
# Programación de Aplicaciones Móviles Nativas

Tournamaker

**Autores**
José Manuel Díaz Hernández
Francisco Malillos Castellano

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción

Las competiciones deportivas y los torneos amistosos se han popularizado gracias a la facilidad para organizar encuentros entre grupos de amigos, equipos aficionados y ligas locales, tanto presenciales como en línea. Sin embargo, la gestión manual de calendarios, emparejamientos y resultados sigue siendo una tarea compleja para muchos organizadores, que suelen apoyarse en hojas de cálculo o aplicaciones generales que no se adaptan bien a las particularidades de cada torneo. En este contexto surge *TournaMaker*, una aplicación móvil orientada inicialmente al fútbol, pero diseñada para ser aplicable a cualquier deporte o disciplina basada en eliminatorias o rondas.

La aplicación ofrece a los usuarios la posibilidad de crear y gestionar equipos, partidos y torneos de forma flexible, permitiendo desde competiciones profesionales hasta torneos rápidos entre amigos. Cada torneo cuenta con información asociada como fecha, lugar, premio, tasa de inscripción y una descripción donde se pueden incluir las normas, mientras que los partidos incorporan datos como hora, localización y marcador. Los equipos se pueden configurar sin restricciones de número de jugadores, lo que permite tanto plantillas reales como equipos ficticios creados únicamente para registrar resultados de partidas informales, por ejemplo en videojuegos como *FIFA* u otros de otros deportes.

TournaMaker se centra en facilitar el trabajo del organizador y, al mismo tiempo, ofrecer una experiencia clara a los participantes. Para participar en la aplicación es necesario registrarse y validar el correo electrónico, de forma que cada usuario puede gestionar su propio perfil, modificar su contraseña y consultar estadísticas sobre el número de torneos, partidos y equipos en los que participa. A partir de su cuenta, el usuario puede crear equipos, inscribirlos en torneos o partidos y recibir notificaciones cuando otros usuarios se unen a su equipo o se unen equipos a su torneo o partido de manera que se mantiene un historial ordenado de la actividad de los equipos, partidos o torneos que el usuario organiza.

Uno de los aspectos clave de la aplicación es la gestión de la lógica de competición. Los torneos y partidos solo pueden comenzar cuando se ha alcanzado el número de equipos requerido, pero el inicio real puede adaptarse al nivel de formalidad deseado por el organizador. De este modo, es posible usar la aplicación tanto para competiciones con horarios estrictos como para torneos improvisados donde simplemente se desea registrar los emparejamientos y los resultados finales. En el caso de los torneos por eliminatorias, los equipos se colocan de forma aleatoria en un *bracket* que se va actualizando según los resultados, lo que permite usar la aplicación también como herramienta de sorteo.

La pantalla principal de TournaMaker muestra un resumen de torneos, partidos y equipos disponibles, con accesos directos para ver listados completos y crear nuevos ele-

mentos. El usuario puede crear tantos equipos, partidos y torneos como desee, y al unirse a un torneo o partido puede elegir con cuál de sus equipos propietarios desea participar. El usuario puede unirse con su equipo a un torneo que organiza el mismo, garantizando eso mismo de que el torneo pueda ser lo más informal que quiera el usuario.

Desde el punto de vista tecnológico, la aplicación se ha desarrollado en Android Studio utilizando Kotlin y una arquitectura basada en el patrón MVVM siguiendo las recomendaciones oficiales de la arquitectura de Android [1], apoyada en *ViewModels*, repositorios y fragmentos especializados para cada funcionalidad. La persistencia y la autenticación de usuarios se gestionan mediante Firebase Firestore intengrando Firebase Authentication y Cloud Firestore[2, 3], de manera que toda la información relevante se almacena en la nube y se sincroniza entre dispositivos. Esta base tecnológica permite que la solución sea escalable y ofrece margen para incorporar funcionalidades futuras, como estadísticas detalladas de jugadores (goles, asistencias, tarjetas) o métricas avanzadas de equipos y torneos.

La idea original de TournaMaker surgió en una asignatura previa de Programación Web y Móvil, donde se desarrolló un prototipo web con un conjunto reducido de funcionalidades y unos primeros *mockups* diseñados en Figma para web, tablet y móvil. En la asignatura de Programación de Aplicaciones Móviles Nativas se ha retomado y ampliado ese concepto, rediseñando por completo la experiencia de usuario y la interfaz gráfica para mejorar la accesibilidad, el contraste de colores y la claridad de los flujos de navegación. Los diseños iniciales, basados en una paleta predominantemente verde asociada al fútbol, han evolucionado hacia una interfaz más neutra y elegante con tonos violeta y blanco, pensada para ser usable en cualquier tipo de disciplina y en contextos tanto casuales como profesionales.

El propósito de esta memoria es documentar el contexto, el diseño y el desarrollo de TournaMaker, así como las decisiones técnicas y de experiencia de usuario que se han tomado durante el proyecto. En los capítulos siguientes se describen con mayor detalle los objetivos de la aplicación, el diseño de la interfaz y la experiencia de usuario, la arquitectura software y el modelo de datos, el proceso de desarrollo y las interfaces finales, para concluir con una reflexión sobre los resultados obtenidos y las posibles líneas de trabajo futuro.

Durante el desarrollo se ha utilizado apoyo puntual de asistentes de inteligencia artificial para resolver dudas de sintaxis y refinar algunos textos de la memoria, contrastando siempre las respuestas con la documentación oficial. En este caso Perplexity[4].

## 1.1. Organización del documento

El resto del documento se estructura de la siguiente manera. En el capítulo 2 se presenta una descripción detallada de la aplicación, incluyendo el público objetivo y los objetivos específicos del proyecto. A continuación, el capítulo 3 recoge las principales decisiones de diseño de la interfaz y la experiencia de usuario, así como los prototipos utilizados durante el proceso de ideación.

El capítulo 4 describe la arquitectura software adoptada, las capas y módulos que la componen y las tecnologías empleadas. Posteriormente, en el capítulo 5 se expone el modelo de datos, detallando las entidades principales de la aplicación y sus relaciones. El capítulo 6 resume el proceso de desarrollo, la organización del código y la estrategia de pruebas.

En el capítulo 7 se muestran las interfaces finales de la aplicación y los principales flujos

de navegación disponibles para el usuario. Por último, el capítulo 8 recoge las conclusiones del trabajo, las limitaciones detectadas y varias propuestas de mejora y ampliación, entre ellas la incorporación de estadísticas avanzadas de jugadores y equipos.

# Capítulo 2

# Descripción de la aplicación incluyendo objetivos

TournaMaker es una aplicación móvil para Android diseñada para facilitar la creación y gestión de torneos y partidos de forma flexible, tanto en contextos recreativos como en competiciones más organizadas. Aunque el proyecto nació pensando principalmente en el fútbol, la aplicación está concebida para ser independiente del deporte y puede emplearse en cualquier disciplina que se organice mediante rondas o eliminatorias, incluidos videojuegos deportivos y otros tipos de enfrentamientos por equipos.

La aplicación permite a los usuarios crear equipos, partidos y torneos sin restricciones en el número de elementos, ofreciendo un entorno unificado en el que organizar tanto eventos puntuales como eliminatorias más estructuradas. Los equipos no tienen límite de jugadores, lo que facilita tanto la representación de plantillas reales como la creación de equipos ficticios para torneos rápidos, por ejemplo para anotar resultados de partidas de videojuegos entre amigos.

El flujo de participación se basa en la relación entre organizadores, equipos y participantes. Cada usuario, tras registrarse y verificar su correo electrónico, puede crear sus propios equipos y emplearlos para unirse a torneos o partidos en los que actúa como propietario. Cuando un equipo se une a un torneo, partido, el organizador correspondiente recibe una notificación con la información y la fecha de unión. De esta manera, la aplicación sirve tanto para torneos profesionales, donde se requiere un control preciso de horarios y plazas, como para encuentros informales donde únicamente se desean registrar emparejamientos y resultados.

La pantalla principal de TournaMaker ofrece una visión resumida del ecosistema de la aplicación, mostrando listados destacados de torneos, partidos y equipos disponibles, junto con accesos directos para explorar todos los elementos de cada categoría. Desde esta pantalla es posible crear nuevos torneos, partidos y equipos, así como navegar a pantallas específicas para gestionar cada entidad. Esta organización pretende reducir la carga cognitiva del usuario, ofreciendo un punto de entrada único desde el que controlar toda su actividad competitiva y encontrar rápidamente la información relevante sobre dónde y cuándo participa.

En el caso de los torneos que por ahora son siempre eliminatorias, la aplicación genera un cuadro de enfrentamientos (*bracket*) donde los equipos se colocan de forma aleatoria al unirse y avanzan en función de los resultados introducidos por el organizador. Este mecanismo permite utilizar TournaMaker como herramienta de sorteo y seguimiento de eliminatorias, evitando tener que recurrir a herramientas externas para decidir empareja-

mientos. En los partidos individuales, el sistema permite iniciar el encuentro, actualizar el marcador en tiempo real o de forma diferida y finalizar el partido cuando el organizador lo considere oportuno, adaptándose tanto a un uso profesional como a un uso más informal.

La aplicación incorpora una pantalla de perfil en la que cada usuario puede consultar sus datos personales, modificar su contraseña mediante el envío de correos de verificación y revisar estadísticas sobre el número de torneos, partidos y equipos en los que participa, o visitar desde ahí mismo cada torneo, partido o equipo en el que ahora mismo está, haciendo que sea más fácil para el usuario consultar cualquier detalle que desee sin buscar en la pantalla principal un torneo infinitamente, ahorrando muchísimo tiempo. Esta vista resume la actividad del usuario dentro del sistema y aporta una perspectiva global de su implicación en diferentes competiciones. Además, posibilita escenarios en los que el propio organizador se inscribe en sus torneos o partidos con varios equipos creados por él mismo, por ejemplo para gestionar eliminatorias rápidas entre amigos sin necesidad de que cada participante disponga de una cuenta propia.

## 2.1. Objetivos

- Facilitar la creación y gestión de torneos y partidos para cualquier deporte o disciplina basada en enfrentamientos por rondas o eliminatorias.
- Proporcionar a los organizadores una herramienta unificada para registrar equipos, gestionar inscripciones y definir la información clave de cada evento (fechas, localizaciones, normas, premios y tasas de inscripción).
- Permitir a los participantes consultar de forma clara en qué torneos, partidos y equipos están involucrados, así como los horarios y lugares correspondientes.
- Automatizar la generación y actualización de cuadros de eliminatorias, de manera que los emparejamientos se asignen de forma sencilla y transparente.
- Ofrecer una experiencia de usuario accesible e intuitiva, adaptable tanto a competiciones informales entre amigos como a torneos con un grado mayor de formalidad.
- Sentar las bases para futuras extensiones de la aplicación, como el registro de estadísticas avanzadas de jugadores y equipos, sin necesidad de rediseñar la arquitectura general del sistema.

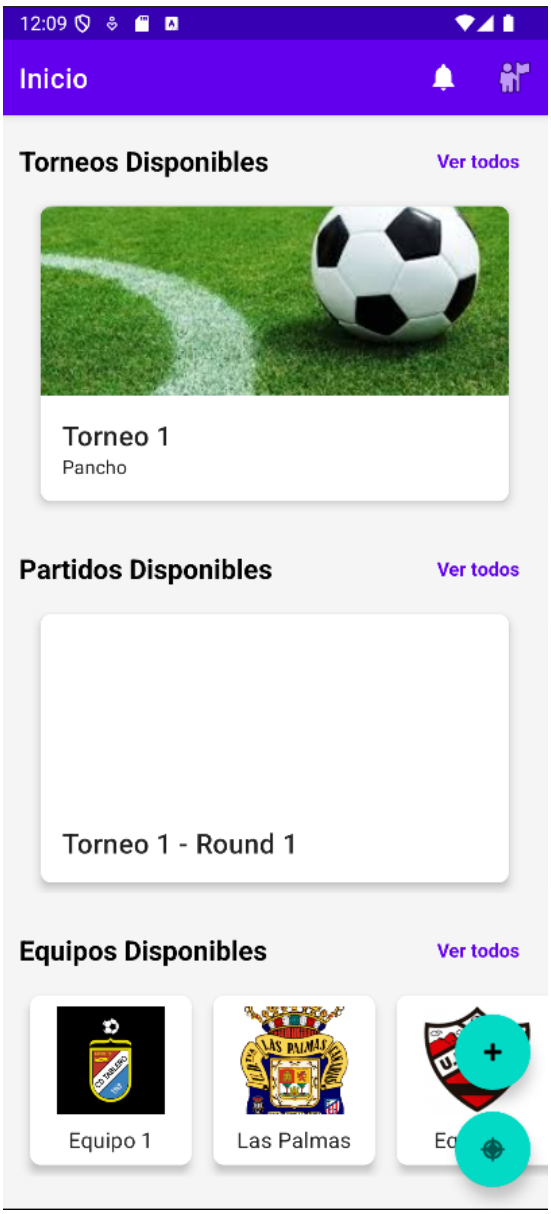
# Capítulo 3

# Diseño

El diseño de TournaMaker se ha planteado con el objetivo de ofrecer una experiencia de usuario sencilla e intuitiva tanto para organizadores como para participantes. La aplicación estructura sus funcionalidades en torno a una pantalla principal que actúa como punto de entrada, desde la que se accede a los listados de torneos, partidos y equipos, así como a las opciones de creación de nuevos elementos. De esta forma se reduce el número de pasos necesarios para realizar las acciones más frecuentes y se proporciona una visión global del estado de la competición.

En comparación con el prototipo web desarrollado en una asignatura previa, la interfaz se ha rediseñado para adaptarse al contexto móvil y cumplir mejor los criterios de accesibilidad, siguiendo los principios de Material Designsiguiendo los principios de Material Design [5]. Los diseños iniciales, basados en una paleta de verdes muy asociada al fútbol y con contrastes limitados, se han sustituido por una combinación de tonos violeta y blanco que mejora la legibilidad y resulta más neutra respecto al deporte o disciplina utilizada. Se han revisado también tamaños de tipografía, espaciados y jerarquías visuales para facilitar la lectura en pantallas pequeñas y hacer más evidente qué acciones están disponibles en cada momento, como el botón de crear torneo, partido o equipo que está siempre presente como se ve en las figuras que muestran el cambio entre nuestro mockup inicial del año pasado de la pantalla principal de la app, frente a la que se rediseño este año.


(a) Mockup inicial de la pantalla *landing* (año anterior).




(b) Pantalla *landing* actual de TournaMaker.

Figura 3.1: Comparación entre el diseño inicial y el diseño actual de la pantalla de inicio.

El flujo de navegación se organiza mediante fragmentos especializados que responden a las principales tareas del usuario: exploración de torneos y partidos, gestión de equipos, visualización de detalles y administración del perfil. Desde la pantalla de inicio, el usuario puede acceder a listados completos de torneos, partidos o equipos, o bien seleccionar directamente una tarjeta destacada para consultar sus detalles. En la vista de detalle de torneo o partido se muestran los datos clave (fecha, lugar, descripción, premio o tasa de inscripción) y se proporciona un botón claro para solicitar la participación con uno de los equipos de los que el usuario es propietario.

En el diseño se ha tenido en cuenta que la aplicación debe soportar tanto torneos estructurados como encuentros rápidos entre amigos. Por ello, las pantallas de creación de torneos, partidos y equipos permiten introducir solo la información imprescindible, dejando el resto de campos como opcionales para no sobrecargar al usuario cuando el uso

es más informal. Al mismo tiempo, las vistas de detalle muestran de forma compacta la información relevante, de manera que cada participante pueda saber de un vistazo dónde, cuándo y con qué condiciones se disputa cada evento.

## 3.1. Patrones y decisiones

A nivel de diseño lógico, la aplicación sigue una arquitectura basada en el patrón Modelo–Vista–VistaModelo (MVVM), en la que las vistas se implementan mediante fragmentos y actividades, los *ViewModels* encapsulan la lógica de presentación y los repositorios gestionan el acceso a los datos almacenados en Firebase Firestore. Esta organización favorece la separación de responsabilidades, facilita las pruebas unitarias sobre la lógica de negocio y simplifica la reutilización de código entre diferentes pantallas que comparten información.

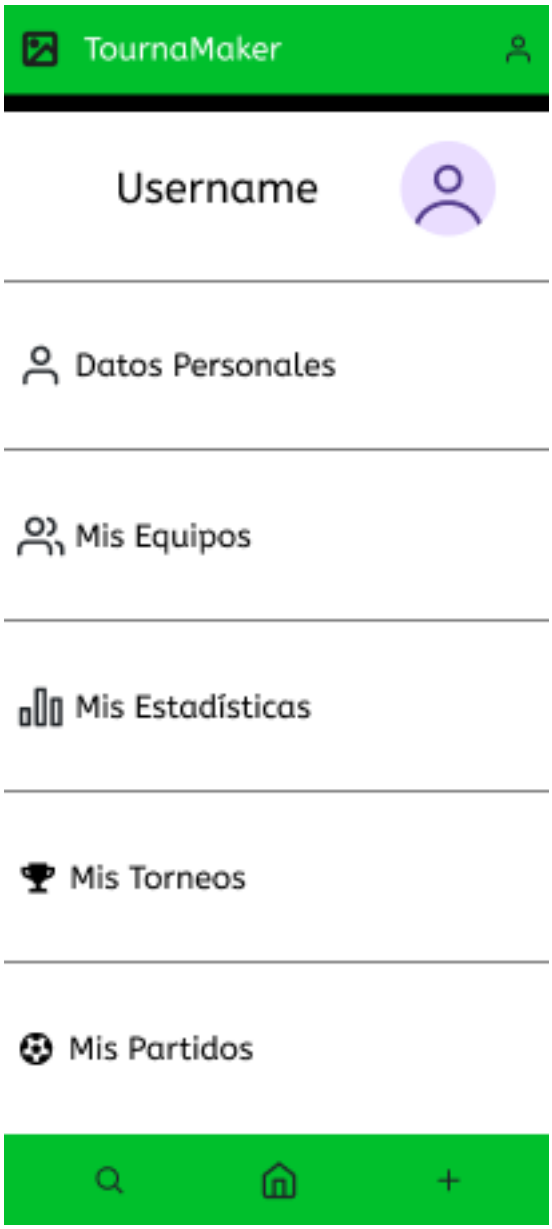
La comunicación entre vistas y *ViewModels* se apoya en mecanismos reactivos, de forma que los cambios en los datos (por ejemplo, la actualización de un torneo o la llegada de nuevas notificaciones) se reflejan automáticamente en la interfaz sin necesidad de que el usuario recargue manualmente la información. Las decisiones de diseño incluyen también la utilización de adaptadores personalizados para mostrar listas de torneos, partidos y equipos en componentes de tipo *RecyclerView*, lo que permite manejar de forma eficiente grandes volúmenes de elementos y aplicar diseños de tarjetas coherentes en toda la aplicación.

## 3.2. Prototipos/Mockups

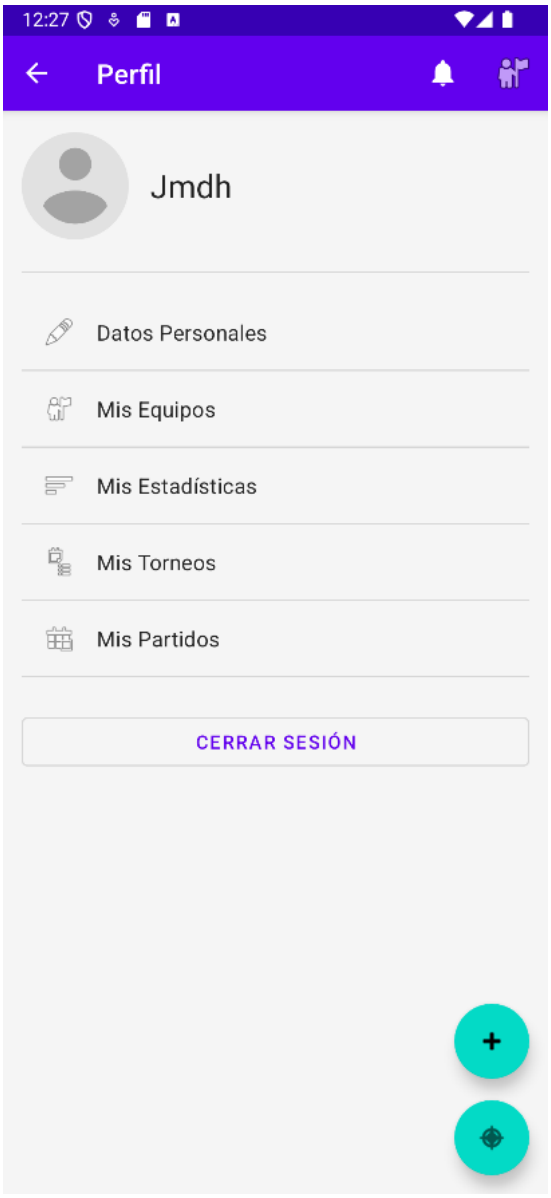
El proceso de diseño partió de una serie de *mockups* creados en Figma durante el desarrollo del prototipo web inicial, que incluían versiones para web, tablet y móvil. Estos bocetos se han reutilizado y refinado para la versión nativa de Android, realizando ajustes significativos en la estructura de navegación, la disposición de los componentes y la paleta de colores para mejorar la usabilidad y la accesibilidad. En particular, se han simplificado las cabeceras, reorganizado los menús y revisado los contrastes de color para cumplir mejor las recomendaciones de diseño de interfaces móviles.

A partir de los *mockups* actualizados se han definido mapas de navegación que describen las transiciones entre pantallas principales, incluyendo el flujo de registro e inicio de sesión, la navegación desde la pantalla principal a los listados de torneos, partidos y equipos, y el acceso a las vistas de detalle y al perfil de usuario. Estos prototipos han servido como guía durante la implementación, reduciendo la necesidad de cambios estructurales en fases avanzadas del desarrollo y asegurando una coherencia visual entre las distintas secciones de la aplicación.

En muchas de las pantallas como la de perfil se ha mantenido la estructura general propuesta en los *mockups* iniciales, sustituyendo la paleta verde por tonos violeta y ajustando el contraste y la jerarquía visual para mejorar la legibilidad y la accesibilidad, pero sin muchas modificaciones, sin embargo en otras pantallas como la de inicio de sesión si que ha cambiado un poco la estructura de algunas partes como se puede ver en las siguientes figuras.



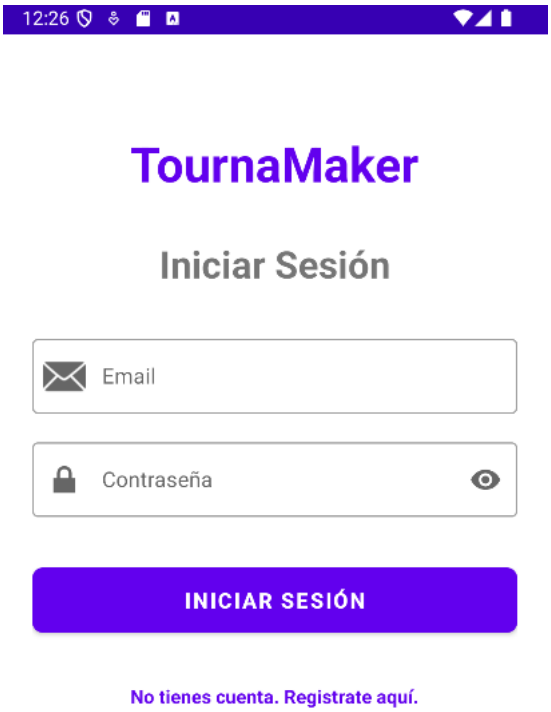

(a) Mockup inicial de perfil




(b) Pantalla actual de perfil en TournaMaker.

Figura 3.2: Evolución del diseño de la pantalla de perfil.


(a) Mockup inicial de inicio de sesión




(b) Pantalla actual de inicio de sesión en TournaMaker.

Figura 3.3: Evolución del diseño de la pantalla de inicio de sesión.

# Capítulo 4

# Arquitectura

La arquitectura de TournaMaker se ha diseñado siguiendo un enfoque por capas que separa claramente la interfaz de usuario, la lógica de presentación, la lógica de dominio y el acceso a datos. Este enfoque facilita el mantenimiento del código, la incorporación de nuevas funcionalidades y la reutilización de componentes en distintas partes de la aplicación. La elección de esta estructura responde también a la necesidad de trabajar con datos remotos almacenados en la nube mediante Firebase.

En la capa de presentación se encuentran las actividades y fragmentos de Android, responsables de mostrar la información y gestionar la interacción con el usuario. Cada pantalla principal (inicio, listas de torneos, partidos y equipos, detalle de torneo o partido, creación de entidades, perfil, etc.) se implementa mediante fragmentos especializados que se comunican con la capa de lógica de presentación a través de *ViewModels*. Estos fragmentos se centran en tareas como configurar adaptadores de listas, mostrar mensajes de error o éxito y reaccionar a cambios en el estado de la interfaz.

La capa de lógica de presentación está formada por los *ViewModels*, que actúan como intermediarios entre las vistas y los datos de la aplicación. Cada *ViewModel* expone al fragmento la información necesaria (por ejemplo, el torneo seleccionado, la lista de equipos disponibles o las notificaciones pendientes) y ofrece operaciones para gestionar acciones del usuario, como unirse a un torneo, crear un partido o actualizar el marcador. De esta forma, las vistas no necesitan conocer los detalles de cómo se obtienen o se almacenan los datos, sino únicamente sus representaciones listas para mostrar.

En la capa de acceso a datos se encuentran los repositorios, encargados de comunicarse con Firebase Firestore y, en su caso, con otros servicios de Firebase relacionados con la autenticación. Cada repositorio gestiona un conjunto de entidades (usuarios, equipos, torneos, partidos, notificaciones, etc.) y ofrece métodos de alto nivel para realizar operaciones como crear documentos, actualizar campos, consultar colecciones filtradas o escuchar cambios en tiempo real. Esta organización permite centralizar la lógica de acceso a datos y facilita la sustitución o ampliación del backend en el futuro sin afectar directamente a las vistas ni a los *ViewModels*.

La autenticación de usuarios se apoya en los servicios de Firebase, que permiten registrar cuentas, verificar correos electrónicos, gestionar inicios de sesión y recuperar contraseñas mediante enlaces enviados por correo. La información adicional de perfil y la estructura de torneos, partidos y equipos se almacena en colecciones de Firestore, lo que permite acceder a los datos desde distintos dispositivos y mantener sincronizada la información de forma transparente para el usuario. La combinación de esta infraestructura en la nube con la arquitectura MVVM facilita que la aplicación pueda escalar en número

de usuarios y volumen de competiciones sin cambios drásticos en el diseño del código y permite también seguir teniendo una alta seguridad de los datos delicados almacenados por el usuario, como por ejenplo su contraseña.

## 4.1. Diagrama de arquitectura

En la figura 4.1 se muestra un diagrama de alto nivel de la arquitectura de TournaMaker, donde se representan las principales capas y componentes de la aplicación, así como su relación con los servicios de Firebase. La capa de interfaz agrupa actividades y fragmentos, la capa de lógica de presentación está formada por los *ViewModels*, y la capa de datos incluye los repositorios y las colecciones de Firestore responsables de persistir usuarios, equipos, torneos, partidos y notificaciones.

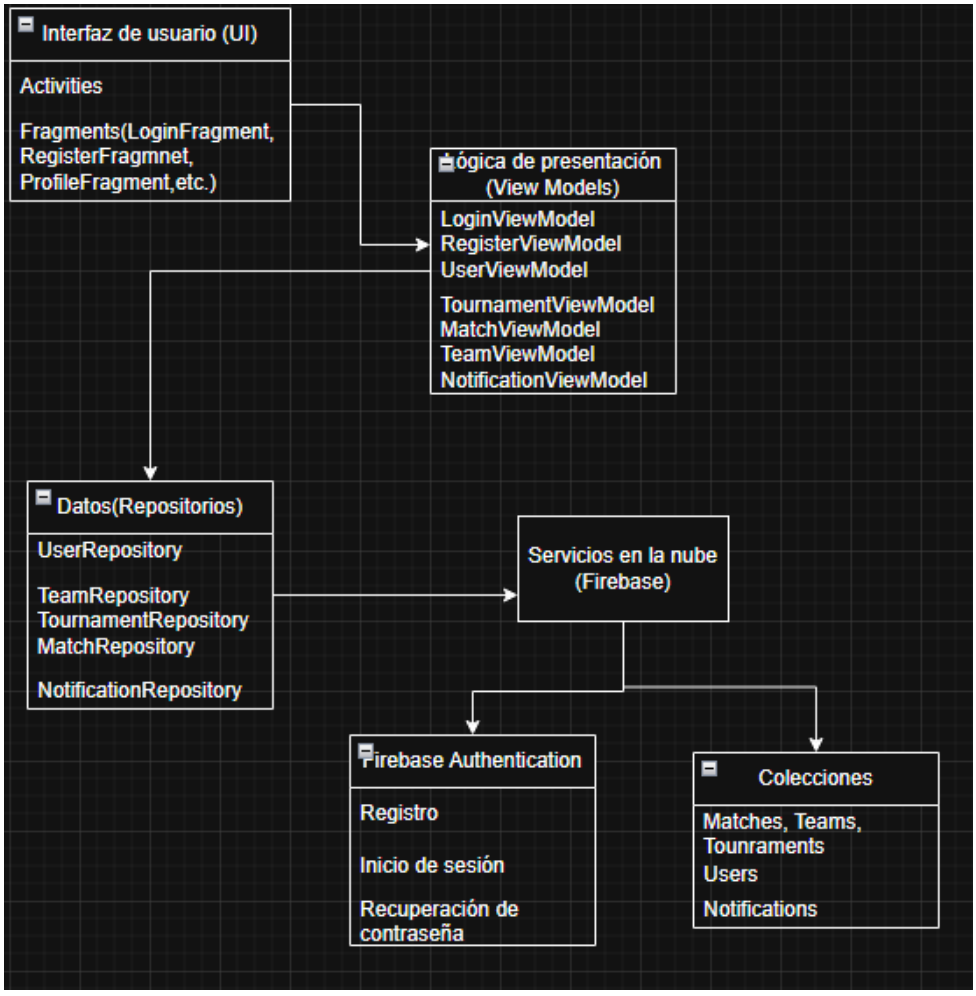



Figura 4.1: Diagrama de arquitectura de TournaMaker, mostrando las capas principales y su interacción con Firebase.

# Capítulo 5

# Modelo de Datos

El modelo de datos de TournaMaker se ha diseñado para representar de forma explícita las relaciones entre usuarios, equipos, partidos, torneos y notificaciones. Dado que la aplicación se apoya en Firebase Firestore, la información se organiza en colecciones y documentos en lugar de en tablas relacionales tradicionales, pero la estructura lógica mantiene conceptos similares a entidades y relaciones. El objetivo principal del modelo es permitir que un mismo usuario pueda participar en múltiples equipos, torneos y partidos, y que un equipo pueda intervenir en distintos eventos a lo largo del tiempo.

La entidad **Usuario** almacena los datos básicos de cada persona registrada en la aplicación, incluyendo identificador único, nombre de usuario y correo electrónico verificado. A partir de esta entidad se derivan las relaciones con equipos, torneos y partidos, ya sea como organizador o como participante. Cada usuario puede ser propietario de varios equipos, crear torneos y partidos, unirse a competiciones con sus equipos y recibir notificaciones cuando se unen otros usuarios o equipos a cada uno de los torneos, partidos y equipos en los que es organizador.

La entidad **Equipo** representa un conjunto de jugadores gestionado por un usuario propietario. En el contexto actual de la aplicación, no se almacenan todos los datos individuales de cada jugador, lo que permite usar equipos tanto reales como ficticios para torneos rápidos. Cada equipo contiene información como nombre, descripción, posible escudo o imagen asociada y referencia al usuario que actúa como organizador. La relación entre usuarios y equipos es de tipo uno a muchos (un usuario puede gestionar varios equipos), y los equipos se relacionan posteriormente con partidos y torneos a través de sus identificadores.

La entidad **Torneo** describe una competición compuesta por varias rondas o eliminatorias. Entre sus atributos se incluyen nombre, descripción, fecha y lugar de celebración, premio previsto, tasa de inscripción y número máximo de equipos participantes. Además, cada torneo referencia al usuario organizador y mantiene el estado de la competición (pendiente, en curso, finalizada). La relación entre torneos y equipos se materializa mediante listas de identificadores de equipos inscritos y, en su caso, estructuras adicionales que representan el *bracket* o cuadro de eliminatorias.

La entidad **Partido** representa un enfrentamiento individual entre dos equipos dentro o fuera de un torneo. Cada partido almacena información sobre los equipos implicados, la fecha y hora prevista, el lugar de celebración y el estado del marcador, que puede actualizarse en tiempo real o al instante según lo que quiera el usuario. En el modelo actual, los partidos pueden estar asociados a un torneo concreto o definirse de manera independiente para encuentros amistosos. La relación entre torneos y partidos es de tipo

uno a muchos, ya que un torneo puede estar formado por varios partidos distribuidos en distintas rondas.

Para gestionar la comunicación entre organizadores y participantes se utiliza la entidad auxiliar **Notificación**. Las notificaciones de unión registran qué equipo se ha unido a qué torneo o partido y qué usuario se ha unido a qué equipo, junto con la fecha de la unión. Esto permite que los organizadores reciban avisos cuando se incribe un equipo a su torneo o partido, o en el que caso de un equipo si un usuario se ha unido al mismo.

## 5.1. Esquemas

En la figura 5.1 se muestra un esquema simplificado del modelo de datos de TournaMaker, donde se representan las principales entidades y sus relaciones lógicas. Los usuarios se relacionan con equipos, torneos y partidos tanto como organizadores como participantes; los equipos pueden estar vinculados a múltiples torneos y partidos; y las notificaciones actúan como elementos intermedios que registran las interacciones entre estos componentes.

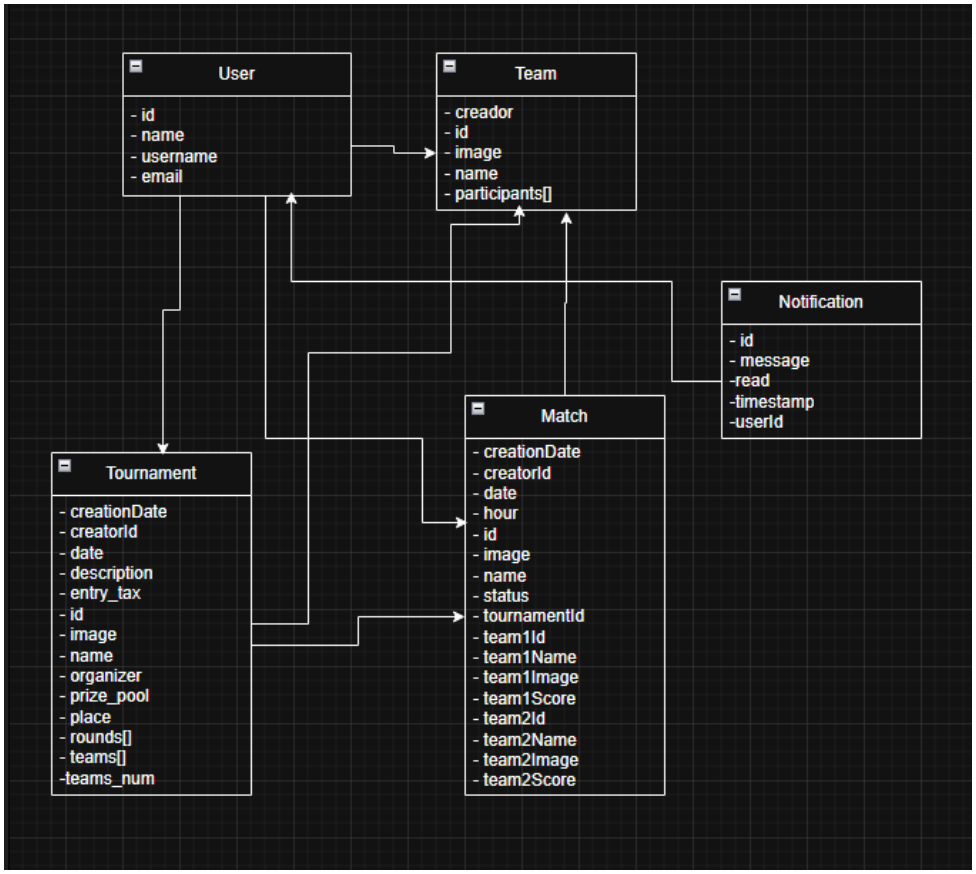



Figura 5.1: Esquema del modelo de datos de TournaMaker, mostrando las entidades principales y sus relaciones.

# Capítulo 6

# Desarrollo

El desarrollo de TournaMaker se ha llevado a cabo utilizando Android Studio como entorno principal de trabajo y el lenguaje Kotlin para la implementación de la lógica de la aplicación. El proyecto se estructura siguiendo las recomendaciones habituales para aplicaciones Android modernas, organizando el código en paquetes según responsabilidades: capa de datos (modelos y repositorios), capa de lógica de presentación (ViewModels) y capa de interfaz de usuario (activities y fragments). Esta organización facilita tanto la navegación por el código como la localización de componentes concretos durante las fases de implementación y mantenimiento.

En cuanto a dependencias, la aplicación se apoya en las bibliotecas oficiales de Android Jetpack, incluyendo componentes de arquitectura como ViewModel y LiveData, así como Navigation para gestionar los flujos entre pantallas mediante un grafo de navegación. Además, se utilizan RecyclerView y adaptadores personalizados para mostrar listas de torneos, partidos y equipos, junto con otras librerías de soporte habituales para gestionar compatibilidad, diseño de interfaces y acceso a recursos. Para la capa de datos y la autenticación se integran los SDK de Firebase Authentication y Firebase Firestore, que permiten gestionar el ciclo de vida de las cuentas de usuario y almacenar la información de la aplicación en la nube.

La estructura del proyecto en Android Studio refleja esta separación en capas. Dentro del paquete principal se distinguen subpaquetes como `data` (modelos de dominio y repositorios), `viewmodel` (clases ViewModel encargadas de la lógica de presentación) y `ui` (dividido a su vez en secciones como `auth`, `main`, `team`, `tournament`, `match` o `notification`, cada una con sus fragmentos correspondientes). Esta organización permite que cada funcionalidad de la aplicación tenga asociados de forma clara sus fragmentos, ViewModels y repositorios, reduciendo la duplicación de código y facilitando la reutilización de componentes entre diferentes pantallas.

Durante el desarrollo se han seguido iteraciones cortas en las que primero se definía el comportamiento esperado a partir de el rediseño de los *mockups* y, a continuación, se implementaban los fragmentos y ViewModels necesarios para dar soporte a ese flujo concreto. Una vez que la funcionalidad básica quedaba operativa (por ejemplo, creación de un torneo, unión de un equipo o actualización del marcador de un partido), se refinaba la interfaz para mejorar la usabilidad y se añadían detalles como mensajes de error, validaciones de formularios o notificaciones al usuario. Este enfoque incremental ha permitido ir validando el comportamiento de la aplicación de forma continua y ajustar decisiones de diseño a medida que se probaban los flujos reales.

## 6.1. Estrategia de pruebas

La estrategia de pruebas se ha centrado principalmente en pruebas funcionales manuales sobre las distintas pantallas y flujos de la aplicación, ejecutadas en emuladores de Android. Para cada funcionalidad principal (registro y verificación de cuenta, inicio de sesión, creación y edición de equipos, torneos y partidos, unión a competiciones, gestión de notificaciones y actualización de marcadores) se han definido casos de prueba informales que comprueban el comportamiento esperado en situaciones normales y ante entradas incorrectas, como campos obligatorios vacíos o combinaciones de datos inconsistentes.

Además de estas pruebas funcionales, se han realizado comprobaciones específicas relacionadas con la integración con Firebase, verificando que las operaciones de registro, autenticación y acceso a Firestore se comportan correctamente en distintos escenarios, incluyendo errores de conexión o credenciales inválidas. También se han probado las restricciones de negocio más importantes, como la imposibilidad de iniciar un torneo o un partido mientras no se alcance el número mínimo de equipos requeridos, y el correcto avance de los equipos en el cuadro de eliminatorias según los resultados introducidos. Aunque no se ha desplegado una batería exhaustiva de pruebas unitarias automatizadas, la separación en capas y el uso de ViewModels y repositorios deja preparada la base para incorporar este tipo de pruebas en futuras iteraciones del proyecto.

## 6.2. Distribución de casos de uso

Tabla 6.1: Resumen de casos de uso, prioridad y esfuerzo dedicado.

| **Caso de uso** | **Prioridad** | **Descripción** | **Tiempo estimado** | **Tiempo real** | **Responsable principal** |
|---|---|---|---|---|---|
| Registro de usuario y verificación de correo | Must | El usuario crea una cuenta introduciendo sus datos y recibe un correo para verificarla antes de poder acceder a la aplicación. | 6 h | 10 h | José Manuel Díaz Hernández |
| Inicio de sesión y cierre de sesión | Must | El usuario inicia sesión con correo y contraseña, y puede cerrar sesión desde la pantalla de perfil para proteger su cuenta. | 4 h | 8 h | José Manuel Díaz Hernández |

Continúa en la siguiente página

| Caso de uso | Prioridad | Descripción | Tiempo estimado | Tiempo real | Responsable principal |
|---|---|---|---|---|---|
| Gestión de perfil de usuario | Should | El usuario consulta y actualiza sus datos personales básicos y accede a la opción de cambio de contraseña. | 4 h | 7 h | José Manuel Díaz Hernández |
| Cambio y recuperación de contraseña | Should | El usuario solicita un correo para restablecer la contraseña y completa el proceso siguiendo el enlace enviado por Firebase. | 4 h | 7 h | José Manuel Díaz Hernández |
| Creación y edición de equipos | Must | El usuario crea equipos nuevos, ajusta nombre, descripción e imagen y actúa como organizador de dichos equipos. | 6 h | 11 h | José Manuel Díaz Hernández |
| Listado y búsqueda de equipos | Should | El usuario consulta el listado de equipos disponibles, filtra y accede al detalle de cada equipo para ver su información. | 3 h | 6 h | Francisco Malillos Castellano |
| Creación y edición de torneos | Must | El organizador define torneos indicando nombre, descripción, fechas, lugar, premio, tasa de inscripción y número máximo de equipos. | 8 h | 14 h | José Manuel Díaz Hernández |
| Creación y edición de partidos | Must | El organizador crea partidos individuales, elige los equipos implicados y configura fecha, hora y lugar. | 6 h | 11 h | José Manuel Díaz Hernández |

Continúa en la siguiente página

| Caso de uso | Prioridad | Descripción | Tiempo estimado | Tiempo real | Responsable principal |
|---|---|---|---|---|---|
| Unión de equipos a torneos y partidos | Must | El usuario selecciona uno de sus equipos para solicitar la participación en un torneo o partido determinado. | 5 h | 9 h | José Manuel Díaz Hernández |
| Generación y actualización del bracket de torneo | Should | La aplicación genera automáticamente el cuadro de eliminatorias con los equipos inscritos y actualiza el avance según los resultados. | 7 h | 13 h | José Manuel Díaz Hernández |
| Actualización del marcador de partidos | Must | El organizador inicia un partido, modifica el marcador en tiempo real o al finalizar y marca el encuentro como terminado. | 5 h | 9 h | José Manuel Díaz Hernández |
| Pantalla de inicio con torneos, partidos y equipos destacados | Must | El usuario ve en la pantalla principal un resumen de torneos, partidos y equipos disponibles, con accesos a los listados completos. | 5 h | 10 h | Francisco Malillos Castellano |
| Listados completos de torneos, partidos y equipos | Must | El usuario navega a pantallas específicas donde se muestran todos los torneos, partidos o equipos, con tarjetas y acciones asociadas. | 5 h | 9 h | Francisco Malillos Castellano |

Continúa en la siguiente página

| Caso de uso | Prioridad | Descripción | Tiempo estimado | Tiempo real | Responsable principal |
|---|---|---|---|---|---|
| Notificaciones de actividad | Should | El organizador recibe notificaciones cuando un equipo se une a un torneo, partido o un usuario se une a un equipo y el usuario ve el historial de avisos. | 4 h | 8 h | José Manuel Díaz Hernández |
| Pantallas de autenticación y perfil (diseño frontend) | Could | Se implementan las pantallas de login, registro y perfil siguiendo los diseños de Figma y adaptándolos a la paleta final violeta/-blanco. | 4 h | 7 h | Francisco Malillos Castellano |
| Navegación principal y menú | Should | Se configura la navegación entre fragmentos (inicio, listas, detalles, perfil) utilizando el grafo de navegación y barras de acción. | 4 h | 8 h | Francisco Malillos Castellano |
| Pantalla de estadísticas de usuario | Could | El usuario consulta un resumen del número de torneos, partidos y equipos en los que participa como jugador u organizador. | 3 h | 6 h | José Manuel Díaz Hernández |
| Mejora de accesibilidad y ajustes visuales | Could | Se revisan contrastes de color, tamaños de fuente y disposición de componentes para mejorar la legibilidad en dispositivos móviles. | 3 h | 6 h | Francisco Malillos Castellano |

Continúa en la siguiente página

| Caso de uso | Prioridad | Descripción | Tiempo estimado | Tiempo real | Responsable principal |
|---|---|---|---|---|---|
| Refactorización y corrección de errores | Must | Se unifica la estructura de paquetes, se elimina código duplicado y se corrigen errores detectados durante las pruebas manuales. | 6 h | 12 h | José Manuel Díaz Hernández |

# Capítulo 7

# Interfaces

Las interfaces de TournaMaker se han diseñado para ofrecer un acceso rápido a las funcionalidades principales de la aplicación, manteniendo una estructura coherente entre pantallas. La navegación se organiza en torno a una pantalla de inicio que muestra un resumen de torneos, partidos y equipos disponibles, junto con accesos a los listados completos y a las acciones de creación. Desde esta pantalla el usuario puede abrir el menú de notificaciones, acceder a su perfil y acceder a la creación tanto de equipos, como de torneos o partidos, lo que convierte a la vista inicial en el punto central de entrada a la aplicación, tal cual se muestra en la Figura 3.1b.

Las pantallas destinadas a la autenticación incluyen vistas específicas para el inicio de sesión y el registro de nuevos usuarios. En la pantalla de inicio de sesión se muestran campos para correo electrónico y contraseña, acompañados de un botón principal para acceder a la aplicación y un enlace secundario para pasar al registro en caso de no tener cuenta. La pantalla de registro amplía estos campos con información adicional como nombre de usuario y nombre completo, y guía al usuario de vuelta al inicio de sesión una vez completado el proceso. Ambas pantallas comparten la misma paleta de colores violeta y blanco, lo que refuerza la consistencia visual de la identidad de la aplicación, como se puede observar en la Figura 3.3b y en la Figura 7.1

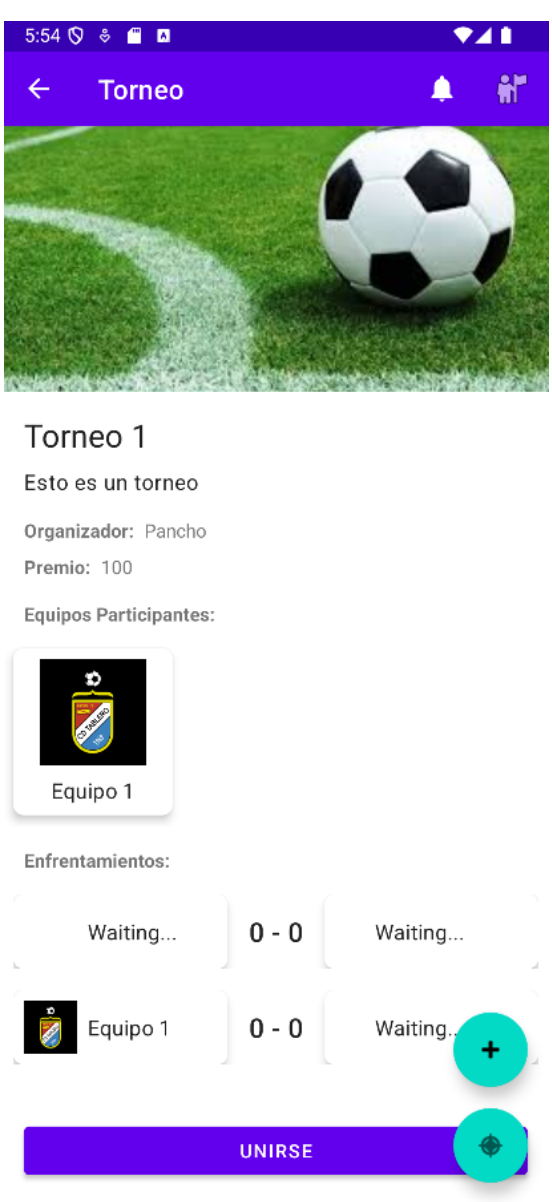

Figura 7.1: Pantalla actual de registro en TournaMaker.

La sección de perfil permite al usuario consultar y gestionar su información personal y su actividad dentro de la aplicación. Desde esta pantalla se accede a opciones como la edición de datos personales, la gestión de equipos propios, la consulta de estadísticas de participación y el acceso rápido a los torneos y partidos en los que interviene. También se incluye la opción de cerrar sesión. El diseño de esta vista se basa en una lista de entradas con iconos representativos, lo que facilita la identificación de cada apartado y reduce el tiempo necesario para localizar una funcionalidad concreta. Esto se puede ver en la Figura 3.2b

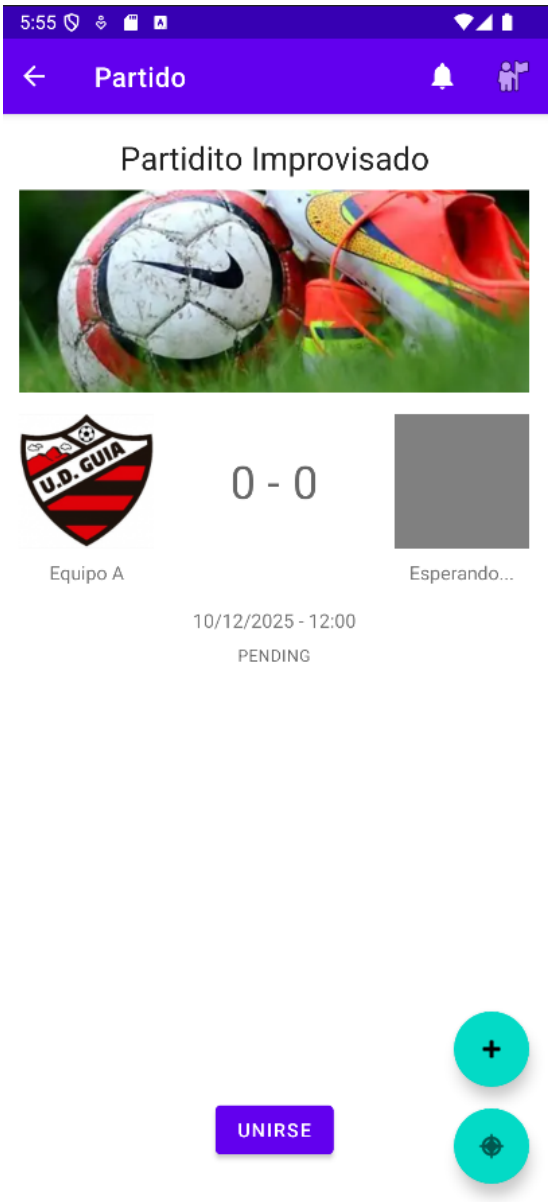
Las pantallas de listado de torneos, partidos y equipos presentan la información agrupada en tarjetas dentro de listas desplazables. Cada tarjeta muestra los datos más relevantes (por ejemplo, nombre del torneo, descripción corta, fecha y lugar, o nombre y escudo del equipo) junto con acciones contextuales como ver detalles o solicitar la participación. Desde los listados se puede navegar a vistas de detalle, donde se ofrece información ampliada y botones específicos para unirse a un evento con uno de los equipos del usuario. Este diseño basado en tarjetas permite reutilizar el mismo patrón visual en distintas

secciones de la aplicación.

En la vista de detalle de torneo se muestra la información completa del evento, incluyendo su descripción, fechas clave, localización, premio, tasa de inscripción y número de plazas disponibles. Cuando se trata de torneos por eliminatorias, la pantalla incorpora además una representación del cuadro de enfrentamientos, donde se reflejan los equipos que participan y su avance según los resultados de cada partido. La vista de detalle de partido ofrece una estructura similar, centrada en la información de los dos equipos implicados, la fecha y hora del encuentro y el marcador, que puede ser actualizado por el organizador durante el partido. Tanto en la Figura 7.2a como en la Figura 7.2b se puede ver tanto la visulización de los detalles de un torneo como la de un partido.




(a) Pantalla de visualización de un torneo.




(b) Pantalla de Visualización de un partido.

Figura 7.2: Diseño de visualización de los detalles de un torneo y un partido.

Por último también existe la pantalla de notificaciones en la que el usuario podría ver todo el historial de los últimos usuarios o equipos que se han unido a torneos, partidos

o equipos que el organiza, para tener un mayor control de como van estos, y saber si ya están las plazas o si ya va todo correctamente. Además se incluye que cuando las notificaciones llegan al usuario le salta un adevertencia encima del icono de notificacuión de cuántas de ellas hay nuevas, o no ha leído. El formato de la pantalla de notificación es totalmente nueva, ya que esta funcionalidad no había pensado en el proyecto que empezó el año pasado y se puede ver como ha quedado en la Figura 7.3


Figura 7.3: Pantalla de notificaciones.

# Accesibilidad

En el diseño de las interfaces se han tenido en cuenta criterios básicos de accesibilidad para mejorar la experiencia de usuarios con distintas necesidades y en diferentes condiciones de uso. Se ha revisado la paleta de colores inicial, basada en tonos verdes, sustituyéndola por una combinación de violeta y blanco con mayor contraste, lo que facilita la lectura del texto y la identificación de botones y elementos interactivos. Asimismo, se han ajustado tamaños de fuente y márgenes para garantizar que los contenidos principales resulten legibles en pantallas de distintos tamaños.

La aplicación utiliza textos claros en los botones y mensajes, evitando abreviaturas ambiguas y proporcionando retroalimentación inmediata cuando se completa una acción relevante, como la creación de un torneo o la unión a un equipo, partido o torneo. Aunque en la versión actual solo se soporta el idioma español, la estructura de recursos de la aplicación está preparada para incorporar otros idiomas en futuras versiones mediante el uso de ficheros de cadenas localizadas. Estos aspectos sientan las bases para seguir mejorando la accesibilidad y la internacionalización en evoluciones posteriores del proyecto.

# Capítulo 8

# Conclusiones

El desarrollo de TournaMaker ha permitido diseñar e implementar una aplicación móvil nativa capaz de gestionar torneos y partidos de manera flexible, cubriendo tanto escenarios informales entre amigos como competiciones con un mayor grado de organización. A lo largo del proyecto se han definido y construido las funcionalidades principales necesarias para este tipo de aplicación: registro y autenticación de usuarios, creación y administración de equipos, torneos y partidos, notificaciones de unión, generación de cuadros de eliminatorias y actualización de marcadores. Todo ello se ha integrado en una interfaz coherente que unifica en una sola herramienta tareas que habitualmente requerían combinar varias aplicaciones distintas.

Desde el punto de vista técnico, el uso de Kotlin, la arquitectura basada en el patrón MVVM y la integración con Firebase Authentication y Firestore han permitido estructurar la aplicación en capas bien diferenciadas y apoyarse en servicios en la nube para el almacenamiento y la sincronización de datos. Esta organización ha facilitado la evolución progresiva del proyecto, permitiendo incorporar nuevas funcionalidades sin necesidad de reestructurar el código por completo. El trabajo realizado sobre el diseño de la interfaz, pasando de los mockups iniciales en Figma a las pantallas finales en Android, ha supuesto además una mejora significativa en términos de accesibilidad, consistencia visual y claridad en los flujos de navegación.

El proyecto presenta, no obstante, algunas limitaciones. En la versión actual no se gestionan todavía estadísticas detalladas de jugadores y equipos (por ejemplo, goleadores, asistencias o tarjetas), ni se ofrece un módulo avanzado de configuración de normas específicas para cada deporte o disciplina. Asimismo, aunque la aplicación soporta de forma genérica distintos tipos de torneos y partidos, aún no se han implementado variantes de competición más complejas, como ligas largas con fases de grupos o sistemas de puntuación personalizados. Tampoco se han desarrollado pruebas automatizadas extensivas, por lo que la validación se ha basado principalmente en pruebas manuales y en la detección progresiva de errores durante el uso.

A pesar de estas limitaciones, la arquitectura y el modelo de datos diseñados dejan el proyecto en una buena posición para su ampliación futura. Entre las líneas de trabajo más interesantes se encuentran la incorporación de estadísticas avanzadas por jugador y por equipo, la generación automática de informes de torneos, la inclusión de más tipos de formatos de competición y la internacionalización de la aplicación mediante la incorporación de nuevos idiomas. También resultaría valioso reforzar la estrategia de pruebas mediante suites automatizadas que cubran los casos de uso más críticos y la creación de mecanismos de análisis de uso que permitan identificar qué funcionalidades son más utilizadas por los

usuarios.

En conjunto, TournaMaker demuestra que es posible ofrecer en una sola aplicación móvil un soporte completo para la organización y seguimiento de torneos y partidos en contextos muy diversos. El trabajo realizado ha servido para aprender conocimientos sobre desarrollo nativo en Android, diseño de interfaces centradas en el usuario y uso de servicios en la nube, y ha dejado abiertas múltiples posibilidades de evolución que podrían convertir la aplicación en una solución aún más completa y versátil para la gestión de competiciones deportivas y otros tipos de eventos basados en eliminatorias o rondas, además de que da un abanico más grande para en futuras aplicaciones usar este mismo sistema que tanto nos ha gustado y servido.

# Bibliografía


[1] Android Developers, *Guía de arquitectura para apps Android*, 2025. https://developer.android.com/topic/architecture.

[2] Google LLC, *Firebase Authentication*, 2025. https://firebase.google.com/docs/auth.

[3] Google LLC, *Cloud Firestore*, 2025. https://firebase.google.com/docs/firestore.

[4] Google LLC, *IA Perplexity*, 2025. https://www.perplexity.ai/.

[5] Google LLC, *Material Design Guidelines*, 2025. https://m3.material.io/.