Universidad Zaragoza
1542

Trabajo Fin de Grado

Aplicación Android basada en Firebase para la gestión y evaluación de comportamientos antideportivos en competiciones de tenis

Autor

Víctor de Lera Plaza

Director

José García Moros

Escuela de Ingeniería y Arquitectura

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

2019

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe entregarse en la Secretaría de la EINA, dentro del plazo de depósito del TFG/TFM para su evaluación).

D./Dª. Víctor de Lera Plaza ,en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza, Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado (Título del Trabajo)

Aplicación Android basada en Firebase para la gestión y evaluación de comportamientos antideportivos en competiciones de tenis

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 19 de noviembre de 2019

Fdo: Víctor de Lera

RESUMEN

Este Trabajo Fin de Grado consiste en el desarrollo de una aplicación Android, basada en la plataforma de Google para desarrollo de aplicaciones móviles **Firebase**, que, fundamentalmente, facilite la gestión y evaluación de los comportamientos deportivos y antideportivos en las competiciones de tenis de base. La aplicación permite a los organizadores de torneos de tenis crear formularios para evaluar distintos aspectos de la competición y a los jugadores de estos torneos contestar las preguntas propuestas en sus dispositivos móviles al finalizar los partidos.

El sistema desarrollado se basa en una arquitectura cliente-servidor donde la plataforma **Firebase** actúa como servidor del sistema y la aplicación es el cliente que, haciendo uso de las diferentes APIs de la plataforma, se comunica con ella.

La aplicación implementada en este proyecto permite el registro, inicio y cierre de sesión de usuarios mediante el uso de la API **Firebase Authentication**.

Los administradores de la aplicación pueden crear torneos asignando a los jugadores, así como los formularios que contestarán al finalizar sus partidos. Para agilizar la generación de formularios, estos se pueden crear desde fuera de la aplicación importando un archivo Excel en el almacenamiento en la nube **Cloud Storage**.

Una función, desarrollada en **Cloud Functions** (funciones desarrolladas en **Javascript** y ubicadas en la nube que actúan como backend de la aplicación)**,** lee la información del archivo y fija los datos en la base de datos **NoSQL** de **Firebase**, **Cloud Firestore** haciendo uso de su API para **NodeJs.** Una vez que la información está en la base de datos es accesible desde la aplicación.

La plataforma envía notificaciones a los jugadores de los torneos haciendo uso de la API de **Cloud Messaging** desde otra función en **Cloud Functions** que se activa al producirse ciertos eventos en **Cloud Firestore**.

Los jugadores contestan los formularios, las respuestas se envían y se guardan en **Cloud Firestore** activando otra función en **Cloud Functions** que crea un archivo Excel con las respuestas de los jugadores y lo sube a **Cloud Storage,** en una carpeta con el nombre del torneo, separando, de esta manera, las respuestas de cada uno de los torneos.

Con la aplicación se facilita la recogida de las respuestas de los competidores y con las respuestas almacenadas en archivos Excel se facilita el tratamiento y evaluación de los datos.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El conocimiento y el análisis de la incidencia de comportamientos deportivos y no deportivos en el tenis base competitivo en la Comunidad Autónoma de Aragón, primeramente, y la posterior aportación de algunas acciones de prevención primaria que permitiesen mejorar el comportamiento y la concienciación de jugadores, padres y entrenadores acerca de no estar implicados en incidentes no deportivos, fueron los dos propósitos objeto de estudio de la realización de la tesis doctoral (Lacambra, 2015) de donde procede la aplicación realizada en este TFG.

Para la consecución del primer objeto de estudio de esta investigación diseñaron un instrumento de evaluación que midiese los comportamientos deportivos y no deportivos que acontecían en los partidos del tenis de base, especialmente en las figuras de los jugadores y de los padres. Éste fue el Registro de Evaluación de la Deportividad en los Partidos de Tenis (REDPT).

1.1 Contexto

Este TFG surge fruto de otras colaboraciones previas en proyecto de eSalud entre Fernando Gimeno Marco (Profesor de Psicología de la Actividad Física y del Deporte) y el grupo Communications Networks and Information Technologies (CeNIT).

La Dirección Técnica de la Escuela Territorial de la Federación Aragonesa de Tenis (FAT) se encarga de que los jugadores de tenis de categorías base de la Comunidad Autónoma de Aragón cumplimenten el Registro de Evaluación de la Deportividad en los Partidos de Tenis (REDPT) cuando finalizan los partidos de una determinada competición, con la finalidad de evaluar los comportamientos deportivos y antideportivos que acontecen durante un partido de tenis. El REDPT se cumplimenta en formato papel, por lo que posteriormente, uno de los miembros de la Dirección Técnica

de la FAT, es el encargado de realizar el traslado de los datos recogidos a un formato digital para su posterior evaluación.

Una de las propuestas de mejora en la tesis doctoral de Lacambra (2015), fue la de la posibilidad de que el REDPT presentase alguna adaptación en el futuro, añadiendo posibles conductas que pudiesen aparecer en este deporte y que cobraran la importancia necesaria para ser incluidas dentro del mismo, adaptándose, a su vez este registro para los padres, entrenadores o jueces árbitros, con la finalidad de completar la información aportada por los jugadores.

Además, el traspaso de datos de papel a un archivo como Excel puede resultar un trabajo muy laborioso y en el cual se puede cometer un error humano que comprometa los datos recogidos en el estudio.

Se vio la necesidad de automatizar todo esto proponiendo una aplicación Android que permitiera a los jugadores registrarse para que pudieran recibir notificaciones para contestar formularios al finalizar los distintos partidos de cada torneo creados por la Federación Aragonesa de Tenis desde la aplicación con distintos tipos de preguntas.

1.2 Estado del arte

En el mercado no existen actualmente aplicaciones que sean capaces de crear torneos con los jugadores registrados y añadir formularios a esos torneos para que contesten los participantes limitando las respuestas de cada uno de los formularios por cada ronda que participan del torneo.

Existen varias herramientas para la creación de formularios, sobre todo en entornos web, que guardan las respuestas en hojas de cálculo. Ninguna de las analizadas tiene la posibilidad de crear una competición a la que asociar determinados formularios para que únicamente los participantes contesten a las preguntas. Las más conocidas son:

- **Google Forms (web):** Herramienta gratuita para la creación de formularios, solo se necesita una cuenta de Google para poder usarla. Guarda automáticamente los

resultados del formulario en una hoja de cálculo de Google Sheets. Para contestar el formulario necesitas una invitación por correo o disponer del enlace.

- **Microsoft Forms (web):** Es la herramienta de Microsoft para la creación de formularios y funciona en gran medida de la misma manera que Google Forms. Es gratuita, solo se necesita una cuenta de Microsoft para usarla.

- **JotForm (web, Android, iOS):** Los formularios de JotForm son mucho más personalizables que los formularios creados en Google Forms y Microsoft Forms. Es gratuita pero dispone de varios planes de pago que aportan más funcionalidades.

- **Formstack (Web, Android, iOS):** Formstack dispone de una función de flujos de trabajo para enviar los datos de los formularios a las personas correctas para revisarlos en línea o por correo electrónico antes de enviarlo a la siguiente persona que los trate. Está orientado a empresas e industrias para la recolección de datos de los clientes y su posterior análisis. No es gratuita pero se puede probar gratuitamente durante 14 días.

Para la creación y gestión de torneos existen otras herramientas muy potentes capaces de crear y gestionar torneos de todo tipo de deportes aunque ninguna otorga la posibilidad de asociar preguntas de evaluación sobre los partidos disputados a los participantes. Las más conocidas son:

- **Xporty (web, Android, iOS):** Herramienta para la creación de ligas y torneos de todo tipo de deportes, disponible en aplicación web y móvil. Ofrece una página web a la organización del torneo con multitud de funcionalidades para la gestión de las competiciones y para el seguimiento de los resultados. Los usuarios pueden inscribirse en los torneos. Permite crear formularios para la inscripción en los torneos pero no permite que los participantes contesten preguntas sobre los partidos disputados. Dispone de un plan gratuito con funcionalidades limitadas y varios planes de pago.

- **Leverade (web, Android, iOS):** Herramienta similar a Xporty pero con alguna funcionalidad menos. Es gratuita con funcionalidades limitadas y dispone de

varios planes de pago, algunos te permiten crear fácilmente una web con el software integrado para crear las competiciones en distintos ámbitos.

- **DoLeague (web, Android, iOS):** Herramienta para la creación de torneos y ligas similar a las anteriores. Permite la prueba gratuita con limitación en sus funcionalidades y varios planes de pago.

1.3 Motivación y objetivos

El objetivo principal es informatizar el sistema de Registro de Evaluación de la Deportividad en los Partidos de Tenis (REDPT) de la Federación Aragonesa desarrollando una aplicación en Android para que los jugadores instalen en sus dispositivos móviles y en la que se tendrán que registrar para utilizar su cuenta de usuario, que les permitirá recibir notificaciones, recordándoles que pueden realizar la evaluación del partido que acaban de disputar.

Como objetivo secundario la aplicación tiene que ser capaz de replicar el sistema REDPT y de crear otro tipo de formularios similares de este contenido para poder cubrir las propuestas de mejora en la tesis doctoral de Lacambra (2015). Para ello, la aplicación es capaz de crear formularios con cuatro tipos de preguntas distintas y con posibilidad de separarlas en secciones que se almacenan en la base de datos no relacional Cloud Firestore de Firebase para su posterior visualización en la aplicación.

La Dirección Técnica de la Escuela Territorial de la FAT analiza los datos por torneo y los compara con otros torneos realizados, por ello la aplicación tiene que ser capaz de crear estos torneos para poder asignar los distintos formularios creados en la aplicación y también que jugadores serán los que disputen el torneo para así poder recoger los datos de cada torneo por separado.

De esta manera las respuestas de los jugadores se almacenarán organizadas por carpetas, con los nombres de los torneos, en Cloud Storage, el servicio de almacenamiento de archivos que proporciona Firebase, facilitando la recogida de datos para su estudio.

Otro de los aspectos destacados que tiene la realización de esta herramienta de evaluación, es la de su carácter novedoso, ya que en el deporte del tenis no existen instrumentos de medida que reflejen los aspectos que se van a medir a través de esta aplicación. En otros deportes, como por ejemplo, en el fútbol, sí que existen cuestionarios que midan factores relevantes a los comportamientos deportivos y no deportivos de los deportistas, pero en el tenis este hecho es completamente novedoso hasta la aparición del REDPT, por lo que su mejora a través de la aplicación "Tennis Match Assessment and Counselling" va a dotarle de una calidad mucho mayor y de una expansión probablemente desconocida hasta el momento.

1.4 Cronograma

En la figura 1.1 se puede observar un diagrama de Gantt mostrando el tiempo de desarrollo de cada una de las etapas del proyecto (requisitos, análisis, diseño, implementación y pruebas).

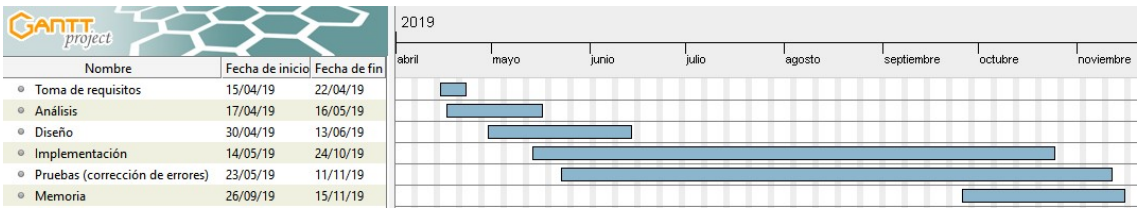


Figura 1.1: Diagrama de Gantt del proyecto

1.5 Estructura del documento

El contenido de la memoria sigue la siguiente estructura:

- En el **Capítulo 1** se introduce el Trabajo de Fin de Grado hablando del contexto que lo rodea así como del estado del arte, la motivación y objetivos del TFG.
- En el **Capítulo 2** se analizan los requisitos funcionales y no funcionales del sistema así como las herramientas que se han decidido utilizar.

- En el **Capítulo 3** se detalla el desarrollo del sistema y su funcionamiento, la arquitectura de la aplicación y los componentes del servidor.

- En el **Capítulo 4** se especifican las pruebas realizadas para validar el desarrollo del sistema.

- En el **Capítulo 5** se explican las conclusiones obtenidas con la resolución del trabajo y las posibles líneas futuras.

- En el **Capítulo 6** se incluye la bibliografía consultada en el desarrollo del trabajo.

En la parte final del trabajo se encuentran varios Anexos que complementan la información expuesta en la memoria.

- En el **Anexo A** se especifica la forma de distribución de la aplicación entre los distintos usuarios.

- En el **Anexo B** se detalla la guía de instalación de la aplicación.

- En el **Anexo C** se incluyen las pantallas de la plataforma de desarrollo Firebase.

- En el **Anexo D** se explican las actividades, fragmentos y métodos desarrollados en la aplicación.

Capítulo 2

Definición de requisitos y especificaciones

En esta sección se especifican los requisitos de la aplicación tras estudiar las necesidades en las diferentes reuniones con la dirección de la FAT.

2.1 Análisis y requisitos de la aplicación

2.2.1 Stakeholders

- **Administrador:** Usuario que crea los formularios del sistema REDPT y los torneos que se van a jugar. Se encarga de administrar los torneos en juego para que los jugadores puedan recibir notificaciones que les permitan contestar a las preguntas propuestas.
- **Jugador:** Usuario que utiliza la aplicación en su dispositivo Android para contestar a las preguntas propuestas por el Administrador.

2.2.2 Requisitos funcionales

- **RF1.** La aplicación permite registrarse, iniciar y cerrar sesión a un usuario.
- **RF2.** El sistema permite la creación de formularios mediante la lectura de un archivo Excel con el formato establecido.
- **RF3.** El sistema almacena las respuestas de los jugadores en un archivo Excel para que los organizadores puedan acceder a los datos.
- **RF4.** La aplicación permite la edición del perfil de usuario.
- **RF5.** La aplicación es capaz de aplicar distintos roles a los usuarios.
- **RF6.** La aplicación muestra distintas funcionalidades según el rol del usuario (administrador y jugador).

Administrador:

- **RF7.** La aplicación permite crear formularios.
- **RF8.** La aplicación permite ver y eliminar los formularios creados.
- **RF9.** La aplicación permite crear torneos con nombre, lugar, fecha y número de rondas.
- **RF10.** La aplicación permite asignar formularios y jugadores a los torneos en su creación.
- **RF11.** La aplicación permite administrar los torneos creados gestionando los vencedores de cada ronda.
- **RF12.** La aplicación permite ver los usuarios registrados y editar el rol del resto de usuarios.

Jugador:

- **RF13.** La aplicación es capaz de recibir notificaciones informando de la posibilidad de contestar un formulario.
- **RF14.** La aplicación es capaz de mostrar los formularios que el jugador tiene pendientes por contestar.
- **RF15.** La aplicación permite que el jugador sea capaz de seleccionar que formulario quiere contestar.
- **RF16.** La aplicación muestra las preguntas del formulario seleccionado y permite su envío.

2.2.3 Requisitos no funcionales

- **RNF1.** La aplicación solo mostrará los torneos en curso para su gestión en el mismo día de la ejecución del torneo.
- **RNF2.** La aplicación solo permitirá contestar los formularios el mismo día del la realización del torneo.
- **RNF3.** La aplicación permite el envío de formularios incompletos.
- **RNF4.** La aplicación solo permite una única respuesta a un jugador por cada ronda y formulario del torneo.
- **RNF5.** La aplicación no debe suponer un coste económico.

2.2.4 Materiales y herramientas utilizadas

Tras la recolección de los requisitos de la aplicación se realizó una búsqueda para elegir las herramientas y materiales que eran convenientes utilizar en el desarrollo de la aplicación. Uno de los puntos más importantes para el desarrollo de esta aplicación es la obligatoriedad de utilizar un servidor de backend para la gestión de usuarios, almacenamiento de datos y envío de notificaciones. Se valoraron varias posibilidades como la creación y configuración de un servidor y una base de datos en Amazon Web Services pero tanto el coste económico como el temporal supondrían un problema para el desarrollo.

El descubrimiento de Firebase como plataforma de desarrollo de la aplicación fue un hito importante ya que permitió la ejecución de este TFG pudiendo satisfacer todos los requisitos que se solicitan para la aplicación.

A continuación se describen las herramientas utilizadas en el desarrollo:

- **Android**: Sistema operativo pensado para teléfonos móviles con pantalla táctil que está basado en el kernel de Linux, un núcleo de sistema operativo libre, multiplataforma y de código abierto.

- **Java**: Lenguaje de programación orientado a objetos con posibilidad de ejecutarse en cualquier dispositivo que tenga una maquina virtual de Java. La maquina virtual en Android desde la versión 5.0 Lollipop es Android Runtime (ART). En este lenguaje se desarrolla la aplicación.

- **Android SDK [1]:** Kit de desarrollo de Software de Android, incluye un conjunto de herramientas de con las que desarrollar la aplicación.

- **AndroidStudio [1]:** Entorno de desarrollo integrado (IDE) oficial de Android para el desarrollo de aplicaciones, está basado en IntelliJ IDEA.

- **Firebase [2]:** Plataforma para el desarrollo de aplicaciones web y aplicaciones móviles de Google, ubicada en la nube. Está integrada con Google Cloud Platform. Nos permite crear este proyecto sin necesidad de un servidor. Las herramientas se incluyen en los SDK para los dispositivos móviles y web, por lo que no es necesario crear un servidor para el proyecto. Usa la infraestructura de Google y escala automáticamente para cualquier tipo de aplicación. Es compatible con diversas plataformas, entre ellas se encuentra Android. Las herramientas de Firebase utilizadas para este proyecto son Firebase Authentication, Cloud.

- **Firebase Authentication [3]:** Proporciona los servicios de backend y el SDK para autenticar a los usuarios de la aplicación. Admite la autenticación de diversas formas. Se integra con otros servicios de Firebase y aprovecha los estándares como OAuth 2.0 y OpenID Connect. Se utiliza para el registro, inicio de sesión y gestión de los usuarios de la aplicación.

- **Cloud Firestore [4]:** Base de datos NoSQL flexible, escalable y en la nube a fin de almacenar y sincronizar datos para la programación en el lado del cliente y del servidor. Ofrece integración con otros productos de Firebase y Google Cloud Platform. Se utiliza para almacenar todos los datos generados por la aplicación, formularios creados, torneos, usuarios registrados, respuestas de los jugadores y también para disparar los eventos de Cloud Functions que permiten enviar las notificaciones a los jugadores.

- **Cloud Storage[5]: S**ervicio de almacenamiento construido para la escala de Google. Los SDK de Firebase para Cloud Storage agregan la seguridad de Google a las operaciones de carga y descarga de archivos para la aplicación. Se utiliza para almacenar los archivos Excel con las respuestas de los jugadores.

- **Cloud Messaging[6]:** Solución de mensajería multiplataforma que permite enviar mensajes de forma segura y gratuita. Se utiliza para enviar notificaciones a los jugadores integrándolo con Cloud Functions.

- **Cloud Functions[7]:** Permite ejecutar automáticamente el código de backend en respuesta a eventos activados por funciones de Firebase y solicitudes HTTPS. El código se almacena en la nube de Google y se ejecuta en un entorno administrado sin necesidad de administrar ni escalar servidores. Se utiliza para enviar notificaciones a los usuarios, almacenar las respuestas de los jugadores en Cloud Storage y crear formularios en Cloud Firestore cuando se disparan distintos eventos generados por la aplicación.

- **Javascript:** Lenguaje de programación multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa. Es más conocido como lenguaje para páginas web aunque también es utilizado en entornos sin navegador como Node.js. Las funciones de Cloud Functions se han desarrollado en este lenguaje.

- **Node.js [8]:** Entorno de ejecución de JavaScript orientado a eventos asíncronos. Se utiliza para desarrollar las funciones de Cloud Functions.

- **Npm [9]:**Node Package Manager, es el gestor de paquetes por defecto para Node.js, con el podemos obtener cualquier librearía de manera simple. Se utiliza para instalar Firebase CLI.

- **Firebase CLI [10]:** Es la interfaz de línea de comandos de Firebase. Se necesita para inicializar el SDK de Firebase para Cloud Functions y para desplegar las funciones desarrolladas en Firebase.

- **SheetJS [11]:** Librería en Javascript puro que es capaz de leer y escribir en varios formatos de hojas de cálculo. Se utiliza para leer archivos Excel con un formato definido en una de las funciones de Cloud Functions.

- **ExcelJS [12]:** Librería en Javascript para leer y escribir archivos en formatos XLSX y JSON. Se utiliza para crear y modificar los archivos Excel almacenados en la plataforma con las respuestas de los jugadores.

Capítulo 3

Desarrollo del sistema

En este capítulo se detalla el desarrollo y funcionamiento de los componentes del sistema, la aplicación Android y el servidor de la plataforma Firebase, así como su arquitectura, los diagramas de clases y componentes y los diagramas de la base de datos.

3.1 Descripción General del Sistema

En este apartado, se describen los componentes que conforman el sistema desarrollado. Para satisfacer los requisitos, el sistema está compuesto por dos componentes: la aplicación Android y la plataforma de desarrollo Firebase.

La aplicación es el componente cliente del sistema mientras que Firebase es el servidor que se necesita para almacenar datos, gestionar usuarios y enviar notificaciones.

El diagrama de componentes del sistema se observa en la Figura 3.1.

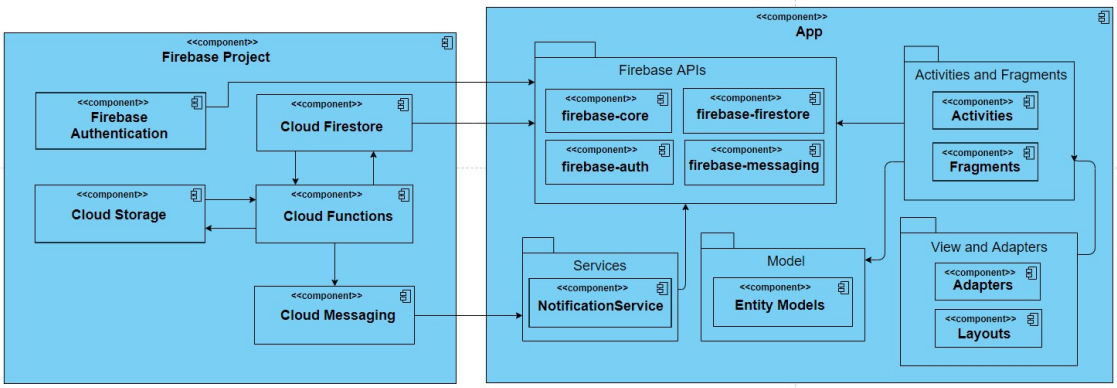


Figura 3.1: Diagrama de componentes

Como se ha comentado anteriormente, se requiere de un servidor para la gestión de los usuarios y para almacenar los datos proporcionados por estos. Este servidor nos

lo proporciona Firebase de manera gratuita con algunas limitaciones en número de lecturas y escrituras en la base de datos pero que son suficientes para la cantidad de datos manejados por la aplicación y que, además, se podría ampliar con un plan de pago en el futuro si la Dirección Técnica de la FAT lo considerase oportuno.

Es necesario tener una cuenta de Google para utilizar Firebase, se ha creado una para el desarrollo de este TFG que se otorga a la Dirección Técnica de FAT para poder obtener los datos analíticos de la aplicación que directamente nos proporciona la plataforma como por ejemplo el número de usuarios activos.

Algunas de las funcionalidades que se han desarrollado para la aplicación residen en la plataforma y se activan con eventos generados desde la aplicación al modificar la base de datos de Firebase.

Desde la aplicación, con la creación de torneos y formularios, se alimenta la base de datos no relacional que reside en Firebase utilizando la API Cloud Firestore. Una vez que los datos están en la plataforma los usuarios pueden acceder a ellos utilizando la aplicación y haciendo uso de la misma API. Adicionalmente, para hacer el registro de usuarios, inicio y cierre de sesión, se utiliza la API Firebase Authentication, y para recibir las notificaciones en la aplicación se hace uso de la API Cloud Messaging.

Para poder utilizar Firebase, es necesario crear un proyecto en la plataforma y realizar una configuración inicial en la aplicación para poder comunicarnos con la plataforma. Todo el desarrollo relativo a la configuración inicial de Firebase y la comunicación entre la aplicación y Firebase se ha realizado siguiendo la documentación de la plataforma.

La seguridad del sistema es proporcionada por Google Cloud Platform en la que confían multitud de empresas ya que tienen una de las mejores medidas de seguridad de la industria. Los datos se almacenan en esta plataforma y solo se puede acceder al proyecto con los correos electrónicos autorizados. La aplicación no almacena ningún dato personal en el dispositivo móvil. Los datos que se solicitan son el nombre completo y el correo electrónico. El correo electrónico únicamente es utilizado para iniciar sesión como nombre de usuario de la cuenta y el nombre completo se utiliza para firmar las respuestas enviadas.

3.2 Visión y arquitectura de la aplicación

Tennis Match Assessment and Counselling es la aplicación que permite a los organizadores de torneos de tenis crear formularios para evaluar distintos aspectos de la competición y a los jugadores de estos torneos contestar las preguntas propuestas en sus dispositivos móviles al finalizar los partidos. Se hace uso de las APIs de Firebase para gestionar los usuarios, almacenar los datos generados y recibir notificaciones.

La aplicación se ha desarrollado siguiendo el patrón de arquitectura MVC, separando los elementos que conforman la aplicación de tal manera que, el **controlador** recibe los eventos generados por los usuarios desde la **vista** o por las funciones de Cloud Functions (actividades, fragmentos y servicios), para así enviar la acción correspondiente al **modelo** que actualizará los datos en la **vista** (layouts de Android representados en ficheros XLM).

El modelo representa los datos de la aplicación almacenados en la base de datos no relacional de Firebase. Los datos se identifican de forma única por su ubicación dentro de la base de datos. Para gestionar los accesos a la base de datos desde la aplicación, se utiliza la API de Cloud Firestore que permite instanciar un objeto de la clase FirebaseFirestore con el que obtener las referencias de los datos guardados en la base de datos de Firebase, una vez obtenidas, se utilizan llamadas asíncronas para acceder a los datos.

El modelo se representa en el diagrama de clases de la aplicación en la Figura 3.2

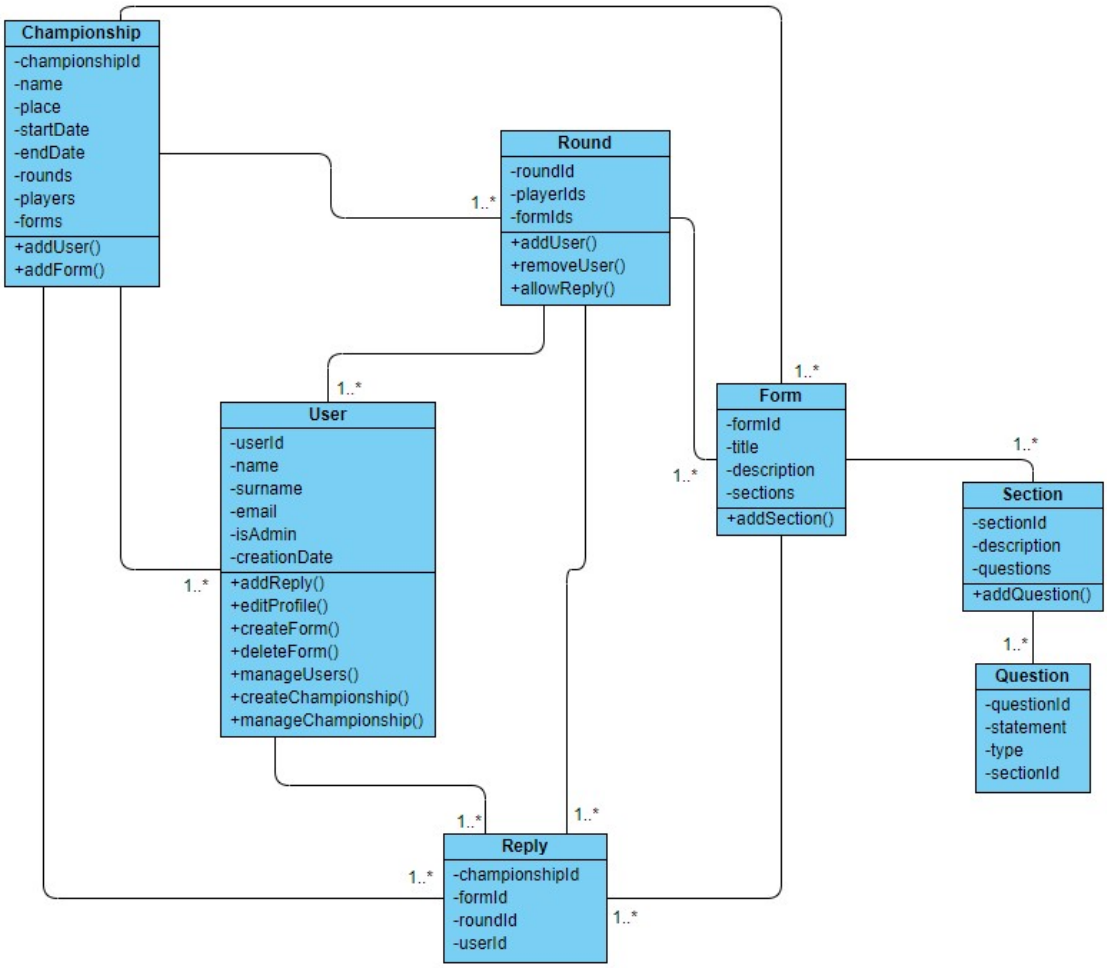


Figura 3.2: Diagrama de Clases

La comunicación con Firebase se realiza mediante el uso de sus APIs, para poder utilizarlas se necesita registrar la aplicación con el proyecto de Firebase. Para ello, desde la plataforma se solicita el identificador de la aplicación que se puede encontrar en el fichero **build.gradle** de la aplicación.

Una vez la aplicación es registrada, se descarga el archivo de configuración que proporciona la plataforma para Android (**google-services.json**). Este fichero se mueve al directorio *root* de la aplicación, es un JSON con información relativa al proyecto de Firebase, como la URL que proporciona Firebase para el proyecto o la URL del repositorio de almacenamiento de Cloud Storage.

Para utilizar el complemento de los servicios de Google para Gradle es necesario añadir, en el fichero build.gradle del proyecto, el repositorio Maven de Google y el

classpath de las dependencias. Además, en el fichero build.gradle de la aplicación se necesita indicar que se va a utilizar el plugin de los servicios de Google. Por último, se agregan en el fichero build.gradle de la aplicación las dependencias de los SDK de Firebase que son utilizados por la aplicación.

Se ha desarrollado una actividad que administra el registro de los usuarios y otra para iniciar sesión. Estas actividades hacen uso de la API Firebase Authentication que instanciando un objeto de la clase FirebaseAuth permite registrar usuarios en el proyecto de Firebase e iniciar sesión con los datos introducidos por los usuarios, además de otras funcionalidades como cambio de contraseña, eliminación de cuenta o cierre de sesión.

La actividad principal se implementa con un diseño de menú lateral utilizando el *Navigation Drawer* de Android. Esta actividad administra los distintos fragmentos que controlan cada una de las pantallas de la aplicación. Algunos fragmentos muestran en sus layouts listas de torneos, rondas, formularios o usuarios. Para poder visualizar estas listas se han desarrollado varios adaptadores extendiendo de la clase ArrayAdapter de Android para los modelos *Championships, Rounds, Forms y Users*, que adaptan la visualización de los datos obtenidos de Firebase.

Para poder recibir las notificaciones utilizando Cloud Messaging es obligatorio añadir un servicio en el fichero AndroidManifest.xml de la aplicación, este servicio extiende de FirebaseMessagingService, clase que es proporcionada por la API de Cloud Messaging.

Cuando la aplicación se inicia por primera vez, el SDK de Cloud Messaging genera un token de registro para la instancia de la aplicación. Cuando un usuario introduce sus credenciales e inicia sesión, el token se almacena en la base de datos para poder enviar las notificaciones al dispositivo correcto. Si el usuario cierra sesión, el token se elimina de la base de datos y este usuario no recibirá notificaciones hasta que no inicie sesión en el mismo u otro dispositivo y se vuelva a almacenar el token que utiliza en la base de datos. Como el token que se genera puede cambiar, por ejemplo si un usuario cambia de dispositivo, es necesario sobrescribir la función onNewToken en el servicio de notificaciones para poder actualizar el nuevo valor del token en base de datos cuando este cambia.

3.3 Firebase como servidor para el desarrollo

En este apartado se detalla la arquitectura de la base de datos del proyecto creado en la plataforma de desarrollo Firebase, los componentes de autenticación, almacenamiento en la nube y notificación de mensajes de Firebase que utiliza la aplicación así como las funciones de backend desarrolladas en Firebase.

3.3.1 Componentes de Firebase

Firebase posee multitud de funcionalidades, además del componente que nos proporciona la base de datos no relacional, la aplicación hace uso de cuatro funcionalidades más:

- **Firebase Authentication:** la aplicación necesita identificar a los usuarios, utilizando el componente de Firebase, se almacenan los datos de los usuarios en la nube de forma segura y permite que los usuarios puedan cambiar de dispositivo sin perder su cuenta de usuario.

  La autenticación de usuarios que se ha elegido para el desarrollo de la aplicación es mediante correo electrónico y contraseña. Para utilizar este método se tiene que activar en la plataforma de desarrollo e importar la dependencia en el archivo build.gradle de la aplicación.

- **Cloud Firestore:** es el componente que nos proporciona la base de datos explicada en el apartado anterior. Para poder realizar las consultas que muestran si existen torneos en curso desde la aplicación es necesario crear un índice compuesto de dos campos del torneo. Si no hacemos esto la plataforma no nos permite realizar este tipo de consultas compuestas.

- **Cloud Storage:** el componente que proporciona almacenamiento en la nube para guardar las respuestas de los jugadores en ficheros Excel es Cloud Storage. No se comunica directamente con la aplicación ya que es utilizado por las funciones de Cloud Functions almacenadas en la plataforma.

Se ha creado una carpeta dentro del sistema de almacenamiento en la nube para guardar las respuestas de los jugadores separando las respuestas de cada torneo en carpetas con el nombre del torneo. Siguiendo el siguiente esquema:

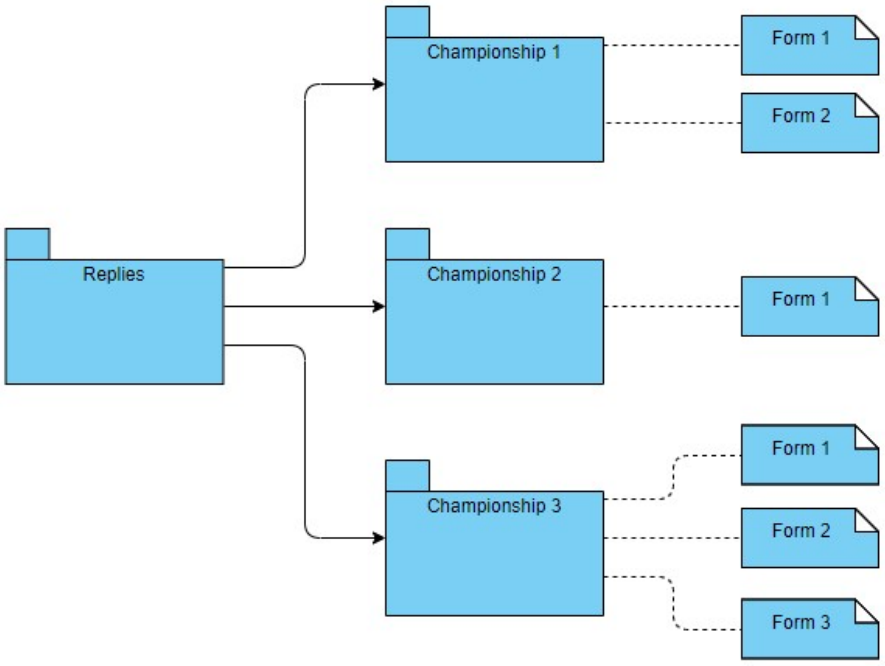


Figura 3.3: Esquema repositorio almacenamiento de las respuestas

Adicionalmente existe otra carpeta para que la Dirección Técnica de la FAT pueda crear formularios importando un Excel con el siguiente formato:

	A	B	C	D	E
1	**nombre**	**descripcion**	**seccion**	**enunciado**	**tipo**
2	nombre del formulario	descripcion del formulario	descripcion seccion 1	pregunta 1 seccion 1	Texto
3				pregunta 2 seccion 1	Escala
4			descripcion seccion 2	pregunta 1 seccion 2	Escala
5				pregunta 2 seccion 2	Si/No
6				pregunta 3 seccion 2	Si/No
7			descripcion seccion 3	pregunta 1 seccion 3	Texto
8				pregunta 2 seccion 3	Escala
9				pregunta 3 seccion 3	Escala
10				pregunta 4 seccion 3	Si/No

Figura 3.4: Formato formulario en Excel

La primera columna representa el nombre del formulario y la segunda la descripción del mismo. En estas columnas se pueden combinar las celdas para

que las filas engloben todo el formulario pero no es necesario. La siguiente columna es para crear las secciones con la descripción de la sección. Las filas de esta columna tienen que combinarse de tal manera que agrupen todas las respuestas de la sección como se puede ver en la Figura 3.4. Las siguientes dos columnas son los enunciados de las preguntas y su tipo, las filas de estas columnas no pueden combinarse y cada una representa una pregunta del formulario. Las funciones desarrolladas en Cloud Functions se encargan de leer este fichero y con los datos crear un formulario en la colección *forms* como se explica en la sección "Funciones Cloud Functions".

- **Cloud Messaging:** el componente con el que se puede enviar a cada jugador las notificaciones indicándole que tiene un nuevo formulario para responder. Para utilizar este módulo se tiene que activar Google Analytics en el proyecto de Firebase. Desde la plataforma se podrían enviar notificaciones en masa a todos los usuarios. Se utiliza desde las funciones de Cloud Functions como se explica en la sección "Funciones Cloud Functions".

- **Cloud Functions:** con este componente se pueden crear funciones para la lógica del backend del sistema. El código se almacena en la nube y se ejecuta en un entorno administrado por Google. No es necesario administrar ni escalar ningún servidor, de este proceso se encarga la plataforma. Las funciones responden a eventos generados por algunos de los otros componentes de Firebase. Para desarrollar es necesario instalar Node.js y npm con el fin de descargar Firebase CLI que nos permite desplegar las funciones desarrolladas en la plataforma.

3.3.2 Arquitectura de la base de datos

Firebase ofrece dos bases de datos en la nube: Cloud Firestore y Realtime Database. Las dos son bases de datos NoSQL, Cloud Firestore almacena los datos en documentos y Realtime Database lo hace como un gran árbol JSON. Para el desarrollo de la aplicación se ha optado por Cloud Firestore ya que además de ser más reciente, escala automáticamente y permite operaciones avanzadas de escritura y transacción.

El diagrama de clases utilizado en la aplicación difiere con la estructura de la base de datos alojada en la plataforma de desarrollo Firebase ya que la base de datos Cloud Firestore es NoSQL orientada a documentos. Las diferencias con una base de datos SQL son que no hay tablas ni filas. Para almacenar los datos se utilizan *documentos* [13] que se organizan en *colecciones*. Los documentos contienen conjuntos de pares clave-valor y pueden tener *subcolecciones* y objetos anidados, incluyendo campos primitivos o tipos de objetos complejos como mapas o listas. Los nombres de los documentos de una colección son únicos y se utilizan como identificadores. Estos identificadores pueden ser generados automáticamente por Firebase pero en algunos casos son gestionados por la aplicación. No hay límite de documentos que pueden ser almacenados en una colección. Firebase está diseñado para escalar con grandes cantidades de datos.

Las colecciones únicamente alojan estos documentos, para desarrollar la aplicación, se han utilizado tres colecciones principales (*championships*, *forms*, y *users*) que alojan los datos generados por la aplicación y una colección más que se utiliza para disparar el evento que envía la notificación a los jugadores. Los documentos que se agrupan en estas colecciones poseen subcolecciones para almacenar el resto de datos y hacer que los documentos sean ligeros.

Todas las colecciones y documentos de la base de datos están identificados de forma única por su ubicación. Para comunicarnos con Cloud Firestore desde la aplicación se utilizan *referencias* [13]. Las referencias son los objetos proporcionados por las librerías de Cloud Firestore que apuntan a una ubicación en la base de datos. Se obtienen del objeto FirebaseFirestore indicando la ruta del documento o colección.

La base de datos se ha diseñado para satisfacer los requisitos de la aplicación teniendo en cuenta la escalabilidad de los datos generados. Las colecciones principales se definen a continuación:

- **Users**: En esta colección se guardan los documentos con la información que representa a un usuario en la aplicación.

  Los documentos son creados con el identificador que genera Firebase Authentication al registrarse un usuario en la aplicación. Estos documentos que

identifican a los usuarios poseen una subcolección *replies* para almacenar todas las respuestas que dan los jugadores.

Los documentos de esta subcolección guardan toda la información relativa a la respuesta como lo son los identificadores de los documentos que representan el torneo, el formulario contestado y la ronda en la que se está respondiendo el formulario. Además poseen otra subcolección *questions* cuyos documentos almacenan las preguntas del formulario contestado y la respuesta dada por el jugador.

Cada vez que se crea un documento en la subcolección *replies* de cualquier documento de la colección *users*, se activa el evento que ejecuta una función de Cloud Functions para almacenar la respuesta del jugador en el Excel correspondiente en Cloud Storage.

La colección *users* y las relaciones de los documentos de la subcolección *replies* con las colecciones de la base de datos se representan en la Figura 3.5.

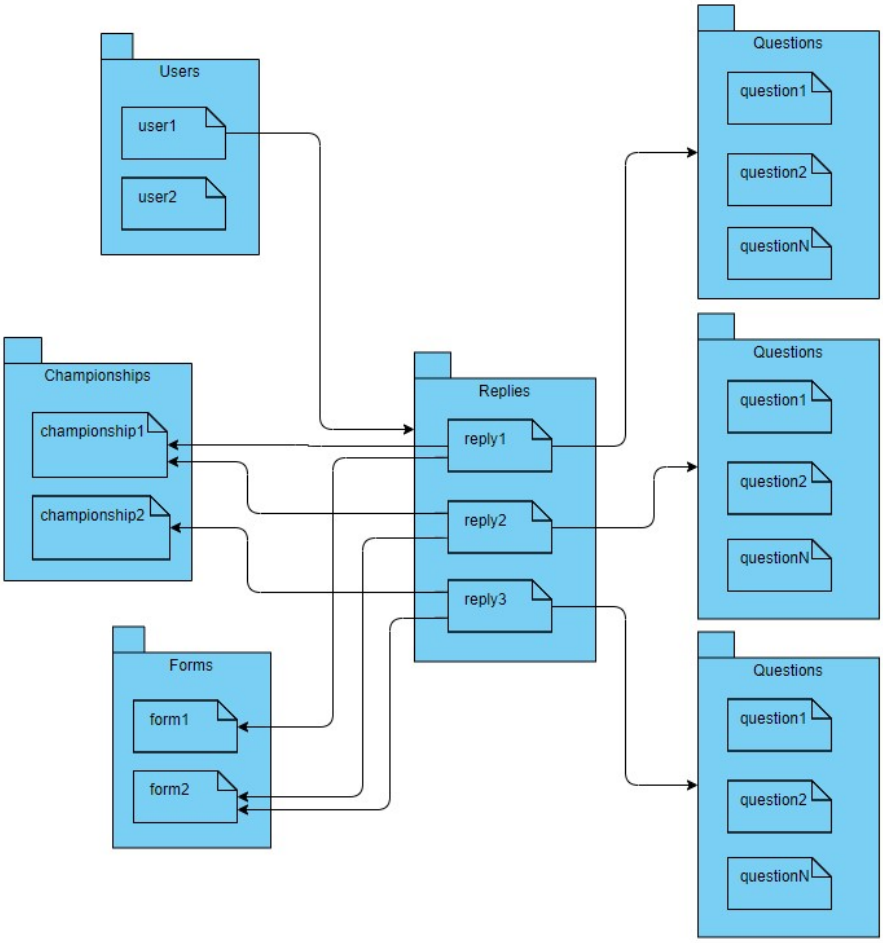


Figura 3.5: Modelo de la colección users y sus relaciones

- **Forms**: Esta colección almacena los documentos con la información de los formularios generados tanto por la aplicación como por la función desarrollada en Cloud Functions.

  Los documentos de esta colección poseen dos subcolecciones (*sections* y *questions*) con documentos para almacenar las secciones y las preguntas del formulario. Los documentos de estas subcolecciones se crean con identificadores propios utilizados como índices para poder representarlas posteriormente en el orden adecuado.

  Los documentos de la subcolección *questions* almacenan el identificador de la sección a la que pertenecen. En esta ocasión no se ha utilizado una subcolección

dentro de los documentos de la colección *sections* para facilitar la representación de las preguntas dentro de sus secciones en la aplicación.

La colección *forms* y las relaciones entre los documentos de las subcolecciones *sections* y *questions* se representan en la Figura 3.6.

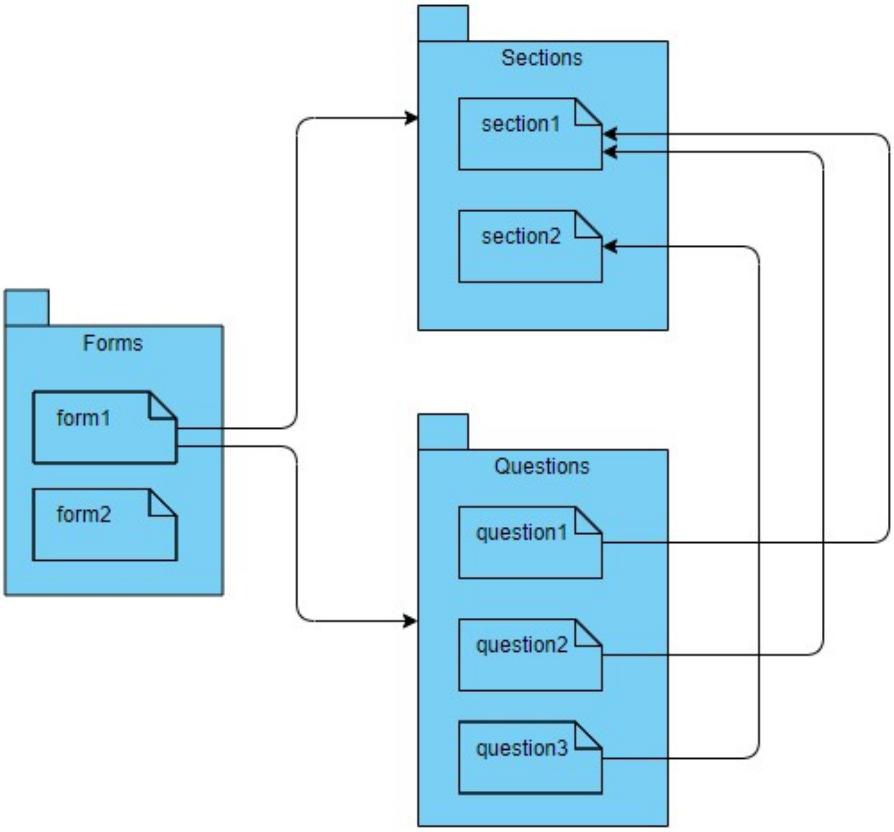


Figura 3.6: Modelo de la colección forms y sus relaciones

- **Championships**: Es la colección que almacena los documentos de los torneos creados por los administradores de la aplicación.

  Estos documentos guardan la información de qué formularios y qué jugadores forman parte del torneo guardando los identificadores de los documentos de las colecciones *forms* y *users* en dos listas.

Además, estos documentos poseen una subcolección llamada *rounds* cuyos documentos están creados con identificadores que representan el número de ronda. Estos documentos almacenan información en forma de mapa que sirve para gestionar qué jugadores continúan en cada ronda y para limitar a cada jugador una única respuesta por formulario y ronda.

La colección *championships* y sus relaciones se muestran en la Figura 3.7.

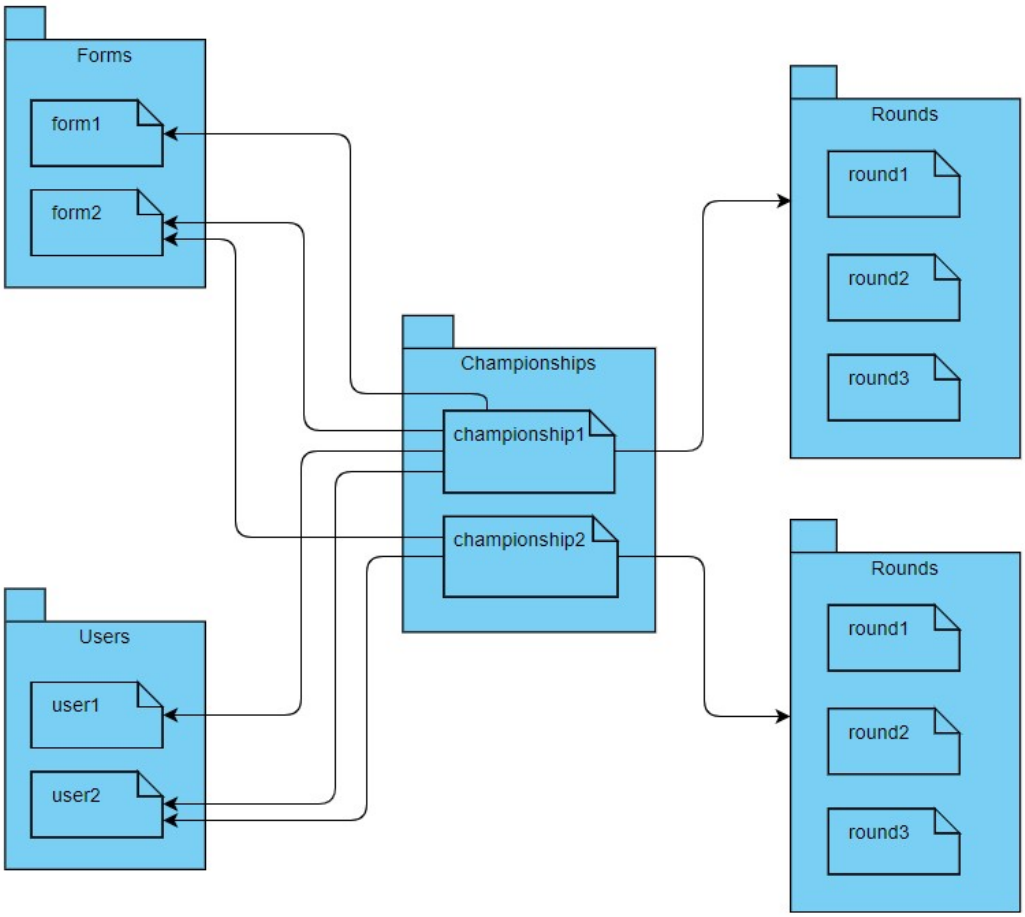


Figura 3.7: Modelo de la colección championships y sus relaciones

3.3.3 Funciones de Cloud Functions

Para satisfacer algunos de los requisitos de la aplicación es necesario utilizar lógica de backend, esta lógica nos la proporcionan las funciones de Cloud Functions.

Se han desarrollado 3 funciones en Javascript que potencian la aplicación desde la plataforma de desarrollo. Estas funciones se activan con distintos eventos que ocurren en la plataforma como la creación de datos en la base de datos o la subida de ficheros al almacenamiento en la nube de Firebase. Para la creación y lectura de los archivos Excel se han utilizado las librerías SheetJS y ExcelJS. La concurrencia de estas funciones es gestionada por la plataforma de desarrollo.

Las funciones desarrolladas se describen a continuación:

- **createForm**: es la función que permite crear formularios desde la plataforma. Se ha creado una carpeta en Cloud Storage llamada *formularios* para que la FAT pueda crear formularios en la base de datos subiendo un archivo Excel a esta carpeta con el formato definido en la sección anterior.

  La función se activa cuando sucede este evento, esta lee las dos primeras columnas ya que estas contienen la información del formulario (nombre y descripción) y con esta información crea el documento en la colección *forms* alojada en Cloud Firestore. Ejecutando las promesas que nos proporcionan las librerías de Firebase para Javascript podemos esperar a que la función acabe de crear el documento para continuar leyendo del archivo Excel las secciones y las preguntas de las otras columnas y así crear las subcolecciones del formulario. El esquema de la función se representa en la Figura 3.8.

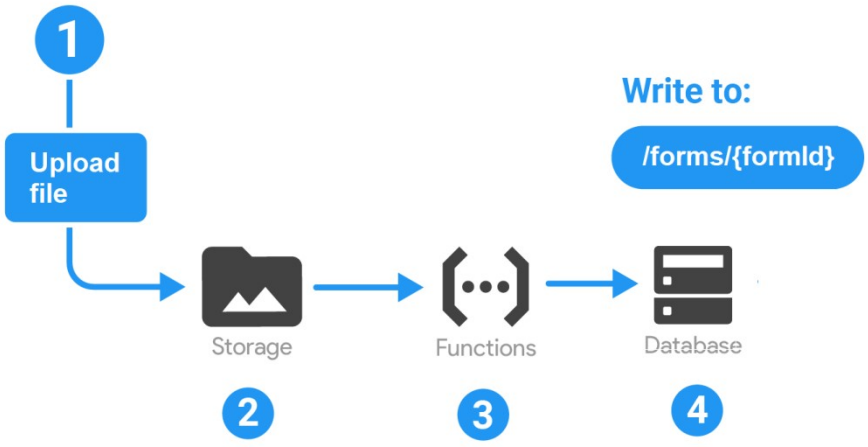


Figura 3.8: Esquema del método createForm.

- **saveReply**: es la funcion que recoge las respuestas de los jugadores y las guarda en archivos Excel que son almacenados en el sistema de almacenamiento de archivos de la plataforma. Cuando un jugador envia las respuestas del formulario de un torneo, estas se guardan en la subcolección *replies* de los documentos de la coleccion *users*. Esta acción dispara el evento que ejecuta esta función.

  Se lee el documento de la subcolección *replies* que como se explica en la sección anterior contiene las respuestas de las preguntas del formulario además de otra información como el nombre del torneo y el nombre del formulario. El nombre del torneo nos sirve para crear una carpeta dentro del repositorio con este nombre para que las respuestas a los formularios estén divididas de esta manera. El nombre del formulario contestado es el nombre del fichero Excel que se sube a la carpeta del torneo. Las respuestas del formulario se iteran para colocar las respuestas en las columnas correspondientes. Para poder identificar dentro del archivo Excel que respuesta se corresponde a cada jugador, la primera columna se corresponde con el nombre del jugador que contesta el formulario y la segunda es la ronda del torneo. Las columnas siguientes son las respuestas a cada una de las preguntas. El esquema de la función se representa en la Figura 3.9.

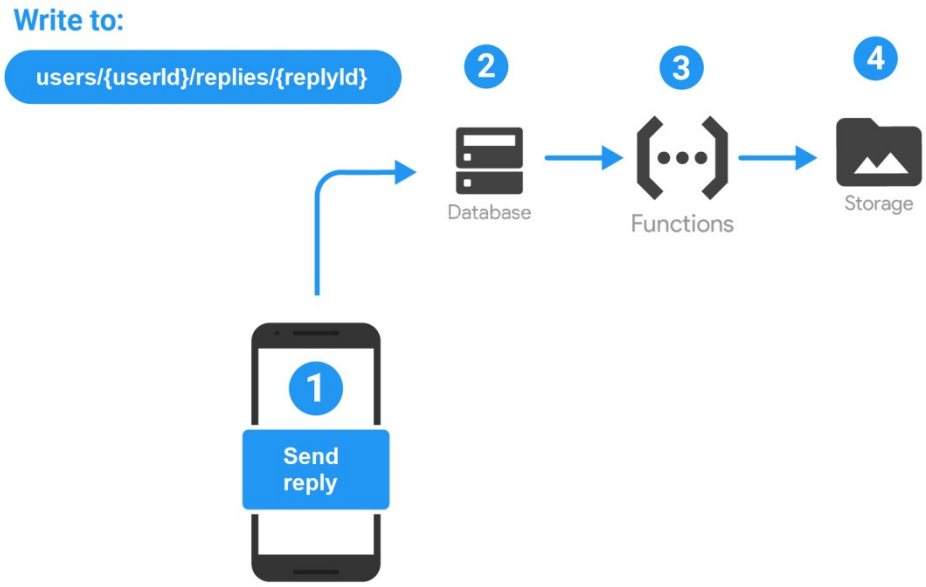


Figura 3.9: Esquema del método saveReply.

- **sendNotification**: es la función que envía notificaciones a los jugadores indicando que pueden contestar un formulario. Cuando el administrador está gestionando las rondas del torneo e indica el ganador de un partido, desde la aplicación se crean dos documentos, uno por cada jugador, en una colección llamada *notifications*. Estos documentos contienen el identificador del jugador, el nombre del torneo y el número de ronda para confeccionar el mensaje de la notificación. El evento que dispara esta función es la escritura de cualquier documento en la colección *notifications*.

  La función lee los datos del documento, con el identificador de usuario consulta en la colección *users* el token de registro que obtuvo cuando inició sesión en la aplicación. A través del token de usuario y haciendo uso de la API de Firebase Messaging esta función envía las notificaciones a los dispositivos correspondientes. El esquema de la función se representa en la Figura 3.10.

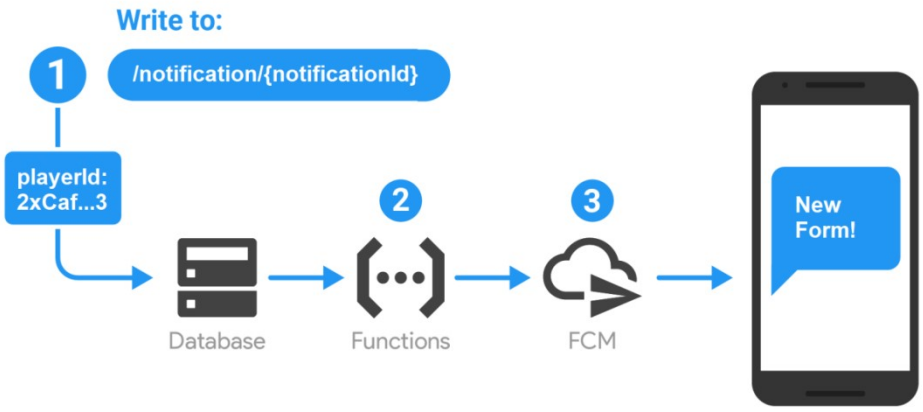


Figura 3.10: Esquema del método sendNotification.

3.4 Funcionamiento de la aplicación

En esta sección se explican las diferentes funcionalidades de la aplicación y las pantallas correspondientes con las que interactúan los distintos usuarios. Para el registro de usuarios, inicio y cierre de sesión se hace uso de la API Firebase Authentication así como para la lectura y escritura en la base de datos se hace uso de los métodos proporcionados por la API de Cloud Firestore.

En el Anexo E se detallan las actividades y fragmentos de la aplicación mostrando las funcionalidades de sus métodos y qué tipo de usuario es capaz de acceder a estos métodos.

3.4.1 Registro e Inicio de sesión

Al abrir por primera vez la aplicación se visualiza la pantalla de inicio de sesión (Figura 3.11), a través de la actividad *LoginActivity.java*. Esta consta de dos campos para introducir los credenciales de usuario si ya disponemos de cuenta, un botón para iniciar la sesión una vez introducidos los datos y un mensaje indicando al usuario que si no tiene cuenta se puede registrar.

Los datos solicitados por esta pantalla son obligatorios, si no se introducen, y se pulsa el botón de inicio, se alerta al usuario con un mensaje. Si el usuario introduce sus credenciales y pulsa el botón de inicio de sesión, se comprueban que los datos introducidos son correctos con el método *userLogin.*

. Si los datos son correctos el usuario accede a la aplicación haciendo uso del método *signInWithEmailAndPassword* que nos proporciona la API de Firebase Authentication, cuando se completa el inicio de sesión se utiliza el método *updateToken* que haciendo uso de la API de Cloud Firestore actualiza el token de usuario en la base de datos que se utilizará para poder enviarle notificaciones. Si los datos introducidos son erróneos, la aplicación avisa al usuario a través de una notificación en la pantalla (*Toast*).

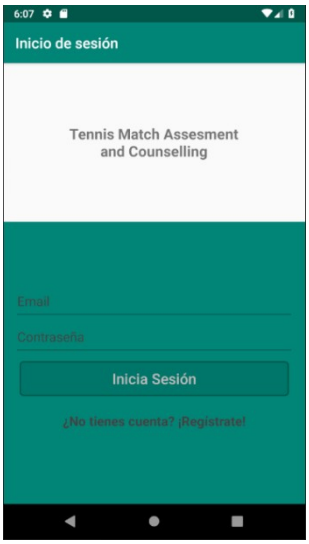


Figura 3.11: Pantalla de Inicio de sesión

Al pulsar sobre el mensaje de registro de la pantalla de inicio de sesión nos lleva a la pantalla de registro (Figura 3.12), a través de la actividad *SignUpActivity.java* donde aparecen los campos requeridos para poder crear una cuenta, el botón de registro que activa el método *registerUser* y un mensaje para volver a la pantalla de inicio de sesión si el usuario ya dispone de cuenta.

El método *registerUser* valida los campos para que los datos introducidos cumplan una serie de requisitos. El nombre y el apellido no pueden estar vacios, el campo email tiene que ser una dirección de correo electrónico, la contraseña tiene que tener al menos 6 caracteres y es obligatorio marcar el checkbox aceptando la política de privacidad. Si alguno de estos campos no es correcto el campo solicita el foco y muestra un error con la advertencia correspondiente. Una vez los datos introducidos son validados se llama al método *createUserWithEmailAndPassword* proporcionada por la API de Firebase Authentication y después se añade el registro del usuario en la colección *users* de la base de datos utilizando la API de Cloud Firestore.

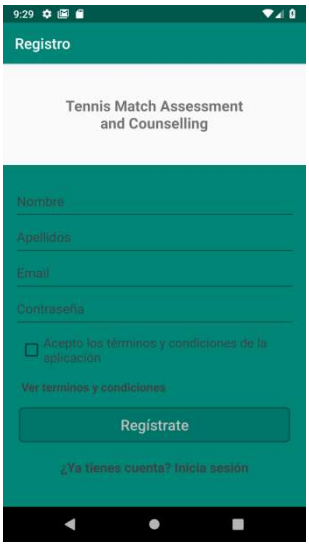


Figura 3.12: Pantalla de Registro

Una vez finaliza el proceso completo de registro, incluyendo la adición de los datos de usuario en la base de datos, se informa al usuario que su cuenta ha sido creada mediante un *Toast* y automáticamente la aplicación vuelve a la pantalla de inicio de sesión para que el usuario acceda a su cuenta.

La política de privacidad se puede consultar pulsando sobre el texto "Ver política de privacidad" que inicia la actividad *PolicyPrivacyActivity.java* para mostrar la pantalla con esta información (Figura 3.13).

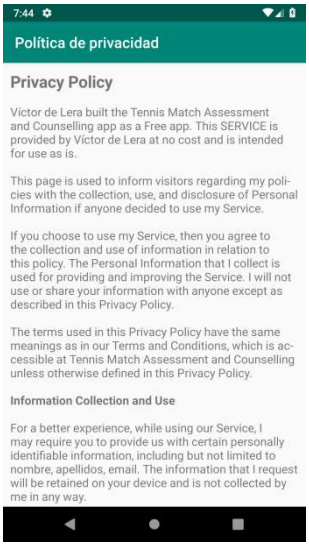


Figura 3.13: Política de privacidad

3.4.2 Menú Administradores y Menú Jugadores

Para gestionar las distintas funcionalidades de la aplicación se ha utilizado un *Navigation Drawer* con un menú lateral de navegación en la actividad principal *MainActivity.java* implementando la clase *NavigationView* de Android. Las opciones *Inicio, Perfil, Desconectar y Política de privacidad* del menú son comunes para los dos tipos de usuario.

Los administradores también tienen las opciones de *Crear Formulario, Administrar Formularios, Crear Torneo, Administrar Torneos y Administrar Usuarios* (Figura 3.14). Para mostrar estas opciones, al iniciar la aplicación se realiza una consulta a la base de datos con el método *showAdminExtraButtons* para comprobar si el usuario que ha iniciado sesión es administrador o es un jugador.

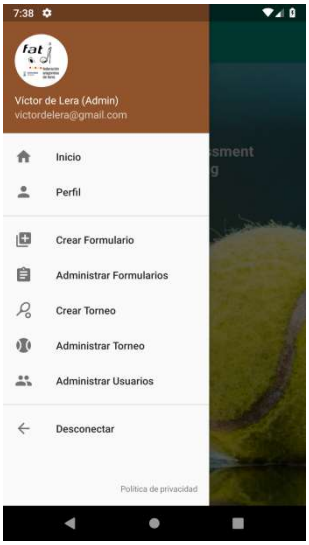


Figura 3.14: Menú Administrador

El menú de los jugadores (Figura 3.15) contiene, además de las opciones comunes, la opción *Contestar Formulario* que solo se muestra si el jugador que ha iniciado sesión se encuentra en algún torneo en curso. Para mostrar este botón se hace una consulta a la base de datos con el método *thereAreFormsToReply* que utilizando la API de Cloud Firestore filtra los resultados con los métodos *whereGreaterThanOrEqualTo* para comparar la fecha de finalización del torneo con la fecha actual y *whereArrayContains* introduciendo el identificador del usuario para

obtener solo los torneos donde el jugador participa. Una vez obtenidos los resultados se filtran para no incluir torneos que todavía no han empezado.

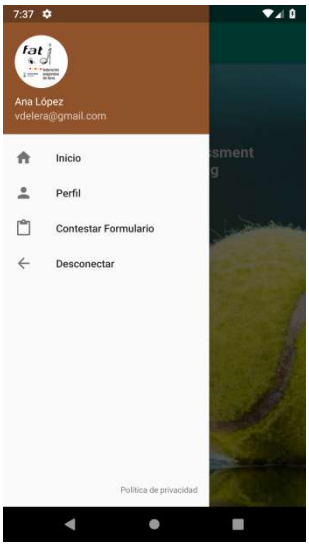


Figura 3.15: Menú Jugador.

3.4.3 Crear Formulario

Esta opción sirve para que los administradores de la aplicación puedan crear formularios desde la aplicación. El fragmento *CreateFormFragment.java* infla este layout que contiene el titulo del formulario, su descripción, un botón para añadir secciones al formulario y otro botón para crear el formulario en la base de datos (Figura 3.16). El botón *Añadir Sección* llama al método *addSectionToLayout* que infla el layout de la sección (section.xml) en un *LinearLayout* definido en el layout del formulario (add_form.xml). Se pueden añadir tantas secciones como requiera el formulario. El layout de la sección contiene un campo para definir el título de la sección y un botón para añadir preguntas a esa sección (Figura 3.17).

Figura 3.16: Crear Formulario

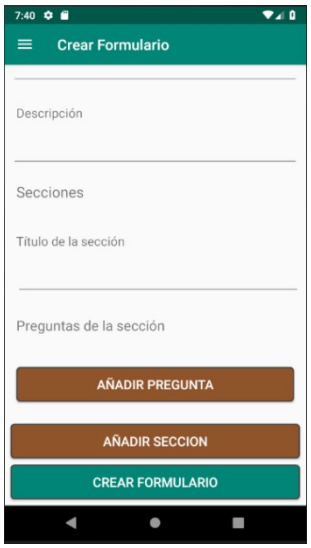


Figura 3.17: Formulario con sección.

Este botón llama al método *addQuestionToSection* que, con el mismo procedimiento que al añadir secciones en el formulario, infla el layout de la pregunta (question_item.xml) en el *LinearLayout* que se encuentra definido en el layout de la sección (Figura 3.18). Las preguntas tienen un campo para definir el enunciado y otro para declarar el tipo de pregunta mediante un *Spinner* de Android en el layout (Figura 3.19).

Figura 3.18: Pregunta de la sección

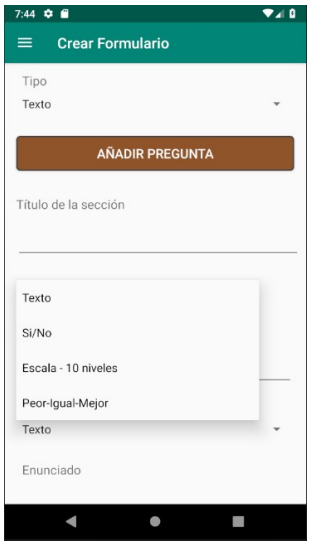


Figura 3.19: Tipos de preguntas

Finalmente cuando el usuario pulsa el botón *Crear Formulario* se llama al método *addFormToFirebase* que recorre todas las instancias de las secciones y en cada una de ellas itera sobre sus preguntas para ir conformando el modelo del formulario que se escribe en la base de datos de Cloud Firestore mediante la clase *WriteBatch,* que nos

permite crear el documento del formulario con las subcolecciones de las secciones y las preguntas ejecutando todas las operaciones de escritura como una unidad atómica.

3.4.4 Administrar Formularios

Los administradores pueden ver y eliminar los formularios que han sido creados. Los formularios se representan en una lista mediante el elemento *ListView* en el layout (form_list.xml).

Para acceder a los formularios, el fragmento *ListManageFormsFragment.java* llama al método *loadFormList* cuando este se inicia. Este método realiza una consulta a la base de datos para obtener todos los formularios, haciendo uso del método *orderBy* se ordenan los resultados por nombre para poder localizar los formularios de manera rápida.

Los resultados obtenidos con la consulta se añaden a una lista que se utiliza para instanciar el adaptador *FormAdapter.java* que nos permite visualizar los datos principales del formulario en los elementos de la lista (Figura 3.20).

Figura 3.20: Listado de formularios

Pulsando sobre cualquiera de los elementos de la lista, se invoca al fragmento *ManageFormFragment.java* que al iniciarse llama al método *loadSectionsData* para cargar las secciones del formulario consultando la base de datos con la API de Cloud

Firestore. Una vez obtenemos los resultados de las secciones se llama al método *loadQuestionsData* para cargar las preguntas del formulario. Cuando tenemos los datos de las preguntas, se llama al método *placeQuestionsInSection* que recorre las secciones para pintarlas en el layout inflando el layout de la seccion (section_content.xml) e iterando las preguntas de la seccion se infla el layout del tipo de pregunta correspondiente (Figura 3.21). Ademas existe un boton al final del formulario para eliminarlo si los administradores lo consideran oportuno (Figura 3.22).

Figura 3.21: Gestión formulario

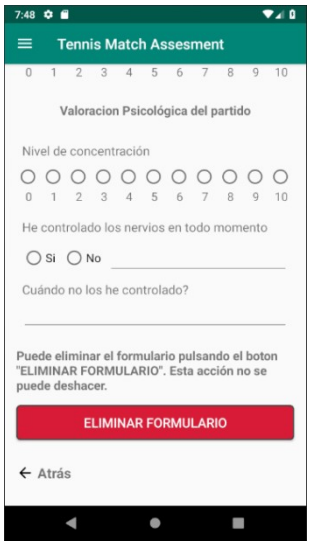


Figura 3.22: Elminar formulario

3.4.5 Crear Torneo

Los administradores de la aplicación son capaces de crear torneos con esta opción. El proceso consta de tres pasos. En el primero se describen los datos generales del torneo (Figura 3.23). Este proceso lo administra el fragmento *CreateChampionship.java* que se encarga de recoger los datos introducidos cuando el usuario pulsa el botón *Siguiente* y se los envía al siguiente fragmento *AddPlayersToChampionship.java*. Este fragmento muestra la lista de los jugadores utilizando el mismo procedimiento que al obtener los formularios en el apartado anterior (Figura 3.24).

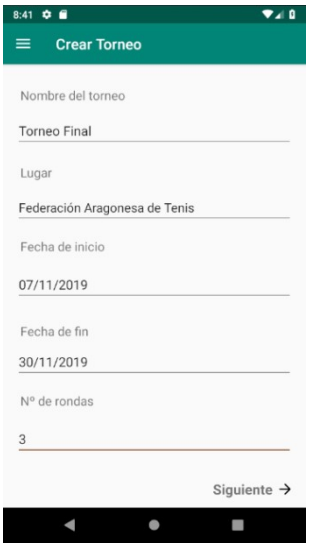


Figura 3.23: Crear Torneo

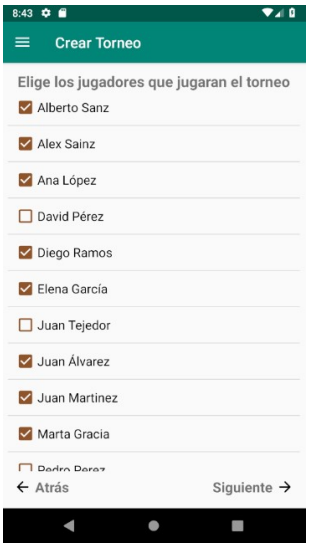


Figura 3.24: Agregar jugadores

El fragmento valida que el número de jugadores seleccionados sea correcto dependiendo del número de rondas de las que consta el torneo, dato que se introduce en el paso anterior. Cuando el administrador pulsa el botón *Siguiente* se invoca al fragmento *AddFormsToChampionship.java* para mostrar los formularios creados en el sistema y seleccionar los deseados por el usuario (Figura 3.25). Pulsando el botón *Crear Torneo* el torneo se guarda en la base de datos.

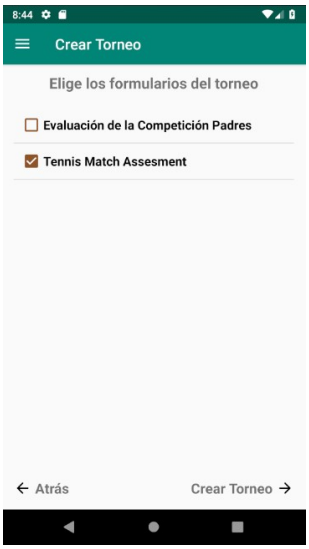


Figura 3.25: Agregar formularios

3.4.6 Administrar Torneos

Esta opción muestra una lista de torneos por orden de fecha de finalización. Se utiliza la misma lógica que para obtener la lista de formularios. El fragmento *ManageChampionshipsFragment.java* se encarga de recuperar los registros de la base de datos y mostrarlos en el *ListView* del layout para que el usuario elija el torneo que quiere gestionar (Figura 3.26). Al pulsar en los elementos de la lista se invoca al fragmento *ManageChampionshipRoundsFragment.java* que obtiene las rondas del torneo y las muestra de nuevo en una lista (Figura 3.27).

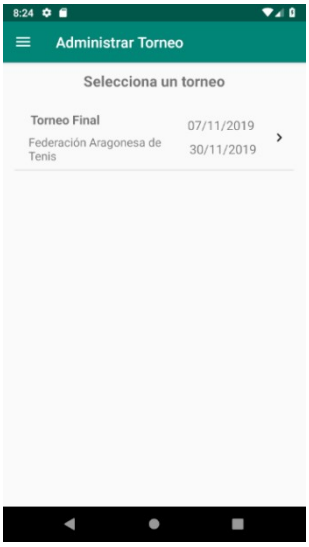


Figura 3.26: Seleccionar Torneo

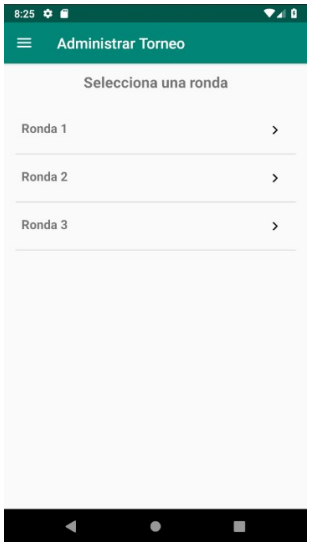


Figura 3.27: Seleccionar Ronda

En un torneo en el que todavía no se haya jugado ningún partido, todos los jugadores del torneo aparecerán en la ronda 1. Esta pantalla es gestionada por el fragmento *ManageChampionshipPlayersFragment.java*. Cuando dos jugadores finalizan su partido, el administrador debe seleccionarlos (Figura 3.28) y pulsar el botón *Siguiente* para designar al ganador (Figura 3.29). Esto invoca al fragmento *SelectWinnerFragment.java* que recibe los jugadores seleccionados y los muestra en otro *ListView*. El ganador pasará a la lista de la siguiente ronda y los dos desaparecerán de la ronda actual.

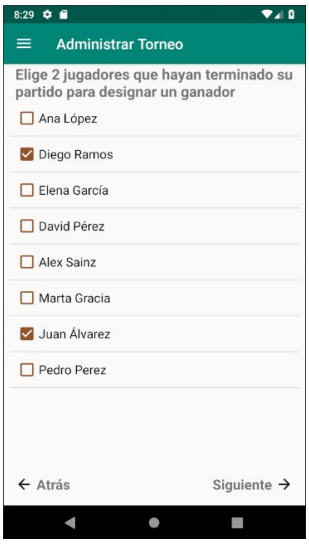


Figura 3.28: Elegir jugadores

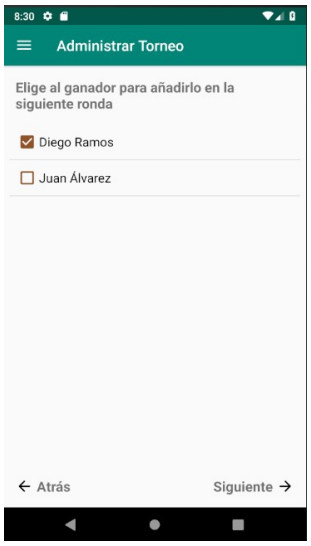


Figura 3.29: Designar ganador

Cuando el administrador pulsa el botón *Siguiente* se crea un documento por cada jugador en la colección notificaciones para activar el evento de la función *sendNotification* explicado en la sección 3.3.3 Funciones de Cloud Functions.

3.4.7 Administrar usuarios

Los administradores de la aplicación pueden hacer administrador a otro usuario para que le ayude con la gestión de los torneos y la creación de formularios. Para esto, el fragmento *ListManageUsersFragment.java* obtiene la lista de usuarios por orden alfabético para facilitar su búsqueda (Figura 3.30). Al pulsar en el usuario, se invoca al fragmento *ManageUserFragment.java* para visualizar sus datos y aparece un checkbox para convertir al usuario en administrador pulsando el botón *Guardar* (Figura 3.31).

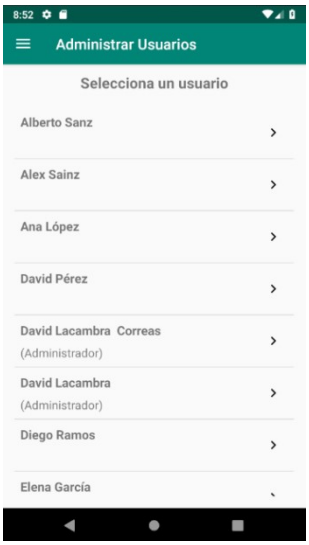


Figura 3.30: Seleccionar usuario

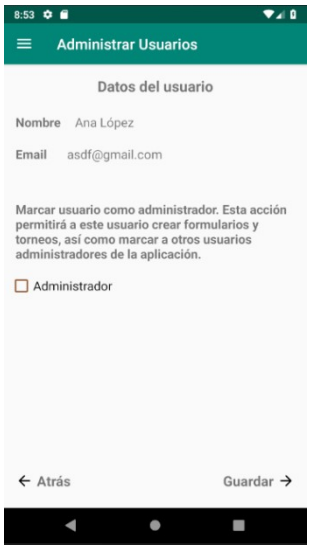


Figura 3.31: Administrar usuario

3.4.8 Notificaciones

Las notificaciones llegan al dispositivo del jugador indicando en el título de la notificación que tiene nuevos formularios para responder. En el cuerpo de la notificación se informa del nombre del torneo y la ronda a la que pertenece esa instancia del formulario (Figura 3.32).

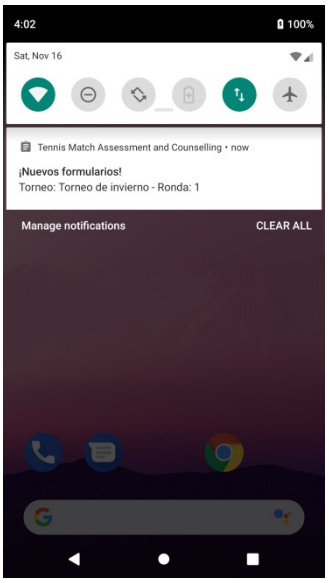


Figura 3.32: Notificación

Si el jugador pulsa sobre la notificación, la aplicación se abre y muestra la opción de “Contestar Formularios”.

3.4.9 Contestar Formularios

Está opción permite a los jugadores contestar los formularios de cada ronda de los torneos en curso. Para ello el fragmento *ListChampionshipFragment.java* obtiene la lista de los torneos que está disputando el jugador de la misma manera que en el apartado *3.4.6 Administrar Torneos* pero filtrando además por el identificador de usuario con el método *whereArrayContains* en el campo *playerIds* del documento del torneo (Figura 3.32). Al seleccionar el torneo, se invoca al fragmento *ListFormFragment.java* que, en la siguiente pantalla, muestra una lista con los formularios no contestados de ese torneo, indicando en cada uno de ellos la ronda a la que pertenece el cuestionario utilizando el adaptador *FormAdapter.java* (Figura 3.33).

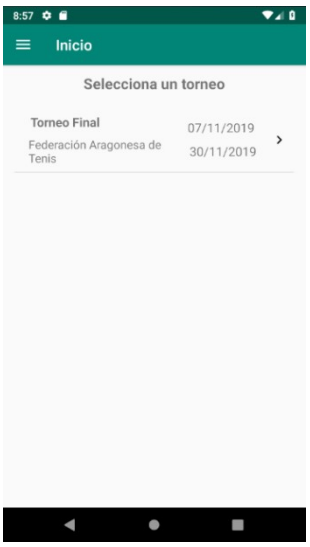


Figura 3.32: Listado Torneos

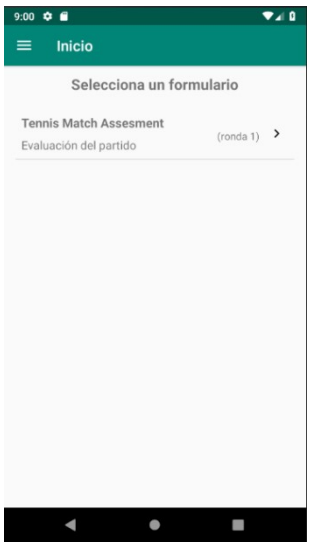


Figura 3.33: Elegir Formulario

Seleccionando el formulario que se quiere contestar, la aplicación invoca al fragmento ReplyFormFragment.java que obtiene y muestra las preguntas de este iterando los documentos de sus subcolecciones e inflando el layout correspondiente al tipo de pregunta iterada (Figura 3.34). El jugador puede contestar a las preguntas y finalmente enviarlas pulsando el botón *Enviar* almacenando las respuestas en la base de datos.

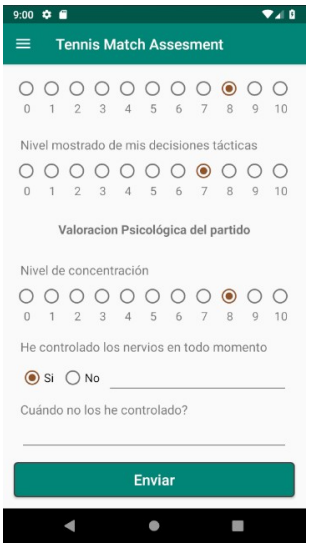


Figura 3.34: Formulario contestado

3.4.10 Obtener respuestas

Las respuestas se almacenan en archivos Excel en el repositorio de almacenamiento en la nube de la plataforma. Para descargar estos archivos es necesario iniciar sesión en la plataforma y acceder al módulo de Cloud Storage en el menú lateral (Figura 3.35).

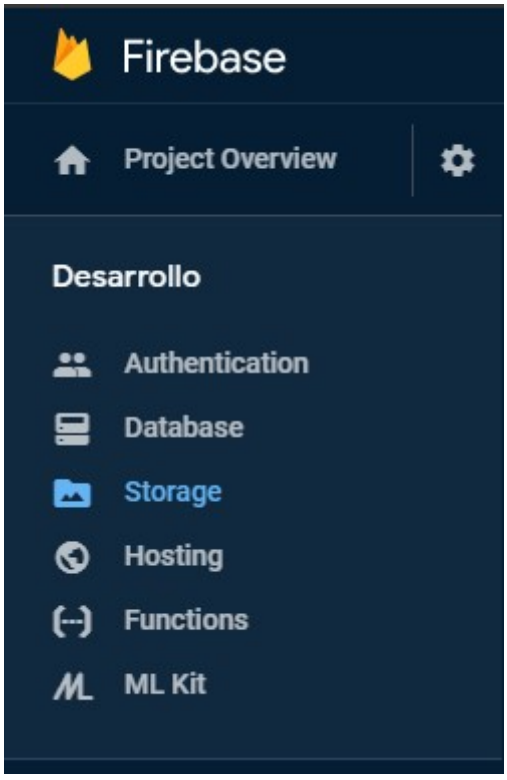


Figura 3.35: Menú lateral Firebase

En este módulo se encuentran las dos carpetas comentadas en la sección 3.3.1: formularios y respuestas. En la carpeta respuestas se ubican las carpetas con los nombres de los torneos que ya tienen respuestas (Figura 3.36).

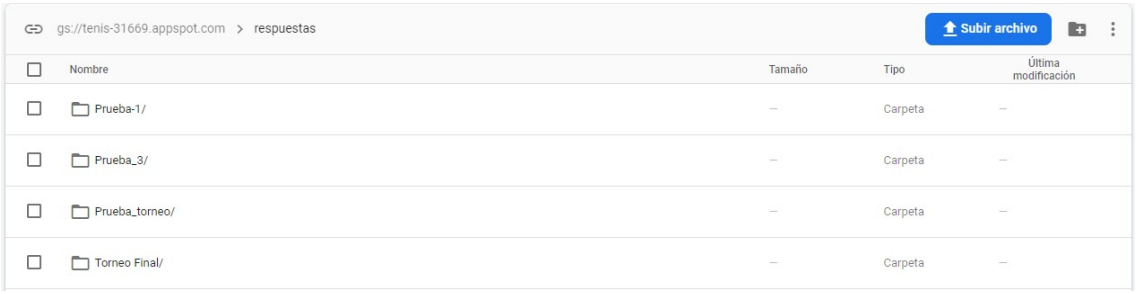


Figura 3.36: Carpeta “respuestas” del repositorio de almacenamiento

En estas carpetas se crea un archivo Excel por cada uno de los formularios del torneo con las respuestas de los jugadores por cada una de las rondas (Figura 3.37) y Figura 3.38).

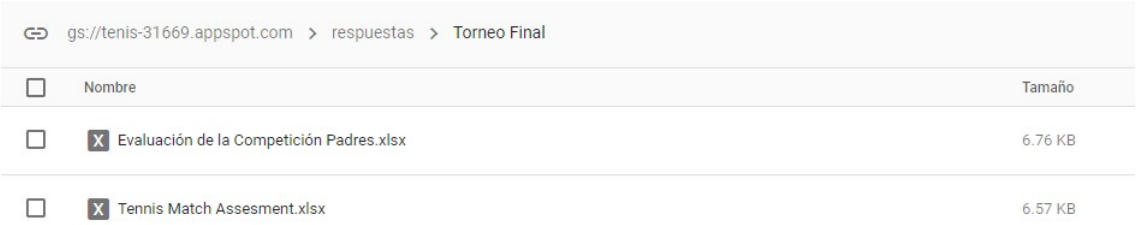


Figura 3.37: Respuestas del torneo

Estos archivos contienen las respuestas de los jugadores para ese formulario en todo el torneo, indicando en la primera columna el nombre del jugador y en la segunda columna la ronda contestada (Figura 3.37).

	A	B	C	D	E	F	G	H	I	J	K	L
1	Jugador	Ronda	El resultad	Este parti	Valoració	Nivel mos	Nivel mos	Nivel de c	He contro	Cuándo no los he controlado?		
2	Pedro López	1	Mejor	10	10			0	Si			
3	Juan Álvarez	1	Igual	7	6	6		6	Si			
4	Pedro López	2	Igual	7	6	2	4	8	No	En los puntos decisivos		
5	Juan Álvarez	2	Igual	3	3	3						
6	Pedro López	3		5		4		7				
7	Juan Álvarez	3	Peor		4			0	Si			

Figura 3.38: Respuestas del formulario

Capítulo 4

Pruebas y evaluación.

En este capítulo se detallan las pruebas que se han realizado para evaluar el correcto funcionamiento del sistema implementado.

4.1 Pruebas durante el desarrollo

La implementación de la aplicación se ha llevado a cabo en los terminales Xiaomi Mi A1 y Redmi Note 8T, cuyas especificaciones se observan en la Tabla 4.1 y la Tabla 4.2.

Modelo	Redmi Note 8T
Pantalla	6,3 pulgadas
Resolución	2340 x 1080 pixels
Versión Android	MIUI V10 (Android 9 Pie)
Procesador	Qualcomm Snapdragon 665/ 4x2.0 GHz + 4x 1.7 GHz Kryo 260
Memoria RAM	4GB

Tabla 4. 1: Especificaciones Técnicas Redmi Note 8T

Modelo	Xiaomi Mi A1
Pantalla	5,5 pulgadas
Resolución	1080 x 1920 pixels
Versión Android	Android 9.0 P
Procesador	Qualcomm Snapdragon 625/ Octa-Core 2GHz
Memoria RAM	4GB

Tabla 4. 2: Especificaciones Técnicas Xiaomi Mi A1

Para comprobar algunas de las funcionalidades de la aplicación como el envío de notificaciones era necesario tener varios dispositivos ya que el administrador envía las notificaciones a los dispositivos de los jugadores.

El terminal Redmi Note 8T realizaba la función de administrador mientras el Xiaomi Mi A1 recibía las notificaciones y contestaba a los formularios.

Para comprobar que las respuestas del dispositivo se guardan en el almacenamiento en la nube de Firebase se accede a la cuenta del proyecto y se descarga el archivo generado comprobando que las respuestas enviadas son las mismas que las que almacena el fichero.

4.2 Pruebas tras el desarrollo

Tras el desarrollo, la FAT ha recibido la aplicación y la ha distribuido entre algunos de sus jugadores para realizar algunas pruebas. No se ha realizado ningún torneo de manera oficial pero en su defecto se han realizado varias simulaciones de torneos y los resultados han sido positivos.

Hasta 9 usuarios procedentes de la FAT han probado la aplicación. Además, se han generado cuentas falsas para probar la aplicación en un torneo con 16 jugadores. Para las pruebas se han utilizado 9 dispositivos móviles Android de distintas marcas, 8 para ejercer la función de jugadores del torneo y otro más que ejercía de administrador de la aplicación.

Los dispositivos móviles recibían las notificaciones segundos después de que el administrador gestionara las rondas del torneo.

Aunque la concurrencia en las funciones de Cloud Functions es gestionada por Firebase se ha probado a contestar de manera simultánea varios formularios para comprobar la concurrencia en la función que genera el fichero Excel con las respuestas y en todos los casos se han guardado todas las respuestas en el archivo Excel sin perder ninguna respuesta ni sobrescribir alguna existente.

Capítulo 5

Conclusiones y líneas futuras

En este capítulo se describe una pequeña opinión personal sobre el proyecto realizado y las conclusiones extraídas de su realización. También se incluye un pequeño análisis de posibles mejoras del sistema en el futuro.

5.1 Opinión personal

El desarrollo de este trabajo ha supuesto todo un desafío desde el principio, tras las distintas reuniones con la Dirección Técnica de la FAT (los usuarios finales de la aplicación) para la toma de requisitos y el posterior análisis del problema planteado han supuesto que el desarrollo de este TFG se parezca mucho al de una situación en el mundo laboral.

El primer problema al que me enfrentaba era la necesidad de un servidor gratuito para gestionar los usuarios y todos los datos generados. En la fase de análisis investigué varias opciones, incluso llegando a realizar alguna pequeña prueba de concepto para ver la viabilidad de la solución, ninguna podía satisfacer todos los requisitos recogidos.

Sinceramente, creo que el descubrimiento de Firebase como plataforma de desarrollo que nos proporciona un servidor de backend gratuito y que requiere la mínima configuración hizo posible el desarrollo de la aplicación.

No tener unas especificaciones técnicas para implementar el sistema me ha aportado libertad en ese sentido, pudiendo elegir tanto las herramientas utilizadas como el diseño de la aplicación, pero a su vez he tenido que ir resolviendo los problemas encontrados con una plataforma completamente desconocida para mi hasta este trabajo.

Como conclusión final, considero que la realización de este trabajo ha sido algo muy útil a nivel personal por las tecnologías que he aprendido y también a nivel profesional por el hecho de haber tratado directamente con el cliente en todas las fases del desarrollo.

5.2 Conclusiones

Al concluir el desarrollo se puede observar cómo se alcanza el objetivo principal del TFG al informatizar el sistema REDPT pudiendo replicar las preguntas del sistema desde la aplicación con un diseño similar al que disponían en papel.

Los objetivos secundarios de este trabajo también se alcanzan al permitir crear formularios de contenido similar al del sistema REDPT para poder evaluar distintos factores de la competición.

La creación de torneos y la asignación de jugadores y formularios a estos torneos permiten a la FAT distribuir las instancias de los formularios de evaluación a través de los dispositivos móviles de los jugadores sin tener que repartir estos formularios en papel al finalizar los partidos.

Los jugadores posteriormente contestan los formularios y las respuestas se almacenan en archivos Excel lo que permite obtener las respuestas de los jugadores de manera más rápida para su posterior evaluación.

5.3 Líneas futuras

Aunque la aplicación desarrollada es completamente funcional para los requisitos propuestos solo está disponible para dispositivos Android. El proyecto creado en Firebase admite varias aplicaciones conectadas por lo que se podría desarrollar la aplicación para dispositivos iOS utilizando la misma base de datos. Además se podrían añadir algunas mejoras funcionales a la aplicación.

Los jugadores podrían ser notificados también cuando se les añade a un torneo para recordarles el lugar y la fecha del evento en sus dispositivos móviles.

Además, se podría realizar una integración con los calendarios de Google para que los jugadores tengan sincronizados sus calendarios en la nube con los torneos que van a jugar.

Capítulo 6

Bibliografía y referencias

En este capítulo se citan los artículos y páginas web consultadas para el desarrollo de este Trabajo de Fin de Grado.

[1] Android Studio and SDK tools. [ONLINE]. HYPERLINK "https://developer.android.com/studio" https://developer.android.com/studio, última fecha de visita: 13/11/2019

[2] Firebase [ONLINE]. HYPERLINK "https://firebase.google.com/" https://firebase.google.com/, última fecha de visita 13/11/2019

[3] Firebase Authentication [ONLINE] HYPERLINK "https://firebase.google.com/docs/auth" https://firebase.google.com/docs/auth/, última fecha de visita: 13/11/2019

[4] Cloud Firestore [ONLINE] HYPERLINK "https://firebase.google.com/docs/firestore" https://firebase.google.com/docs/firestore, última fecha de visita: 13/11/2019

[5] Cloud Storage [ONLINE] HYPERLINK "https://firebase.google.com/docs/storage" https://firebase.google.com/docs/storage, última fecha de visita: 13/11/2019

[6] Cloud Messaging [ONLINE] HYPERLINK "https://firebase.google.com/docs/cloud-messaging" https://firebase.google.com/docs/cloud-messaging, última fecha de visita: 13/11/2019

[7] Cloud Functions [ONLINE] HYPERLINK "https://firebase.google.com/docs/functions" https://firebase.google.com/docs/functions, última fecha de visita: 13/11/2019

[8] Node.js [ONLINE] HYPERLINK “https://nodejs.org/” https://nodejs.org/, última fecha de visita: 13/11/2019

[9] NPM [ONLINE] HYPERLINK “https://www.npmjs.com/” https://www.npmjs.com/, última fecha de visita: 13/11/2019

[10] Firebase CLI [ONLINE] HYPERLINK “https://firebase.google.com/docs/cli” https://firebase.google.com/docs/cli, última fecha de visita: 13/11/2019

[11] SheetJS [ONLINE] HYPERLINK “https://github.com/SheetJS/sheetjs” https://github.com/SheetJS/sheetjs, última fecha de visita: 13/11/2019

[12] ExcelJS [ONLINE] HYPERLINK https://github.com/exceljs/exceljs https://github.com/exceljs/exceljs, última fecha de visita: 13/11/2019

[13] Modelo de datos de Cloud Firestore [ONLINE] “https://firebase.google.com/docs/firestore/data-model” https://firebase.google.com/docs/firestore/data-model, última fecha de visita: 13/11/2019

Anexo A. Distribución de la aplicación

En este apartado se explica cual es el procedimiento que se ha seguido para la distribución de la aplicación entre los usuarios.

La dirección técnica de la FAT no quiere publicar la aplicación en Google Play que es la plataforma de distribución digital de aplicaciones móviles para Android. La alternativa por la que se ha optado es subir el archivo APK de la aplicación, que es el fichero donde se encuentra el código compilado y que sirve para instalar la aplicación en un dispositivo Android, al sistema de almacenamiento en la nube Google Drive con la cuenta de Google creada con la que también se ha generado el proyecto en Firebase.

De esta manera, la FAT envía el enlace de descarga del archivo a los jugadores para que se lo instalen en sus dispositivos móviles y así poder participar en los formularios de los torneos.

Anexo B. Guía de instalación

En este anexo se describen los pasos a seguir para instalar la aplicación en un dispositivo móvil Android.

Cuando el usuario se descarga el fichero de instalación de la aplicación desde Google Drive, el dispositivo nos avisa con un mensaje indicando que no se permite instalar aplicaciones desconocidas desde Google Drive (Figura B.1).

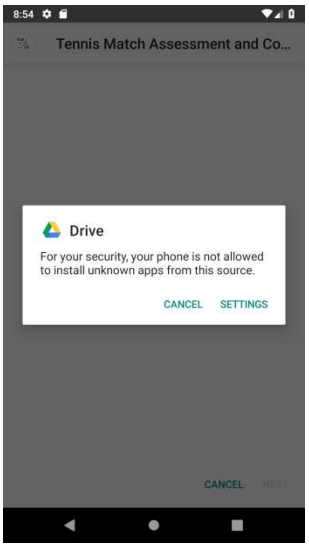


Figura B.1: Aplicaciones desconocidas

Pulsando en el botón *Ajustes* nos llevara a una pantalla donde se permite modificar esta restricción y autorizar la instalación. (Figura B.2).

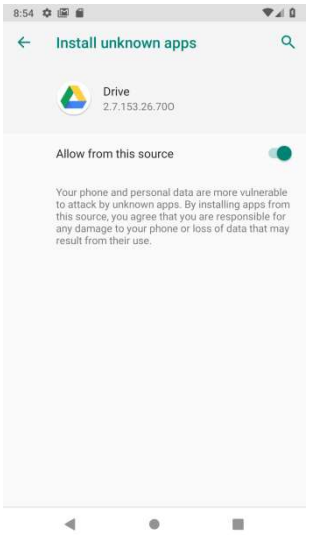


Figura B.2: Autorizar instalación

Por último, el proceso de instalación se inicia en la siguiente pantalla pulsando el botón *Instalar* (Figura B.3) y finaliza cuando la pantalla muestra un mensaje informando que la aplicación está instalada (Figura B.4).

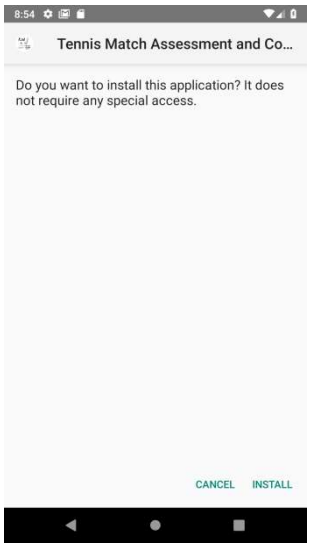


Figura B.3: Instalar aplicación

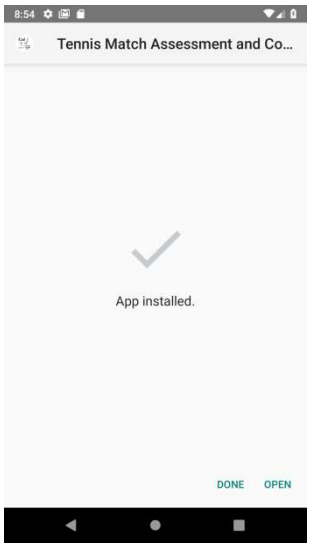


Figura B.4: Abrir aplicación

Anexo C. Pantallas de Firebase

En este anexo se incluyen el resto de pantallas de la plataforma de desarrollo Firebase.

El inicio de sesión en Firebase se hace a través de una cuenta de Google. Al iniciar sesión se muestran los proyectos creados en Firebase y una opción para crear nuevos proyectos (Figura C.1).

Figura C.1: Proyectos en Firebase

Para crear un proyecto hay que seguir los pasos que indica la plataforma introduciendo el nombre del proyecto y parámetros de configuración del proyecto como el uso de Google Analytics. Entrando en el proyecto creado y habiendo registrado la aplicación en el proyecto de Firebase tal y como se explica en la sección 3.2, la pantalla principal muestra cierta información analítica de la aplicación (Figura C.2).

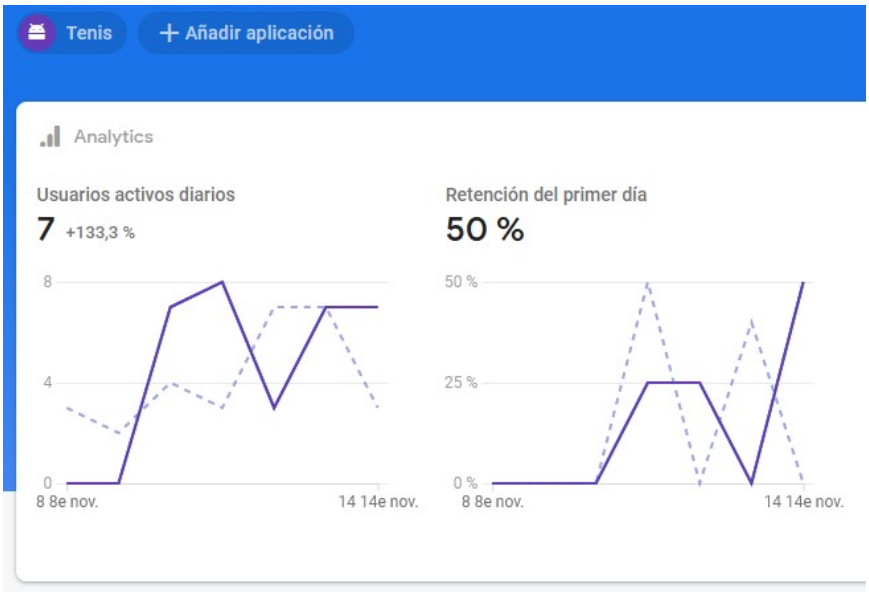


Figura C.2: Información analítica de la aplicación

En el módulo de Authentication se puede ver una lista con los usuarios registrados en la aplicación (Figura C.3), desde este módulo se pueden eliminar cuentas de usuario o inhabilitarlas.

Identificador	Proveedores	Fecha de creación	Inicio de sesión	UID de usuario
@gmail.com	✉	2 oct. 2019	16 nov. 2019	2CXUShJGvdM9S
ı@gmail.com	✉	27 may. 2019	15 nov. 2019	3i2n4PHwRfcdVS
@mail.com	✉	4 nov. 2019	16 nov. 2019	6iCC99sCxwfMRF
@yahoo.es	✉	11 nov. 2019	11 nov. 2019	FNA4L8Gk2dXH0
@gmail.com	✉	12 nov. 2019	12 nov. 2019	JENPyieT54hnDY
@hotmail.com	✉	6 nov. 2019	11 nov. 2019	L1YcVEIDcgXYT7
@gmail.com	✉	12 nov. 2019	12 nov. 2019	LvZIKdwmo7RcJr
@mail.com	✉	4 nov. 2019	4 nov. 2019	OMKwJh0Pc1YkI
@hotmail.com	✉	17 jul. 2019	28 oct. 2019	Qn8BD1r6PwfUjd
⁊@gmail.com	✉	6 nov. 2019	6 nov. 2019	RPR1aqkZM4fFsc
@aragontenis.com	✉	11 nov. 2019	15 nov. 2019	lqpxUgELCCXRaY
@gmail.com	✉	12 nov. 2019	12 nov. 2019	mZIqUEIEjuQakxy

Figura C.3: Usuarios registrados

En otro apartado de este módulo se configuran los métodos de inicio de sesión de la aplicación (Figura C.4).

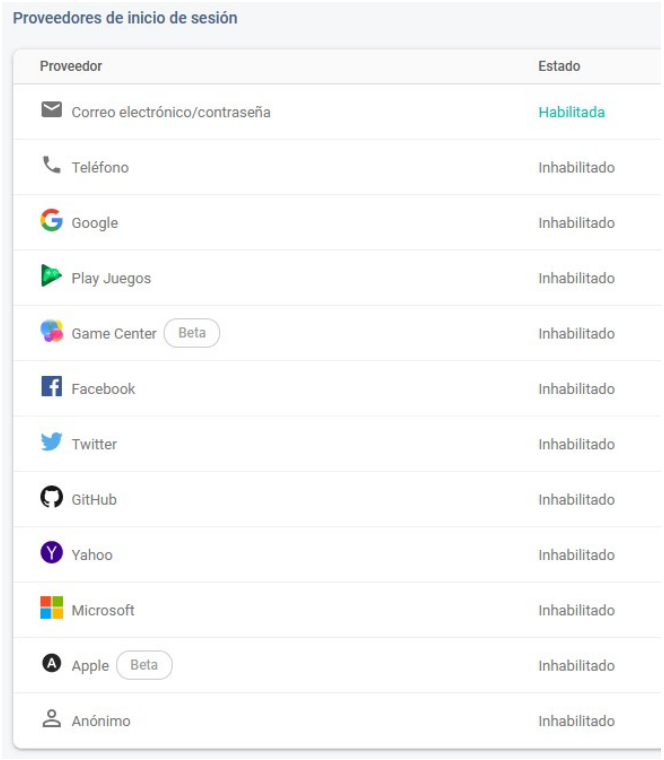


Figura C.4: Métodos de inicio de sesión

En el módulo de la base de datos se pueden ver, eliminar y modificar las colecciones y los documentos con los datos generados por los usuarios de la aplicación (Figura C.5).

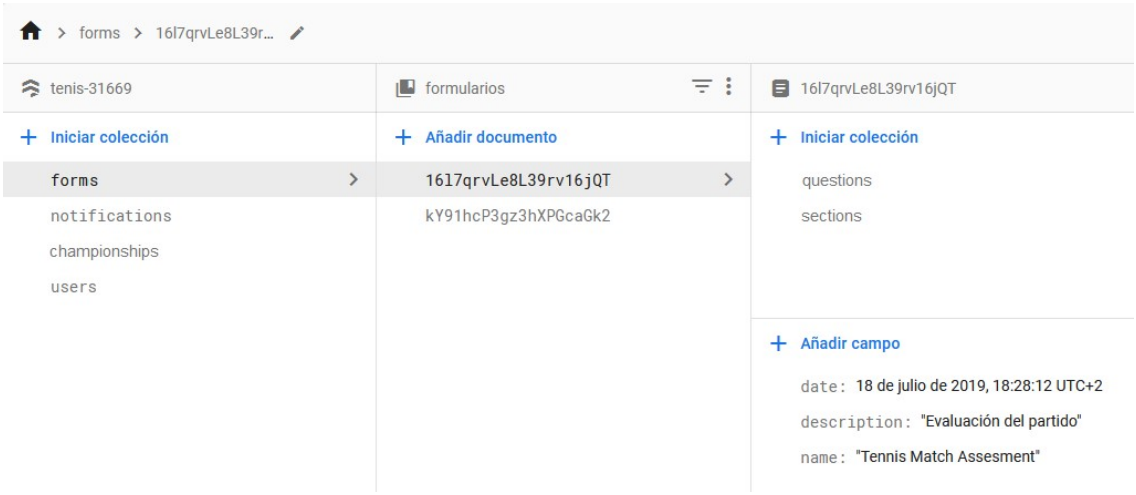


Figura C.5: Base de datos en Firebase

En el módulo de Cloud Functions encontramos las funciones desarrolladas en este módulo indicando en una tabla los activadores de las mismas junto a otra información como el *timeout* de ejecución o la región donde se ejecuta (Figura C.6).

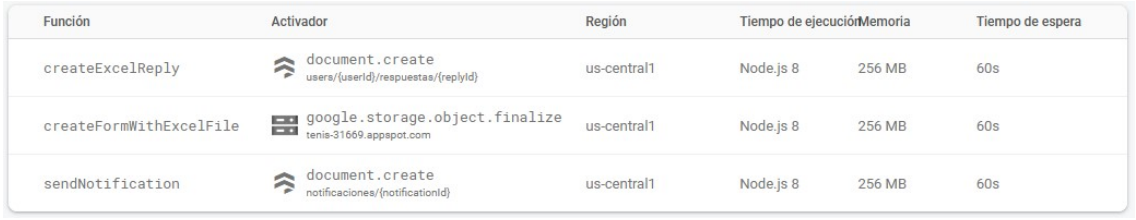
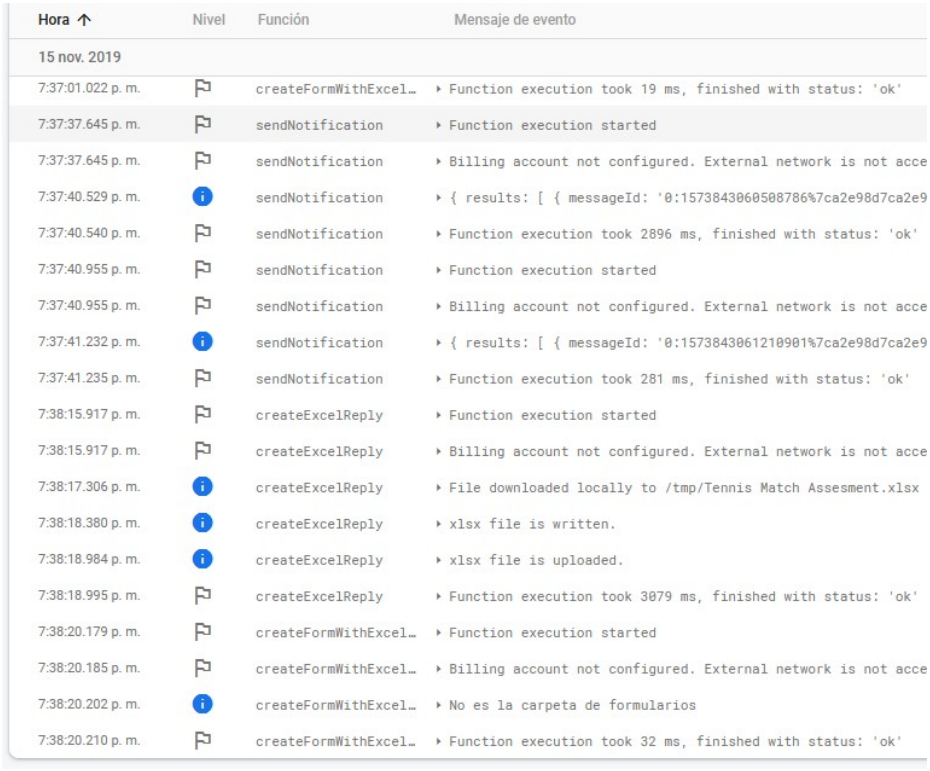


Figura C.6: Funciones de Cloud Functions

En otra sección de este módulo podemos ver las trazas de registro que dejan las funciones desarrolladas. De esta manera se puede comprobar si ha habido algún error en la ejecución de alguna de las funciones (Figura C.7). La plataforma nos permite filtrar la traza de registro para localizar la información fácilmente.



Hora ↑	Nivel	Función	Mensaje de evento
15 nov. 2019			
7:37:01.022 p. m.		createFormWithExcel…	▸ Function execution took 19 ms, finished with status: 'ok'
7:37:37.645 p. m.		sendNotification	▸ Function execution started
7:37:37.645 p. m.		sendNotification	▸ Billing account not configured. External network is not acce
7:37:40.529 p. m.		sendNotification	▸ { results: [{ messageId: '0:1573843060508786%7ca2e98d7ca2e9
7:37:40.540 p. m.		sendNotification	▸ Function execution took 2896 ms, finished with status: 'ok'
7:37:40.955 p. m.		sendNotification	▸ Function execution started
7:37:40.955 p. m.		sendNotification	▸ Billing account not configured. External network is not acce
7:37:41.232 p. m.		sendNotification	▸ { results: [{ messageId: '0:1573843061210901%7ca2e98d7ca2e9
7:37:41.235 p. m.		sendNotification	▸ Function execution took 281 ms, finished with status: 'ok'
7:38:15.917 p. m.		createExcelReply	▸ Function execution started
7:38:15.917 p. m.		createExcelReply	▸ Billing account not configured. External network is not acce
7:38:17.306 p. m.		createExcelReply	▸ File downloaded locally to /tmp/Tennis Match Assesment.xlsx
7:38:18.380 p. m.		createExcelReply	▸ xlsx file is written.
7:38:18.984 p. m.		createExcelReply	▸ xlsx file is uploaded.
7:38:18.995 p. m.		createExcelReply	▸ Function execution took 3079 ms, finished with status: 'ok'
7:38:20.179 p. m.		createFormWithExcel…	▸ Function execution started
7:38:20.185 p. m.		createFormWithExcel…	▸ Billing account not configured. External network is not acce
7:38:20.202 p. m.		createFormWithExcel…	▸ No es la carpeta de formularios
7:38:20.210 p. m.		createFormWithExcel…	▸ Function execution took 32 ms, finished with status: 'ok'

Figura C.7: Traza de registro en Cloud Functions

Anexo D. Fragmentos y métodos

En este anexo se detallan las actividades, fragmentos y métodos desarrollados dividiéndolos por el tipo de usuario que puede utilizarlos.

Actividades y Fragmentos comunes:

En la aplicación existen actividades y fragmentos que pueden ser utilizados por los dos tipos de usuarios que existen en la aplicación. Estas actividades y fragmentos se describen a continuación indicando los métodos que contienen:

- **SignUpActivity.java:** Esta actividad se utiliza para registrar usuarios nuevos en la aplicación.

  - Métodos de la actividad:

    - **registerUser**: Valida los datos introducidos y llama al método *createUserWithEmailAndPassword* de Firebase para registrar al usuario.

- **LoginActivity.java:** Es la actividad utilizada para iniciar sesión en la aplicación.

  - Métodos de la actividad:

    - **userLogin**: Valida los datos introducidos y llama al método *signInWithEmailAndPassword* de Firebase para iniciar la sesión del usuario. Cuando esta se completa se llama al método *updateToken* pasándole el token que ha obtenido el usuario al iniciar sesión.

    - **updateToken**: Actualiza el token de usuario en la base de datos.

- **MainActivity.java:** Es la actividad principal de la aplicación, implementa la clase *NavigationView* de Android. Cuando esta se inicia comprueba si el usuario tiene la sesión iniciada con la API de Firebase. Si no hay sesión iniciada, se llama a la actividad LoginActivity.java.

  - Métodos de la actividad:

- **placeNavigationHeader**: Obtiene el nombre y el correo electrónico del usuario que ha iniciado sesión y lo coloca en la cabecera del menú de navegación.

- **thereAreFormsToReply**: Consulta en la base de datos a través del identificador de usuario, si este se encuentra en alguno de los torneos que todavía no han finalizado.

- **showAdminButtons**: Muestra el conjunto de opciones que disponen los administradores consultando en la base de datos si el usuario es administrador.

- **onNavigationItemSelected:** Este método se anula de la implementación de *NavigationView* y se implementa para iniciar las funcionalidades principales de la aplicación desde el menú de navegación lateral.

- **EditUserProfileFragment.java:** Este fragmento es utilizado para cambiar la contraseña de la cuenta y para permitir la eliminación de la misma.

  - Métodos del fragmento:

    - **editProfile:** Comprueba si el campo contraseña ha sido introducido, mostrando en caso afirmativo un dialogo de progreso con un mensaje informativo para validar los datos introducidos con el método *reauthenticate* de Firebase. Si el método *reauthenticate* finaliza con éxito, se llama al método *modifyPassword*. En caso contrario se llama al método *showErrorUpdatingPassword*.

    - **modifyPassword:** Valida el campo "Nueva Contraseña" para que contenga al menos 6 caracteres y utiliza el método *updatePassword* de Firebase para actualizar la contraseña del usuario.

    - **showErrorUpdatingPassword:** Este método informa al usuario que la contraseña introducida para el cambio de la misma no es correcta.

- **onDeleteClick:** Este método se activa pulsando el botón eliminar cuenta. Muestra un dialogo con la clase *AlertDialog* de Android preguntando al usuario si está seguro de que desea eliminar su cuenta y obligándole a introducir la contraseña de la cuenta para proceder a su eliminación. Si el usuario prosigue con la eliminación de la cuenta, introduciendo la contraseña y pulsando el botón "Si", se llama al método *deleteAccount*.

- **deleteAccount:** Este método recibe la contraseña introducida por el usuario en el dialogo de aviso y utilizando el método *reauthenticate* de Firebase se comprueban si la contraseña introducida es correcta. En caso afirmativo, se procede a borrar el documento del usuario en la base de datos y en el sistema de registro de la plataforma con el método con el método *delete* de Firebase

Fragmentos de los administradores:

Los administradores de la aplicación poseen funcionalidades a las que los jugadores no pueden acceder. Estas funcionalidades son proporcionadas por los siguientes fragmentos:

- **CreateFormFragment.java:** Con este fragmento se crean los formularios que serán añadidos a los torneos.

  - Métodos del fragmento:

    - **addSectionToLayout:** Al pulsar el botón "Añadir Sección" añade una nueva sección al formulario inflando el layout *section.xml* en el *LinearLayout* que actúa como contenedor de las secciones.

    - **addQuestionToSection:** Con el mismo procedimiento que el método anterior, este método añade preguntas a la sección correspondiente pulsando el botón "Añadir Pregunta".

    - **addFormToFirebase:** Este método recoge las secciones y las preguntas creadas validando los datos introducidos. Si los datos son correctos, crea

el documento en la base de datos de Firebase con las subcolecciones que contienen las secciones y las preguntas de estas.

- **ListManageFormsFragment.java:** Este fragmento muestra la lista de formularios creados.

  - Métodos del fragmento:

    - **loadFormList:** Consulta los formularios creados en la base de datos y los muestra en una lista del layout utilizando el componente *ListView* y el adaptador de la vista. Cuando uno de los elementos de la lista es pulsado se inicia el fragmento *ManageFormFragment.java*.

- **ManageFormFragment.java:** Muestra los datos del formulario seleccionado en el fragmento anterior permitiendo al administrador obtener una vista previa del formulario. Además permite eliminar el formulario que se está visualizando.

  - Métodos del fragmento:

    - **loadSectionsData:** Obtiene y almacena en una lista las secciones del formulario consultando en la subcolección del documento

    - **loadQuestionsData:** Obtiene y almacena en una lista las preguntas del formulario consultando en la subcolección del documento

    - **placeQuestionsInSection:** Recorre la lista de secciones y la lista de preguntas para colocar estas últimas en la sección correspondiente a través del identificador que almacenan en la base de datos.

    - **deleteForm:** Elimina el formulario de la base de datos.

- **CreateChampionshipFragment.java:** Con este fragmento se inicia la creación de un torneo.

  - Métodos del fragmento:

    - **showDatePickerDialog:** Muestra un dialogo con un calendario para facilitar la introducción de las fechas de inicio y fin del torneo.

- **nextStepCreateChampionship:** Valida los datos introducidos y si son correctos inicia el fragmento *AddPlayersToChampionshipFragment.java*

- **AddPlayersToChampionshipFragment.java:** Este fragmento muestra en una lista los jugadores de la aplicación.

  - Métodos del fragmento:

    - **loadUserList:** Obtiene los jugadores de la base de datos y los carga en la *ListView* del layout utilizando el adaptador de la vista.

    - **checkSelectedPlayersAndContinue:** Comprueba el número de jugadores seleccionados sea correcto según el número de rondas introducido en el paso anterior.

- **AddFormsToChampionshipFragment.java:** Este fragmento muestra en una lista los formularios disponibles en la aplicación.

  - Métodos del fragmento:

    - **loadFormList:** Obtiene los formularios de la base de datos y los muestra en la *ListView* del layout utilizando el adaptador de la vista.

    - **createChampionship:** Comprueba que al menos se ha seleccionado un formulario para el torneo que se está creando. Crea el documento del torneo y la subcolección *rounds* en la base de datos con la información obtenida en los pasos anteriores. Define en los documentos de las rondas un mapa de datos que sirve para gestionar cuando un jugador puede contestar un formulario.

- **ManageChampionshipsFragment.java:** Permite gestionar los torneos en curso colocando cada uno de los torneos en curso en una lista.

  - Métodos del fragmento:

    - **loadChampionshipList:** Obtiene de la base de datos una lista con los torneos en curso utilizando el método *whereGreaterThanOrEqualTo* de Firebase. Utilizando este método se compara la fecha de finalización de

cada uno de los torneos con la fecha en la que se realiza la consulta. Si la fecha de finalización del torneo es posterior al día actual se incluirá en la lista. Después de realizar esta consulta, se filtran los torneos que todavía no han empezado, es decir, los que la fecha de inicio es posterior a la fecha en la que se ejecuta la consulta. Si el usuario pulsa sobre cualquier elemento de la lista de torneos se llama al fragmento *ManageChampionshipRoundsFragment.java.*

- **ManageChampionshipRoundsFragment.java:** Este fragmento carga una lista con las rondas del torneo seleccionado.

  - Métodos del fragmento:

    - **loadRounds:** Obtiene de la base de datos las rondas del torneo seleccionado y las muestra en un *ListView* de la vista a través del adaptador. Al pulsar sobre cualquier elemento de la lista se inicia el fragmento *ManageChampionshipPlayersFragment.java***.**

- **ManageChampionshipPlayersFragment.java:** Este fragmento carga la lista de jugadores que está disputando esa ronda para que el administrador seleccione a los jugadores que han finalizado su partido.

  - Métodos del fragmento:

    - **loadUserList:** Obtiene de la base de datos los jugadores de la ronda que se está gestionando y los muestra en un *ListView* en la vista con el adaptador.

    - **checkSelectedPlayersAndContinue:** Comprueba que el número de jugadores seleccionado es 2 y los envía al fragmento *SelectWinnerFragment.java.*

- **SelectWinnerFragment.java:** Este fragmento recibe los jugadores seleccionados en el paso anterior y permite designar al ganador de un partido.

  - Métodos del fragmento:

    - **addWinnerToNextRound:** Comprueba que solo se ha seleccionado un jugador y en caso afirmativo modifica el mapa creado en el método *createChampionship* para que estos dos jugadores puedan contestar los formularios de este partido. Elimina de la ronda a los dos jugadores y, si no es la última ronda del torneo, añade al ganador del partido a la siguiente ronda del torneo.

- **ListManageUsersFragment.java:** Este fragmento permite ver los usuarios de la aplicación.

  - Métodos del fragmento:

    - **loadUserList:** Obtiene de la base de datos los usuarios de la aplicación y los muestra en un *ListView* de la vista. Pulsando sobre cualquier elemento de la lista se inicia el fragmento *ManageUserFragment.java*.

- **ManageUserFragment.java:** Permite dar y quitar permisos de administrador a otros usuarios.

  - Métodos del fragmento:

    - **changeUserRole:** Actualiza en la base de datos el rol del usuario seleccionado.

Fragmentos de los jugadores:

Los jugadores poseen otras funcionalidades distintas a las de los administradores. Estas funcionalidades las proporcionan los siguientes fragmentos:

- **ListChampionshipsFragment.java:** Muestra en una lista los torneos en curso.

  - Métodos del fragmento:

    - **loadChampionships:** Obtiene de la base de datos los torneos en curso del jugador utilizando con su identificador de usuario el método *whereArrayContains* y, de la misma manera que en el método

loadChampionshipList, el método *whereGreaterThanOrEqualTo*. Cuando se pulsa sobre cualquier elemento de la lista se incia el fragmento *ListFormFragment.java.*

- **ListFormFragment.java:** Muestra en una lista los formularios del torneo no contestados por el jugador.

  - Métodos del fragmento:

    - **loadRounds:** Obtiene de la base de datos las rondas del torneo y las guarda en una lista.

    - **loadAllForms:** Obtiene de la base de datos los formularios del torneo y los almacena en una lista.

    - **loadFormsCanReplyFromRounds:** Obtiene la lista de formularios que los jugadores pueden contestar. Para ello, recorre las rondas y comprueba en base de datos si el jugador ha contestado a los formularios de la ronda. Coloca los resultados en un *ListView* de la lista. Al seleccionar cualquier elemento de la lista se inicia el fragmento *ReplyFormFragment.java.*

- **ReplyFormFragment.java:** Muestra las preguntas del formulario seleccionado y permite a los jugadores responder a las preguntas.

  - Métodos del fragmento:

    - **loadFormData:** Obtiene de base de datos los datos básicos del formulario y los coloca en la cabecera del layout del formulario.

    - **loadSectionsData:** Obtiene de base de datos las secciones del formulario y las guarda en una lista.

    - **loadQuestionsData:** Obtiene de base de datos las preguntas del formulario y las almacena en una lista.

    - **placeQuestionsInSection:** Coloca las preguntas del formulario en la sección correspondiente, inflando los layouts correspondientes al tipo de pregunta.

- **retrieveResponses:** Recorre las preguntas del formulario recogiendo las respuestas y almacenándolas en una lista.

- **addReplyToFirebase:** Al pulsar el botón “Enviar Respuesta” se llama a este método. Este método llama a *retrieveResponses* para recuperar las respuestas en una lista y recorrerlas para fijar los resultados en la base de datos. Cuando los datos son almacenados en la base de datos se llama al método *userCanNoReplyThisFormAnymore*.

- **userCanNoReplyThisFormAnymore:** Este método actualiza el mapa de la ronda del torneo que gestiona que jugadores del torneo pueden acceder al formulario. De esta forma cuando un jugador contesta el formulario de su partido ya no puede volver a contestarlo.