jugador podrá realizar la reserva de una pista de forma rápida y sencilla.
RF- 009	Una vez que el usuario se dé de alta en la plataforma se deberá enviar un código de confirmación para confirmar su email.
RF-010	El usuario deberá saber en todo momento qué tipo de usuario es dentro de la aplicación.
RF-011	El usuario podrá cerrar sesión desde cualquier pantalla de la aplicación.

Tabla 1 - Requisitos funcionales de la aplicación.

REQUISITOS NO FUNCIONALES	
Código del requisito	**Descripción del requisito**
RNF-001	La aplicación debe ser intuitiva. El usuario no debe perderse en ningún momento.
RNF-002	El tiempo de respuesta de las funcionalidades de la aplicación debe ser breve.
RNF-003	El *backend* de la aplicación deberá estar alojado en Cloud.
RNF-004	La base de datos deberá estar alojado en Cloud y deberá ser NoSQL.
RNF-005	Tiene que existir una misma base de código para ejecutar la aplicación tanto en iOS como en Android, para que haya sólo un proyecto.

[2] El Tie-Break tanto en el tenis como en el pádel es el sistema de puntuación que sirve para desempatar el set o el partido si está igualado (empate a 6-6). Se juega a 7 puntos . Gana el contrincante/pareja que llegue antes a los 7 puntos con 2 puntos de ventaja sobre la puntuación del rival. Es un sistema de puntuación típico en entrenamientos.

RNF-006	La apariencia de la aplicación tiene que ser sencilla y moderna.
RNF-007	La aplicación debe mantener una sincronización de los datos.

Tabla 2 -Requisitos no funcionales de la aplicación.

5. Descripción del trabajo

Debido a la magnitud que implica el desarrollo de una app para la gestión integral de un club de pádel se ha optado por desarrollar aquellas características de las que, hasta el momento, no se tiene constancia de su existencia y que, según los requisitos de usuario recabados, son necesarias para que la aplicación tenga sentido.

En algunos de los siguientes apartados se incluirán algunas breves extracciones del código para completar la explicación.

5.1 Medios

En este apartado se enumeran los medios necesarios para la construcción del proyecto:

5.1.1 Lenguajes de programación

Los lenguajes de programación utilizados:

- React Native (Framework) [5]:
  - JavaScript
  - React
  - TypeScript

5.1.2 Lenguaje de peticiones y consultas

En este Proyecto se ha utilizado una base de datos NoSQL DynamoDB y el lenguaje de consultas se llama GraphqQL que se explicará en un apartado posterior, véase GraphQL [6].

5.1.3 Herramientas

A continuación, se enumeran las herramientas que se necesitan para llevar a cabo el desarrollo completo de la aplicación, desde la programación hasta la visualización de las demostraciones y pruebas de funcionamiento:

- VS Code
- Expo Go
- Expo CLI app para iPhone

5.1.4 Dispositivos Físicos

Para realizar el proyecto se va a disponer del portátil personal y del móvil personal (iPhone) para testear en dispositivos con SO iOS.

5.1.5 Versionado.

El software que se ha elegido para el control de versiones ha sido Git y su consola. Se inició creando el repositorio en el perfil de GitHub [7] y en este momento es un repositorio público pudiendo

acceder cualquier persona. Se creó una rama *master* en la que se realizan todos los *commits*, sin creación de ninguna rama alternativa.

Ilustración 2 - Icono de la aplicación Git

Ilustración 4 - Icono de la aplicación GitHub

5.2 Proceso

Según la norma ISO/IEC 12207:2008 los procesos y actividades relacionados con la evolución del software pueden organizarse en etapas. Esta aplicación ha seguido el modelo de ciclo de vida en cascada. Durante el proyecto se han producido reiteradas subidas y bajadas en este ciclo sin llegar a la etapa de Mantenimiento; esta comenzará una vez sea lanzada la aplicación a producción.

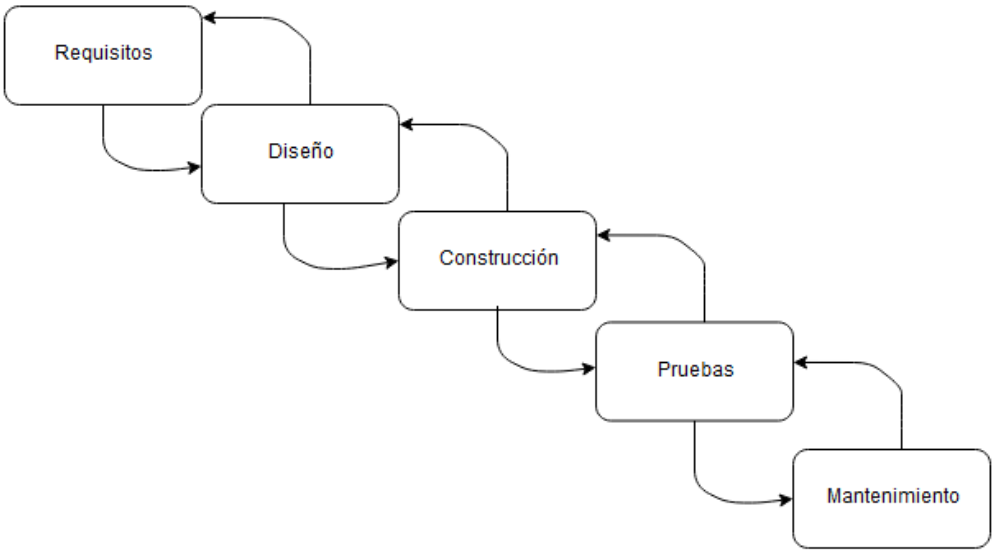

Ilustración 5 - Modelo de ciclo de vida de software en cascada

El ciclo de vida del software se ha construido en cascada. A continuación, se va a desglosar cada una de las etapas, explicando brevemente los pasos y decisiones que se han ido tomando a medida que se ha avanzado en el proyecto:

5.2.1 Requisitos

Se han realizado diferentes entrevistas, como se indicó más arriba, con personas relacionadas con el mundo de este deporte y que además tenían experiencia en el uso de otras aplicaciones, lo que ha servido para inspirar aquellos puntos que había que desarrollar para que esta aplicación cobrase un sentido particular frente a otras del mercado. Por otro lado, los requisitos no funcionales

establecidos van acorde a los estándares de UI (User interface) de las aplicaciones multiplataforma para dispositivos móviles (veáse Requisitos de la aplicación).

5.2.2 Diseño y Construcción

Existen una serie de aspectos que se deben considerar cuando se pretende diseñar y construir una aplicación móvil moderna:

- Interfaz de Usuario: Cómo los usuarios consumen e interactúan con la aplicación.
- Ruta: Cómo los usuarios navegan entre las diferentes pantallas de la aplicación.
- Tratamiento de datos: Dónde se alojan y como se traen y se exponen los datos
- Renderizado: Cuándo y dónde se renderiza contenido estático y dinámico
- Integraciones: Que tipo de servicios va a tener la aplicación (p.ej. Autenticación, pagos, etc) y cómo se conectan entre sí.
- Infraestructura: Dónde se despliega, se almacena y se ejecuta la aplicación (Serverless, CDN, Edge, etc).
- Escalabilidad: Cómo la aplicación es capaz de adaptarse debió al crecimiento de usuarios y datos.

5.3 Node.js

5.3.1 ¿Qué es Node.js?

Ideado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para crear aplicaciones network escalables [8].

Ilustración 6 -Node.js Logo.

5.3.2 ¿Cómo utilizar Node.js?

npm es el Node Package Manager que viene incluido y ayuda a cada desarrollo asociado a Node. Node ha sido ampliamente utilizado por los desarrolladores de JavaScript para compartir herramientas, instalar módulos y administrar sus dependencias.

Esta herramienta funciona de dos formas:

1. Como un repositorio ampliamente utilizado para la publicación de proyectos Node.js de código abierto. Lo que significa que es una plataforma en línea donde cualquiera puede publicar y compartir herramientas escritas en JavaScript.

2. Como una herramienta de línea de comandos que ayuda a interactuar con plataformas en línea, como navegadores y servidores. Esto ayuda a instalar y desinstalar paquetes, gestión de versiones y gestión de dependencias necesarias para ejecutar un proyecto.

Para usar paquetes, el proyecto debe contener un archivo llamado **package.json**. Dentro de ese archivo, se encuentran metadatos específicos para los proyectos. En este proyecto se descarga la versión para Windows de Node.js y se añade a las variables de entorno, habilitando *npm* para que la Shell de Windows lo reconozca como comando. Esta información ha sido extraída del artículo [9].

Ilustración 7 - npm Logo.

5.4 React Native

5.4.1 ¿Qué es React Native?

React Native es un framework de interfaz de usuario de código abierto desarrollado por Meta Platforms, Inc (lo que era conocido como Facebook) en marzo de 2015. Es utilizado para desarrollar aplicaciones para iOs, Android, macOs, Android TV, tvOS, Web, Windows [5].

Ilustración 8 - ReactNative Logo.

React Native combina las mejores partes del desarrollo nativo con React, la mejor clase de JavaScript para construir interfaces de usuario. Utiliza componentes nativos en vez de componentes web como bloques de construcción. Por lo tanto, para entender la estructura de una "React Native App", se necesita entender algunos conceptos básicos de React como *JSX*[3], *componentes*[4], *state*[5] and *props*[6] [10], [11].

---

[3] Es una extensión de la sintaxis de JavaScript. Se recomienda usarlo con React para describir cómo debería ser la interfaz de usuario. JSX puede recordar a un lenguaje de plantillas, pero viene con todo el poder de JavaScript.

[4] Conceptualmente, los componentes son como las funciones de JavaScript. Aceptan entradas arbitrarias (llamadas "props") y retornan elementos de React que describen lo que debe aparecer en la pantalla.

[5] Es mutable, es decir, que el estado puede ser actualizado en el futuro mientras que los *props* no. Podemos inicializar el estado en el constructor, y luego llamar a *setState* cuando se quiera cambiar.

[6] Son inmutables y son establecidos por el padre y son fijos durante toda la vida de un componente.

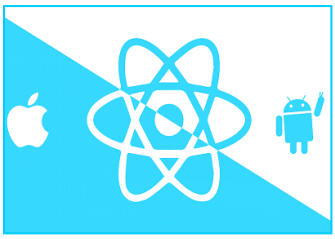

Ilustración 9 - Imagen representativa de React Native como framework para desarrollo multiplataforma.

5.4.2 Funcionalidades de React Native

Estas son las funcionalidades principales que se le atribuyen a React Native:

- Compatibilidad cross-plattform (para distintos SO).
- Funcionalidad nativa[7].
- Actualizaciones instantáneas (para desarrollo y/o test).
- Experiencia positiva para el desarrollador.

[7] Las Apps nativas son aquellas aplicaciones que están desarrolladas para un equipo o plataforma determinada. Es decir, funciona en el equipo sin necesidad de ningún programa externo ya que se ha desarrollado en el lenguaje de programación específico de cada equipo. El término de App Nativa está habitualmente asociado a los dispositivos móviles y por tanto hay Apps Nativas para cada sistema operativo como iOS o Android.

5.4.3 Diferencias con Flutter y elección del framework

Flutter también es un framework open source desarrollado por Google en 2018 y su lenguaje de programación es Dart[8], siendo este relativamente nuevo y fácil de aprender para programadores experimentados en Java y C#. Flutter funciona sin el uso de APIs[9] adicionales a diferencia de React Native, de hecho, el framework hace todo el trabajo para realizar la conexión entre el dispositivo y la base de código. Cuando se trata de la apariencia nativa, React Native es más dinámico con una API potente y flexible [12], [13], [14].

El puente JS en React Native es la capa que interpreta en tiempo de ejecución la lógica asociada a las

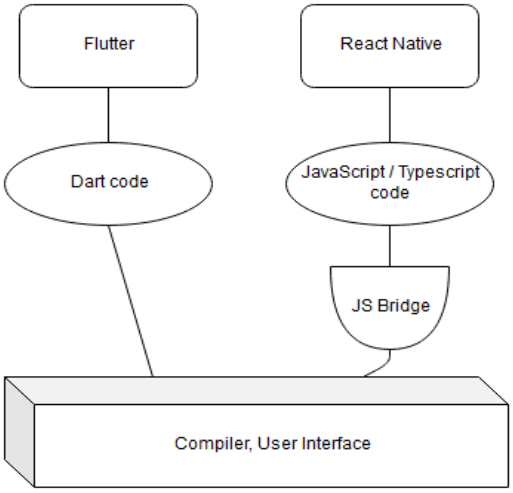

Ilustración 10 – Diagrama de Frameworks de desarrollo multiplataforma

vistas nativas e interactúa con las mismas.

Con Flutter no se ha podido testear la app en el dispositivo físico móvil de iOS, un iPhone, debido a que desarrollando la app con VSCode y con un portátil con SO Windows no es posible. Para contrastar este impedimento se realizó una pregunta [15] en StackOverflow, que fue contestada a los pocos minutos por una persona con gran experiencia en este campo, verificando dicha información e inidicando la necesidad de una herramienta llamada XCode, que sólo puede ser descargada en macOS. Por tanto, el framework elegido para la implementación ha sido React Native.

5.4.4 ¿Cómo utilizar React Native?

Las aplicaciones React Native se pueden ejecutar en dispositivos móviles iOS a partir de la versión 9.0 y Android a partir 4.1. Para desarrollar con este framework se puede usar cualquier sistema

---

[8] Dart es un lenguaje de programación que fue desarrollado por Google. Lanzó su primera versión en 2011. Es una alternativa a reemplazar Javascript y convertirse en el lenguaje prioritario para los navegadores actuales, aunque sigue en proceso de mejora. Se puede utilizar para desarrollar aplicaciones web, servidores, aplicaciones de consola, aplicaciones móviles.

[9] Application programming interface.

operativo Windows, macOs o Linux. En este trabajo se ha utilizado Visual Studio Code como editor de código fuente. Para la creación y ejecución de aplicaciones React Native en iOs es necesario un macOS o en su defecto, una herramienta llamada Expo.

Se instaló en el proyecto con el siguiente comando:

```
npm install @react-navigation/native
```

React Native Framework ha sido instalado en la herramienta de desarrollo de código fuente VisualStudio Code [16].

5.5 Expo

5.5.1 ¿Qué es Expo?

Expo es un framework para aplicaciones universales React. Es un conjunto de herramientas y servicios construidos en torno a React Native y a plataformas nativas, que ayuda a desarrollar, construir, desplegar e iterar **rápidamente** en iOs, Android, y aplicaciones web desde el mismo código JavaScript/TypeScript [17].

Ilustración 11 - Icono de la aplicación Expo.

5.5.2 ¿Cómo utilizar Expo?

Se comienza ejecutando el comando en la raíz del proyecto, instalando la librería. A continuación, se descarga la aplicación Expo CLI en *App Store* o *Play Store* en el dispositivo móvil, que permitirá al desarrollador testear la aplicación en tiempo real, visualizando el resultado del código.

Una vez se instala la librería Expo para Node.js se debe iniciar el proyecto con el comando *expo init*. Cuando finalice la ejecución de este comando se podrá comenzar a comprobar el desarrollo en el dispositivo móvil. La aplicación se actualizará y aplicará los cambios efectuados desde el código en el dispositivo móvil cada vez que se guarde un archivo o se recargue el proyecto pulsando la letra 'r' desde la terminal de VSCode, estando este en ejecución a partir del comando *npm start* [1].

En adición, Expo ofrece subir la versión de la aplicación a su plataforma, para que desde su app movil se pueda acceder a esta versión sin necesidad de que la aplicación sea ejecutada desde el ordenador. Esta será utilizada para la demostración debido a que publicar la aplicación en AppStore conllevaría un coste monetario adicional.

5.6 AWS

AWS o Amazon Web Services es un proveedor de servicios en la nube. Ofrecen desde tecnologías de infraestructura como cómputo, almacenamiento y bases de datos, hasta tecnologías emergentes como inteligencia artificial y aprendizaje automático. Permite una gran rapidez llevando aplicaciones existentes a la nube, siendo fácil y rentable [18].

Ilustración 12 - Icono de la aplicación de AWS.

5.6.1 Servicios de AWS

A continuación, se detalla una breve explicación, a partir de la propia documentación de AWS, de los servicios que han sido utilizados para la construcción de la aplicación.

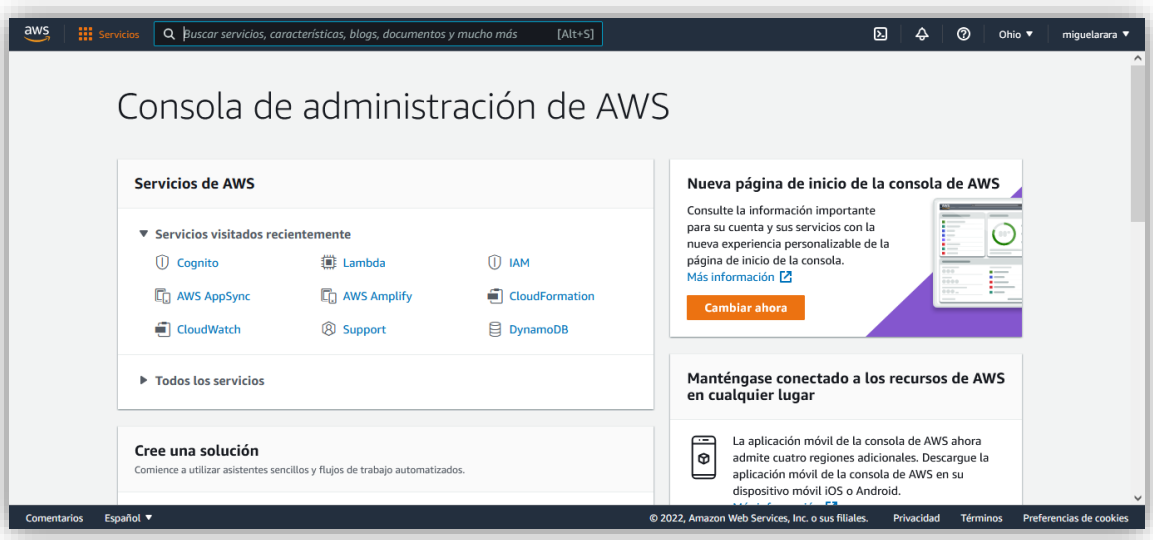

Ilustración 13 - Consola de administración de AWS

Una vez se ha iniciado sesión en AWS, desde la consola de administración se puede acceder a los distintos servicios que se han visitado recientemente. Se comienza con AWS amplify, debido a que, a partir de este servicio, se han explotado los servicios que se comentarán posteriormente:

- **AWS Amplify**

Es un conjunto de herramientas (marco de código abierto, entorno de desarrollo visual, consola) y servicios (alojamiento de sitios web estáticos y aplicaciones web) para acelerar el desarrollo de aplicaciones web y móviles en AWS. Se puede utilizar estas herramientas de AWS Amplify **juntas o por separado** para crear y enviar una aplicación extensible y con muchas características [19]:

- Herramientas
  - Amplify Studio: Capaz de crear una aplicación de pila completa con interfaz de usuario *frontend* y *backend* [20].
  - Amplify CLI: Configura todos los servicios que se necesita para potenciar el *backend* de la app de una forma fácil y sencilla a partir de la interfaz de línea de comandos y siguiendo unos pasos [21].

- Amplify Libraries: Conecta una aplicación a servicios AWS nuevos o existentes (Cognito, S3 y más) [22].
- Amplify Hosting: Aloja aplicaciones web o sitios web rápidos, confiables y seguros a través de la red de entrega de contenido de AWS [23].

- Casos de uso
  - Incorporación de autenticación de usuarios [24].
  - Creación de un importante modelo de datos de *backend* [25].
  - Creación visual de un *backend* y una interfaz de usuario de aplicación.
  - Alojar una aplicación web progresiva o un sitio web estático.

❖ DynamoDB

Es un servicio de base de datos NoSQL totalmente administrado que ofrece un rendimiento rápido y predecible, así como escalabilidad. En el aparto *DynamoDB -> Tablas* de la la siguiente ilustración se visualizan las tablas que tiene la aplicación desde la consola [26]:

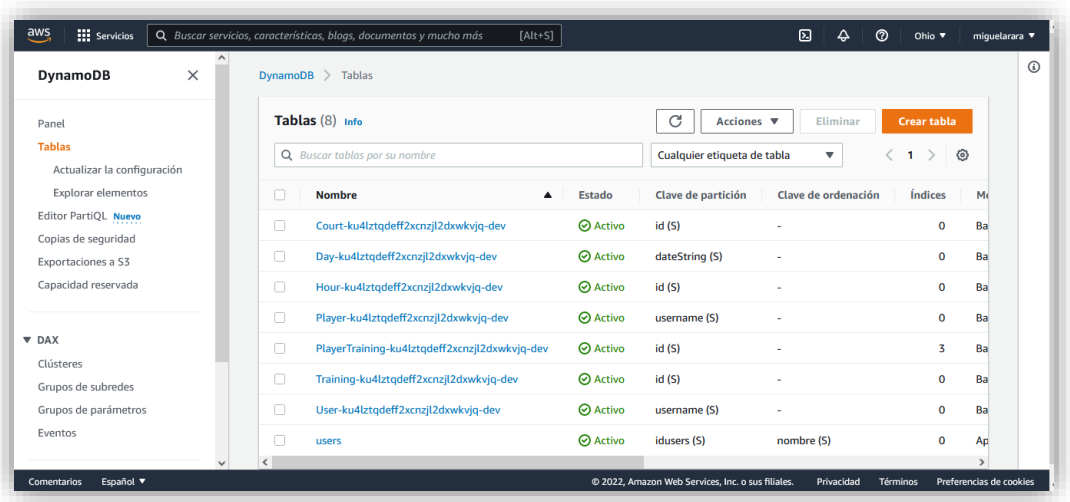

Ilustración 14 - DynamoDB en la consola de AWS.

❖ Cognito

Es un servicio que ofrece autenticación, autorización y administración de usuarios para las aplicaciones móviles y web. Los usuarios pueden iniciar sesión directamente con un nombre de usuario y una contraseña [27].

Ilustración 15 - Pantalla inicial de Cognito en la consola de AWS.

Si se pulsa en “Administrar grupos de usuarios” se abre una pantalla de configuración del grupo de usuario seleccionado.

Ilustración 16 - Pantalla de configuración de grupos de usuarios en Amazon Cognito.

Se pueden ver y gestionar los usuarios que se han registrado en la app, así como los grupos a los que pertenecen dichos usuarios, el estado de su cuenta (CONFIRMADO o NO CONFIRMADO), la fecha de creación y actualización de cada usuario, número de teléfono.

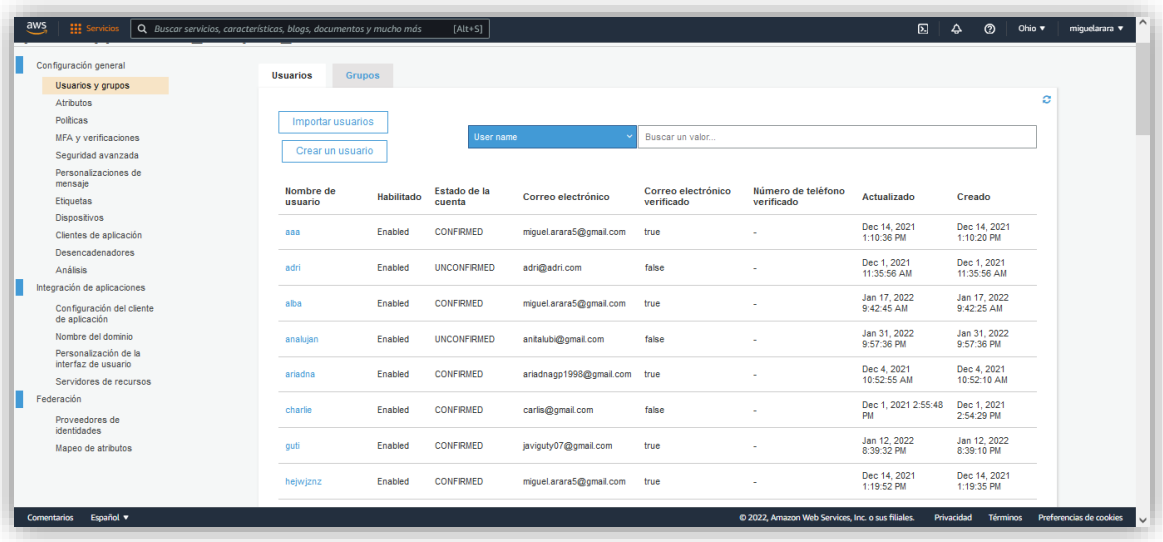

Ilustración 17 - Usuarios registrados en Cognito en la consola de AWS

Existe un apartado muy importante llamado "Desencadenadores" o "Triggers". Se pueden realizar personalizaciones avanzadas con las funciones AWS Lambda (se explican más adelante), personalizando el flujo de trabajo y la experiencia de usuario. En este proyecto se ha creado una función lambda propia que se lanza en el momento en que el usuario ha confirmado un código OTP[10] que ha sido enviado a su correo electrónico ,véase Lambda Functions.

❖ AWS AppSync

Proporciona una interfaz GraphQL sólida y escalable para que los desarrolladores de aplicaciones combinen datos de varias fuentes, como Amazon DynamoDB, AWS Lambda y las API HTTP.

AWS AppSync permite que su aplicación reciba actualizaciones en tiempo real para los datos que quiera mantener sincronizados. Su aplicación necesita mantenerse conectada al servicio de AWS AppSync para recibir actualizaciones en tiempo real [28]. En la ilustración se presenta el apartado *Esquema* con GraphQL de la aplicación:

[10] El código OTP es una password o contraseña de un único uso (sus siglas en inglés: One-Time Password), también conocida como password o contraseña dinámica. Se utiliza como segundo factor de autenticación, además del nombre de usuario y la contraseña comúnmente utilizados

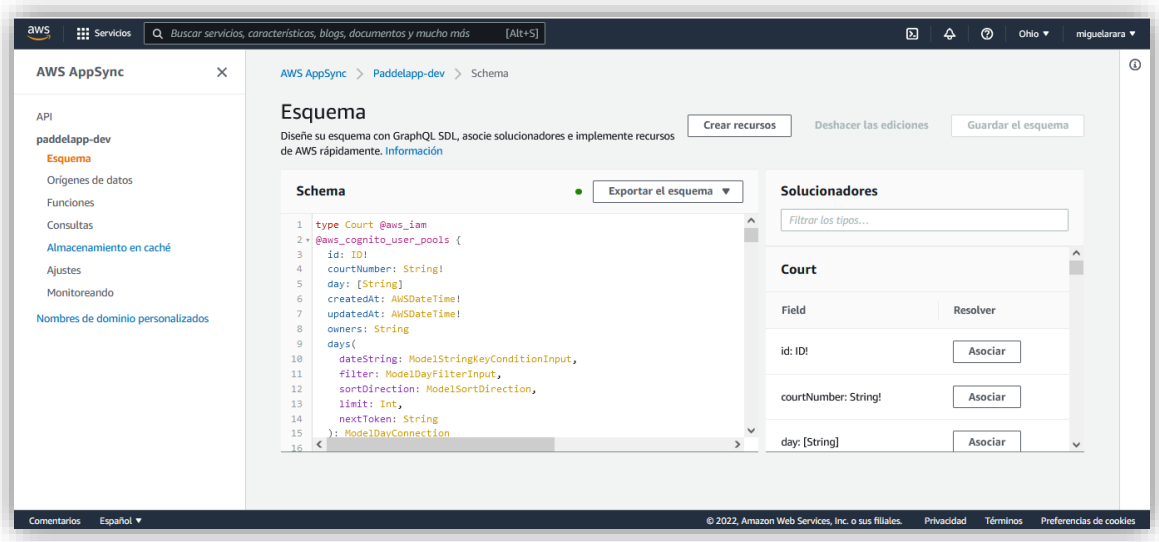

Ilustración 18 - AWS AppSync Esquema de PaddelApp.

❖ CloudWatch

Recopila datos operativos y de monitoreo en forma de registros, métricas y eventos, y permite su visualización mediante paneles automatizados para obtener una vista unificada de los recursos, aplicaciones y los servicios de AWS que se ejecutan en AWS y en las instalaciones [29].

En la ilustración se muestra el monitoreo de la función *postConfirmation*:

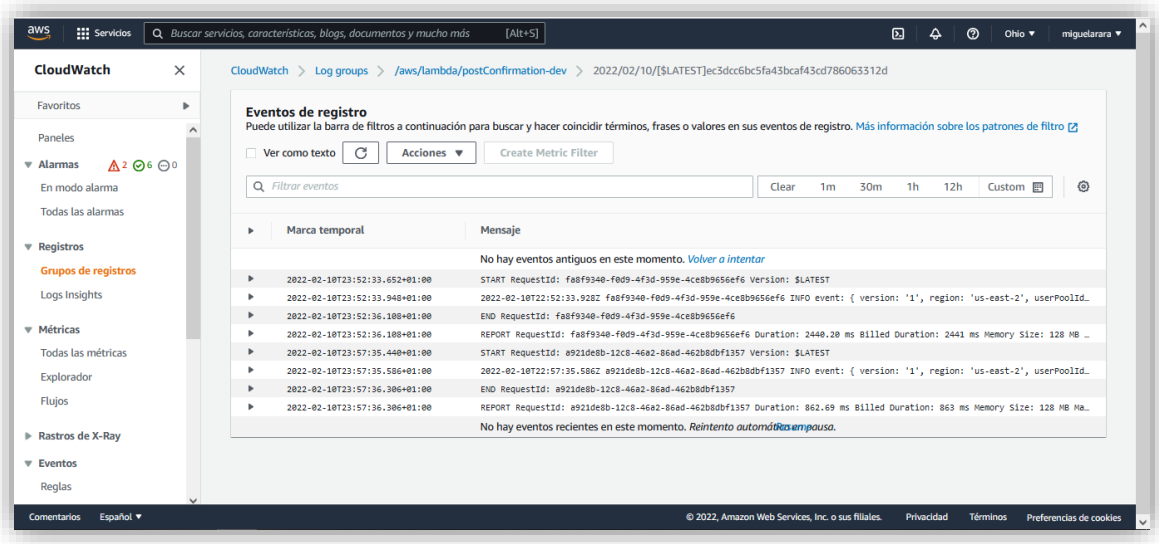

Ilustración 19 - Registros de monitoreo de la función Lambda PostConfirmartion.

❖ Lambda

Es un servicio que permite ejecutar código sin aprovisionar ni administrar servidores. Lambda ejecuta el código en una infraestructura informática de alta disponibilidad y realiza todas las tareas de administración de los recursos informáticos [30].

En la consola de AWS, el servicio Lambda se presenta como se puede ver en la ilustración enumerando primero las funciones que han sido creadas en el proyecto:

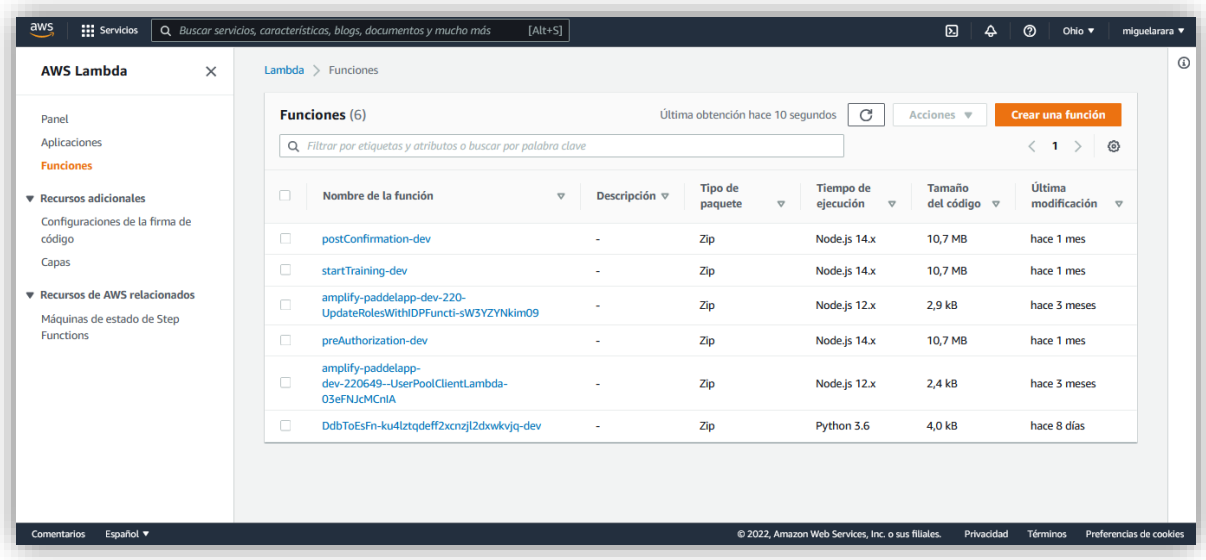

Ilustración 20 - Pantalla principal AWS Lambda

Si se accede a una de las funciones, se muestra el código fuente y una serie de acciones a realizar sobre dicha función, pudiendo monitorear, probar o configurar la misma:

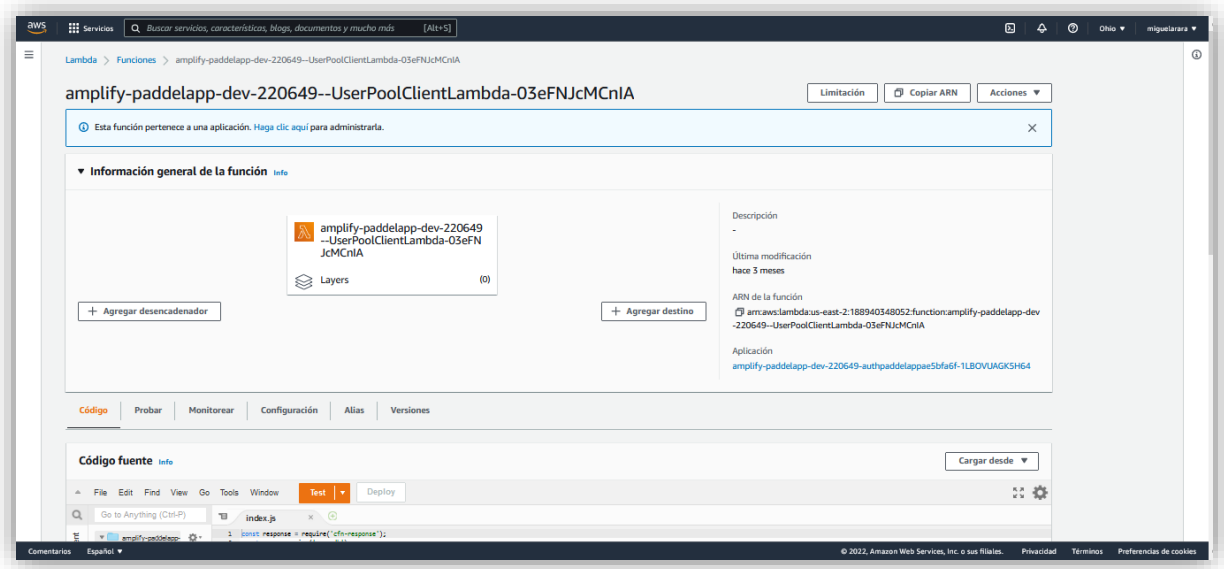

Ilustración 21 - Función Lambda creada automáticamente con AWS Amplify.

5.6.1.1 Casos de éxito

Existen varios casos de éxito, mencionados en la plataforma de Amazon [31] que cabe mencionar para **valorar el uso de AWS Amplify** y la importancia que tiene en el desarrollo de aplicaciones web y móvil en la actualidad:

- NeimanMarcus[11] : Para acelerar el tiempo de desarrollo de su aplicación, el equipo de Neiman Marcus optó por crear una arquitectura sin servidor utilizando AWS Amplify y finalmente aumentó la velocidad de comercialización de su aplicación en un 50% [32].

Ilustración 22 - NeimanMarcus Logo

- Noom[12] " [33]: Citando textualmente las palabras de Patrick Lee, Ingeniero Senior del Equipo de Crecimiento de la empresa Noom, "el uso de Amplify resuelve múltiples retos en nuestro proceso de desarrollo. El principal problema que tratábamos de resolver era mejorar la velocidad para construir y desplegar nuevas versiones de nuestro sitio web. La consola de Amplify proporciona muchas características que queremos sin tener que construirlas desde cero, lo que mejora drásticamente nuestra velocidad de desarrollo".

Ilustración 23 - Noom Logo

- Amazon Music [34]: La plataforma global de música en streaming utiliza AWS Amplify y AWS AppSync para permitir a millones de usuarios sincronizar su lista de

[11] Neiman Marcus, un nombre muy conocido en la venta minorista de lujo, gestiona 38 grandes almacenes en EE.UU y un servicio digital de primera clase para clientes de todo el mundo. Es la marca insignia del Grupo Neiman Marcus, fundado en 1907.

[12] La plataforma Noom combina la ciencia de la inteligencia artificial, la tecnología móvil y la psicología con entrenadores humanos para ayudar a las personas a crear nuevos hábitos saludables. Más de 50 millones de personas en todo el mundo han utilizado Noom para mejorar su salud.

reproducción de música y acceder a ella sin conexión desde sus aplicaciones web y móviles a escala.

Ilustración 24- Amazon Music Logo

- Branch[13] [35]: Citando textualmente a Adithya Reddy, Full Stack Engineer de Branch Insurance "Elegimos las bibliotecas de Amplify porque creíamos que acelerarían nuestra velocidad de desarrollo y el tiempo de comercialización, y así fue. Las librerías de Amplify, en particular la función de autenticación, nos llevaron mucho más rápido a la fase de MVP y a la incorporación de los clientes."

Ilustración 25 - Branch Logo

5.6.2 Consola de AWS

En la consola de AWS se puede acceder e interactuar con los servicios mencionados anteriormente. Para utilizar la consola y sus servicios es necesario crearse una cuenta de AWS. En este proyecto ha sido creada una cuenta con el correo de la universidad. El proceso de creación fue sencillo, y no se va a explicar.

Es importante destacar que, actualmente, la cuenta está bajo una ***Free tier*** (tarifa gratuita) por la cual AWS ofrece sus servicios a estudiantes que quieran aprender y familiarizarse con ellos. Estos servicios están limitados a unos rangos mensuales. Cuando el límite de este rango está próximo, AWS envía un correo electrónico automáticamente, avisando para que se reduzca la carga de trabajo en este servicio, si no se cobrará según los excesos que se hayan producido.

En esta ilustración se puede visualizar los servicios utilizados, uso real y el de uso previsto según lo utilizado en el mes.

---

[13] Branch es una aseguradora de EE. UU que pretende romper el mercado tradicional de los seguros ofreciendo un paquete de seguros de hogar y automóvil en línea a un precio mucho más bajo que el de los proveedores o corredores existentes. Branch proporciona un mayor ahorro al vender directamente y trasladar el ahorro al cliente.

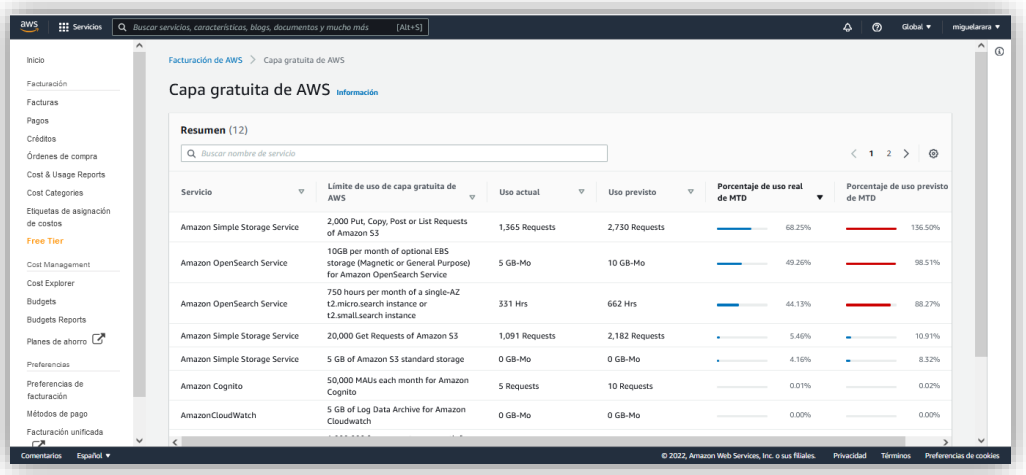

Ilustración 26 - Capa gratuita de AWS

5.6.3 Instalación y uso de AWS en el proyecto

En este apartado se va a especificar los pasos que se han seguido para la instalación de AWS Amplify y el uso de los servicios de AWS en el proyecto. Para ello se comenzó instalando la librería AWS Amplify. Los pasos se siguieron en concordancia con la documentación de AWS Amplify [36].

```
PS C:\Users\migue\PaddelApp> npm i -g aws-amplify@3.3.22
```

En una primera instancia, se instaló la última versión estable habilitada por aws-amplify, pero la sintaxis aplicada en el archivo *schema.graphql* obligó a re-instalar esta librería a la versión 3.3.22, debido a que la documentación e información que se encontró funcionaba únicamente con dicha versión, y no con una posterior. Se mencionará en Sistema de consultas y peticiones GraphQL.

Una vez instalada la librería se debe configurar para el proyecto. Se van a mencionar los comandos ejecutados más relevantes:

```
PS C:\Users\migue\PaddelApp>amplify configure
```

1. En el primer paso se debe elegir la *AWS Region* que ha sido asignada en el momento de crear la cuenta de AWS. Como se puede visualizar en la ilustración del apartado Cognito dentro de Servicios de AWS, la región de este proyecto será *us-east-2.*

```
Sign in to your AWS administrator account:
https://console.aws.amazon.com/
Press Enter to continue

Specify the AWS Region
? region:  (Use arrow keys)
❯ us-east-1
  us-east-2
  us-west-2
  eu-west-1
  eu-west-2
  eu-central-1
  ap-northeast-1
(Move up and down to reveal more choices)
```

Ilustración 26 Configuraciín de AWS Amplify - Capt 1

2. El siguiente paso es escribir el IAM User que ha sido creado para que sea el administrador de la consola de AWS. Previamente se creó un IAM User. Si cabe mencionar que, una vez creado y asociado a la política de *Adminitrator Access* , AWS provee a este usuario una *Access Key ID* y una *Scecret Access Key ID,* para que **este usuario sea el único administrador del proyecto** y va a ser la única ocasión que el usuario creado va a poder descargar el archivo con dichas credenciales (aunque se pueden volver a crear nuevas claves):

```
Specify the username of the new IAM user:
? user name:  (amplify-lfQGp)
```

Ilustración 27 - Introducir el IAM Username para amplify configure. Captura 2

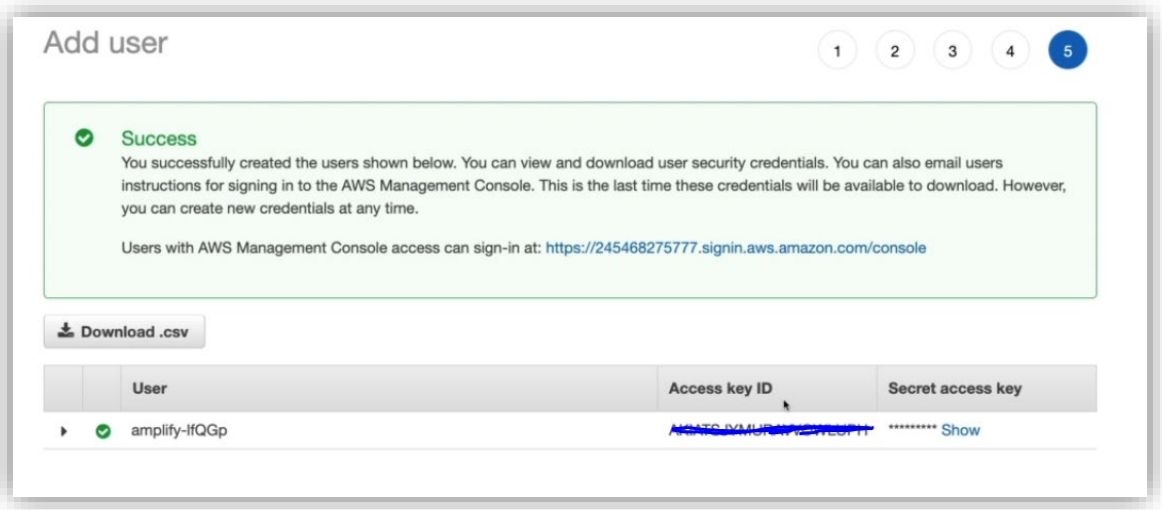

Ilustración 28 - Claves IAM User de AWS

Se deben incluir las claves dadas en la configuración de AWS Amplify:

```
Enter the access key of the newly created user:
? accessKeyId:  ********************
? secretAccessKey:  ****************************************
```

Ilustración 29 - Introducción de las claves en Amplify configure - Captura 3

3. Una vez introducido el nombre que se quiere dar al IAM User, se procede a la creación del *backend* en el cloud de AWS de nuestra app:

```
PS C:\Users\migue\PaddelApp> amplify init
```

Se va a mostrar únicamente una imagen con la configuración en la ejecución del comando de arriba, los parámetros elegidos y una vez introducidas las credenciales se crea el ***backend***:

```
PS C:\Users\migue\padel_app> amplify init
Note: It is recommended to run this command from the root of your app directory
? Enter a name for the project padelapp
The following configuration will be applied:

Project information
| Name: padelapp
| Environment: dev
| Default editor: Visual Studio Code
| App type: flutter
| Configuration file location: ./lib/

? Initialize the project with the above configuration? No
? Enter a name for the environment dev
? Choose your default editor: Visual Studio Code
? Choose the type of app that you're building javascript
Please tell us about your project
? What javascript framework are you using react-native
? Source Directory Path:  src
? Distribution Directory Path: ./
? Build Command:  npm.cmd run-script build
? Start Command: npm.cmd run-script start
Using default provider  awscloudformation
? Select the authentication method you want to use: AWS access keys
? accessKeyId:  [hidden]
```

Ilustración 30 - Configuración Amplify Init

Y se muestra un mensaje como se ve en la siguiente ilustración, mostrando los comandos que ofrece AWS Amplify para personalizar el *backend* desde la consola:

```
✔ Successfully created initial AWS cloud resources for deployments.
✔ Initialized provider successfully.
Initialized your environment successfully.

Your project has been successfully initialized and connected to the cloud!

Some next steps:
"amplify status" will show you what you've added already and if it's locally configured or
 deployed
"amplify add <category>" will allow you to add features like user login or a backend API
"amplify push" will build all your local backend resources and provision it in the cloud
"amplify console" to open the Amplify Console and view your project status
"amplify publish" will build all your local backend and frontend resources (if you have ho
sting category added) and provision it in the cloud

Pro tip:
Try "amplify add api" to create a backend API and then "amplify publish" to deploy everyth
ing
```

Ilustración 31 - Creacion del backend de AWS Amplify con éxito

5.6.3.1 Backend con AWS Amplify en el Cloud de Amazon

Desde la consola de Amazon se puede visualizar la app accediendo al servicio AWS Amplify *backend* de la app:

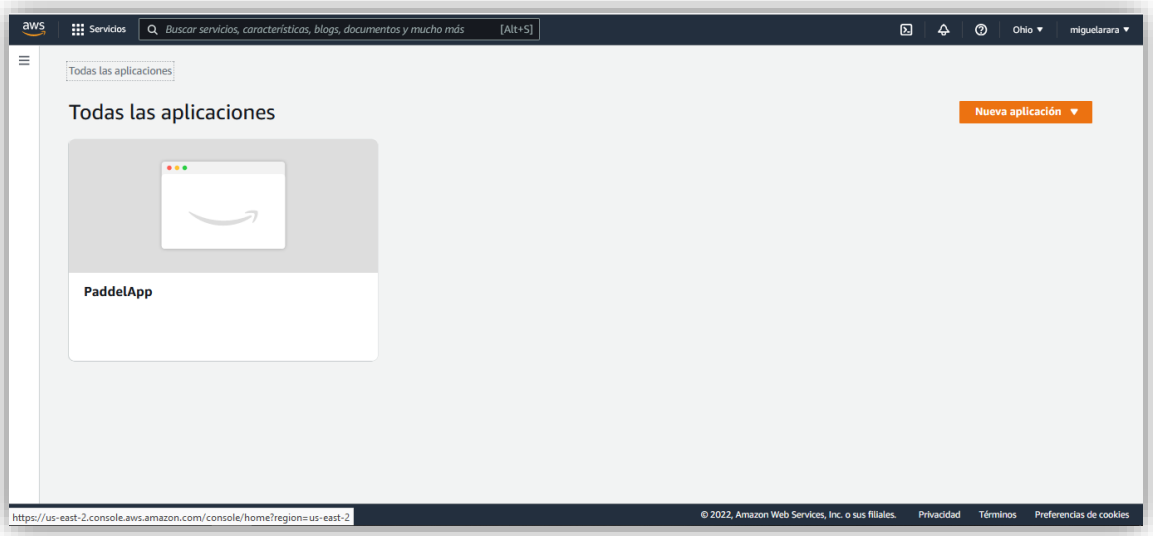

Ilustración 32 - *Backend* de la aplicación en AWS Amplify

Si se pulsa en la *app*, y a continuación en *dev*, aparecen las categorías que han sido construidas a base de los comandos que serán explicados brevemente. También se puede ver la actividad de implementación más reciente y navegar entre las categorías creadas:

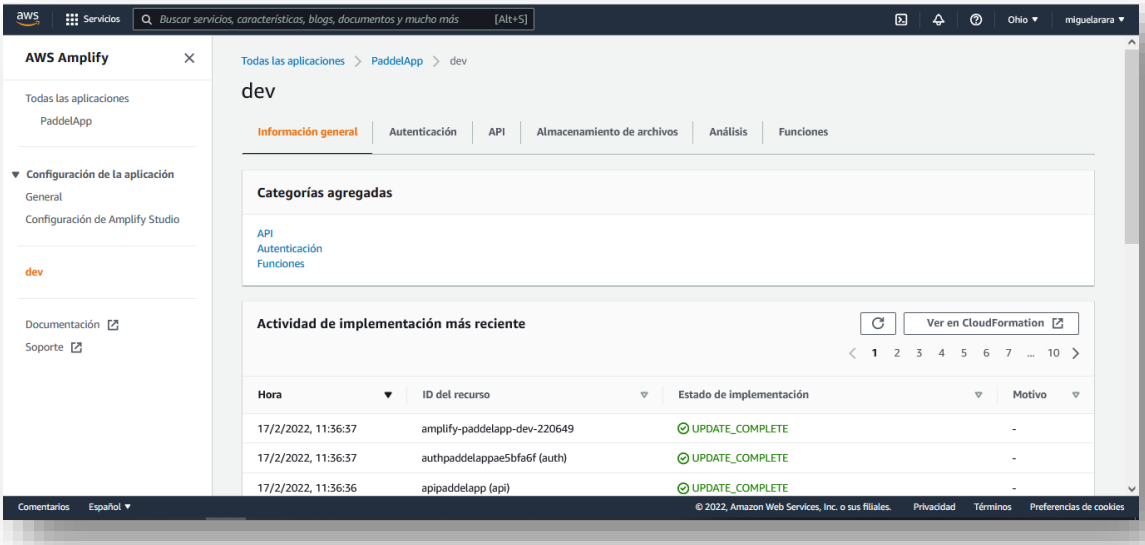

Ilustración 33 - Dev de la aplicación PaddelApp en AWS amplify.

- API - Autenticación

Primero se creó la API mediante el siguiente comando:

```
PS C:\Users\migue\PaddelApp> amplify add api
```

Ilustración 34 - Configuración de amplify add api desde consola.

Se escogieron los parámetros elegidos en la ilustración de arriba, y escogiendo la configuración por defecto para la autenticación y la seguridad:

```
 Warning: you will not be able to edit these selections.
 How do you want users to be able to sign in? Username
 Do you want to configure advanced settings? Yes, I want to make some additiona
l changes.
 Warning: you will not be able to edit these selections.
 What attributes are required for signing up? Email, Name
 Do you want to enable any of the following capabilities?
Successfully added auth resource tictactoe3f7cfa69 locally

Some next steps:
"amplify push" will build all your local backend resources and provision it in
the cloud
"amplify publish" will build all your local backend and frontend resources (if
you have hosting category added) and provision it in the cloud

? Do you want to configure advanced settings for the GraphQL API No, I am done.


? Do you have an annotated GraphQL schema? (y/N)
```

Ilustración 35 - Configuración de la API y la Autenticación.

Se eligió el **nombre de usuario** como el parámetro para que inicien sesión los usuarios, y que el **email** y el **nombre** sean los campos requeridos para registrarse. Ejecutándose el siguiente comando se pueden ver las categorías que han sido creadas y si están versionadas con el cloud del AWS:

```
PS C:\Users\migue\PaddelApp> amplify status
```

Y se obtiene por pantalla el siguiente resultado:

```
Current Environment: dev
```

Category	Resource name	Operation	Provider plugin
Api	paddelapp	No Change	awscloudformation
Auth	paddelappae5bfa6f	No Change	awscloudformation
Function	postConfirmation	No Change	awscloudformation
Function	preAuthorization	No Change	awscloudformation
Function	startTraining	No Change	awscloudformation

```
GraphQL endpoint: https://smccwaptoncrxeundldckkp7pa.appsync-api.us-east-2.amazonaws.com/graphql
GraphQL API KEY: da2-mzh7kb3bpnbw3dv2ehbcawunla

Hosted UI Endpoint: https://paddelapp24419f35-24419f35-dev.auth.us-east-2.amazoncognito.com/
Test Your Hosted UI Endpoint: https://paddelapp24419f35-24419f35-dev.auth.us-east-2.amazoncognito.com/login?response_type=code&client_id=6bp60v0j6815s736f9gg
tsdfr&redirect_uri=http://localhost:5000/

PS C:\Users\migue\PaddelApp>
```

Ilustración 36 - Amplify status - Situación PrePush.

La primera vez que se realizó la creación de la Api y de la autenticación el campo *Operation* de la tabla era *Create*. Para actualizar algún parámetro de los escogidos en la creación de cada una de las categorías es:

```
PS C:\Users\migue\PaddelApp> amplify update <category>
```

Creada la api y la autenticación, se debe construir el frontend [37]de la aplicación. La documentación encontrada para el control de errores que se pueden producir en un registro de un usuario en la siguiente referencia [38].

- Lambda Functions

Se crearon tres funciones lambda, siendo ***postConfirmation*** la única función activa en este momento, por tanto, solo se comentará brevemente la implementación de ésta:

```
PS C:\Users\migue\PaddelApp> amplify add function
```

Se siguió la siguiente configuración para la creación de la función:

```
PS C:\Users\migue\PaddelApp> amplify add function
? Select which capability you want to add: Lambda function (serverless function)
? Provide an AWS Lambda function name: paddelappe94de13b
? Choose the runtime that you want to use: NodeJS
? Choose the function template that you want to use: Hello World

Available advanced settings:
- Resource access permissions
- Scheduled recurring invocation
- Lambda layers configuration
- Environment variables configuration
- Secret values configuration

? Do you want to configure advanced settings? Yes
? Do you want to access other resources in this project from your Lambda function? Yes
? Select the categories you want this function to have access to. api, auth, function
? Select the operations you want to permit on paddelapp Query, Mutation, Subscription
? Select the operations you want to permit on paddelappae5bfa6f create, read, update, delete
? Function has 3 resources in this project. Select the one you would like your Lambda to access

You can access the following resource attributes as environment variables from your Lambda function
        API_PADDELAPP_GRAPHQLAPIENDPOINTOUTPUT
        API_PADDELAPP_GRAPHQLAPIIDOUTPUT
        API_PADDELAPP_GRAPHQLAPIKEYOUTPUT
        AUTH_PADDELAPPAE5BFA6F_USERPOOLID
        ENV
        REGION
? Do you want to invoke this function on a recurring schedule? (y/N) n
```

Ilustración 37 - Configuración de la Lambda function de AWS

- PostConfirmation

Una vez creada la función, desde la consola de AWS, se navega a “Desencadenadores” o “Triggers” dentro del grupo de usuario creado anteriormente, y como se marca en la siguiente ilustración, en el desplegable (después de haber ejecutado el comando *amplify push*) se elige la función creada.

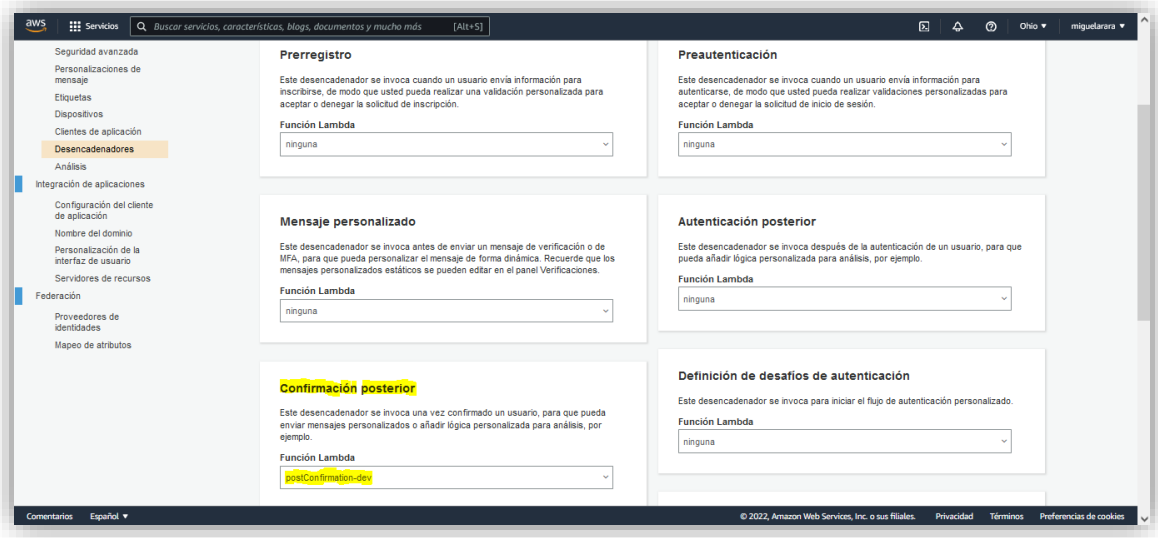

Ilustración 38 - Apartado desencadenadores en el grupo de usuarios de Cognito

Desde este subapartado, se van a incluir partes del código que deben estar en este documento para un mejor entendimiento del desarrollo que se ha implementado.

Esta es la declaración de la variable *graphqlClient* para la comunicación del *backend* con el servidor cloud de Amazon:

```
const graphqlClient = new appsync.AWSAppSyncClient({
        url: process.env.API_PADDELAPP_GRAPHQLAPIENDPOINTOUTPUT,
        region: process.env.REGION,
        auth: {
            type: "AWS_IAM",
            credentials: {
                accessKeyId: process.env.AWS_ACCESS_KEY_ID,
                secretAccessKey: process.env.AWS_SECRET_ACCESS_KEY,
                sessionToken: process.env.AWS_SESSION_TOKEN
            }
        },
        disableOffline: true
    });
```

Pudiéndose crear el usuario en la base de datos, llamando a la función **mutate** del *graphqlClient*:

```
if (event.request.userAttributes['custom:typeUser'] == "Entrenador"){

            await graphqlClient.mutate({
                mutation:mutationUser,
                variables: {
                    name: event.request.userAttributes.name,
                    username: event.userName,
                    cognitoID: event.request.userAttributes.sub,
                    email: event.request.userAttributes.email,
                    typeUser: event.request.userAttributes['custom:typeUser']
                }
            });
```

Para la anterior mutación, la creación del usuario en la DynamoDB fue necesaria la búsqueda de cómo utilizar atributos personalizados creados en AWS [39], y verificando su funcionamiento en estas preguntas en *StackOverFlow* [40], [41].

5.7 GraphQL

5.7.1 ¿Qué es GraphQL y por qué se ha elegido este lenguaje?

GraphQL es un lenguaje desarrollado por Facebook para APIs que tengan que realizar consultas y operaciones (mutaciones) con los datos existentes. Proporciona una descripción completa y comprensible de los datos de su API. Da a los clientes el poder de pedir exactamente lo que necesitan y nada más, facilita la escalabilidad de las APIs a lo largo del tiempo y permite potentes herramientas para desarrolladores [6].

Se puede tener una sintaxis flexible y muy intuitiva. Si se quiere crear un usuario la operación se llamará *createUser*, permitiendo a personas no técnicas comprender que hace el código, obviando que la petición HTTP es un POST y llega una respuesta con la información. En GraphQL la idea es modelar la capa de negocio mediante un grafo, que estará definido por un esquema.

Por tanto, con este lenguaje, se unifican las consultas a la base de datos y las peticiones red bajo un mismo servidor:

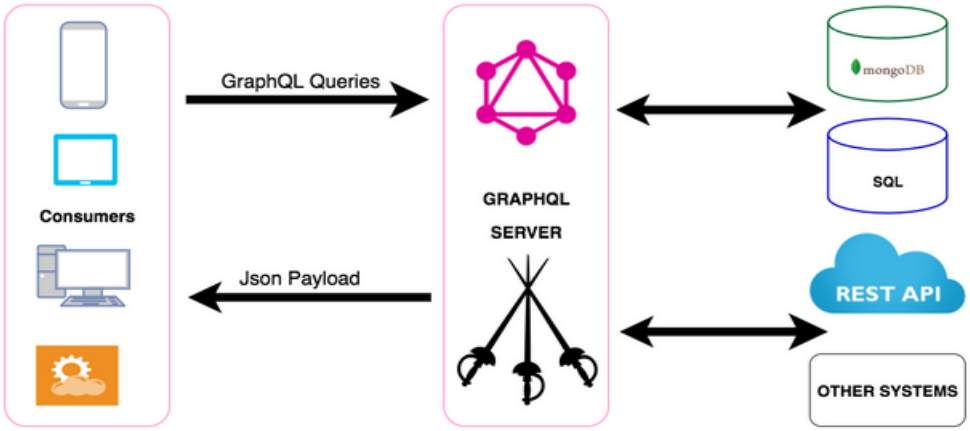

Ilustración 39 - Unificación de las peticiones y consultas y operaciones en GraphQL.

Esta información ha sido extraída de la documentación propia de GraphQL y del artículo de J. Manuel García Rozas en el blog de la empresa *Paradigma Digital* [42].

5.7.2 ¿Cómo utilizar GraphQL?

GraphQL permite que las consultas a la base de datos y las peticiones a la API se realicen bajo esta única herramienta. La implementación se explicará en Sistema de consultas y peticiones GraphQL.

En un sistema API REST[14], sólo se puede pasar un único conjunto de argumentos: los parámetros de consulta y los segmentos de la URL en tu petición. En GraphQL, sin embargo, cada campo y objeto anidado puede obtener su propio conjunto de argumentos, lo que convierte a GraphQL en un sustituto completo para realizar múltiples recuperaciones de la API. Incluso puede pasar argumentos a campos escalares, para implementar transformaciones de datos una vez en el servidor, en lugar de en cada cliente por separado. Para entender el funcionamiento se deben explicar los dos tipos principales de operaciones de esta tecnología.

5.7.2.1 Queries

Este tipo de operaciones [43] son comparables a la sigla 'R' de las operaciones CRUD[15], es decir, consultas de lectura que devuelven o listan elementos. Un ejemplo es el siguiente extraído de la página web oficial de Apollo [44]:

```
query GetAllPets {
  pets {
    name
    petType
  }
}
```

5.7.2.2 Mutations

Este tipo de operaciones [45] son comparables a las siglas 'C', 'U' y 'D' de las operaciones CRUD, es decir consultas que añaden, actualizan o borran elementos. Un ejemplo es el siguiente también extraído de la página web oficial de Apollo:

```
mutation AddNewPet ($name: String!, $petType: PetType) {
  addPet(name: $name, petType: $petType) {
    id
    name
    petType
  }
}
```

[14] Una API REST define un conjunto de funciones que los desarrolladores pueden realizar solicitudes y recibir respuestas a través del protocolo HTTP, como GET y POST.

[15] CRUD (Create, Read, Update, Delete) es un acrónimo para las maneras en las que se puede operar sobre información almacenada. Es un nemónico para las cuatro funciones del almacenamiento persistente. CRUD usualmente se refiere a operaciones llevadas a cabo en una base de datos o un almacén de datos, pero también pude aplicar a funciones de un nivel superior de una aplicación como soft deletes donde la información no es realmente eliminada, sino es marcada como eliminada a través de un estatus.

5.8 Estructura

5.8.1 Organización de carpetas

El proyecto tiene la siguiente organización de carpetas. Los archivos que existen dentro de ellas no están incluidos en este esquema. Se explicarán brevemente el contenido de las carpetas más importantes:

- PaddelApp
  - .expo
  - .expo-shared
  - .vscode
  - Amplify: Se crea automáticamente con amplify configure donde se almacena todo el *backend* y los archivos necesarios para su correcto funcionamiento.
    - Backend
      - Api/paddelapp: Contiene el archivo *schema.graphql*, siendo este el más determinante de los archivos, ya que contiene la declaración de las tablas *Type* Véase Sistema de consultas y peticiones GraphQL
      - Auth:
      - Function: Contiene las funciones lambda creadas.
  - Assets: Contiene todas las imágenes e iconos que se utilizan en la aplicación.
  - node_modules
  - src: Contiene la mayor parte del código, se encuentra la programación de todos los componentes que se han desarrollado.
    - Components
    - Config
    - Contexts
    - Graphql: Contiene los archivos .ts (TypeScript) que contienen las queries, mutaciones y subscripciones generadas automáticamente.

- Screens: En cada subcarpeta se encuentran los archivos .tsx[16] correspondientes a las pantallas y los archivos .ts .
    - Booking
    - Calendar
    - drawerContent
    - home
    - score
    - signIn
    - signUp
    - statistics
    - training
    - welcome

5.9 Desarrollo

En este apartado se incluyen partes del código necesarias para explicar el funcionamiento de este, y los comandos más relevantes que han sido ejecutados, así como la explicación de ciertas herramientas utilizadas, y visualización de los flujos de la información que tiene la app.

5.9.1 Inicio del proyecto

Para la creación del proyecto y la ejecución de los comandos[17] se ha utilizado la terminal de VS Code. El primer paso que se realiza es la instalación de Node.js descargando el paquete desde su página web. Una vez instalado Node.js se instala el paquete de Expo de forma global utilizando Node con el comando.

```
PS C:\Users\migue\PaddelApp>npm i -g expo-cli
```

El siguiente comando es el primer comando que se debe ejecutar para crear un proyecto Expo:

---

[16] Un archivo TSX es un archivo TypeScript (. TS) escrito utilizando la sintaxis JSX. Contiene código que probablemente forme parte de una aplicación móvil o de una sola página. Los archivos TSX pueden abrirse en cualquier editor de texto, pero están pensados para ser abiertos en editores de código fuente.

[17] *Todos los comandos ejecutados son desde la ruta raíz del proyecto salvo que se haga alguna instalación en alguna carpeta específica.*

```
PS C:\Users\migue\PaddelApp> expo init PaddelApp
```

Una vez ejecutado el comando se abre un cuestionario para responder una serie de preguntas acerca de la configuración del proyecto Expo. Se elige la opción marcada en azul, en lenguaje TypeScript:

```
? Choose a template: » - Use arrow-keys. Return to submit.

    ----- Managed workflow -----

    blank               a minimal app as clean as an empty canvas

>   blank (TypeScript)  same as blank but with TypeScript
configuration

    tabs (TypeScript)   several example screens and tabs using react-
navigation and TypeScript

    ----- Bare workflow -----

    minimal             bare and minimal, just the essentials to get
you started
```

Una vez creado el proyecto expo en TypeScript aparece el siguiente mensaje:

```
√ Choose a template: » blank (TypeScript) same as blank but with
TypeScript configuration

√ Downloaded and extracted project files.

📦 Using npm to install packages.

√ Installed JavaScript dependencies.


✅ Your project is ready!


To run your project, navigate to the directory and run one of the
following npm commands.


- cd prueba

- npm start # you can open iOS, Android, or web from here, or run them
directly with the commands below.

- npm run android

- npm run ios # requires an iOS device or macOS for access to an iOS
simulator

- npm run web
```

El siguiente comando se debe ejecutar cada vez que el código deba ser testeado en el dispositivo móvil. Aparece un código QR, que se debe escanear con dicho dispositivo:

```
PS C:\Users\migue\PaddelApp> npm start
```

```
> @ start C:\Users\migue\PaddelApp
> expo start

   There is a new version of expo-cli available (5.0.3).
   You are currently using expo-cli 4.12.10
   Install expo-cli globally using the package manager of your choice;
   for example: `npm install -g expo-cli` to get the latest version

Starting project at C:\Users\migue\PaddelApp
Developer tools running on http://localhost:19002
Opening developer tools in the browser...
Some dependencies are incompatible with the installed expo package version:
 - @react-native-community/netinfo - expected version: 6.0.2 - actual version installed: 7.1.6
 - @react-native-picker/picker - expected version: 2.1.0 - actual version installed: 2.2.1
Your project may not work correctly until you install the correct versions of the packages.
To install the correct versions of these packages, please run: expo install [package-name ...]
Starting Metro Bundler

› Metro waiting on exp://192.168.1.48:19000
› Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

› Press a | open Android
› Press w | open web

› Press r | reload app
› Press m | toggle menu
```

Ilustración 40 - Pantallazo de la consola mostrando el código QR para acceder a la app desde el móvil.

Y se publica en la dirección localhost:19002:

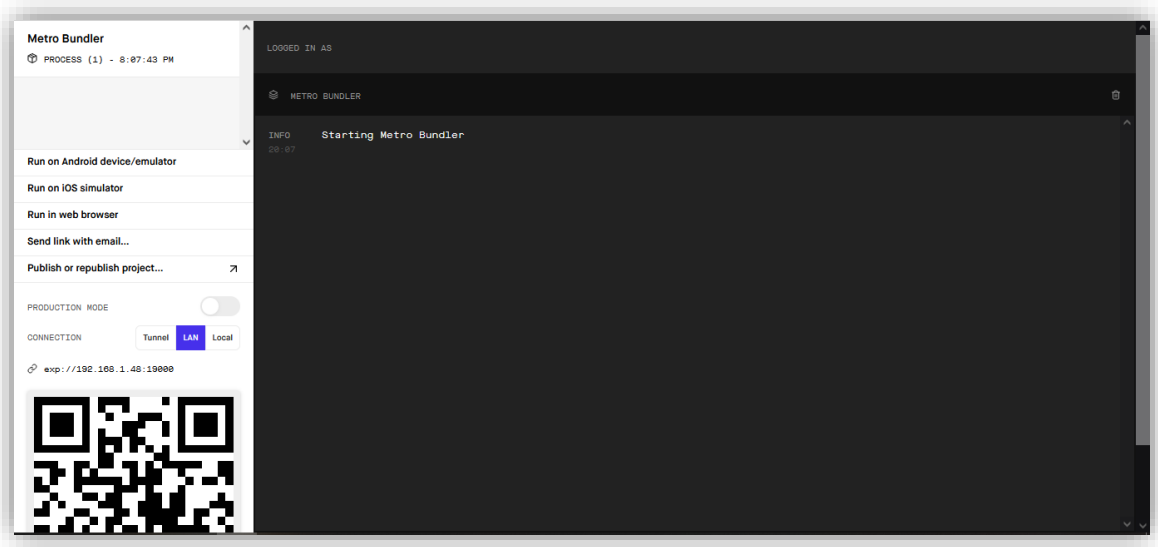

Ilustración 41 - Imagen de la ventana del navegador que se abre cuando se ejecuta el comando npm start.

Una vez ejecutada y publicada la aplicación inicial, ya se puede comenzar a testear el código visualizando el resultado en el dispositivo físico:

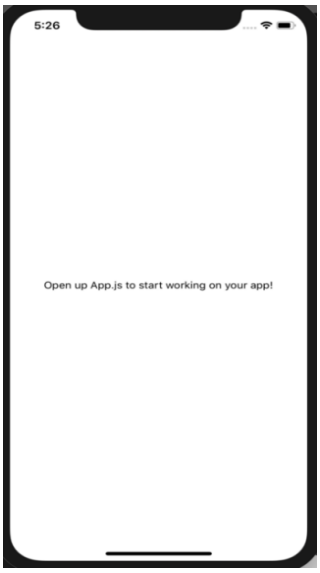

Ilustración 42 - Pantalla inicial de Expo y React Native

5.9.2 Diagrama de Navegación

El diagrama adecuado para la comprensión de este proyecto es un diagrama de navegación entre pantallas. Cada una de estas pantallas tienen asociadas un archivo escrito en TypeScript (.tsx) con sus funcionalidades, y un archivo TypeScript(.ts) para los estilos de cada uno de los componentes. Los diagramas que se van a mostrar han sido con la herramienta online *Diagram Editor* [46].

5.9.2.1 Diagrama de navegación del Entrenador

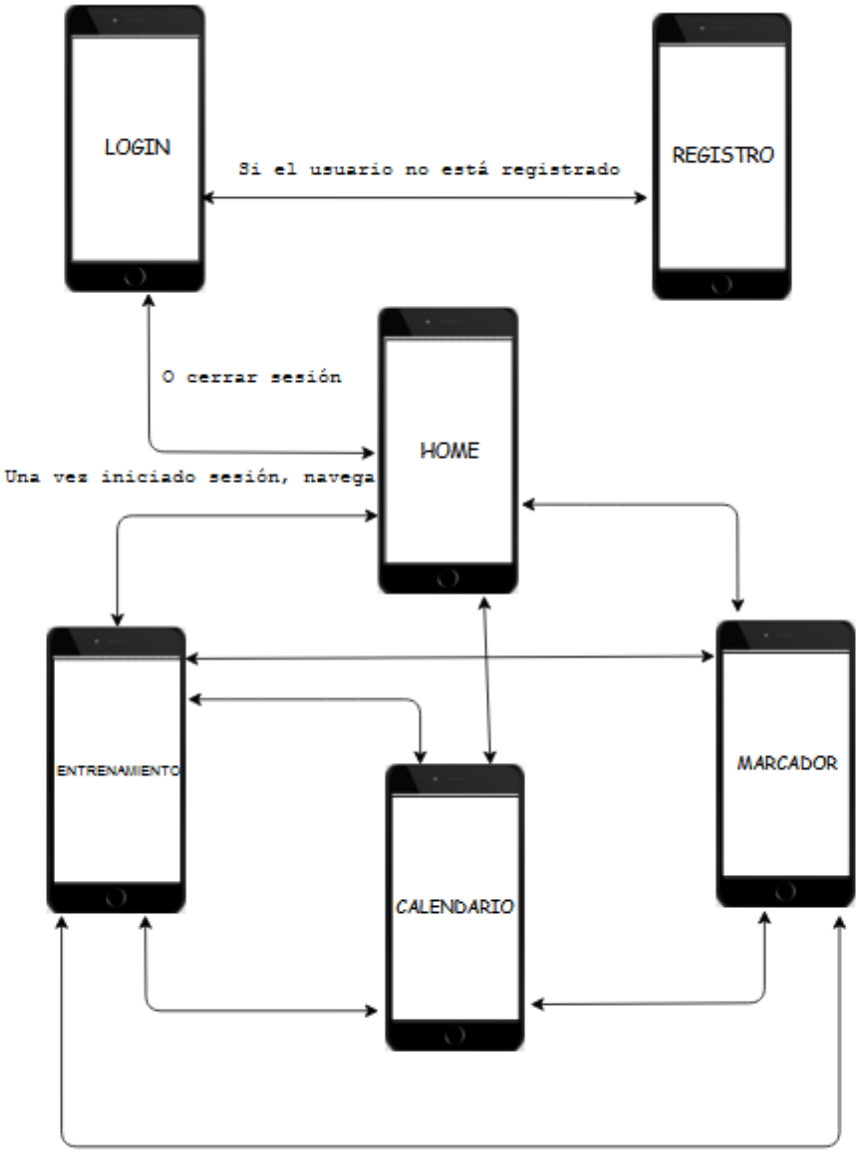

Ilustración 43 - Diagrama de navegación del Entrenador.

5.9.2.2 *Diagrama de navegación del Jugador*

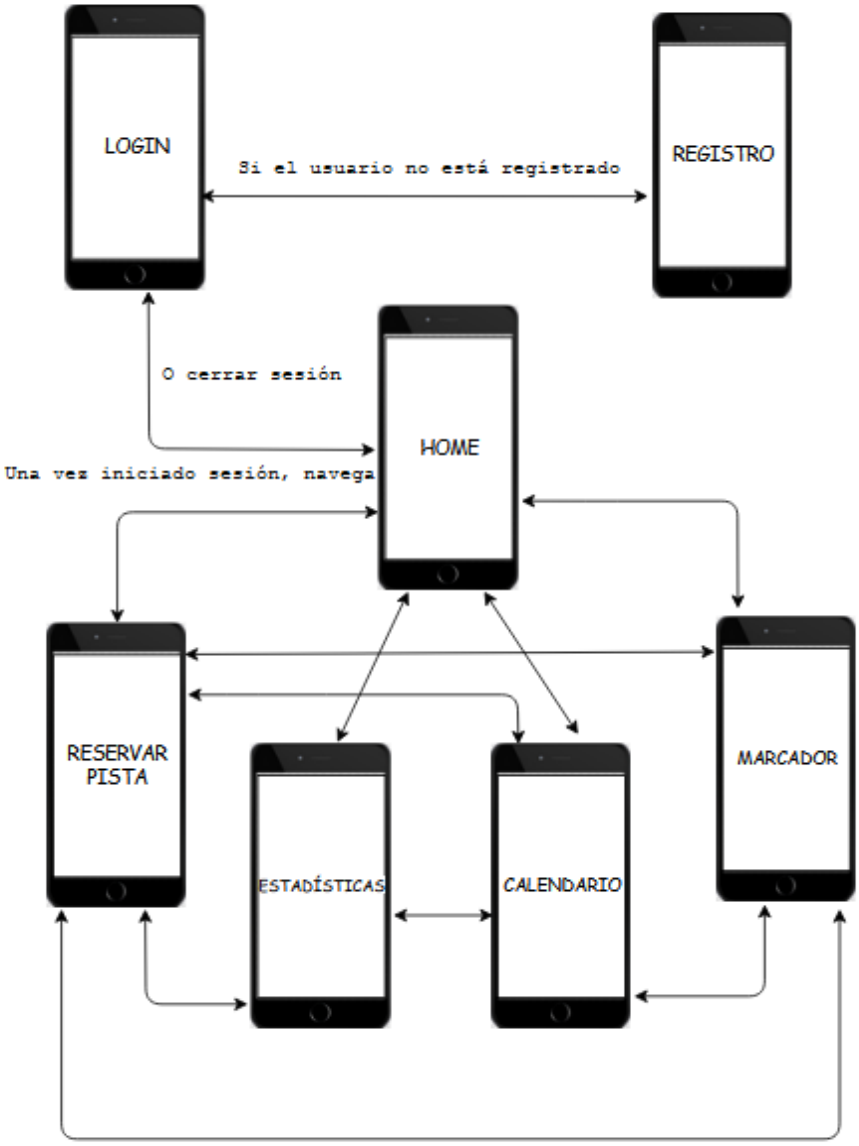

Ilustración 44 - Diagrama de navegación del Jugador.

5.9.3 Funcionamiento

Para un mayor entendimiento de la construcción la aplicación se van a explicar algunas funcionalidades del código que pueden ser interesantes. Para realizar la aplicación ha sido necesaria la investigación y aprendizaje de las tecnologías mencionadas anteriormente. Aprender a implementarlas, utilizarlas y analizarlas ha sido uno de los mayores retos de este proyecto:

- Para la construcción y el desarrollo de la aplicación en React Native en lenguaje TypeScript, sirvió de apoyo este blog [47] .
- Para el uso de AWS DynamoDB y su implementación en una aplicación React Native, ha sido muy importante este post en la página mundialmente conocida *Medium* [26].

- Aunque no se utilizara MySQL, este artículo de AsapDevelopers fue esencial para comprender el funcionamiento y conexión de React Native con un backend en Node.js [48].

5.9.3.1 DataFlow del Registro de un usuario

Cuando un usuario se registra en la aplicación, se da de alta en el grupo de usuarios de Cognito. A continuación, el usuario introduce el código OTP en la app que ha sido enviado al e-mail; en ese momento se lanza la función lambda que, gracias al lenguaje GraphQL, es capaz de crear el usuario en la tabla de usuarios correspondiente en la DynamoDB:

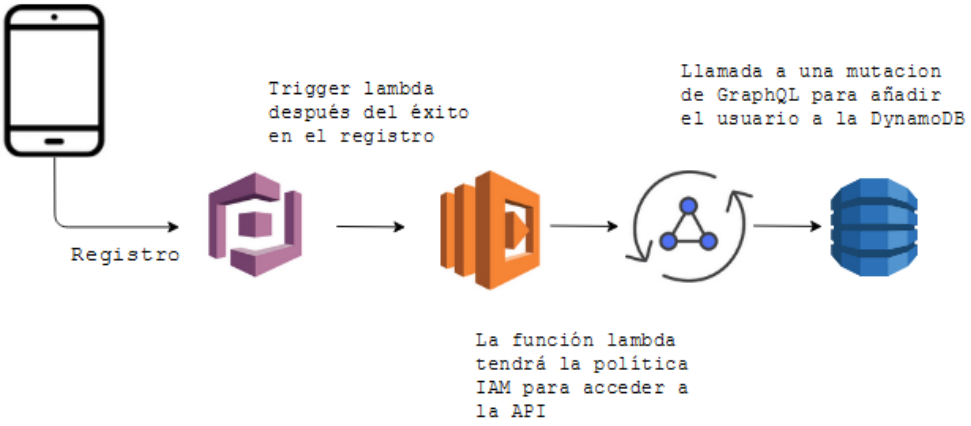

Ilustración 45 - Flujo de registro de un usuario en la aplicación.

Para diferenciar entre los tipos de usuario en la aplicación se creó un atributo personalizado desde la consola de Amazon, dentro de Cognito, en el área Atributos dentro del grupo de usuarios de la aplicación:

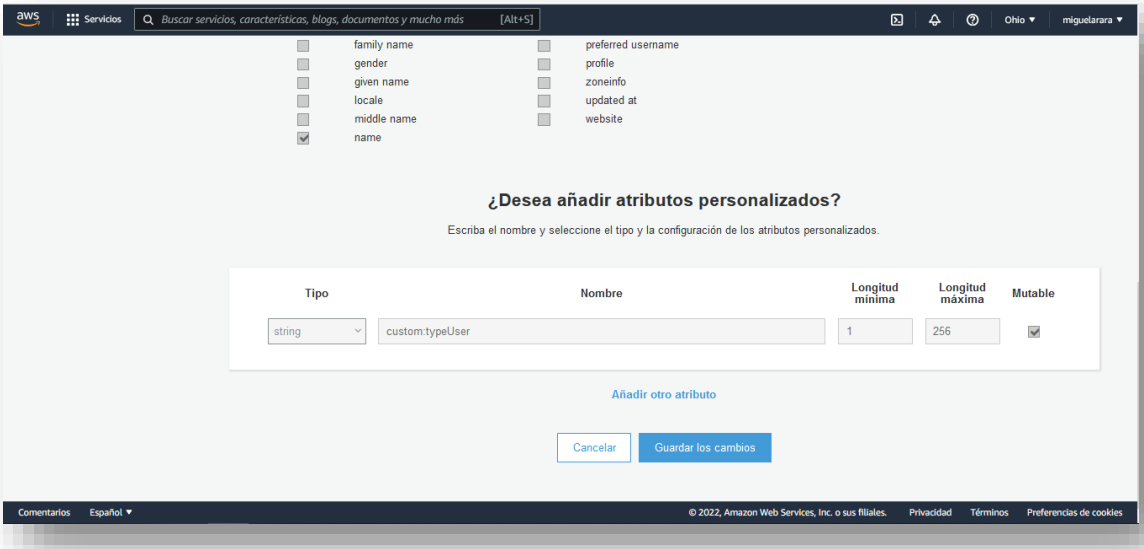

Ilustración 46 - Atributo personalizado de Cognito

5.9.3.2 Sistema de consultas y peticiones GraphQL

Este apartado puede que sea de los más importantes para entender el funcionamiento del modelo de datos. En la siguiente extracción del código que pertenece al archivo **schema.graphql** dentro de la ruta del proyecto *PaddelApp/amplify/backend/api/paddelapp*, un ejemplo completo de la sintaxis utilizada para la inclusión de una tabla en la base de datos.

```
type User
  @model
  @searchable
  @key(fields: ["username"])
  @auth(
    rules: [
      { allow: private, operations: [read,create,update,delete] }
      { allow: owner, ownerField: "username", operations: [update] }
      { allow: private, provider: iam, operations: [read, create, update,
delete] }
    ]
  ) {
  id: ID!
  cognitoID: String!
  username: String!
  name: String!
  email: AWSEmail!
  typeUser: String!
  trainings: [PlayerTraining] @connection (keyName:"byUser", fields:
["username"])
}
```

El tipo User (en la aplicación es el usuario Entrenador), se declara **type** en el lenguaje GraphQL y es similar a una tabla o entidad. Cada uno de los tipos contienen atributos (pudiendo ser otros tipos como se muestra por ejemplo con el último atributo *trainings,* que es de tipo PlayerTraining).

Para solucionar la relación de dos tablas N:M (muchos usuarios pueden tener muchos entrenamientos) se debe, como en SQL, crear un tipo nuevo intermedio. El atributo *trainings* en el tipo User **guarda los entrenamientos que ha realizado el usuario** (entrenador), al igual que el usuario Player (Jugador), muchos jugadores pueden realizar muchos entrenamientos. A continuación, se presentan los dos tipos, Training y PlayerTraining:

```
type PlayerTraining
  @model (queries: null)
  @key(name:"byTraining", fields: ["trainingID"])
  @key(name:"byPlayer", fields:["playerUsername","createdAt"])
  @key(name:"byUser", fields:["userUsername","createdAt"])
  @auth(
    rules: [
      { allow: owner, ownerField: "owners", operations:
[read,create,update,delete] }
      { allow: private, provider: iam, operations: [read, create, update,
delete] }
    ]
  )
  {
    id: ID!
    trainingID: ID!
    createdAt: String!
    playerUsername: String!
    userUsername: String!
    players: [String!]!
    training: Training! @connection(fields: ["trainingID"])
    user: User @connection(fields: ["userUsername"])
    player: Player @connection(fields: ["playerUsername"])
}

type Training
@model
@auth(
    rules: [
      { allow: owner, ownerField: "owners", operations:
[read,create,update,delete] }
      { allow: private, provider: iam, operations: [read, create, update,
delete] }
    ]
  )
{
  id: ID!
  status: TrainingStatus!
  trainer: String!
  player: String!
  owners: [String!]!
  winner: String
  hit: [TrainingHit]
  users: [PlayerTraining] @connection(keyName: "byTraining", fields: ["id"])
}
```

- Complicaciones

Durante el desarrollo del proyecto, la sintaxis del *schema.graphql* ha sido de las más complejas de implementar, debido a que dependiendo de qué **versión** de la librería **aws-amplify** haya sido instalada la sintaxis cambia. Esto generó complicaciones debido a que:

- La declaración de una clave de partición para un tipo difiere entre la documentación que hay en AWS Amplify y la versión que se tiene instalada en el proyecto.
- Cuando se ejecuta el comando *amplify push* y se produce un error habiendo únicamente modificado el archivo *schema.graphql* en ocasiones no se sabe por qué ha fallado.
- Se tuvo un error que retrasó el proyecto más de una semana; parecía simplemente una falta de permisos del usuario que realizaba la petición. Finalmente se resolvió averiguando que la versión de la librería instalada de **aws-amplify** no era la correcta, y la sintaxis que se había implementado no funcionaba correctamente.

- Petición

A continuación, se presenta una extracción de código de una **petición que realiza un entrenador** para la **mutación** que realiza la **creación** de un entrenamiento:

```
const infoInitialTraining ={
        status: 'ACTIVO',
        trainer: user.username,
        owners: selectedPlayers,
        player: element,
      }

try{
    const newTraining = await API.graphql({
      query:createTraining,
      variables: {input: infoInitialTraining},
      authMode: GRAPHQL_AUTH_MODE.AWS_IAM})
}catch(err){
    console.log('Error creación de entrenamiento',err)
}
```

Esta es otra extracción de una **consulta** (query) que **devuelve la sesión de entrenamiento**, y guardando los *hits*[18] del entrenamiento que se está haciendo:

```
const trainingSession = await API.graphql({
      query:getTraining,
        variables: {id: id},
        authMode: GRAPHQL_AUTH_MODE.AWS_IAM
    })

hits = trainingSession["data"]["getTraining"]["hit"] || []
```

Y por último esta es una **mutación** que **actualiza un entrenamiento**, con los *hits* de un jugador:

[18] *Hits* es el nombre que se ha elegido para definir los aciertos y los fallos que realizan los jugadores en los entrenamientos.

```
const updateInfo = {
      id: id,
      hit: hits
    }

try {
  const updateTrainingPlayer = await API.graphql({
    query:updateTraining,
    variables: {input: updateInfo},
    authMode: GRAPHQL_AUTH_MODE.AWS_IAM})
} catch (error) {
  console.log('error update',error)
}
```

- Codificación

Para aplicar distintos estilos en un SO u otro, porque un componente nativo lo requiera, se utiliza *Platform* API de *React Native* con este código para que la aplicación se muestre al usuario de una forma u otra:

```
import { Platform, StyleSheet } from 'react-native';

const styles = StyleSheet.create({
  container: {
    flex: 1,
    ...Platform.select({
      ios: {
        backgroundColor: 'red'
      },
      android: {
        backgroundColor: 'green'
      }
    })
  }
});
```

React Native proporciona una **gran cantidad de funcionalidades y componentes** que funcionan perfectamente, y con ellos es posible la creación de componentes personalizados. En el siguiente ejemplo se presenta el *ButtonComponent*, **botón reutilizable** que es visualizado en distintas pantallas de la aplicación. Primero la creación de la función que devuelve dicho botón:

```
type ButtonProps = {
    title: string;
    loading?: boolean;
} & TouchableOpacityProps;

export default function ButtonComponent({ title, style, loading, ...props }:
ButtonProps): ReactElement {
    // console.log(loading)
    return (

        <TouchableOpacity style={style} disabled={loading} {...props}>
            {loading ? (
                <ActivityIndicator color="white"></ActivityIndicator>
            ) : (
                <Text style={styles.buttonText}>{title}</Text>
            )}
        </TouchableOpacity>
    );
}
```

Y a continuación mostrando cómo se utiliza en una de las pantallas. Extracción de código de la pantalla de inicio de sesión:

```
<LinearGradient colors={['#6495ED', 'cyan']} style={styles.signIn}>

       <ButtonComponent
                title="Iniciar sesión"
                loading={loading}>
       </ButtonComponent>

</LinearGradient>
```

Este botón se puede ver en el subapartado Inicio de sesión o Login.

5.9.4 Manual de Usuario

En este apartado se va a desglosar el manual de usuario. Se hará una breve explicación de cada una de las pantallas, en qué consisten y cómo se utilizan, y se adjuntará una ilustración, mostrando el diseño para cada una de ellas.

Comenzando por la primera pantalla, para que el usuario comience a interactuar con la aplicación deberá pulsar el botón “Dale caña”. Esta pantalla se ha llamado *Welcome Screen*:

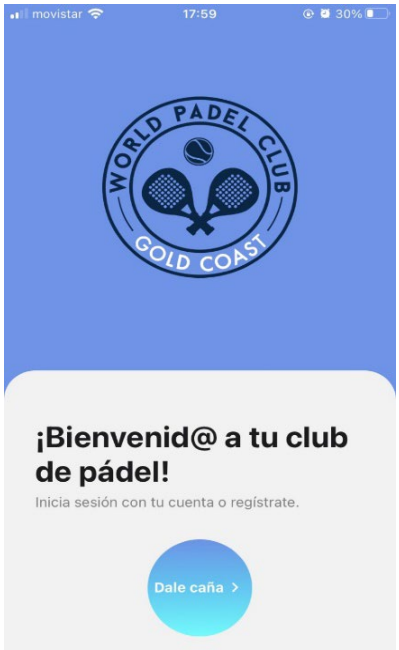

Ilustración 47 Pantallazo Welcome Screen

5.9.4.1 Inicio de sesión o Login

En esta primera pantalla el usuario podrá iniciar sesión, introduciendo el nombre de usuario y contraseña. Si cualquiera de los dos campos introducidos es incorrecto, aparecerá un mensaje de alerta. Si el usuario todavía no se ha registrado, podrá hacerlo pulsando en el botón "Regístrate".

El apartado de la documentación para poder implementar *textInputs* en React Native fue el siguiente [49].

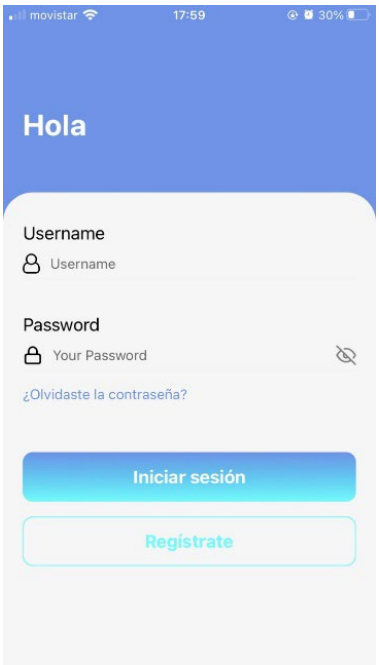

Ilustración 48 - Pantallazo Login Screen

5.9.4.2 Registro

Esta es la pantalla de registro de usuario. La vista es una *Scroll View*, es decir, las tres imágenes siguientes pertenecen a la misma pantalla de registro:

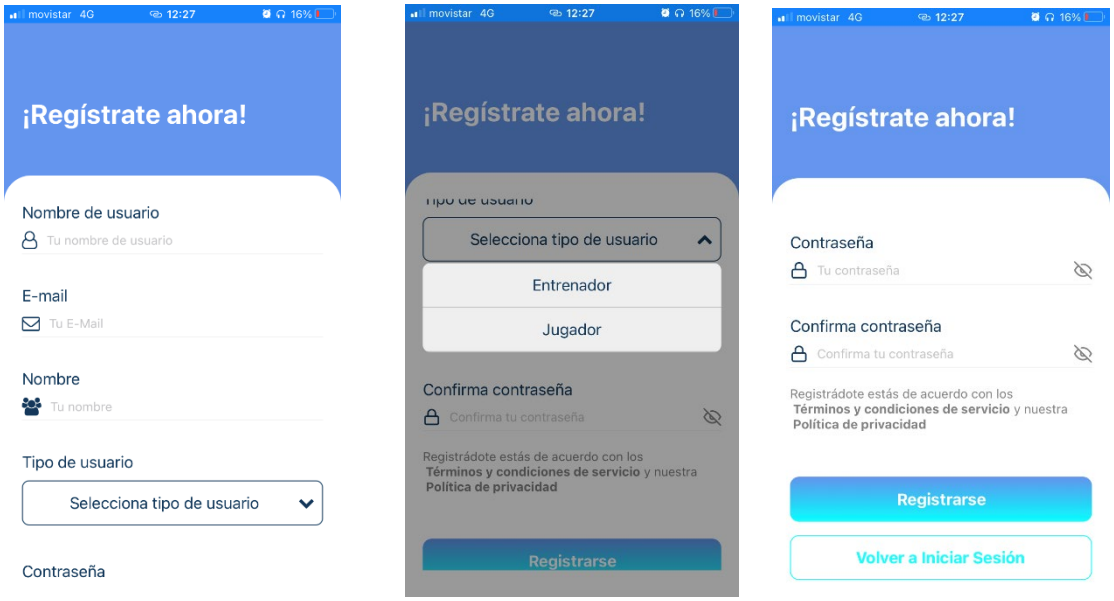

Ilustración 49 - Pantallazos de la pantalla de registro.

Para realizar el registro correctamente el usuario deberá seguir los siguientes pasos:

1. Introducir un nombre de usuario que no exista en la base de datos.
2. Introducir un e-mail.
3. Introducir su nombre.
4. Debe elegir el tipo de usuario que es:
    a. Entrenador.
    b. Jugador.
5. Introducir una contraseña de al menos 8 caracteres.
6. Introducir la misma contraseña de nuevo en el campo “Confirma contraseña”.
7. Pulsar el botón “Registrarse”.
    a. Una vez pulsado el botón y si los datos se han introducido correctamente, la aplicación mandará al usuario a una pantalla OTP, es decir, tendrá que introducir un código de confirmación que se le ha enviado al e-mail que ha introducido.
    b. Si el usuario introduce el código correcto, el registro se habrá realizado correctamente, y volverá a la pantalla de inicio de sesión.

8. Si por el contrario el usuario decide pulsar el botón “Volver a Iniciar Sesión”, el usuario volverá a la pantalla de inicio de sesión.

5.9.4.3 Menú de navegación del jugador

Para implementar la navegación en una aplicación React Native existen distintas opciones que se muestran en la documentación, por ejemplo, una barra en el pie de la pantalla; véase [10]. En esta aplicación se ha optado por desarrollar un *drawerContent*, que se explica en el siguiente parráfo.

Una vez el usuario ha iniciado sesión, se abrirá la pantalla home. En todas las pantallas, si se mueve el dedo de izqda. a dcha. o bien pulsando el icono de las “tres rallas” de la cabecera (de momento en la única pantalla que no hay cabecera es en el marcador) se abrirá un menú desplegable de navegación en el que aparece el nombre de usuario, el tipo de usuario que es, las pantallas por las que puede navegar y cerrar sesión.

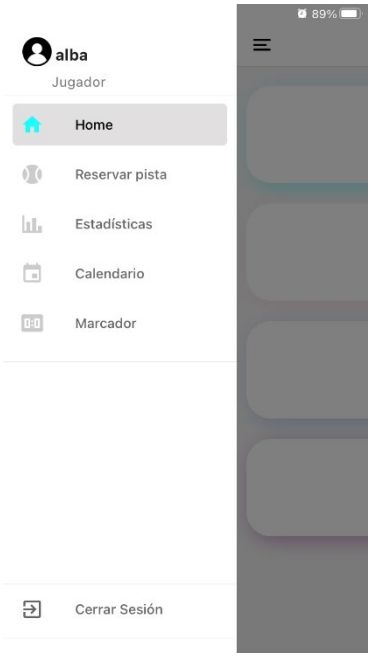

Ilustración 50 - Barra lateral menú.

5.9.4.4 Pantallas

- Pantalla principal o “Home”

Esta es la primera pantalla que aparece una vez que el usuario ha iniciado sesión correctamente. La pantalla Home es una pantalla de navegación; esto quiere decir que desde esta pantalla el usuario podrá navegar por todas las pantallas que existen en la aplicación.

Hay dos tipos de pantalla principal. Dependiendo de si es un Entrenador o un Jugador, el usuario visualizará una pantalla u otra. Dependiendo del usuario, se tiene una navegación u otra, y en la barra del menú lateral aparecen las pantallas de cada tipo de usuario.

➢ Pantalla Home – Entrenador

Esta es la pantalla principal del entrenador. Como se puede ver en la ilustración, el entrenador podrá navegar a las siguientes pantallas:

- Entrenamiento
- Calendario
- Marcador

El usuario puede pulsar el icono que hay arriba a la izquierda (icono de las tres rallas) o también podrá hacer un movimiento con el dedo de izquierda a derecha y se abrirá el menú de navegación de usuario. Puede ver su nombre, el tipo de usuario que es y navegar entre las distintas pantallas que hay en aplicación. En este menú está el botón que debe pulsar el usuario si quiere cerrar su sesión.

Ilustración 39 - Pantalla Home Entrenador

➢ Pantalla Marcador

En esta pantalla el Entrenador o el Jugador tendrá un marcador muy simple pudiendo realizar una cuenta de puntos rápidamente sin que se guarde en la base de datos. La pantalla se divide en dos partes y el funcionamiento consiste en sumar 1 punto cada vez que se toque cada una de las mitades de la pantalla, se restará cuando se pulse el signo de restar y se renovará la cuenta cuando se pulse el botón “reset”.

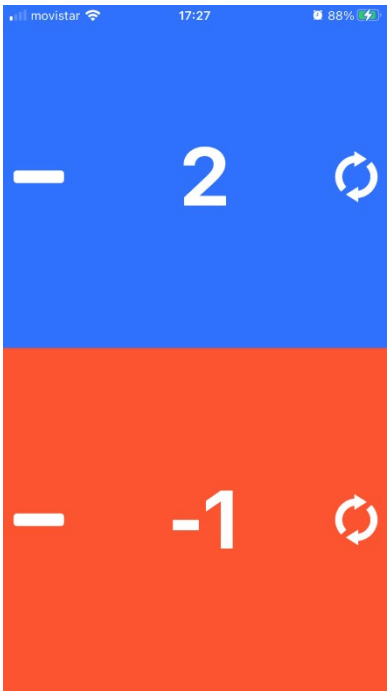

Ilustración 40 - Pantalla marcador

- Pantalla Entrenamiento

La pantalla Entrenamiento, exclusiva del tipo de usuario Entrenador, se podrá realizar los entrenamientos de los jugadores, siendo esta una de las primeras funcionalidades de la aplicación:

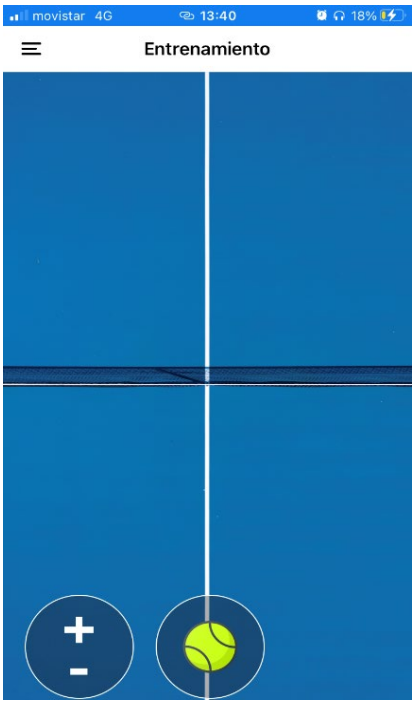

Ilustración 41- Pantalla inicial entrenamiento

En el botón de abajo a la izquierda, el entrenador podrá incluir los jugadores (usuarios que se han registrados en la aplicación con tipo de usuario *Jugador*), que vayan a realizar el entrenamiento,

desde 1 hasta 4 jugadores. Una vez añadidos los jugadores, el entrenador podrá pulsar el botón con el icono de la pelota de tenis, y el entrenamiento habrá comenzado.

Ilustración 42 - Añadir jugadores al entrenamiento.

Para implementar la pantalla modal anterior fue necesario el apoyo en la documentación y la instalación de la librería que se referencia aquí [50].

Y por último, el entrenamiento comenzará, y cuando el usuario pulse STOP parará. Cada vez que se pulse a uno de los jugadores, aparecerá un modal en el que el entrenador podrá pulsar **una** de las opciones, y pulsará el botón "Listo". El timer, que se extrajo de esta librería [51] comenzará a contar hasta que se pulse en el botón "STOP".

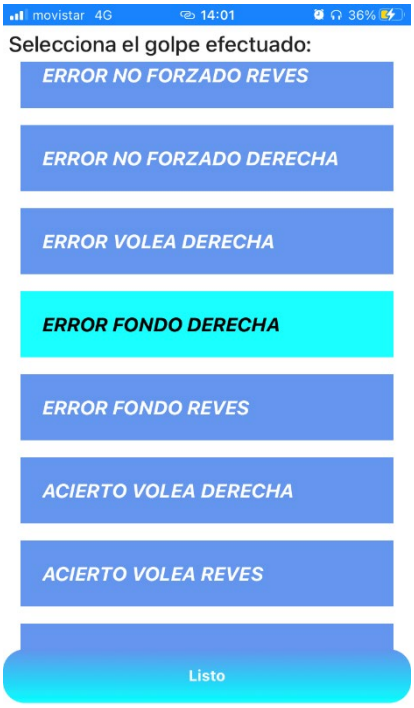

Ilustración 43 - Pantalla Elección del golpe que ha hecho el jugador.

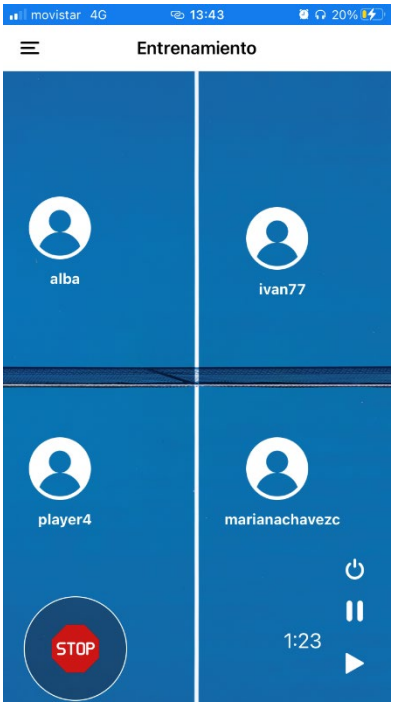

Ilustración 44 - Entrenamiento ACTIVO

➢ Pantalla Home – Jugador

Esta es la pantalla principal del jugador. Como se puede ver en la ilustración, el jugador podrá navegar a las siguientes pantallas:

- Reservar pista.
- Estadísticas.
- Calendario.
- Marcador.

El usuario puede pulsar el icono que hay arriba a la izquierda (icono de las tres rallas) o también podrá hacer un movimiento con el dedo de izquierda a derecha y se abrirá el menú de navegación de usuario. Puede ver su nombre, el tipo de usuario que es y navegar entre las distintas pantallas que hay en aplicación. En este menú está el botón que debe pulsar el usuario si quiere cerrar su sesión.

Ilustración 45 - Pantalla Home Jugador

➢ Pantalla Reservar Pista

El Jugador puede reservar pista desde esta pantalla. Deberá pulsar el botón "*Reservar Pista*", elegir el día en el calendario y una hora de las sugeridas:

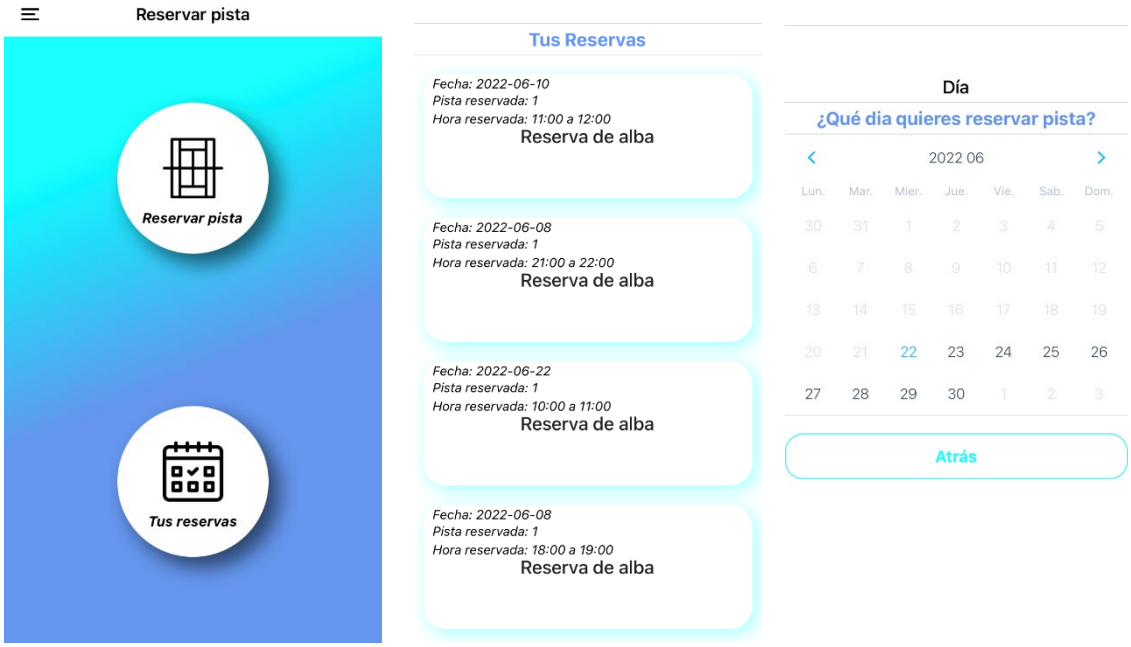

Ilustración 46 - Pantallas de la Reserva de Pista

Si el usuario está realizando una reserva, después de elegir el día deberá elegir cuanto tiempo quiere reservar la pista:

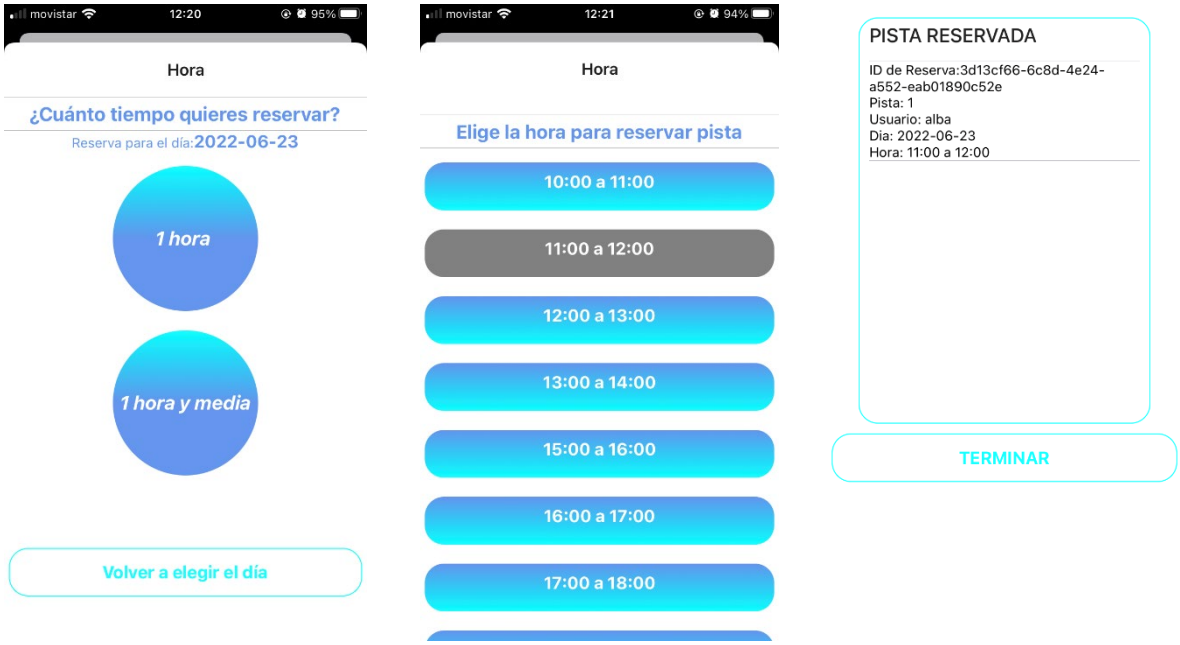

Ilustración 47 - Pantallas de reserva de pistas.

➢ Pantalla Estadísticas

Esta es la pantalla provisional para la pantalla estadísticas. Se deberán aplicar unas modificaciones en estilo de los componentes para mejorar la UI[19]:

Ilustración 48 - Pantallas Estadísticas

[19] User Inteface o Interfaz de Usuario.

6. Presupuesto

El presupuesto que se ha estimado para la creación, construcción y mantenimiento esta aplicación es el siguiente:

En una primera parte distinguimos el coste de los distintos dispositivos utilizados para poder realizar el desarrollo de la aplicación. En la segunda tenemos los gastos de mano de obra. Los servicios de AWS han sido estimados desde *AWS pricing calculator*, herramienta que ofrece Amazon para la estimación de sus precios. Cabe mencionar que es una estimación aproximada, no exacta:

PRESUPUESTO ESTIMADO PARA EL DESARROLLO DE LA APLICACIÓN **(para un año sin contar mantenimiento posterior)**			
Material	**Presupuesto**	**Presupuesto (con IVA general 21%)**	**Información**
Laptop Lenovo 520	520€	629,20€	Equipo informático que se ha utilizado para la construcción de la aplicación.
iPhone SE 2021	400€	484€	Equipo móvil para testear el desarrollo.
Mano de obra	**Presupuesto (salario bruto por hora)**	**Presupuesto (salario bruto por hora con IVA general)**	**Horas trabajadas**
Desarrollador Junior Full-Stack	3750€ (12,50€/hora)	4.537,50€ (15,125€/hora)	300 (horas TFG)
Desarrollador Junior Full-Stack (Horas dedicadas externas al TFG, dedicadas a mantenimiento y perfeccionamiento)	20.000€ (12,50€/hora)	24.200€ (15,125€/hora)	1600 (horas 10 meses)
Servicios de AWS	**Presupuesto estimado**	**Presupuesto estimado (CON I.V.A reducido (10%)**	**1, 13 USD = 1 €**
Amazon EC2 (máquina virtual)	Mensual: 64,54 USD	70,94 USD	
Amazon DynamoDB	Inicial: 180 USD Mensual:27,39 USD	Inicial: 198 USD Mensual:30,12 USD	
AWS Lambda	Incluida en la capa gratuita	Incluida en la capa gratuita	
Consola de AWS Amplify	Mensual: 4380,32 USD	Mensual: 4818,32 USD	
TOTAL	**71.402,25 €**	**89.050,44 €**	

Esta es la estimación de precios que te ofrece AWS si quieres calcular el coste que va a suponer realizar el proyecto software que se desee con los servicios prestados elegidos:

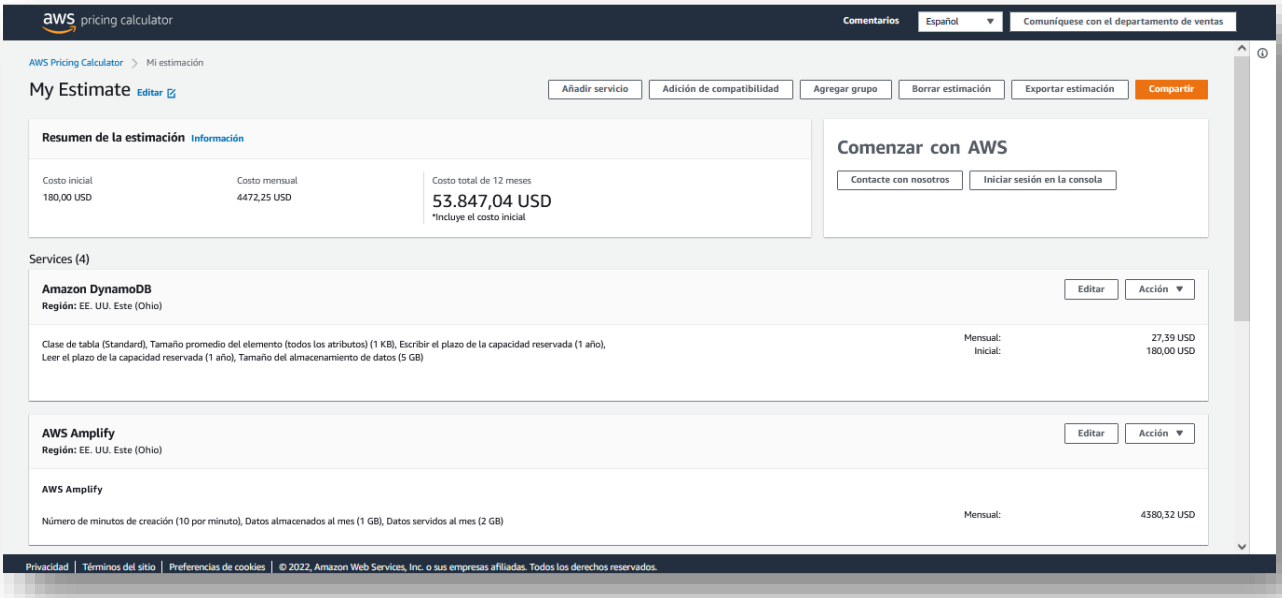

Ilustración 49 -Estimación de precios con AWS pricing

7. Conclusiones

Las primeras impresiones una vez terminada la primera versión de la aplicación son muy positivas. Se han conseguido todos los objetivos enumerados en el apartado Objetivos por conseguir. También ha sido un proceso muy bonito el simple hecho de juntar mis dos pasiones, el pádel y la informática, y con los conocimientos adquiridos todos estos años, poder haber realizado un producto software tangible, útil y a mi parecer, atractivo.

En cuanto a cuestiones técnicas, se debe decir que realizar cambios de estilos en la app puede llegar a ocupar bastante tiempo; una vez se controla el lenguaje y como dividir la pantalla a personalizar este tiempo disminuye.

Abordar un lenguaje nuevo ha sido todo un reto, beneficioso para mi aprendizaje y adentrarme en este nuevo mundo del desarrollo móvil me ha permitido ser contratado por una empresa para el desarrollo y mantenimiento de aplicaciones móviles.

Trabajo futuro

Se han decidido una serie objetivos futuros a realizar para que este proyecto académico derive en un proyecto real para la puesta en producción de la aplicación.

- Funcionalidades generales de la Aplicación:
    - Mostrar un tutorial al iniciar sesión por primera vez, para que el usuario pueda ver para qué se utiliza cada opción en cada una de las pantallas.
    - Pantalla de ajustes para cada tipo de usuario.
    - Presentación y diseños más elaborados de cada una de las pantallas.
    - Realizar pagos a través de la aplicación.
    - Traducible a los lenguajes europeos dónde el pádel está en auge y puede ser clientes potenciales.
    - Apartado para compra de productos que sean vendidos por el club al cual se le ha vendido la app.
- Para el usuario Jugador:
    - Poder introducir su nivel de Juego.
    - Que el Jugador pueda organizar partidos/torneos automáticamente con la aplicación, añadiendo los jugadores que quieran apuntarse.
    - El jugador suba los resultados de los partidos a la aplicación.
    - Añadir foto de perfil.

    - Poder visualizar un histórico de las estadísticas de todos los entrenamientos del año en una gráfica.

- Para el usuario Entrenador:
    - Incluir sus horas trabajadas semanalmente, incluyendo horas extras por clases particulares.
    - Podrán realizar un seguimiento y *feedback* de los alumnos y sus entrenamientos. Para los entrenamientos de alto rendimiento.
    - También podrán incluir su periodo de vacaciones
    - Guía inicial para nuevos usuarios: Titulo y descripción breve de cada pantalla durante 6 segundos.

8. Bibliografía

[1] J. Trullols, «Los otros mercados del pádel: Suecia, Italia, Francia, Bélgica y Países Bajos. http://www.padelspain.net/padel-amateur/16868/El-crecimiento-de-la-popularidad-del-padel-en-el-mundo,» 2021.

[2] F. E. d. Pádel, «Información sobre las lilencias. https://www.padelfederacion.es/Datos_Federacion.asp?Id=0.,» 2022.

[3] R. A. Española, «Definición de pádel. https://dle.rae.es/p%C3%A1del,» 2021.

[4] F. R. Moya, «¿Cuánto ha crecido el pádel en los últimos años? https://www.analistaspadel.com/cuanto-ha-crecido-el-padel-en-los-ultimos-anos/,» 26 diciembre 2021.

[5] R. Native, «React Native Web https:/reactnative.dev/».

[6] GraphQL, «GraphQL. https://graphql.org/».

[7] M. Arará, «Repositorio de la aplicación PaddelApp en GitHub https://www.github.com/reptiliano37/PaddelApp,» 2022.

[8] «https://nodejs.org/es/,» Node JS. [En línea].

[9] D. A., «¿Qué es npm? Una introducción básica para principiantes. https://www.hostinger.es/tutoriales/que-es-npm,» Jul 08, 2021.

[10] «JSX - https://reactjs.org/docs/jsx-in-depth.html».

[11] «Componentes de React Native - https://reactnative.directory/».

[12] «Flutter VS React Native - Enlace 1. https://stackoverflow.com/questions/69750918/can-i-test-my-flutter-app-from-vscode-windows-on-my-physical-iphone-device».

[13] «Flutter VS React Native - Enlace 2 o https://fireart.studio/blog/flutter-vs-react-native-what-app-developers-should-know-about-cross-platform-mobile-development/#:~:text=In%202021%2C%20there%20are%20two,chosen%20by%20only%2018%25!».

[14] «Flutter Vs React Native - Enalce 3 https://nix-united.com/blog/flutter-vs-react-native/».

[15] M. Arará, «Pregunta en StackOverFlow. https://stackoverflow.com/questions/69750918/can-i-test-my-flutter-app-from-vscode-windows-on-my-physical-iphone-device,» 2021.

[16] R. Native, «React Navigation https://reactnavigation.org/docs/getting-started/».

[17] Expo, «Documentación de Expo. https://docs.expo.dev/».

[18] AWS, «¿Qué es AWS? - https://aws.amazon.com/es/what-is-aws/,» [En línea].

[19] «AWS Amplify - https://aws.amazon.com/es/amplify/?c=m&sec=srv».

[20] «Amplify Studio - https://aws.amazon.com/es/amplify/studio/».

[21] «AWS CLI - https://docs.amplify.aws/cli/,» [En línea].

[22] «Amplify Libraries - https://aws.amazon.com/es/amplify/features/#Tools».

[23] «Amplify Hosting - https://aws.amazon.com/es/amplify/hosting/».

[24] «AWS Authentication - https://aws.amazon.com/es/amplify/authentication/».

[25] «Modelo de datos AWS Amplify - https://aws.amazon.com/es/amplify/datastore/».

[26] «Preparar DynamoDB AWS (IAM roles, React Native, access keys) https://betterprogramming.pub/how-to-use-aws-dynamodb-in-react-70b55ffff93e».

[27] «AWS Cognito - https://aws.amazon.com/es/cognito/?did=ft_card&trk=ft_card,» [En línea].

[28] «AWS AppSync - https://aws.amazon.com/es/appsync/resources/,» [En línea].

[29] «AWS CloudWatch - https://aws.amazon.com/es/cloudwatch/?c=m&sec=srv».

[30] «AWS lambda - https://docs.aws.amazon.com/es_es/lambda/latest/dg/welcome.html,» [En línea].

[31] «AWS Casos de éxito - https://aws.amazon.com/es/amplify/customers/?nc=sn&loc=7,» [En línea].

[32] «Neiman Marcus - https://www.neimanmarcus.com/».

[33] «Noom - https://www.noom.com/#/».

[34] «AWS Music - https://www.amazon.com/-/es/music,» [En línea].

[35] «Branch - https://www.ourbranch.com/».

[36] «AWS Amplify https://docs.amplify.aws/start/getting-started/installation/q/integration/react-native/#option-1-watch-the-video-guide».

[37] «Conectar Frontend API con Backend https://docs.amplify.aws/start/getting-started/data-model/q/integration/react/#connect-frontend-to-api».

[38] «Control de errores AWS - https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/API_SignUp.html#API_SignUp_Errors».

[39] «AWS Custom attributes - https://docs.aws.amazon.com/cognito/latest/developerguide/user-pool-settings-attributes.html#user-pool-settings-custom-attributes,» [En línea].

[40] «Error en customType https://stackoverflow.com/questions/61450969/cannot-find-name-attributes-amplify-auth».

[41] «Error en customType - Enlace 2 https://stackoverflow.com/questions/53475094/accessing-cognito-custom-attributes-in-a-post-confirmation-lambda-function».

[42] J. M. G. Rozas, «GraphQL. https://www.paradigmadigital.com/dev/graphql-todos-uno-uno-todos/,» 2018.

[43] G. -. Q. a. Mutations, «Queries and Mutations https://graphql.org/learn/queries/».

[44] A. GrapghQL, «Apollo GrapghQL - https://www.apollographql.com/».

[45] «AWS Amplify - GraphQL Mutations https://docs.amplify.aws/lib/graphqlapi/mutate-data/q/platform/js/#mutations».

[46] «Diagram Editor - https://www.diagrameditor.com/».

[47] «Construir una React App con TypeScript y GraphQL https://blog.logrocket.com/build-graphql-react-app-typescript/».

[48] «Construir una aplicación con React Native https://www.asapdevelopers.com/build-a-react-native-login-app-with-node-js-backend/».

[49] «Validar texto de entrada - Enlace 2 https://reactnative.dev/docs/textinput».

[50] «Multi Select Box - React Native https://www.npmjs.com/package/react-native-multi-selectbox-typescript».

[51] «Implementación del timer en los entrenamientos. https://github.com/hoaphantn7604/react-native-element-timer».

Universidad de Alcalá

Escuela Politécnica Superior

ESCUELA POLITECNICA SUPERIOR

Universidad de Alcalá