UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA DEL SOFTWARE

Aplicación móvil para el encuentro y realización de partidos de fútbol 7

Mobile app for the organisation of football 7 matches

Realizado por
D. Enrique Cañadas Cobo

Tutorizado por
D. Luis Manuel Llopis Torres

Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2023

Septiembre de 2023

Resumen

El fútbol 7 es una variante del fútbol. Como su nombre indica, está formado por equipos de 7 jugadores y normalmente es practicado de una forma más casual. Sin embargo, la simple idea de encontrar a 14 jugadores para poder comenzar un partido puede ser un gran problema y una tarea tediosa.

Este trabajo de fin de grado se enfoca en el desarrollo de una aplicación móvil orientada a resolver este problema, y por tanto, facilitar el encuentro y realización de partidos de fútbol 7. Busca ayudar a aquellas personas amantes de este deporte a que tengan una forma más sencilla, rápida e innovadora de encontrar a otros jugadores con los que llevar acabo estos encuentros.

La aplicación ofrece la oportunidad a los usuarios de registrarse para luego permitirles tanto crear como unirse a partidos cuando y en la posición que deseen. Igualmente, da la posibilidad de formar equipos con sus amigos, para que así puedan incorporarse en conjunto a los partidos. Además, muestra junto a su perfil una sección donde pueden ver estadísticas relacionadas a su desempeño, entre otros apartados.

Palabras clave:

Fútbol 7, Partidos, Aplicación móvil, Android, Java

Abstract

Football 7 is a variant of football. As the name suggests, it is played by teams of 7 players and it is usually practiced in a more casual manner. However, the simple idea of finding 14 players to start a match can be a significant problem and a tedious task.

This final degree project focuses on the development of a mobile application aimed at solving this problem and, therefore, facilitating the encounter and organization of football 7 matches. It aims to help football enthusiasts have a simpler, faster and innovative way to find other players to carry out these matches.

The application offers users the opportunity to register and then allows them to create or join matches at their preferred time and position. It provides the possibility of joining teams with their friends so that they can enter matches as a team. Additionally, it displays a subdivision alongside their profile where they can view statistics related to their performance, among other sections.

Keywords:

Football 7, Matches, Mobile app, Android, Java

Índice

1

Introducción

La idea surge cuando el propio alumno junto a un grupo de amigos comienza a practicar fútbol 7. Durante las semanas siguientes se dan cuenta de que encontrar a los 14 jugadores no es nada fácil, ya que la simple búsqueda, organización y coordinación de todos es una tarea complicada y tediosa.

Por todo esto, la motivación y necesidad que pretende cubrir este proyecto es ayudar a aquellos amantes de este deporte a que tengan una forma más sencilla, amigable y rápida de encontrar a otros jugadores para que así puedan disfrutar todos en conjunto.

El objetivo principal será el desarrollo de una aplicación móvil para organizar y realizar partidos de fútbol 7. La aplicación permitirá a los usuarios registrarse para que puedan posteriormente crear y unirse a partidos. Además, dará la opción de formar equipos a los que los amigos puedan unirse, para así poder entrar en conjunto a un partido. Por último, ofrece junto al perfil de usuario un apartado donde ver las estadísticas de su desempeño como el número de victorias o derrotas.

En cuanto a los entregables principales del proyecto, se encuentran:

- Servidor
- Aplicación móvil
- Manual de usuario
- Guía de instalación
- Documento general de requisitos
- Memoria completa (este documento)

La metodología escogida se basa en un desarrollo iterativo e incremental. El desarrollo iterativo e incremental es una metodología de desarrollo software que se basa en la integración progresiva de las diferentes funcionalidades a lo largo del tiempo. Permite una mayor flexibilidad ya que no sigue un enfoque lineal si no que se adapta a las necesidades concretas del proyecto y del cliente en ese momento concreto. Aporta un avance de manera ágil y ajustada a medida que se progresa y se consigue retroalimentación de todas las partes involucradas.

Por todo esto, se opta por utilizar esta metodología debido a que se trata de un proyecto que realiza una única persona, teniendo que tomar todas las decisiones y realizar todo añadiendo poco a poco cada una de las partes y mejorando las anteriores.

Los apartados en los que se estructurará la memoria serán los siguientes:

- Tecnologías utilizadas. Se ve una breve descripción de aquellas tecnologías, herramientas y frameworks empleadas en el proyecto.

- Fases del desarrollo. Se explican las diferentes etapas en las que se ha dividido el desarrollo de la aplicación. Este apartado incluye:

    - Estudio de las tecnologías. Se estudian las tecnologías y como pueden encajar en el proyecto para su máxima utilización.
    - Análisis de requisitos. Se analiza el problema y se detallan los aspectos más importantes que debe cumplir la aplicación a través de unos requisitos.
    - Diseño. Se diseña y obtiene una posible solución al problema, tomando decisiones importantes relacionadas con la arquitectura o los componentes que serán utilizados.
    - Implementación. Se implementa la solución usando las tecnologías estudiadas, siguiendo los requisitos y el diseño obtenido.
    - Pruebas. Se realizan las pruebas necesarias para verificar el correcto funcionamiento y validar la capacidad de la aplicación.

- Conclusiones y líneas futuras. Se describe como se han cumplido los objetivos, presentando los resultados alcanzados y viendo posibles actualizaciones o líneas futuras.

- Apéndices. Se incluyen documentos que ayudan a comprender el sistema. Se encuentra el manual de usuario, la guía de instalación y el documento general de requisitos.

2

Tecnologías utilizadas

En este capítulo se verán las tecnologías, herramientas y frameworks empleados en el proyecto.

2.1 Java

Java [2] es un lenguaje de programación orientado a objetos y de alto nivel. Es uno de los lenguajes más populares y utilizados en todo el mundo. Gracias a su versatilidad y multiplataforma, puede ejecutarse en la gran mayoría de sistemas operativos sin ningún problema gracias a su propia máquina virtual (JVM) [3].

Entre sus características principales destacan aspectos ya dichos como su versatilidad y multiplataforma, además de su seguridad, robustez, facilidad y su amplia biblioteca de clases, entre otras.

Java ha sido tradicionalmente el lenguaje principal para el desarrollo de aplicaciones móviles, por lo que encaja a la perfeccionan con la línea del proyecto y por eso se ha elegido como lenguaje principal.

Figura 1. Logo de Java.

2.2 Spring Boot

Spring Boot [4] proporciona un enfoque rápido y sencillo para el desarrollo de APIs REST [5][6]. Intenta eliminar el proceso inicial de configuración manual, permitiendo que los desarrolladores puedan centrarse en tareas más importantes en lugar de en la configuración o integración de tecnologías y dependencias. Entre sus características principales se encuentran su configuración automática, gestión de dependencias, monitorización y gestión. Además, proporciona anotaciones y estereotipos para definir componentes y puntos de acceso de forma sencilla.

Gracias a su sencillez y rapidez es ideal para lo que se busca en este proyecto.

Figura 2. Logo de Spring.

2.3 XML

XML (Extensible Markup Language) [7] es un lenguaje de marcado que permite estructurar datos de forma jerárquica y bajo ciertas reglas. En el desarrollo de aplicaciones móviles es ampliamente utilizado para definir la vista o interfaz de usuario. En Android se utiliza principalmente para definir el diseño de las actividades [8] o fragmentos [9]. Además, también se puede usar para definir recursos como las cadenas de texto o estilos, menús y archivos de configuración, entre otros.

En definitiva, XML es una de las partes más importante en el desarrollo de aplicaciones móviles ya que permite definir todos los recursos y vistas necesarios para crear una aplicación más atractiva, visual e innovadora.

2.4 Base de datos Firebase

Cloud Firestore [10] es el servicio de base de datos de tiempo real proporcionado por Firebase. Proporciona sincronización en tiempo real, permitiendo que un cambio en los datos se refleje automáticamente en todos los clientes conectados. Se basa en un modelo NoSQL de datos basados en documentos y colecciones, garantizando que se almacenen y organicen de una manera muy eficiente. Además, destaca por su gran escalabilidad y rendimiento y por su seguridad y reglas, permitiendo establecer quien puede leer y escribir en todo momento. Igualmente, se integra a la perfección con todos los demás servicios de Firebase como la autenticación de usuarios o las notificaciones. Por todo esto, se decide utilizar Cloud Firestore ya que encaja a la perfección con la línea del proyecto.

Figura 3. Logo de Firebase.

2.5 Android Studio

Android Studio [11] es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones Android. Es una herramienta creada por Google para ofrecer a la comunidad de desarrolladores una forma sencilla y rápida de crear aplicaciones en la plataforma Android. Ofrece una interfaz de usuario intuitiva y un diseñador de interfaz gráfica que permite visualizar los cambios en tiempo real. Además, entre sus características más importantes se encuentran su soporte multilenguaje, su depuración avanzada y su integración con Firebase, herramienta que como se ha visto será usada en el proyecto.

Así mismo, uno de los aspectos más importante y más atractivo para su utilización durante el desarrollo es su administrador de emuladores [12], el cual ofrece una herramienta útil y simple para probar y ejecutar la aplicación en diferentes dispositivos Android de distintos modelos y versiones. En definitiva, Android Studio es una herramienta perfecta para el desarrollo de cualquier aplicación móvil y por tanto es ideal para el proyecto.

Figura 4. Logo de Android Studio.

2.6 Visual Studio Code

Visual Studio Code [13] es un editor de código fuente utilizado por programadores y desarrolladores de software. Es un editor de código ligero que da soporte multiplataforma, pudiéndose usar tanto en Windows, Linux y macOS. Además, es multilenguaje, permitiendo todo tipo de lenguajes como C++, Java, Python o HTML. Gracias a su terminal integrada, rendimiento y ligereza es una opción ideal para aquellos proyectos no tan grandes.

Figura 5. Logo de Visual Studio Code.

2.7 Cloudinary

Cloudinary [14] es un servicio en la nube para el almacenamiento y gestión de imágenes y videos. Es una herramienta que ayuda a desarrolladores a almacenar, administrar y recuperar todo tipo de archivos multimedia de forma rápida y sencilla. Entre sus características principales se encuentran su seguridad, su fácil integración gracias a sus librerías, su optimización a la hora de la carga y descarga, y su transformación de archivos de forma dinámica y en tiempo real. Debido a la necesidad de almacenar imágenes en el proyecto para los diferentes objetos, Cloudinary se convierte en una opción ideal y necesaria.

Figura 6. Logo de Cloudinary.

3

Fases del desarrollo

En este capítulo se detallarán las fases o etapas del desarrollo. Se explicarán los pasos seguidos desde el inicio hasta la finalización del proyecto pasando por cada uno de los puntos de inflexión del mismo.

3.1 Estudio de las tecnologías

En primer lugar, como en todo proyecto, se ha de tener una idea clara y exacta de todas las tecnologías que se van a emplear. Al tratarse de una aplicación móvil que requiere que los usuarios estén conectados y compartan datos en todo momento, es necesario que exista una base de datos y un servidor o API que sea el encargado de actuar entre los clientes y la base de datos mencionada. A continuación se verán las tecnologías empleadas en cada uno de los apartados, ya explicadas anteriormente en el apartado 2.

La base de datos elegida fue Cloud Firestore debido a que encajaba a la perfección gracias a su fácil integración tanto con el proyecto como con los demás servicios que se usarían posteriormente.

En el lado del servidor, se optó por usar Spring Boot por la sencillez que propone. Se estudia cómo se configura, integra y desarrolla para poder emplearlo en las siguientes fases de desarrollo.

Por último, en el lado del cliente, se opta por la herramienta proporcionada más utilizada en el ámbito de Android, Android Studio, utilizando como lenguaje principal Java, apoyándose en XML para definir los recursos y vistas correspondientes.

Todas estas herramientas, tecnologías y frameworks se estudian en toda medida posible para garantizar el correcto uso y exprimir al máximo su potencial, todo orientado a completar el proyecto de la mejor manera.

3.2 Análisis de requisitos

Durante esta etapa se analiza el problema y su entorno para definir lo que realmente hay que construir.

El principal problema es la dificultad para reunir a los 14 jugadores, por lo que se debe centrar el enfoque en desarrollar una solución para esto. Por lo tanto, se debe construir una aplicación que lo cubra en su totalidad y no deje aspectos sin resolver.

3.2.1 Identificación de las partes involucradas

En primer lugar se identifican a todos los involucrados afectados directa o indirectamente por el sistema. Normalmente se incluyen partes como los usuarios finales que usaran la aplicación, los desarrolladores, los inversores o patrocinadores, los competidores y los clientes. Sin embargo, al tratarse de un proyecto en el contexto de trabajo de fin de grado, las partes interesadas se ven reducidas a:

- El propio alumno, encargado de desarrollar y completar cada una de las partes del proyecto.
- El tutor académico, encargado de supervisar y orientar al alumno.
- Los usuarios finales, que serán aquellos jugadores que usarán la aplicación cuando esté completada y lanzada.
- La Universidad de Málaga, encargada de proporcionar los recursos necesarios y evaluar, calificar y validar el proyecto.

3.2.2 Recopilación de requisitos

Tras identificar a las partes involucradas, se tiene que tener claro lo que el sistema debe conseguir y cuáles son sus expectativas. Para esto se definen unos requisitos, que no son más que descripciones de las características y funcionalidades de la aplicación. Por cada aspecto que haya que cubrir hay que definir uno o unos requisitos que definan de la forma más clara y corta posible lo que hay que cumplir.

Véase, por ejemplo, una aplicación que requiere el registro de los usuarios mediante un correo y contraseña, su requisito asociado podría ser:

1. Los usuarios deberán registrarse en la aplicación utilizando un correo y una contraseña.

En este caso, se ha definido un requisito que refleja una funcionalidad del sistema, el registro de un usuario, por lo que se habla de requisitos funcionales. Los requisitos funcionales son aquellos que declaran y especifican las diferentes funciones y acciones que tiene que cumplir el sistema para cumplir con las necesidades de los usuarios. En cambio, los requisitos no funcionales definen características que no están relacionadas con la funcionalidad si no con aspectos como la seguridad, el rendimiento o la calidad.

Por ejemplo, un requisito no funcional podría ser:

1. La aplicación debe estar disponible tanto en español como en inglés.

Una vez que queda clara esta división entre funcionales y no funcionales, se deben clasificar según relevancia e importancia para posteriormente ser recogidos en un documento formal que recopila y describe cada uno de ellos, llamado "Documento general de requisitos" (Apéndice C).

Véanse ahora los requisitos principales del sistema.

Requisitos funcionales

Registro e inicio de sesión de usuarios

1. Los usuarios podrán registrarse utilizando un correo, un nombre, unos apellidos y una contraseña.
2. Los usuarios podrán iniciar sesión utilizando su correo y su contraseña.
3. Los usuarios podrán cerrar sesión.

Vista principal

4. Los usuarios podrán visualizar una lista con los partidos disponibles.
   4.1 Los usuarios podrán recargar los partidos disponibles.
5. Los usuarios podrán visualizar una lista con sus partidos reservados y partidos completados.
6. Los usuarios podrán visualizar su perfil.

Partidos disponibles, reservados y completados

7. Los usuarios podrán visualizar un partido y sus características.
8. Los usuarios podrán ver una lista con los jugadores de cada equipo, local y visitante.
9. El creador del partido podrá editar la foto del encuentro.
10. Los usuarios podrán reservar una posición en uno de los equipos.
11. Los usuarios podrán cancelar su reserva de una posición.
12. Los usuarios podrán inscribir a uno de sus equipos.
13. Los usuarios podrán retirar la inscripción de su equipo.
14. Los usuarios podrán acceder al chat del partido.
    14.1. Los usuarios podrán ver los mensajes enviados.
    14.2. Los usuarios podrán enviar mensajes.
    14.3. Los usuarios que sean capitanes o emisores del mensaje podrán eliminar un mensaje.
15. Los usuarios podrán actualizar el resultado del partido una vez haya acabado.

Creación de partidos

16. Los usuarios podrán crear partidos indiciando localización, día, hora y precio total.

Perfil

17. Los usuarios podrán ver su foto de perfil y su nombre.
18. Los usuarios podrán editar su foto de perfil mediante cámara o galería de fotos.
19. Los usuarios podrán ver una lista con los equipos a los que pertenecen.
20. Los usuarios podrán ver estadísticas relacionadas a su número de partidos, incluyendo victorias, derrotas y empates.

Equipos

21. Los usuarios podrán crear equipos indicando únicamente su nombre.
22. Los usuarios podrán ver la foto del equipo.

    22.1 El capitán del equipo podrá editar la foto mediante cámara o galería de fotos.
23. Los usuarios podrán ver una lista con los integrantes del equipo.
24. Los usuarios podrán ver los perfiles de los demás compañeros.
25. Los usuarios podrán invitar a otros jugadores a sus equipos.
26. Los usuarios podrán aceptar o rechazar invitaciones de equipos.
27. Los usuarios podrán dejar un equipo.
28. El capitán del equipo podrá eliminar a un compañero del equipo.

Amigos

29. Los usuarios podrán ver una lista de sus amigos.
30. Los usuarios podrán enviar peticiones de amistad.
31. Los usuarios podrán aceptar y rechazar peticiones de amistad.
32. Los usuarios podrán eliminar a un amigo.

Requisitos No Funcionales

1. La aplicación se encontrará tanto en español como en inglés.
2. La aplicación deberá responder de forma rápida y fluida.
3. La aplicación deberá ser fácil de entender y de aprender su utilización.
4. La aplicación deberá ser compatible con la mayoría de dispositivos Android.
5. La aplicación deberá tener un tiempo de respuesta menor a los 3 segundos.

3.2.3 Casos de uso

Tras especificar y recopilar los requisitos, es momento de dar una descripción más detallada de cómo el usuario final será capaz de interactuar y llevar a cabo los objetivos que se marcan. A continuación, se verán los casos de uso asociados a las tareas más importantes y representativas del sistema.

1. Caso de Uso "Registro de un usuario"

Nombre: Registro de un usuario.
Participantes: Usuario, Sistema.
Descripción: El usuario quiere registrarse en el sistema.
Precondiciones: El usuario debe tener instalada la aplicación y debe tener acceso a internet.
Postcondiciones: El usuario queda registrado en la aplicación.

Escenario principal:

1. El usuario inicia la aplicación.
2. El sistema le muestra la vista inicial para iniciar sesión.
3. El usuario le da al botón "¿No tienes una cuenta? ¡Regístrate!".
4. El sistema cambia la vista y le muestra el formulario para registrarse.
5. El usuario introduce su correo electrónico, su nombre, sus apellidos y su contraseña y pulsa el botón "Registrarse".
6. El sistema valida los datos y le introduce en la aplicación.

Escenarios alternativos:

5.1. El usuario no introduce alguno de los datos y pulsa el botón "Registrarse".

5.1.1. El sistema le indica que le falta algún campo por rellenar.

5.1.2. Vuelta al paso 5.

5.2. El usuario introduce un correo electrónico con un formato incorrecto y pulsa el botón "Registrarse".

5.2.1. El sistema le indica que el formato no es correcto.

5.2.2. Vuelta al paso 5.

5.3. El usuario introduce una contraseña de menos de 6 caracteres y pulsa el botón "Registrarse".

5.3.1. El sistema le indica que la contraseña debe tener al menos 6 caracteres.

5.3.2. Vuelta al paso 5.

2. Caso de Uso "Inicio de sesión de un usuario"

Nombre: Inicio de sesión de un usuario.

Participantes: Usuario, Sistema.

Descripción: El usuario quiere iniciar sesión en la aplicación.

Precondiciones: El usuario debe estar registrado en la aplicación y con conexión a internet.

Postcondiciones: El usuario entra de forma exitosa en la aplicación.

Escenario principal:

1. El usuario inicia la aplicación.
2. El sistema le muestra la vista inicial para iniciar sesión.
3. El usuario introduce su correo y su contraseña y pulsa el botón "Iniciar sesión".
4. El sistema valida los datos y muestra la pantalla inicial con los partidos disponibles, partidos reservados y su perfil.

Escenarios alternativos:

3.1. El usuario no introduce algún dato y pulsa el botón "Iniciar sesión".

3.1.1. El sistema le indica que le falta algún campo por rellenar.

3.1.2. Vuelta al paso 3.

3.2. El usuario introduce un correo no válido y pulsa el botón "Iniciar sesión".

3.2.1. El sistema le indica que el correo no es válido.

3.2.2. Vuelta al paso 3.

4.1. El sistema comprueba que los datos no son correctos.

4.1.1. El sistema le indica que los datos no son correctos.

4.1.2. Vuelta al paso 3.

3. Caso de Uso "Creación de un partido"

Nombre: Creación de un partido.

Participantes: Usuario, Sistema.

Descripción: El usuario quiere crear un partido.

Precondiciones: El usuario ha iniciado sesión en la aplicación.

Postcondiciones: El partido se ha creado con éxito y el usuario ve los detalles del partido.

Escenario principal:

1. El sistema muestra la pantalla inicial con los partidos disponibles, partidos reservados y su perfil.
2. El usuario pulsa sobre el botón flotante con el símbolo "+".
3. El sistema muestra una vista con el formulario para crear el partido.
4. El usuario introduce la localización, la fecha, la hora y el precio total, y pulsa el botón "Crear".
5. El sistema valida los campos y crea el partido.
6. El sistema muestra una vista correspondiente del partido.

Escenarios alternativos:

4.1. El usuario no introduce algún dato y pulsa el botón "Crear".

4.1.1. El sistema le indica que le falta algún campo por rellenar.

4.1.2. Vuelta al paso 4.

4. Caso de Uso " Reserva de una posición en un partido"

Nombre: Reserva de una posición en un partido.

Participantes: Usuario, Sistema.

Descripción: El usuario quiere reservar una posición en un partido concreto.

Precondiciones: El usuario ha iniciado sesión en la aplicación.

Postcondiciones: El usuario reserva con éxito la posición que desea.

Escenario principal:

1. El sistema muestra la pantalla inicial con los partidos disponibles, partidos reservados y su perfil.
2. El usuario pulsa sobre uno de los partidos disponibles.
3. El sistema le muestra una vista correspondiente del partido, enseñándole tanto la fecha, localización y precio como una lista con los jugadores de ambos lados, local y visitante.
4. El usuario pulsa sobre la posición que quiere reservar.
5. El sistema le pregunta mediante un mensaje de confirmación si está seguro que quiere reservar esa posición.
6. El usuario indica que sí está seguro.
7. El sistema reserva esa posición para el usuario.

Escenarios alternativos:

6.1 El usuario indica que no está seguro.

6.1.1 Vuelta al paso 3.

5. Caso de Uso " Cancelación de la reserva de una posición en un partido"

Nombre: Cancelación de la reserva de una posición en un partido.

Participantes: Usuario, Sistema.

Descripción: El usuario quiere cancelar su reserva de una posición en un partido concreto.

Precondiciones: El usuario ha iniciado sesión en la aplicación y tiene realizada la reserva de una posición en un partido.

Postcondiciones: El usuario cancela su reserva con éxito.

Escenario principal:

1. El sistema muestra la pantalla inicial con los partidos disponibles, partidos reservados y su perfil.
2. El usuario pulsa sobre uno de los partidos reservados.
3. El sistema le muestra una vista correspondiente del partido, enseñándole tanto la fecha, localización y precio como una lista con los jugadores de ambos lados, local y visitante.
4. El usuario mantiene pulsado sobre la posición que tiene reservada.
5. El sistema le pregunta mediante un mensaje de confirmación si está seguro que quiere cancelar su reserva de esa posición.
6. El usuario indica que sí está seguro.
7. El sistema cancela la reserva de esa posición para el usuario.

Escenarios alternativos:

6.1 El usuario indica que no está seguro.

6.1.1 Vuelta al paso 3.

6. Caso de Uso "Creación de un equipo"

Nombre: Creación de un equipo.

Participantes: Usuario, Sistema.

Descripción: El usuario quiere crear un equipo.

Precondiciones: El usuario ha iniciado sesión en la aplicación.

Postcondiciones: El usuario crea el equipo con éxito y ve sus detalles.

Escenario principal:

1. El sistema muestra la pantalla inicial con los partidos disponibles, partidos reservados y su perfil.
2. El usuario navega al apartado "Perfil".
3. El sistema muestra el perfil del usuario.
4. El usuario pulsa sobre el botón "Crear equipo".
5. El sistema cambia la vista y enseña un formulario para crear el equipo.
6. El usuario introduce el nombre y pulsa el botón "Crear".
7. El sistema valida los campos y crea el equipo.
8. El sistema muestra una pantalla con los datos del equipo y sus integrantes.

Escenarios alternativos:

6.1 El usuario no introduce el nombre y pulsa el botón "Crear".

6.1.1. El sistema le indica que le falta el campo por rellenar.

6.1.2. Vuelta al paso 6.

7. Caso de Uso "Registro de un equipo en un partido"

Nombre: Registro de un equipo en un partido.

Participantes: Usuario, Sistema.

Descripción: El usuario quiere registrar a uno de sus equipo en un partido.

Precondiciones: El usuario ha iniciado sesión en la aplicación.

Postcondiciones: El equipo queda registrado en el partido.

Escenario principal:

1. El sistema muestra la pantalla inicial con los partidos disponibles, partidos reservados y su perfil.
2. El usuario pulsa sobre uno de los partidos disponibles.
3. El sistema le muestra una vista correspondiente del partido, enseñándole tanto la fecha, localización y precio como una lista con los jugadores de ambos lados, local y visitante.
4. El usuario le da al botón "Entrar como equipo".
5. El sistema muestra una ventana con los equipos del usuario.
6. El usuario selecciona el equipo que quiere inscribir.
7. El sistema registra al equipo en el lado correspondiente.

3.2.4 Diagramas de secuencia

Para ayudar a seguir entendiendo el sistema, se trazarán los diagramas de secuencia asociados a los casos de uso más importantes.

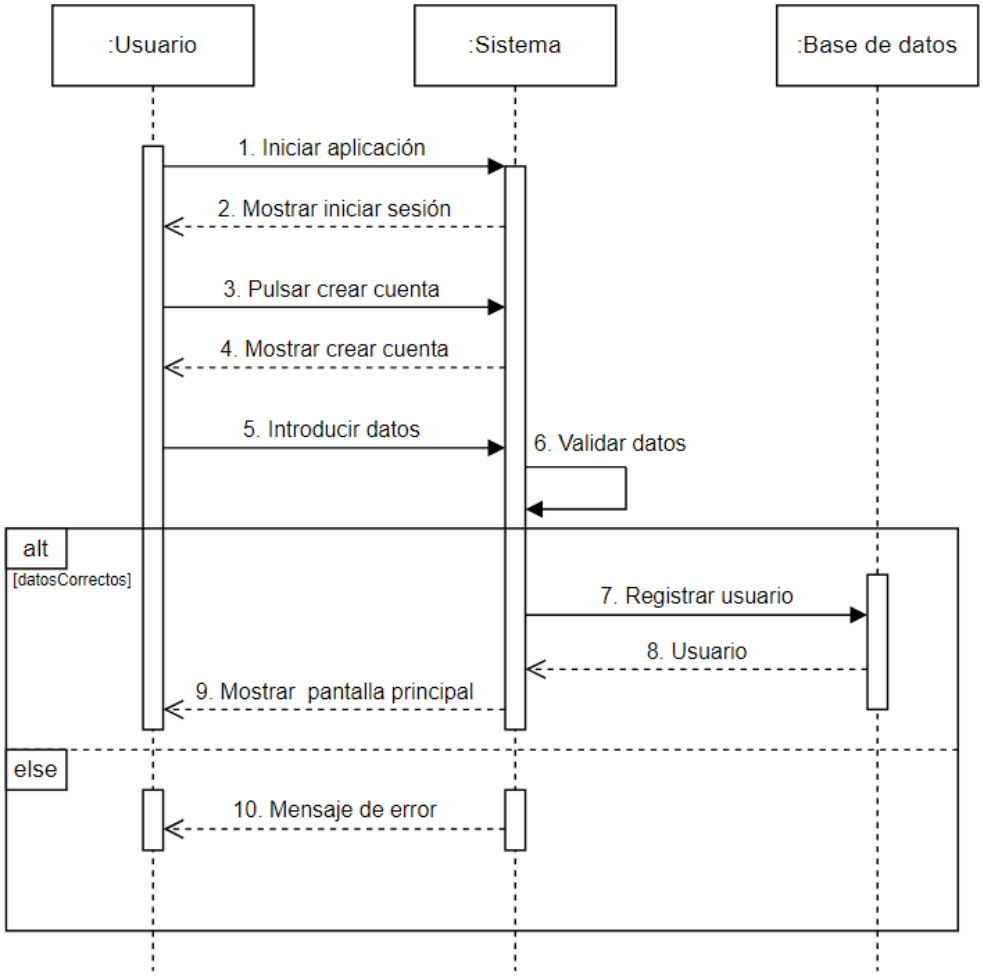

Figura 7. Registro de un usuario.

El primero de ellos es el registro de un usuario. Como se puede observar, existen dos actores principales, el usuario y el sistema, que a su vez también depende de la base de datos. Una vez que el usuario ha iniciado la aplicación, el sistema le muestra la vista correspondiente, donde el usuario podrá pulsar sobre el botón crear cuenta, lo que le llevará al formulario para ello. Tras esto, introducirá todos los datos necesarios y los enviará al sistema, donde se validarán y dependiendo de si son o no correctos, se confirmará el registro, guardándolo en la base de datos, o se informará al usuario mediante un mensaje de error.

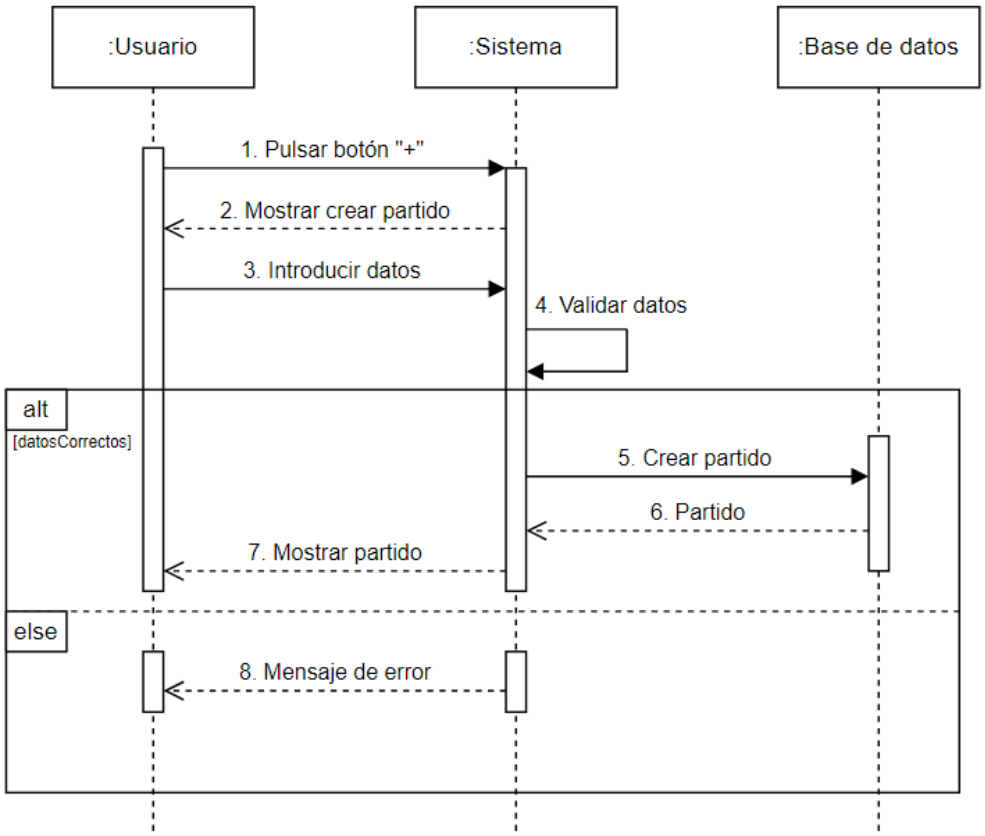

Figura 8. Creación de un partido.

Otro de los casos de uso más importante, es el de la creación de un partido. De igual manera, el usuario tendrá que interaccionar con el sistema. Primeramente, pulsa el botón destinado para crear los partidos, donde se le mostrará un apartado para ello y donde rellenará y enviará todos los datos. El sistema los recibirá y los validará, resultando en la creación del mismo si son correctos, o en un mensaje de error en caso contrario.

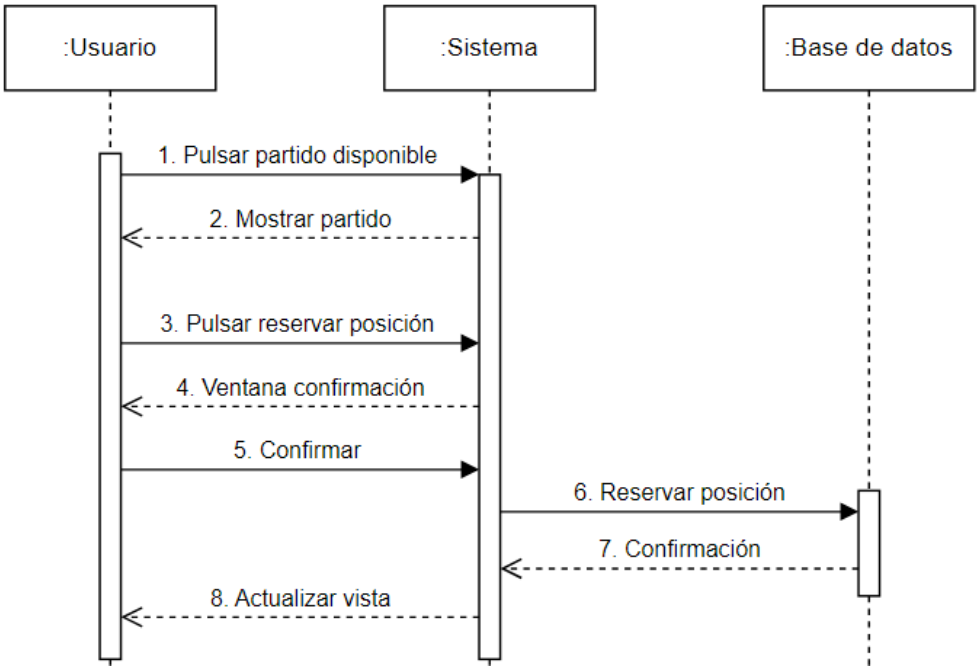

Figura 9. Reserva de una posición en un partido

El último caso de uso es el de la reserva de una posición en un partido. Tras mostrar el sistema los partidos disponibles, el usuario pulsa sobre el que desea realizar la acción para poder ver más detalles sobre este. Tras esto, selecciona la posición que desea, recibiendo un ventana de confirmación donde aceptará la reserva, quedando esta registrada en la base de datos y viendo el usuario los cambios en la vista.

3.3 Diseño

Utilizando los requisitos definidos en la etapa anterior, se diseña una posible solución y se detallan todos los aspectos relacionados con la arquitectura, los componentes o la seguridad, entre otros. Es una etapa muy importante ya que es el punto de inicio del que partirá la implementación, sirviendo de guía en todo momento.

A continuación se explicará el diseño para el lado del servidor, para el lado del cliente y para la base de datos, explicando la interacción entre todas las partes y terminando con un diagrama de clases que ayudará a entender mejor el sistema.

3.3.1 Diseño lado del servidor

Como se ha dicho, en el lado del servidor, se opta por usar Spring Boot para la creación de la API REST por la sencillez que propone a la hora de la configuración y puesta en marcha. Ofrece diferentes estructuras de organización y patrones de diseño, siendo el elegido para el proyecto el conocido como "Controller-Service-Repository".

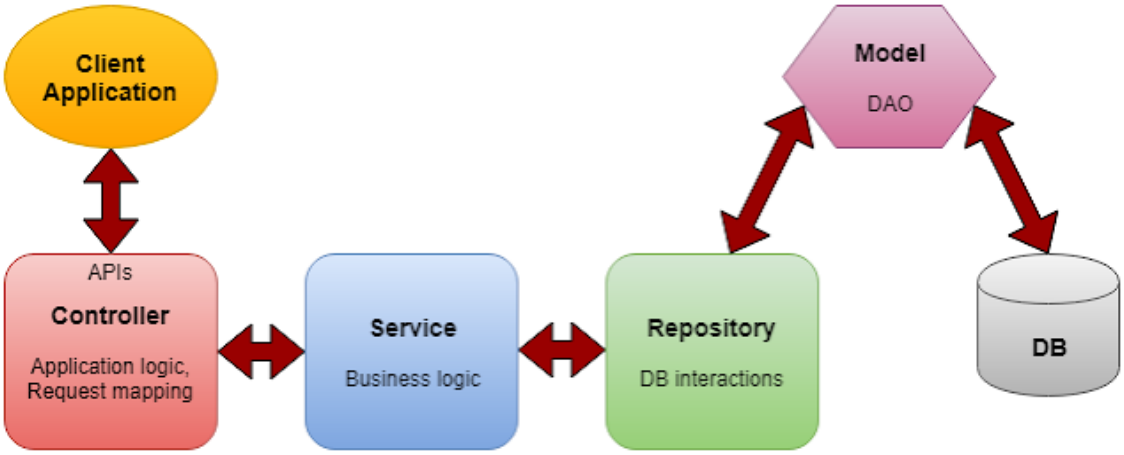

Figura 10. Controller-Service-Repository Architecture.

Fuente: Building a Simple Application with Spring Boot [F1]

Esta estructura se basa en tres componentes o capas diferentes que se conectan entre sí, separando responsabilidades y con propósitos concretos. Estos son controller, service y repository.

El primero de todos es el controller. Es el encargado de recibir las solicitudes del cliente y procesarlas según lo que se requiera. En este caso, se reciben mediante HTTP, haciendo uso de rutas concretas para cada una de ellas. Una vez procesada la solicitud, envía los datos correspondientes a la capa de negocio, es decir, al service.

El service es el encargado de implementar la lógica de negocio, definiendo reglas y operaciones para darle el comportamiento deseado. Recibe los datos del controller, los procesa y los envía al repository.

Por último, el repository es el responsable de acceder y manipular los datos que se encuentren en la base de datos. Recibe la información del service y realiza la operación correspondiente, como podría ser crear o leer un objeto.

Es importante que cada capa o componente ofrezca una abstracción y oculte los detalles de cómo realiza su función, para así poder modularizar y asignar responsabilidades de forma clara y concisa. Al lograr esto, se consigue que cada una se centre en su papel y no se preocupe por como son tratados los datos en las demás. Gracias a esto se consigue facilitar la escalabilidad y mantenimiento de la aplicación.

3.3.2 Diseño lado del cliente

En el lado del cliente, se opta por la herramienta proporcionada más utilizada en el ámbito de Android, Android Studio, utilizando como lenguaje principal Java, apoyándose en XML para definir los recursos y vistas correspondientes.

Para realizar una aplicación Android en Android Studio, hay que entender previamente las partes principales que la componen.

Las protagonistas son las actividades, que son los componentes fundamentales de cualquier aplicación Android. Definen las pantallas que tendrá la aplicación, por lo que se forman por la interfaz o vista y por una clase con la lógica de esta, apoyándose en ciertos recursos.

Las vistas son los elementos visuales con los que el usuario interactúa (botones, imágenes, campos de texto...) definidos utilizando XML.

La clase será la encargada de proporcionar la lógica de esa vista, proporcionando comportamientos específicos a cada elemento y realizando diferentes operaciones según el estado de la aplicación.

Los recursos son archivos que almacenan datos estáticos que usa la aplicación, como cadenas de texto, imágenes o diseños.

Además, existen otros elementos con funcionalidades muy importantes, como lo son los Intents, que sirven para realizar llamadas entre componentes, por ejemplo para iniciar otra actividad y mostrarla, o los fragmentos, que son módulos que definen un comportamiento específico en una actividad.

Entrando en mayor detalle sobre el diseño, al igual que en el lado del servidor, hay muchas estructuras de organización y patrones de diseño, pero el utilizado ha sido el "Repository Pattern" o "Patrón de repositorio" [15].

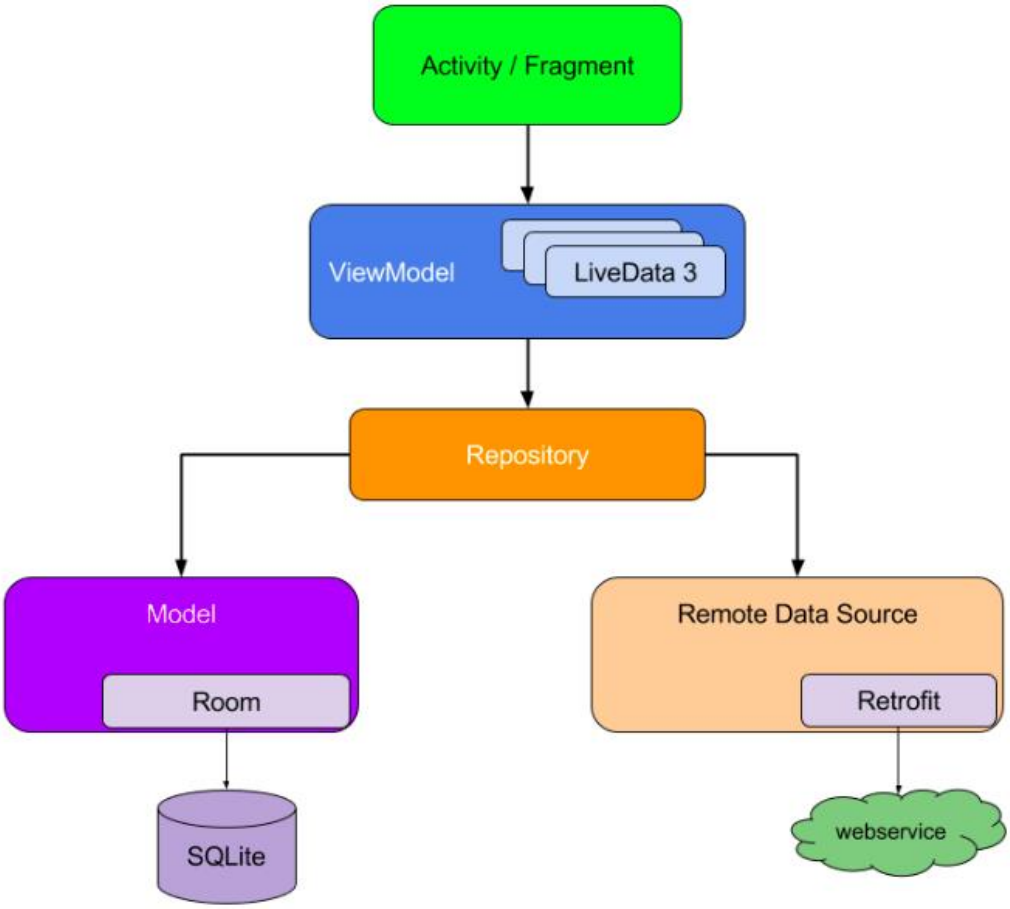

Figura 11. Repository Pattern.

Fuente: Página oficial Android [F2]

Este patrón de diseño surge como parte del patrón MVC (Modelo-Vista-Controlador) [16]. Al igual que su padre, se centra en tres componentes principales, Model, Repository y ViewModel.

Comenzando por el model, es el encargado de representar los objetos y datos que se utilizarán en la aplicación. Por ejemplo, si se utilizan usuarios, el model asociado sería "Usuario" y tendría los atributos necesarios, como podrían ser email, nombre y apellidos.

Siguiendo con el repository, proporciona los métodos y funciones necesarias para acceder a la base de datos o para realizar llamadas a una API, como es nuestro caso.

Por último, el ViewModel se encarga de actuar como intermediario entre el repositorio, obteniendo los datos y preparándolos para la vista o interfaz de la aplicación.

Al igual que su contraparte en el lado del servidor, la separación en diferentes módulos favorece la escalabilidad y seguridad de la aplicación, haciendo también que las pruebas pertinentes sean mucho más simples.

3.3.3 Diseño de la base de datos

La base de datos escogida fue Cloud Firestore. Se trata de una base de datos NoSQL basada en documentos y colecciones.

Un documento es el nivel más bajo de almacenamiento de datos. Representa un objeto concreto con una serie de atributos. Se apoya en una estructura similar a JSON [17], almacenando sus elementos en pares clave-valor. Cada documento tiene asociado un identificador único que lo referencia.

Una colección es un grupo de documentos. A diferencia de las tablas en una base de datos relacional, cada documento de la colección puede tener diferentes campos, lo que genera que pueda haber documentos con elementos totalmente diferentes.

Véase un ejemplo.

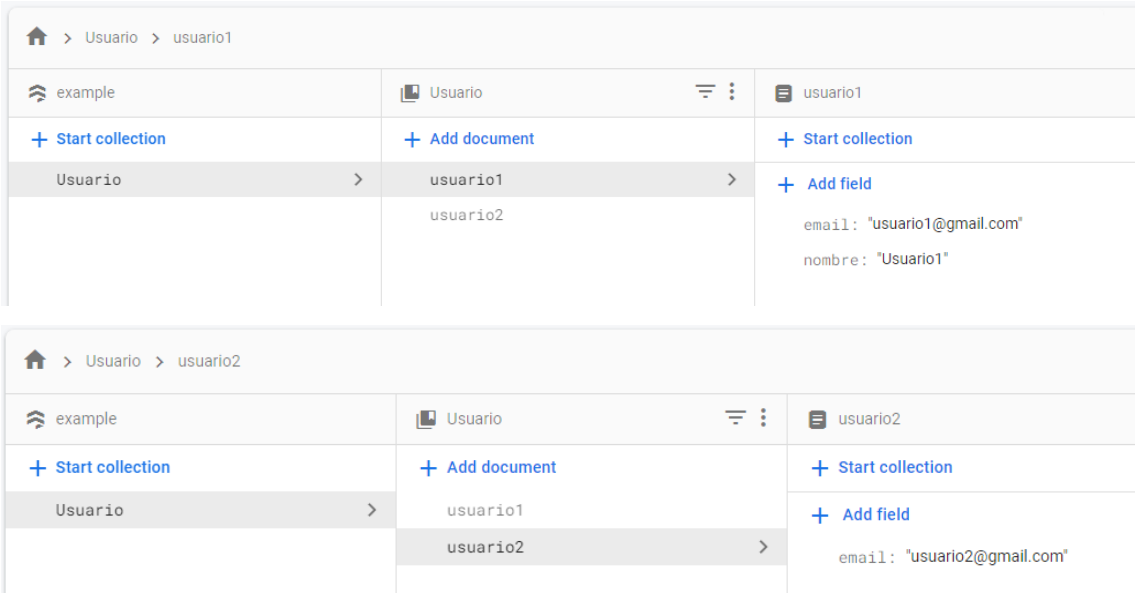

Figura 12. Ejemplo documentos y colecciones.

En la figura se representa una colección Usuario que tiene dos documentos, usuario1 y usuario2. Como se puede observar, usuario1 tiene un email y un nombre, sin embargo, usuario2 solo tiene un email.

Otro aspecto importante es la definición de reglas de seguridad que proporciona. Estas reglas definen y controlan el acceso de lectura y escritura de las colecciones y de los documentos.

```
1  rules_version = '2';
2  service cloud.firestore {
3    match /databases/{database}/documents {
4      match /{document=**} {
5        allow read: if true;
6        allow write: if request.auth != null;
7      }
8    }
9  }
```

Figura 13. Ejemplo reglas.

En este ejemplo, se define que para cualquier documento de la base de datos (línea 4), cualquier usuario tiene permitido leer (línea 5), pero solo aquellos que están autenticados pueden escribir (línea 6).

Este es un ejemplo básico pero que representa la potencia que tiene esta herramienta a la hora de definir reglas para diferentes colecciones y/o documentos, separándolas tanto en lectura como escritura.

3.3.4 Interacción entre lado servidor, lado cliente y base de datos

Al tratarse de una aplicación móvil en el que cada cliente necesita tener conexión en todo momento con los demás clientes y los datos, es decir, necesita acceso a la base de datos, es necesario que haya un encargado de controlar las interacciones y las peticiones con ella. Es por esto, que la base de datos es comunicada únicamente por el lado del servidor. Cuando un cliente necesita acceder a los datos almacenados, realiza una llamada a la API y es esta la encargada de recuperarlos de la base de datos, transformarlos a lo que se requiera y reenviarlos al cliente para su uso.

Este enfoque permite tener un mayor control y seguridad sobre cuándo y quién accede a los recursos compartidos, optimiza las consultas y consigue centralizar los errores y las excepciones proporcionando un mayor manejo de estas. Por último, en el caso de querer modificar, añadir o eliminar alguna funcionalidad no será necesario entrar en la lógica de los clientes sino que simplemente se hará en la API, reduciendo la complejidad del sistema.

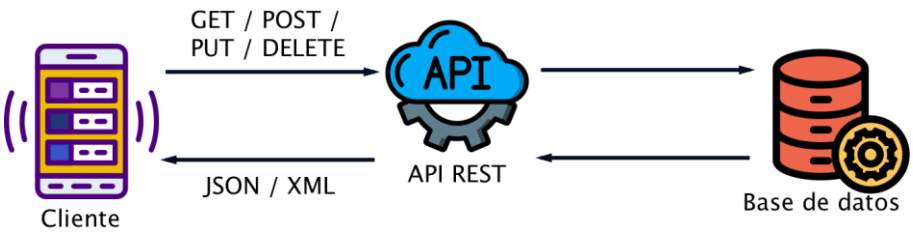

Figura 14. Esquema Cliente-API-Base de datos

Fuente: ¿Qué es una api rest? [F3]

3.3.5 Diagrama de clases

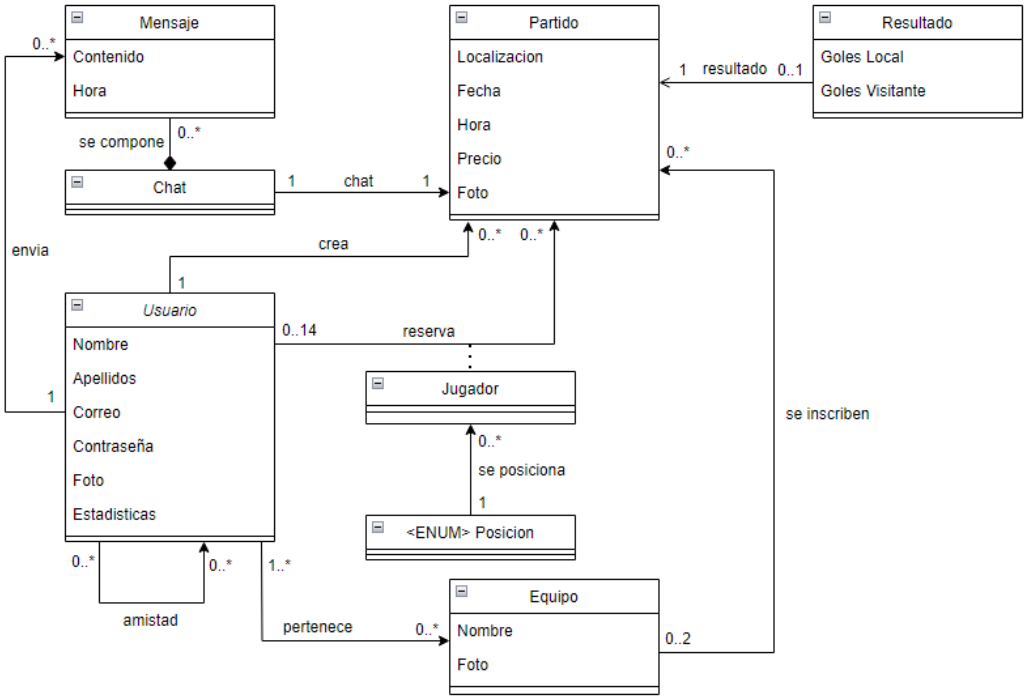

Figura 15. Diagrama de clases

En el diagrama de clases presentado podemos ver las entidades principales del sistema: usuario, partido, equipo, posición, jugador, mensaje, chat y resultado. Además, se observan las relaciones que hay entre ellas. Por ejemplo, un partido puede tener un resultado pero está obligado a tener un chat. Un usuario puede pertenecer a uno o a varios equipo, pero un equipo necesita tener al menos a un usuario para existir. Igualmente, los usuarios pueden crear y reservar en un partido convirtiéndose en un jugador indicando la posición que desean, y pueden enviar mensajes en el chat de ese partido.

Gracias al diagrama se puede obtener una idea de cómo será el sistema final y como sus componentes interaccionarán y se comportarán entre ellos.

3.4 Implementación

Una vez definido y diseñado el enfoque propuesto para el problema se pasa a la siguiente etapa, la implementación. Es la etapa más extensa ya que es donde se implementa cada una de las ideas y conceptos obtenidos anteriormente, de la forma más eficiente y evitando a toda costa los errores.

3.4.1 Implementación lado del servidor

Como se ha visto en el diseño, el objetivo es definir una API REST utilizando como estructura el llamado "Controller-Service-Repository". Para ello, se han de implementar los tres componentes para cada una de las entidades que se encuentran y guardan sus datos en la base de datos, y que por tanto, necesitan ser recuperadas en algún momento.

Sin embargo, antes de poder realizar esto, hay que definir los modelos de cada una de las partes involucradas en este proceso.

Por ejemplo, los usuarios son uno de los actores principales del sistema. Representan a las personas reales que acabarán siendo los jugadores de los partidos. Por ello, es necesario definir previamente un modelo acorde con sus componentes (nombre, apellidos, correo, foto...).

Otro ejemplo pueden ser los partidos, que representan, como su nombre indica, los partidos que jugaran los usuarios, por lo que de igual manera, es necesario establecer un modelo con sus atributos (localización, fecha, jugadores del equipo local, jugadores del equipo visitante...).

Una vez definidos los modelos, se podrá pasar a la estructura ya dicha. Hay que tener en cuenta, que no hay que definir un controller, un service y un repository para cada uno de ellos, si no que dependerá de si el objeto en cuestión se encuentra en la base de datos o no, siendo solamente necesarios cuando si se encuentran.

Este proceso de crear las tres capas comienza con el repository. Este componente es el encargado de acceder a la base de datos para realizar la petición correspondiente, por lo que es necesario que implemente cada una de las acciones básicas. Normalmente, siempre se corresponden con lo que se conoce como CRUD, en inglés, Create (Crear), Read (Leer), Update (Actualizar) y Delete (Borrar).

En definitiva, para el ejemplo de los usuarios, sus funciones principales serán:

- Obtener todos los usuarios.
- Obtener un usuario mediante un identificador.
- Crear un usuario.
- Actualizar un usuario.
- Eliminar un usuario.

El siguiente fragmento de la estructura final es el service. Es el encargado de definir la lógica de negocio, definiendo el comportamiento deseado y actuando de intermediario entre los otros dos componentes. Al igual que el repository, deberá definir cada una de las acciones básicas, por lo que tendrá como mínimo los mismos métodos que este.

Por último, en el controller es donde se definen las rutas de cada una de las peticiones utilizando el protocolo HTTP. Para ello, se indica que tipo de solicitud es, pudiendo ser GET, POST, PUT o DELETE, seguido de la ruta en cuestión. En este caso, además de incluir las acciones básicas, se incluirán un mayor número de métodos ya que será necesaria una ruta para cada petición concreta.

3.4.2 Implementación lado del cliente

En el caso del lado del cliente, el patrón empleado es el Repository Pattern. Como se dijo en el diseño, consta de tres componentes principales, Model, Repository y ViewModel, además del Activity/Fragment correspondiente (ver Figura 2).

Comenzando por el Activity, se define tanto la vista como la lógica de esta. En esta aplicación, un ejemplo podría ser la actividad principal, encargada de mostrar los partidos disponibles, partidos reservados y el perfil del usuario. Cada uno de los apartados es un fragmento diferente, con su vista y su lógica independiente, pudiendo cambiar entre ellos mediante una barra inferior o deslizando hacia los lados.

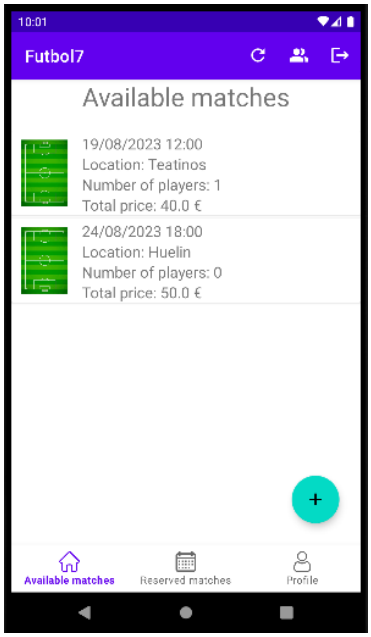

Figura 16a. Partidos disponibles

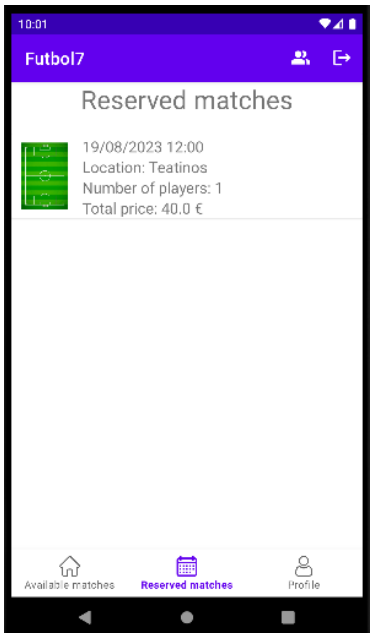

Figura 16b. Partidos reservados

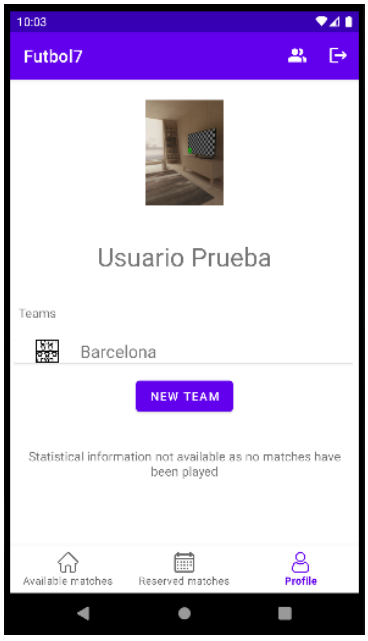

Figura 16c. Perfil

Como se puede observar, la vista utiliza como idioma el inglés ya que el móvil o simulador tiene como idioma principal del sistema este mismo, sin embargo, si estuviese en español, la vista automáticamente se cambiaría al español. Esto se debe a un recurso que existe en Android mediante el cual se pueden definir los idiomas a los que se traducirá la aplicación dependiendo de cual lo sea en el sistema. Esta herramienta es muy útil ya que permite mediante un simple archivo tener centralizados todos los idiomas y que se cambien de forma automática.

Una vez definida la actividad y los fragmentos, para poder mostrar los elementos en cada uno de ellos, es necesario recuperarlos de la base de datos. Es en este momento en el que entra en juego cada uno de los componentes vistos.

El primero es el modelo, que al igual que en el lado del servidor, define el comportamiento de un actor del sistema.

El segundo es el ViewModel, que será el encargado de actuar de intermediario entre ambas partes, obteniendo los datos del repository y preparándolos para la actividad.

Por último, el repository, es el encargado de realizar las llamadas a la API. En este caso, sólo se definen aquellas peticiones que se vayan a utilizar.

Para realizar las llamadas y esperar a la respuesta, se han utilizado interfaces definidas en el repository, llamadas Callback. Estas definen dos métodos, onExito y onError, cada uno encargado de proporcionan una respuesta en función de lo sucedido en la petición. En la actividad se llama a la función correspondiente del ViewModel, pasándole tanto los atributos a utilizar como una definición del Callback correspondiente por entrada, dándole la funcionalidad requerida en cada caso. Una vez que el repository obtiene la respuesta, llama al método del Callback, que devolverá la información a la actividad para su uso.

3.5 Pruebas

Tras realizar la implementación, es necesario comprobar que realmente funciona y cumple con los requisitos según lo previsto. En esta fase se corrigen errores y problemas y se garantiza la calidad del software. Al tener dos partes involucradas, servidor y cliente, se tienen que realizar pruebas apropiadas a cada una de ellas.

En cuanto al servidor, para verificar el correcto funcionamiento se utilizan herramientas como Swagger o Postman, siendo ambas utilizadas tanto para probar como para documentar APIs.

Swagger proporciona una interfaz web donde ver de forma interactiva cada endpoint directamente desde el navegador. Está orientado a pruebas más básicas, como es el caso de las solicitudes simples relacionadas con el CRUD.

En cambio, Postman sí está más especializado en pruebas más complejas y avanzadas. Así, proporciona una amplia gama de funcionalidades orientadas a verificar el correcto funcionamiento de la API, llegando a tratar aspectos como el rendimiento o la seguridad.

Véase un ejemplo de utilización Postman para comprobar la respuesta al intentar obtener un usuario de la base de datos a través de un identificador, en este caso, su correo.

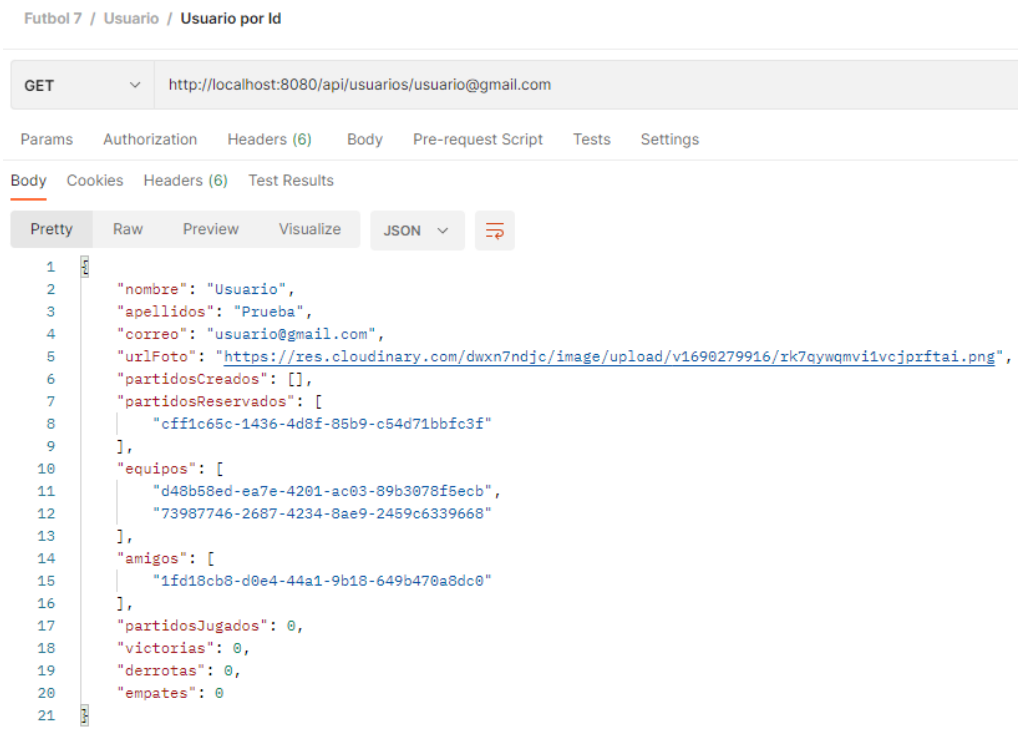

Figura 17. Prueba Postman Usuario por Id

Como se puede observar, devuelve los campos asociados al identificador "usuario@gmail.com" en formato JSON, por lo que se verifica la correcta solicitud y devolución de los datos.

Para el lado del cliente, se han centrado las pruebas en comprobar aquellos casos que tienen un mayor riesgo de ocurrir y con peores consecuencias. Estas pruebas incluyen desde funcionales a de interfaz de usuario, pasando por pruebas de rendimiento y de seguridad.

Para ello se han apoyado estas pruebas en los casos de uso explicados antes, probando cada uno de los escenarios alternativos que normalmente conducen a una situación de error. Por ejemplo, el Caso de Uso 1 "Registro de un usuario", recoge como escenario alternativo el no introducir alguno de los campos. Si se realiza esa prueba sobre la aplicación debería indicar que hay que rellenar esos datos. Véase a continuación la prueba.

Si no se introducen los datos, el sistema indica que se deben introducir todos.

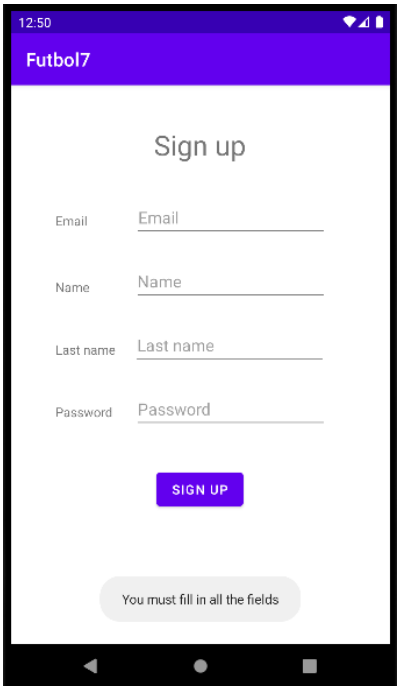

Figura 18a. Prueba ningún campo introducido

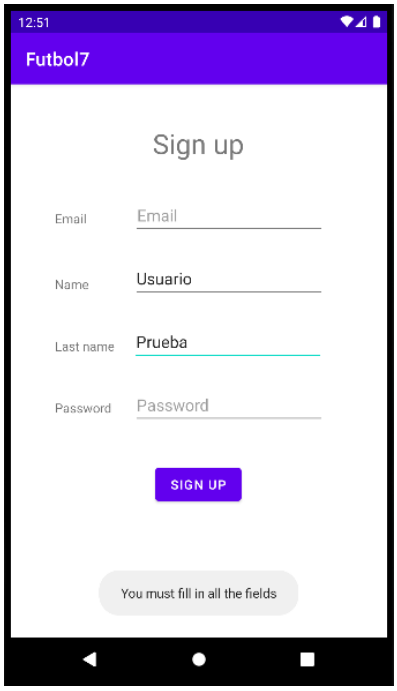

Figura 18b. Prueba algún campo no introducido

De igual forma, el segundo escenario alternativo es introducir un correo electrónico con un mal formato, lo que resulta en que el sistema indica que el correo no tiene el formato correcto.

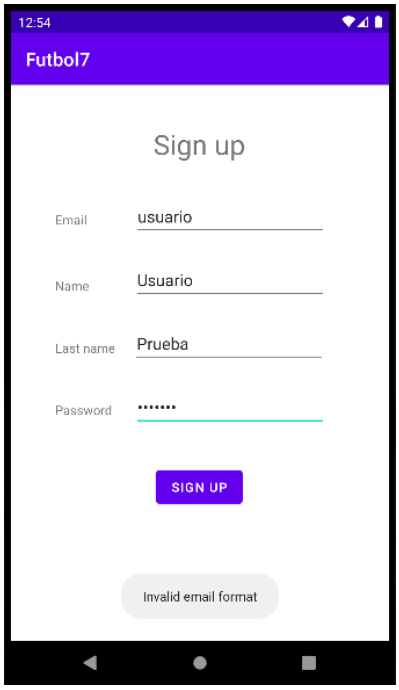

Figura 19. Prueba correo con mal formato

Por último, el último escenario alternativo es introducir una contraseña muy corta, resultando en que el sistema indica que la contraseña debe de tener al menos 6 caracteres.

Figura 20. Prueba contraseña muy corta

En resumen, las pruebas se han basado en verificar que los requisitos, y por tanto, los casos de uso, se han completado con éxito y con los mínimos errores. Al igual que se ha hecho para el caso de uso 1, se han realizado las pruebas correspondientes para todos los demás, consiguiendo así validar el sistema.

4

Conclusiones y líneas futuras

Tras finalizar el proyecto, es momento de echar la vista atrás y ver el proceso hasta este punto. El objetivo era sencillo, realizar una aplicación móvil en la que poder crear y unirse a partidos, además de poder estar en diferentes equipos y poder ver estadísticas relacionadas al desempeño del usuario. Todas estas tareas se han llevado a cabo de forma exitosa, intentando siempre que fuesen lo más intuitivas y sencillas de entender.

De igual manera, en cuanto a los entregables, se ha realizado una API encargada de manejar todas las peticiones y accesos a la base de datos en el lado del servidor, se ha desarrollado una aplicación móvil que actúa como cliente y que cumple con los requisitos impuestos, y se ha completado la documentación complementaria que ayuda a comprender y entender el proceso seguido. Esta documentación se puede ver en el apartado de los apéndices, incluye un manual de usuario, una guía de instalación y un documento general de requisitos.

En cuanto a líneas futuras, es interesante pensar que esta es una primera versión de la aplicación y que se puede actualizar de forma periódica, ya sea para solucionar posibles errores o para añadir nueva funcionalidad. Se podrían implementar características que añadan aspectos como:

- Torneos y ligas. El propio sistema podría crear y gestionar tanto torneos como ligas a los que se podrían unir un número limitado de equipos. Además, podría ser interesante que haya diferentes divisiones en función de factores relacionados con el rendimiento y compromiso de los equipos y de los jugadores.

- Registro de goles. Podría ampliarse el seguimiento de las estadísticas para contemplar la posibilidad de introducir quienes y cuantos goles han metido los jugadores en un partido. Este aspecto podría ser indicado por los capitanes de los respectivos equipos.

- Sistema de puntuación. Podría asignarse un sistema de puntuación asociado a cada jugador, para que, en función de su desempeño en los partidos, pueda optar a encuentros con otros jugadores con un nivel parecido al suyo, fomentando así la competitividad y la seriedad de la competición.

Referencias

[1] **Desarrollo iterativo e incremental.** 22 de septiembre, 2019. Agile.
https://blog.jmbeas.es/2019/09/22/desarrollo-iterativo-e-incremental/

[2] **Java.** Página oficial de Java.
https://www.java.com/es/

[3] **¿Qué es y cómo funciona la Máquina Virtual Java?** Alex Walton.
https://javadesdecero.es/fundamentos/como-funciona-maquina-virtual/

[4] **Spring Boot.** Página oficial Spring.
https://spring.io/web-applications

[5] **¿Qué es una API y cómo funciona?** 20 de enero, 2023. Red Hat.
https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

[6] **¿Qué es REST API?** 31 de julio, 2023. Red Hat.
https://www.redhat.com/es/topics/api/what-is-a-rest-api

[7] **Diseños. XML en Android.** Android para desarrolladores.
https://developer.android.com/guide/topics/ui/declaring-layout?hl=es-419

[8] **Introducción a las actividades.** Android para desarrolladores.
https://developer.android.com/guide/components/activities/intro-activities?hl=es-419

[9] **Fragmentos.** Android para desarrolladores.
https://developer.android.com/guide/components/fragments?hl=es-419

[10] **Cloud Firestore.** Documentación de Firebase.
https://firebase.google.com/docs/firestore?hl=es-419

[11] **Android Studio.** Página oficial Android Studio.
https://developer.android.com/studio

[12] **Cómo ejecutar apps en Android Emulator.** Página oficial Android Studio.
https://developer.android.com/studio/run/emulator?hl=es-419

[13] **Cómo ejecutar apps en Android Emulator.** Página oficial Android Studio.
https://developer.android.com/studio/run/emulator?hl=es-419

[14] **Cloudinary.** Página oficial Cloudinary.
https://cloudinary.com/

[15] **Patrón de repositorio.** Página oficial Android Studio.
https://developer.android.com/codelabs/basic-android-kotlin-training-repository-pattern?hl=es-419#3

[16] **¿Qué es el patrón MVC en programación y por qué es útil?.** 15 de octubre, 2019. José María Aguilar.
https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx

[17] **Archivo JSON: ¿qué es y para qué sirve en las páginas web?.** 19 de agosto, 2021. Ivan de Souza.
https://rockcontent.com/es/blog/archivo-json/

Referencias Figuras

[F1] **Building a Simple Application with Spring Boot.** 4 de septiembre, 2019.

Randika's tech blast.

http://randikatech.blogspot.com/2019/09/get-your-hands-dirty-with-micro-services.html

[F2] **Patrón de repositorio.** Página oficial Android.

https://developer.android.com/codelabs/basic-android-kotlin-training-repository-pattern?hl=es-419#3

[F3] **¿Qué es una api rest?.** Dos Setenta.

https://dossetenta.com/que-es-una-api-rest/

Apéndice A

Manual de usuario

Introducción

Este documento servirá como guía para los usuarios finales sobre cómo interactuar y utilizar de manera efectiva la aplicación. Para ello, se detallará el funcionamiento y las principales características del sistema, dando instrucciones claras y concisas para el uso correcto de esta.

Registro e inicio de sesión de usuarios

Una vez iniciada la aplicación, lo primero que se observa es un formulario para realizar el inicio de sesión. Se deberá introducir un correo electrónico y una contraseña que previamente han tenido que ser registradas en el sistema.

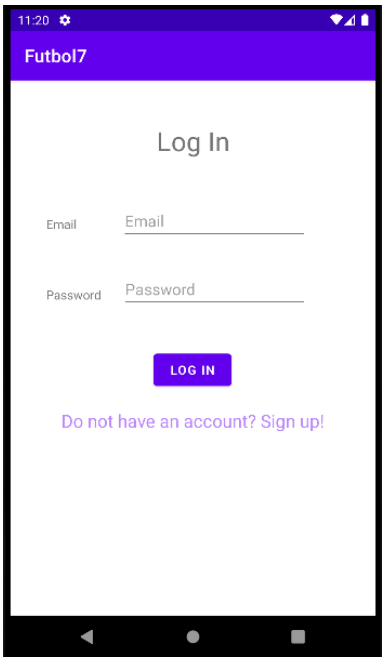

Figura 21. Inicio de sesión

Para el registro, se debe pulsar en el texto "¿No tienes una cuenta? ¡Regístrate!" y aparecerá un nuevo formulario donde introducir un correo electrónico, un nombre, un o unos apellidos y una contraseña, de al menos 6 caracteres. Finalmente, se pulsa sobre el botón correspondiente y si todos los campos son correctos, se pasará a la vista principal.

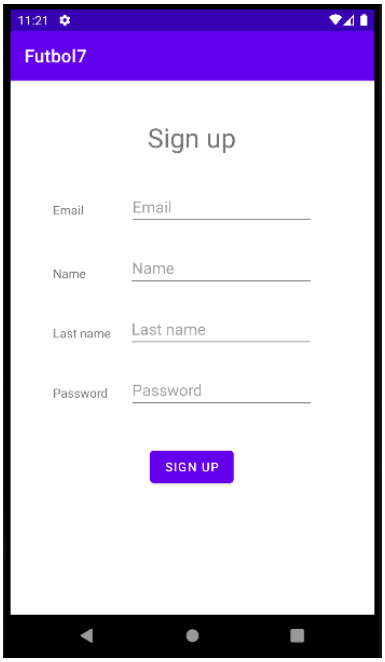

Figura 22. Registro de un usuario

Vista principal

Tras realizar el inicio de sesión o el registro, se muestra una pantalla en la que se pueden realizar casi todas las acciones principales de la aplicación.

En el lado inferior, se puede observar una barra con la que se puede navegar entre los tres fragmentos disponibles, pudiendo hacerlo también deslizando hacia los lados. En el lado superior, se podrán ver iconos que realizarán acciones como cerrar sesión o mostrar las notificaciones o a los amigos.

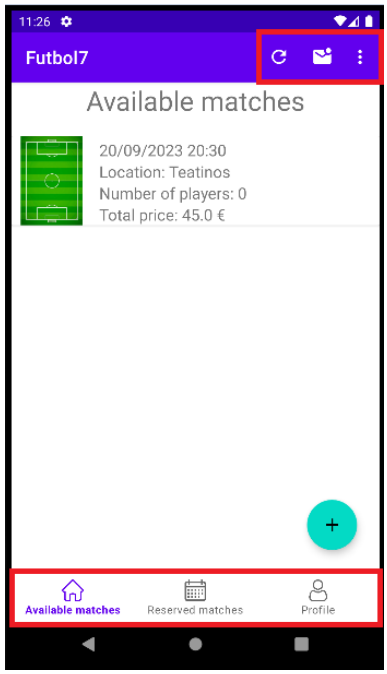

Figura 23. Vista principal

Partidos disponibles

El primero de los fragmentos muestra los partidos disponibles a los que un jugador podrá unirse. Pulsando sobre un partido se verán más características sobre ese encuentro, pudiendo realizar la reserva o cancelación de una posición. Además, si se pulsa sobre el botón flotante con el símbolo "+" podrá crearse un partido (véase más adelante).

Partidos reservados y completados

El segundo enseña tanto los partidos reservados como los partidos completados. Al igual que en el primer fragmento, pulsando sobre algún partido se accederá a él para verlo en detalle.

Perfil

El tercero y último muestra el perfil del usuario que haya iniciado sesión. Se puede observar la foto de perfil, el nombre completo, los equipos y las estadísticas del jugador.

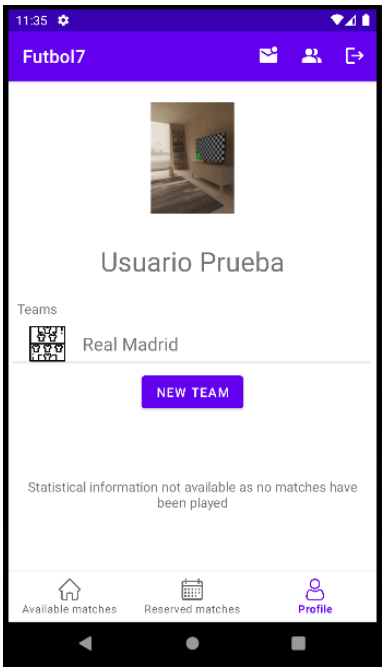

Figura 24. Perfil del usuario

Para editar la foto, bastará con pulsar la imagen e indicar si se quiere usar la cámara o si se quiere seleccionar de la galería. Siguiendo los pasos para cada una de las opciones quedará actualizada la foto de perfil.

En cuanto a los equipos, pulsando sobre cualquiera de ellos se puede acceder a información más detallada, en concreto, se podrá visualizar la imagen y el nombre seguido de una lista con todos los jugadores que lo componen, pudiendo visitar el perfil de cada uno de ellos. Si el usuario que visualiza al equipo es el capitán, podrá editar la imagen de la misma manera que si fuese su propia foto de perfil.

Figura 25. Vista de un equipo

Por último, pulsando sobre el botón "Crear equipo", se podrá formar uno introduciendo el nombre.

Creación de partidos

Una vez pulsado el botón de crear partido, habrá que completar un formulario indicando la localización, el día, la hora y el precio total.

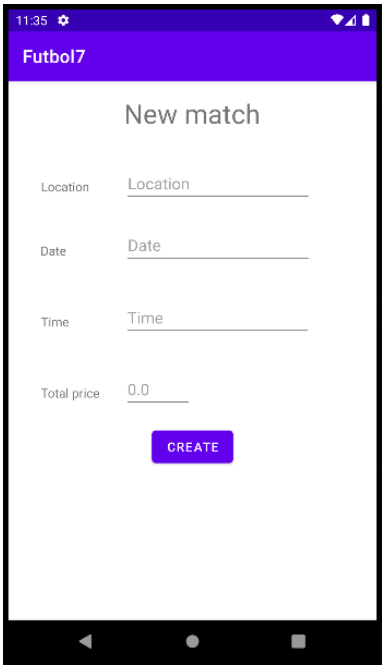

Figura 26. Creación de un partido

Reserva/cancelación de partidos

Cuando se accede a la información de un partido, se puede ver una lista con los jugadores tanto del equipo local como del visitante. Cuando una posición no está ocupada, la aplicación lo indicará. Pulsando sobre esta posición se podrá realizar una reserva.

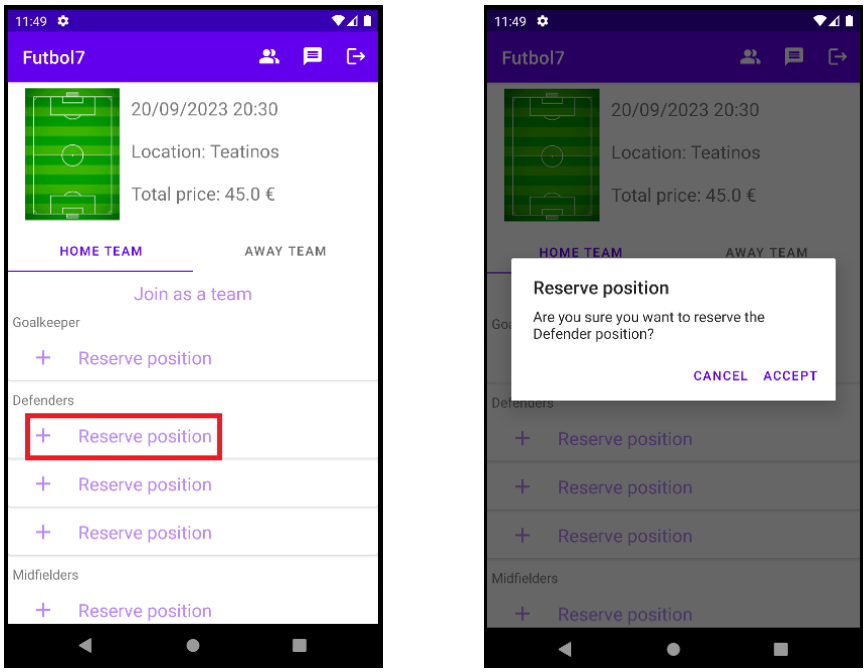

Figura 27. Reserva de una posición

De igual forma, para cancelar una reserva bastará con mantener pulsado sobre la reserva del usuario.

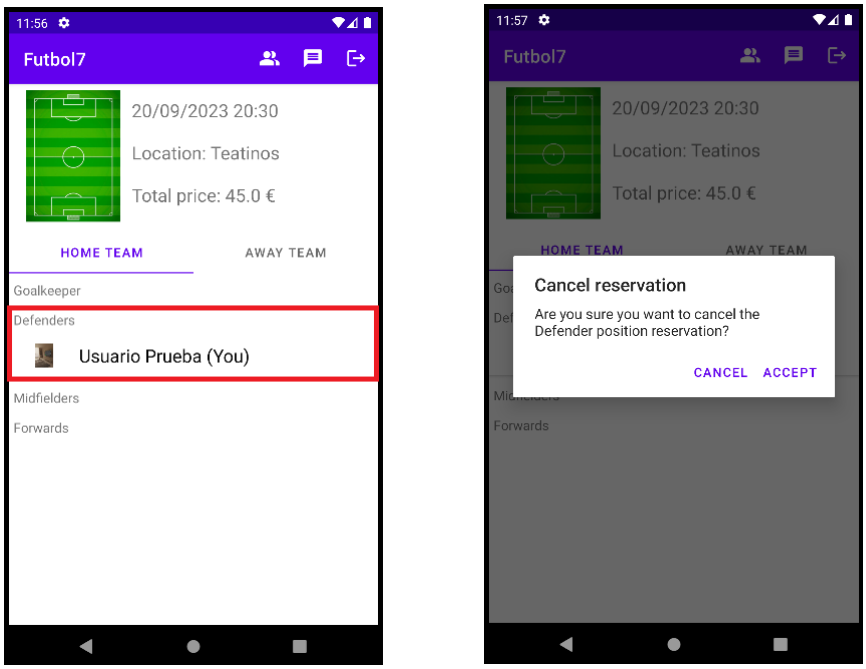

Figura 28. Cancelación de una reserva

Inscripción/cancelación de un equipo a un partido

Estando en la vista de un partido, se puede inscribir a un equipo pulsando sobre el botón indicado. Tras esto, se preguntará que equipo y tras pulsar sobre él quedará inscrito.

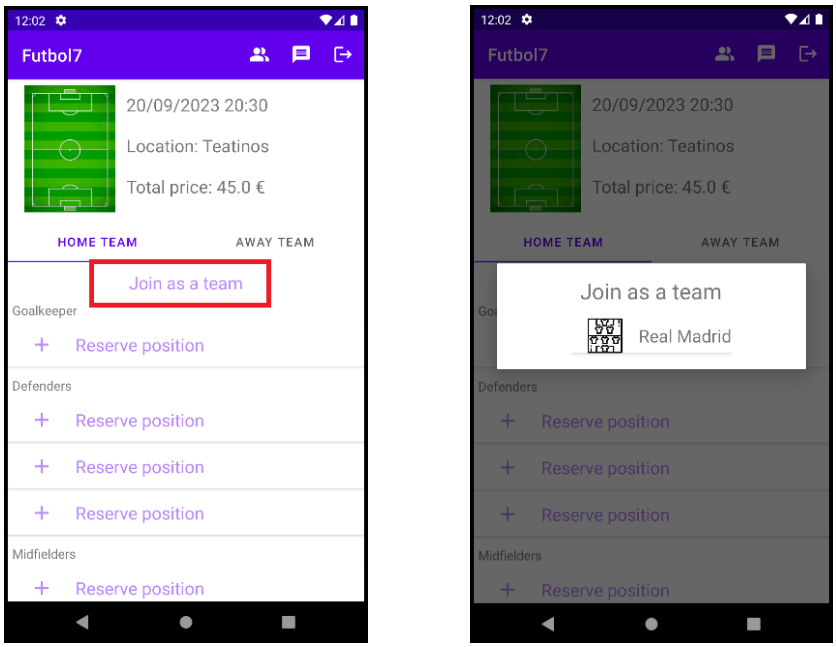

Figura 29. Inscripción de un equipo en un partido

Para eliminar la inscripción del equipo, bastará con pulsar nuevamente sobre el botón señalado, preguntando si se está seguro y confirmando la cancelación.

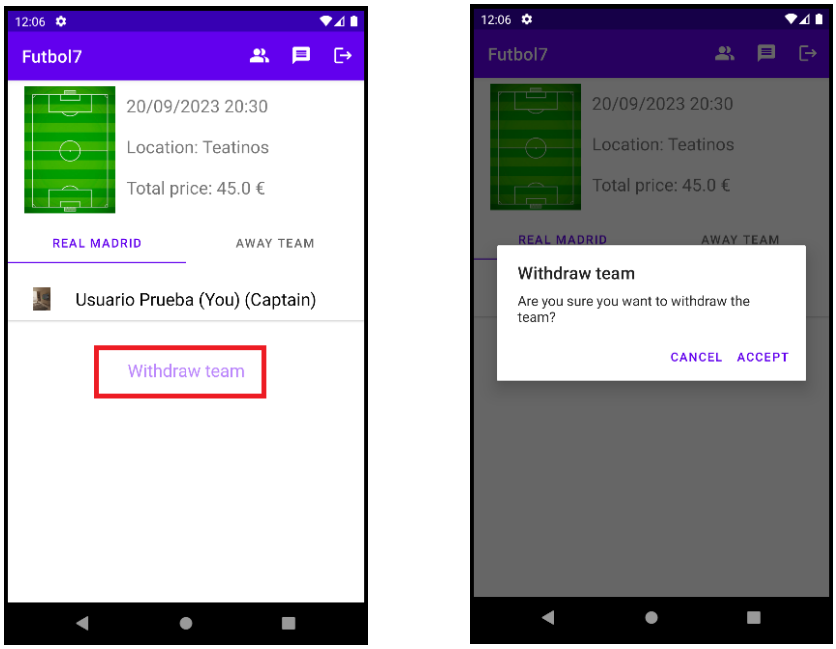

Figura 30. Inscripción de un equipo en un partido

Apéndice B

Guía de instalación

Introducción

Este documento tiene como propósito enseñar y ayudar a cualquier usuario a instalar y configurar la aplicación en su totalidad. Se explicarán tanto los pasos para la parte del servidor como para la parte del cliente.

Servidor

1. Descomprimir la carpeta en la ruta deseada.

2. Abrir Visual Studio Code, si no está instalado puede descargarse en su página oficial https://code.visualstudio.com/.

3. Abrir la pestaña extensiones e instalar "Extension Pack for Java".

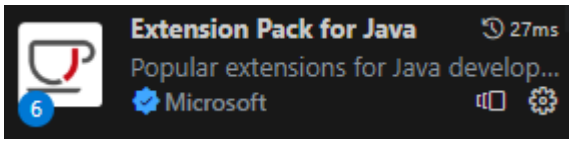

Figura 31. Extension Pack For Java

4. Abrir el menú superior izquierdo "File"(Archivo), pulsar "Open folder" (Abrir carpeta) y seleccionar la carpeta descomprimida.

5. Abrir la carpeta src/main/java/com/api, pulsar con el clic derecho el archivo Main.java y seleccionar "Run Java".

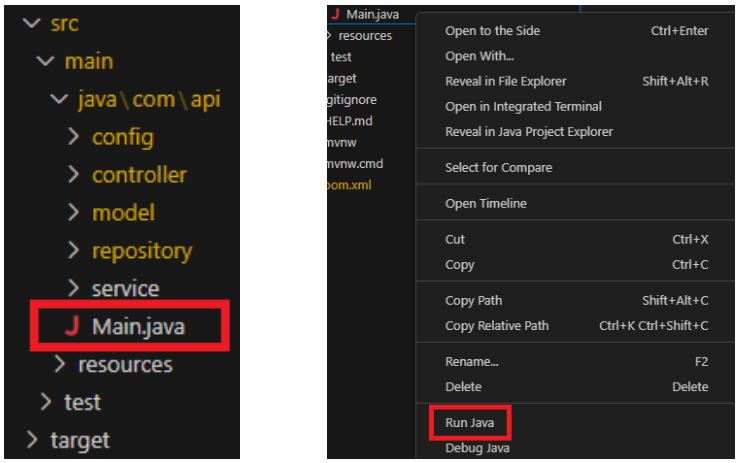

Figura 32. Archivo Main.java

6. En el terminal correspondiente se podrá ver como se ha inicializado y como se está ejecutando la API.

Cliente

1. Descomprimir la carpeta en la ruta deseada.

2. Abrir Android Studio, si no está instalado puede descargarse en su página oficial https://developer.android.com/studio.

3. Seleccionar "Open" en la parte superior y buscar la carpeta descomprimida.

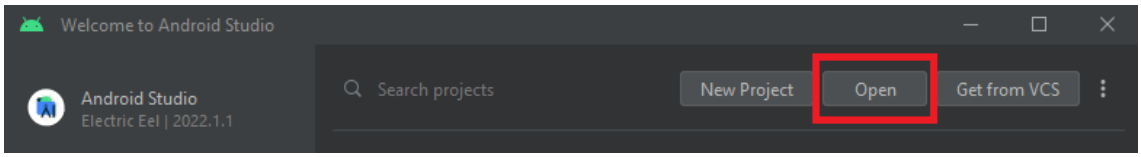

Figura 33. Botón Open

4. Ejecutar en el simulador móvil o dispositivo físico deseado.

5. Si al ejecutar por primera vez el proyecto, tras intentar iniciar sesión aparece el error siguiente, bastará con recompilar y volver a ejecutar nuevamente la aplicación:

Error: Default FirebaseApp is not initialized in this process com.app.futbol7. Make sure to call FirebaseApp.initializeApp(Context) first.

Apéndice C

Documento General de Requisitos

Introducción

La aplicación móvil para el encuentro y realización de partidos de fútbol 7 tiene como objetivo solucionar la dificultad de encontrar los 14 jugadores de un partido. Este documento tiene como propósito definir y establecer los requisitos funcionales y no funcionales para el desarrollo de esta aplicación.

Actores principales

- El propio alumno, encargado de desarrollar y completar cada una de las partes del proyecto.
- El tutor académico, encargado de supervisar y orientar al alumno.
- Los usuarios finales, que serán aquellos jugadores que usarán la aplicación cuando esté completada y lanzada.
- La Universidad de Málaga, encargada de proporcionar los recursos necesarios y evaluar, calificar y validar el proyecto.

Requisitos Funcionales

Registro e inicio de sesión de usuarios

1. Los usuarios podrán registrarse utilizando un correo, un nombre, unos apellidos y una contraseña.
2. Los usuarios podrán iniciar sesión utilizando su correo y su contraseña.
3. Los usuarios podrán cerrar sesión.

Vista principal

4. Los usuarios podrán visualizar una lista con los partidos disponibles.
   4.1 Los usuarios podrán recargar los partidos disponibles.
5. Los usuarios podrán visualizar una lista con sus partidos reservados y partidos completados.
6. Los usuarios podrán visualizar su perfil.

Partidos disponibles, reservados y completados

7. Los usuarios podrán visualizar un partido y sus características.
8. Los usuarios podrán ver una lista con los jugadores de cada equipo, local y visitante.
9. El creador del partido podrá editar la foto del encuentro.
10. Los usuarios podrán reservar una posición en uno de los equipos.
11. Los usuarios podrán cancelar su reserva de una posición.
12. Los usuarios podrán inscribir a uno de sus equipos.
13. Los usuarios podrán retirar la inscripción de su equipo.
14. Los usuarios podrán acceder al chat del partido.
    14.1. Los usuarios podrán ver los mensajes enviados.
    14.2. Los usuarios podrán enviar mensajes.
    14.3. Los usuarios que sean capitanes o emisores del mensaje podrán eliminar un mensaje.
15. Los usuarios podrán actualizar el resultado del partido una vez haya acabado.

Creación de partidos

16. Los usuarios podrán crear partidos indiciando localización, día, hora y precio total.

Perfil

17. Los usuarios podrán ver su foto de perfil y su nombre.
18. Los usuarios podrán editar su foto de perfil mediante cámara o galería de fotos.
19. Los usuarios podrán ver una lista con los equipos a los que pertenecen.
20. Los usuarios podrán ver estadísticas relacionadas a su número de partidos, incluyendo victorias, derrotas y empates.

Equipos

21. Los usuarios podrán crear equipos mediante un nombre.
22. Los usuarios podrán ver la foto del equipo.
    22.1 El capitán del equipo podrá editar la foto mediante cámara o galería de fotos.
23. Los usuarios podrán ver una lista con los integrantes del equipo.
24. Los usuarios podrán ver los perfiles de los demás compañeros.
25. Los usuarios podrán invitar a otros jugadores a sus equipos.
26. Los usuarios podrán aceptar o rechazar invitaciones de equipos.
27. Los usuarios podrán dejar un equipo.
28. El capitán del equipo podrá eliminar a un compañero del equipo.

Amigos

29. Los usuarios podrán ver una lista de sus amigos.
30. Los usuarios podrán enviar peticiones de amistad.
31. Los usuarios podrán aceptar y rechazar peticiones de amistad.
32. Los usuarios podrán eliminar a un amigo.

Requisitos No Funcionales

1. La aplicación se encontrará tanto en español como en inglés.
2. La aplicación deberá responder de forma rápida y fluida.

3. La aplicación deberá ser fácil de entender y de aprender su utilización.
4. La aplicación deberá ser compatible con la mayoría de dispositivos Android.
5. La aplicación deberá tener un tiempo de respuesta menor a los 3 segundos.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA