



MÓDULO PROYECTO

CFGS Desarrollo de Aplicaciones

Multiplataforma

Informática y Comunicaciones

PappDEL – Gestión de ligas deportivas

Tutor individual: José María Rojo Zumel

Tutor colectivo: Cristina Pardo Silván

Año: 2024

Fecha de presentación: 17/06/2024

Nombre y Apellidos: Raúl Pérez de la Calle

Email: raulpdelacalle@gmail.com

Índice

1	Identificación proyecto	4
2	Organización de la memoria.....	4
3	Descripción general del proyecto	5
3.1	Objetivos	6
3.2	Cuestiones metodológicas	7
3.3	Entorno de trabajo (tecnologías de desarrollo y herramientas).....	8
4	Descripción general del producto	10
4.1	Visión general del sistema: límites del sistema, funcionalidades básicas, usuarios y/o otros sistemas con los que pueda interactuar.	10
4.2	Descripción breve de métodos, técnicas o arquitecturas(m/t/a) utilizadas.....	12
4.3	Despliegue de la aplicación indicando plataforma tecnológica, instalación de la aplicación y puesta en marcha.....	14
5	Planificación y presupuesto	14
6	Documentación Técnica: análisis, diseño, implementación y pruebas.	19
6.1	Especificación de requisitos	19
6.2	Análisis del sistema.....	21
6.3	Diseño del sistema:.....	21
6.3.1	Diseño de la Base de Datos.....	21
6.3.2	Diseño de la Interfaz de usuario.....	26
6.3.3	Diseño de la Aplicación.	30
6.4	Implementación:.....	35
6.4.1	Entorno de desarrollo.....	35
6.4.2	Estructura del código.	35
6.4.3	Cuestiones de diseño e implementación reseñables.....	¡Error! Marcador no definido.
6.5	Pruebas.	41
7	Manuales de usuario	45
7.1	Manual de usuario	45
7.2	Manual de instalación.....	50
8	Conclusiones y posibles ampliaciones	52

9	Bibliografía.....	54
10	Anexos	54

1 Identificación proyecto

El proyecto PappDEL tiene como objetivo principal facilitar la organización y gestión de torneos y ligas de pádel a través de una plataforma digitalizada.

Esta aplicación permite a los usuarios crear y administrar torneos, gestionar equipos y jugadores, registrar partidos y resultados, y proporcionar una interfaz intuitiva para la visualización de estadísticas y seguimiento de eventos en tiempo real. Utilizando tecnologías modernas y herramientas de desarrollo ágil, el proyecto busca mejorar la experiencia tanto para los organizadores de torneos como para los jugadores, promoviendo así la práctica y competencia del pádel de manera eficiente y accesible.

2 Organización de la memoria

La estructura de esta memoria está diseñada para proporcionar una visión completa del proyecto de la aplicación de gestión de una liga de pádel, desde la conceptualización inicial hasta la implementación y despliegue final.

La **Descripción General del Proyecto** establece los fundamentos del proyecto, detallando los objetivos, las cuestiones metodológicas y el entorno de trabajo, incluyendo las tecnologías y herramientas utilizadas. Este apartado es importante para entender el contexto y los objetivos del proyecto, así como las metodologías y tecnologías adoptadas para su desarrollo.

La **Descripción General del Producto** proporciona de forma detallada las funcionalidades básicas, y las interacciones con los usuarios y otros sistemas. También se incluyen descripciones breves sobre tecnologías y técnicas de desarrollo implementadas, así como detalles sobre el despliegue de la aplicación y la puesta en marcha para comprender la estructura y arquitectura del proyecto.

La sección de **Documentación Técnica** es uno de los componentes más extensos y detallados, abordando el análisis, diseño, implementación y pruebas del sistema. Dentro de esta sección, se presta especial atención a la especificación de requisitos, el análisis y diseño del sistema (incluyendo la base de datos, la interfaz de usuario y la *API*), así como la implementación y las pruebas sobre estas tres capas.

Manuales de Usuario y el **Manual de Instalación**, que proporcionan instrucciones claras de uso.

La memoria concluye con una sección dedicada a las **Conclusiones y Posibles Ampliaciones**, donde se reflexiona sobre los resultados obtenidos y se proponen posibles mejoras o expansiones futuras del sistema, destacando el potencial de evolución del proyecto dentro del entorno de este deporte en actual crecimiento global.

3 Descripción general del proyecto

Se trata de una aplicación móvil que permite tanto la gestión de una competición deportiva de pádel por equipos, y a su vez da un servicio informativo a los participantes.

Permite visualizar información sobre sus equipos a los participantes de la liga y a su vez, facilita a los administradores de la competición una herramienta para controlar fácilmente la competición.

Se ha desarrollado a través de la tecnología *Flutter*, framework multiplataforma creado por *Google*, , que permite crear aplicaciones nativas de alta calidad para *Android* e *iOS* desde una única base de código. *Flutter* ha sido seleccionado por su capacidad para ofrecer una experiencia de usuario fluida y un diseño atractivo.

3.1 *Objetivos*

El principal objetivo del proyecto es crear una aplicación móvil innovadora y eficiente que permita gestionar una liga de pádel de manera integral, ofreciendo una experiencia optimizada tanto para los administradores de la competición como para los jugadores.

Los objetivos específicos son los siguientes:

1. **Intuitividad y Funcionalidad:** Desarrollar una interfaz de usuario que sea completamente intuitiva, facilitando la navegación y el uso de todas las funcionalidades disponibles, independientemente del perfil del usuario (administrador o jugador).
2. **Acceso a Información para Jugadores:** Ofrecer a los jugadores un acceso fácil y rápido a toda la información relevante sobre su equipo y la liga, incluyendo calendarios de partidos, resultados, clasificaciones, estadísticas personales y del equipo.
3. **Mejora del Rendimiento:** Garantizar que la aplicación tenga un rendimiento óptimo, reduciendo tiempos de carga y asegurando una experiencia de usuario fluida y sin interrupciones.
4. **Interfaz Atractiva y Moderna:** Diseñar una interfaz gráfica moderna y atractiva que se diferencie de las aplicaciones existentes, aportando una sensación de frescura y modernidad que atraiga a los usuarios y mejore su satisfacción.
5. **Escalabilidad y Futuro:** Desarrollar la aplicación con una arquitectura escalable que permita añadir nuevas funcionalidades y adaptarse a las necesidades futuras, facilitando su evolución hacia un producto comercializable y con potencial de crecimiento en el mercado deportivo.

3.2 Cuestiones metodológicas

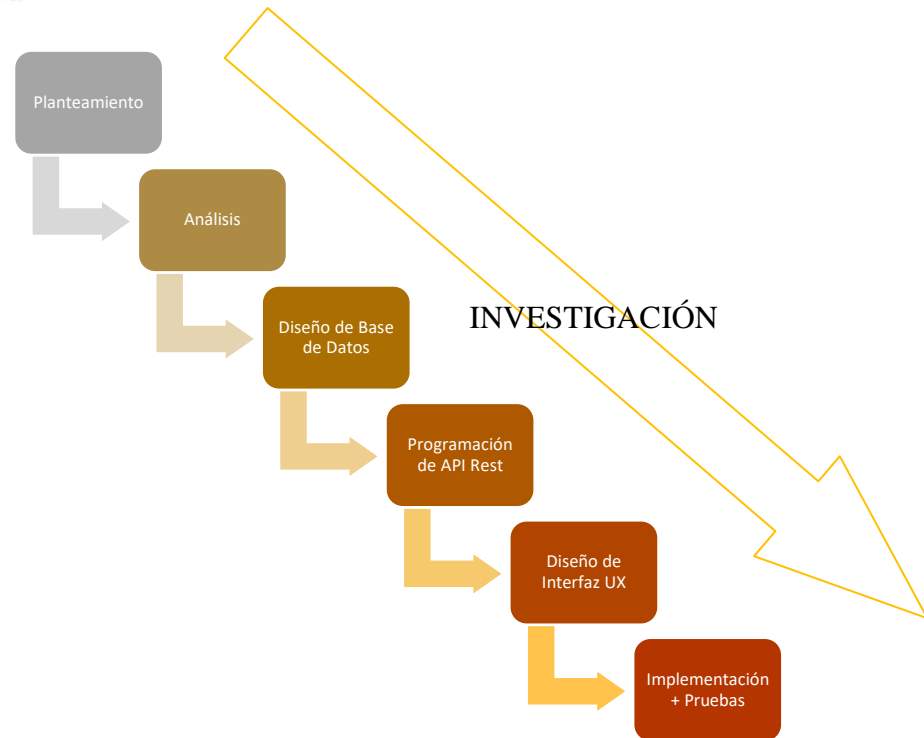
Al no tratarse de un equipo de trabajo, la metodología de trabajo ha sido libre, no se ha utilizado ningún método de trabajo en específico, ya he repartido las tareas en el tiempo por orden de conveniencia.

Se ha utilizado una mezcla de “modelo de cascada” y “gestión por cadena crítica” (CPM). Ya que los comienzos pasan por analizar y plantear el funcionamiento que se va a querer para después llevarlo a cabo de una forma ordenada, también he focalizado algunos puntos críticos que he querido realizar con antelación por encima de otros, por ejemplo, mientras me encontraba en el diseño de la base de datos he ido planteando las características de diseño de cada pantalla.

Durante todo el planteamiento y desarrollo se ha realizado una fase transversal que une cada uno de los puntos del modelo de cascada, la investigación. Ha sido indispensable tanto para reforzar conocimientos ya adquiridos, como para agregar funcionalidades en forma de widgets a través de Flutter. También se ha investigado la posibilidad de desplegar la base de datos junto a la API en un servidor para pasar el proyecto de local a remoto.

Durante la fase de “Implementación + pruebas” se ha revisado el modelo de base de datos y se han modificado algunas características. También se han revisado y añadido los *endpoints* de la API para adaptar la funcionalidad de la aplicación, sobre todo para que las peticiones *HTTP* funcionen de una manera correcta.

Tras finalizar la última fase se han retocado aspectos visuales del diseño de interfaz UX.



3.3 Entorno de trabajo (tecnologías de desarrollo y herramientas)

Al tratarse de un proyecto basado en una aplicación móvil, para el desarrollo y pruebas se ha utilizado:

- Emulador de dispositivo Android (A través de *Android Studio*).
- Dispositivo Android físico (Xiaomi MiA1).

En el entorno de trabajo se han utilizado tres IDE's:

- **Visual Studio Code**: este IDE se ha utilizado durante todo el desarrollo de la aplicación mediante el framework *Flutter*.

Ha sido elegido por su compatibilidad, ligereza y versatilidad mediante plugins.

- **IntelliJ**: para desarrollar la *API Rest* con el framework *SpringBoot*.
- **Android Studio**: principalmente utilizado para emular dispositivos android en los que realizar pruebas de la aplicación.

Tecnologías de desarrollo:

- **Flutter**: es un framework multiplataforma creado por Google que permite el desarrollo de aplicaciones nativas de alta calidad para Android e iOS desde una única base de código. Lo he seleccionado por su capacidad para ofrecer una experiencia de usuario fluida y un diseño atractivo.
- **MySQL**: sistema de gestión de bases de datos relacional de código abierto, utilizado por su rendimiento, fiabilidad y escalabilidad, lo que permite manejar eficientemente grandes volúmenes de datos, crucial para la gestión de información de equipos, partidos y resultados en la aplicación de pádel.
- **SpringBoot**: framework de desarrollo para aplicaciones Java utilizado para construir la *API Rest* de la aplicación.
- **Docker**: plataforma de contenedorización que permite a los desarrolladores empaquetar aplicaciones y sus dependencias en contenedores. Se utilizó para realizar pruebas de despliegue de la base de datos y *API* en servidor remoto.



Al tratarse de un entorno de desarrollo, el proyecto se ha llevado a cabo íntegramente en un entorno de trabajo local. Durante el proceso, se ha utilizado un servidor de base de datos *MySQL* en *localhost*. Paralelamente, se ha desplegado y

ejecutado la *API Rest* desarrollada con *SpringBoot* en un servidor *Tomcat* también configurado en *localhost*.

Esta configuración local ha permitido tener un control total sobre el entorno de desarrollo, facilitando la realización de pruebas exhaustivas y la evolución en el desarrollo de nuevas funcionalidades.

Además, se ha utilizado *Docker* para realizar pruebas de despliegue, garantizando que la transición de un entorno local a uno de producción sea fluida y fiel a la configuración con la que se ha trabajado durante el desarrollo.

Se ha utilizado la herramienta *MySQL WorkBench* para la creación del diseño de la base de datos a través de diagramas de tablas y para gestionar la base de datos de forma visual. Es una gran ayuda ya que permite un diseño, desarrollo y gestión muy eficaz y organizado gracias a la capacidad de exportar diagramas y esquemas de base de datos (para documentar el proyecto) y su entorno visual, a diferencia de la línea de comandos.

4 Descripción general del producto

La idea nace a raíz de mi experiencia personal participando como jugador en ligas o competiciones de pádel, interactuando con las aplicaciones que cada competición pone a disposición de los participantes para mostrar información sobre los partidos. Algunas de estas aplicaciones se encuentran un tanto obsoletas y presentan numerosas deficiencias en términos de rendimiento, usabilidad e interfaz de usuario.

A lo largo de numerosas conversaciones con compañeros de equipo y rivales, hemos compartido nuestra percepción de que las aplicaciones que utilizamos satisfacen nuestras necesidades ni expectativas. Entre los problemas más comunes se encuentran la lentitud y falta de fluidez en la navegación, diseños anticuados y poco intuitivos, y una

gestión deportiva ineficaz por parte de las empresas privadas encargadas de estas aplicaciones.

Frente a esta situación, surge la idea de desarrollar una aplicación móvil innovadora que aporte un valor significativo al entorno deportivo del pádel. El pilar principal es crear una plataforma moderna y eficiente que responda a las necesidades reales de los jugadores y administradores de competiciones, y que se convierta en una referencia en el sector.

Características y objetivos clave:

1. **Interfaz de Usuario Intuitiva y Atractiva**: Una interfaz moderna, limpia y fácil de usar, que permita a los usuarios navegar y acceder a la información de manera rápida y sin complicaciones.
2. **Rendimiento Óptimo**: Una aplicación con un rendimiento ágil y fluido, evitando los problemas de lentitud y tiempos de carga prolongados que afectan a muchas de las aplicaciones actuales.
3. **Gestión Integral para Administradores**: Herramientas avanzadas y eficientes para la gestión de la liga, facilitando la programación de partidos, actualización de resultados, gestión de equipos y comunicación con los jugadores.
4. **Acceso Completo para Jugadores**: Funcionalidades que permitan a los jugadores acceder a toda la información relevante sobre sus equipos, partidos, estadísticas y clasificaciones.
5. **Innovación y Adaptabilidad**: Una arquitectura escalable que permita incorporar nuevas funcionalidades y adaptarse a las necesidades cambiantes del mercado y de los usuarios, con vistas a un futuro crecimiento y expansión.
6. **Potencial de Comercialización**: Considerar la posibilidad de llevar la aplicación a producción y buscar financiación para convertirla en un producto

comercial viable, con el objetivo de alcanzar un público más amplio y consolidarse como una herramienta indispensable en el ámbito de las competiciones de pádel. Naciendo como producto para una sola liga, pero pudiendo adaptarse a la necesidad de cualquier organización.

En resumen, este proyecto busca cambiar la forma en que se gestiona y participa en las ligas de pádel, ofreciendo una solución tecnológica avanzada y centrada en el usuario, que responda a las demandas actuales y futuras de este deporte que actualmente se encuentra en auge en todo el mundo.

4.1 Descripción breve de métodos, técnicas o arquitecturas(m/t/a)

utilizadas.

El proyecto ha sido desarrollado siguiendo una arquitectura modularizada y basada en tres capas, adecuada para aplicaciones que requieren cierta escalabilidad, mantenibilidad y eficiencia en la gestión de datos.

Arquitectura de Tres Capas

1. Capa de Presentación (*Flutter*):

El framework *Flutter* se ha utilizado para desarrollar la interfaz de usuario (*UI*) de la aplicación móvil. Se encarga de presentar la información al usuario final de manera visualmente atractiva y funcional.

- **Características:**

- Utilización de widgets y composición para construir interfaces dinámicas.

- Manejo de estado centralizado a través de `setState` y `Provider` para una gestión eficiente del flujo de datos.

2. Capa de Negocio (*API REST* con *Spring Boot*):

Spring Boot se seleccionó para construir una *API RESTful* que sirve como intermediario entre la capa de presentación y la base de datos. Esta capa maneja la lógica de negocio y asegura que los datos sean accesibles de manera controlada.

A su vez, la *API Rest* también utiliza una arquitectura de tres capas: Controller, Service y Repository.

3. Capa de Datos (*MySQL*):

Se ha utilizado *MySQL* como sistema gestor de base de datos relacional.

- **Características:**
 - Definición de modelos de datos con relaciones entre tablas.
 - Uso de consultas *SQL* y operaciones *CRUD* (Crear, Leer, Actualizar, Eliminar) para interactuar con la base de datos desde la *API REST*.

4.1.1 Modularización y Beneficios

- **Separación de Responsabilidades:** Cada capa tiene un propósito específico, lo que facilita el mantenimiento y la escalabilidad del sistema.
- **Reutilización de Componentes:** Los módulos en Flutter y los servicios en Spring Boot permiten reutilizar lógica y componentes en diferentes partes de la aplicación.

- **Pruebas Unitarias y de Integración:** Se pueden realizar pruebas de manera eficiente en cada capa para garantizar la calidad del software desarrollado.

Esta arquitectura modularizada y basada en tres capas ha proporcionado una base sólida para el desarrollo de la aplicación, asegurando que cumpla con los requisitos funcionales y no funcionales establecidos, además de facilitar su mantenimiento y evolución futura.

4.2 Despliegue de la aplicación indicando plataforma tecnológica, instalación de la aplicación y puesta en marcha

5 Planificación y presupuesto

El desarrollo de esta aplicación móvil de gestión de ligas de pádel se planifica en tres fases, comenzando con una demo básica y evolucionando hacia una aplicación completa con funcionalidades avanzadas.

El presupuesto estimado es de 50,036€, cubriendo tanto los costos de mano de obra como los gastos en infraestructura y herramientas necesarias.

Este enfoque permitirá un desarrollo ágil y escalable, asegurando una base sólida para futuras expansiones y mejoras.

5.1 Planificación por Fases

Fase 0: Búsqueda de inversión

Esta fase va a ser transversal a toda la duración del proyecto, iniciando entorno a 2 meses antes del comienzo del desarrollo, y finalizando con la salida de este proyecto al mercado.

Se trata de una fase transversal ya que puede encontrarse una fuente de inversión antes de comenzar, o durante el desarrollo. Además, esta fase nunca dejaría de estar activa ya que el contacto con la fuente o fuentes de donde proviene la inversión se alargaría hasta el final del proyecto.

Para esta planificación se tiene en cuenta que se ha encontrado un “Business Angel “ que ve futuro que la idea, y decide apostar por el proyecto tras visualizar la planificación y el presupuesto detallados a continuación.

Fase 1: Desarrollo de la Demo (3 meses)

- **Objetivo:** Crear una versión básica de la aplicación que incluya las funcionalidades esenciales.
- **Características:**
 - Visualización de información de partidos.
 - Registro y login de usuarios.
 - Introducción de resultados de partidos.
- **Actividades:**
 1. **Análisis y Diseño** (0.5 meses)
 - Análisis de requisitos.
 - Diseño de la base de datos.
 - Diseño de la interfaz de usuario.
 2. **Desarrollo Inicial** (1.5 meses)
 - Implementación de la base de datos (MySQL).

- Desarrollo del backend (API Rest con SpringBoot).
- Desarrollo del frontend (Flutter).
- Integración del frontend con la API Rest.

3. Pruebas y Ajustes (0.5 meses)

- Pruebas de funcionalidad y usabilidad.
- Corrección de errores.

4. Despliegue Inicial (0.5 meses)

- Configuración de servidores en localhost.
- Documentación de la demo.

Fase 2: Evolución y Añadido de Funcionalidades (6 meses)

- **Objetivo:** Expandir la aplicación con funcionalidades avanzadas y mejoras.
- **Características:**
 - Creación de nuevas ligas.
 - Participación de un equipo en diferentes ligas.
 - Gestión avanzada de ligas y equipos.
- **Actividades:**
 1. **Análisis y Diseño de Nuevas Funcionalidades (1 mes)**
 - Revisión de requisitos adicionales.
 - Diseño de nuevas funcionalidades y mejoras.
 2. **Desarrollo de Nuevas Funcionalidades (3 meses)**
 - Implementación de nuevas características en backend y frontend.
 3. **Pruebas y Ajustes (1 mes)**
 - Pruebas de las nuevas funcionalidades.

- Corrección de errores y ajustes.
- 4. **Despliegue en Entorno de Pruebas** (1 mes)
 - Configuración de servidores de prueba.
 - Documentación y preparación para producción.

Fase 3: Preparación para Producción y Lanzamiento (3 meses)

- **Objetivo:** Finalizar la aplicación y lanzarla al público.
- **Actividades:**
 1. **Optimización y Rendimiento** (1 mes)
 - Optimización de código y base de datos.
 - Mejora del rendimiento de la aplicación.
 2. **Pruebas de Usuario y Feedback** (1 mes)
 - Realización de pruebas con usuarios reales.
 - Recopilación de feedback y ajustes finales.
 3. **Despliegue en Producción y Marketing** (1 mes)
 - Configuración de servidores de producción (AWS o similar).
 - Plan de marketing y lanzamiento.

A lo largo de todas estas fases y principalmente al comienzo y al finalizar cada una de ellas, se incluyen numerosas reuniones de información y revisión del trabajo realizado junto a la persona inversionista (encontrada en la ‘fase 0’).

5.2 Presupuesto Detallado

Mano de Obra (4 personas)

- **Desarrolladores:** 4 personas (yo junto a tres compañeros conocidos en clase/FCT)
- **Costo mensual por desarrollador:** 1000€ (considerando un sueldo base de junior sin experiencia previa).

Si se realizara en un entorno independiente sin ningún tipo de inversor, se buscaría un equipo de desarrollo que realice el proyecto a coste 0€ (por vocación y aprendizaje).

Como se está creando un presupuesto ficticio adaptado al entorno de trabajo real actual, se considera que cada desarrollador debería ser remunerado.

- **Duración del proyecto:** 12 meses
- **Costo total de mano de obra:** 4 desarrolladores * 1000€/mes * 12 meses = 48,000€

Infraestructura y Servicios

- **Servidores de Base de Datos y API (AWS o similar):**
 - Costo mensual: 50€ (DB) + 50€ (API) = 100€
 - Duración del proyecto: 12 meses
 - Costo total: 100€/mes * 12 meses = 1,200€
- **Almacenamiento de Fotos y Archivos (AWS S3 o similar):**
 - Costo mensual: 20€
 - Duración del proyecto: 12 meses
 - Costo total: 20€/mes * 12 meses = 240€

Otros Gastos

- **Licencias de Software y Herramientas:**
 - Visual Studio Code: Gratis
 - IntelliJ (IDE): 149€ anual por desarrollador

- Costo total de licencias: $149€ \cdot 4 = 596€$
- **Gastos en pruebas y dispositivos:**
 - Dispositivos Android/Ios físicos (se utilizan los que tenemos) = 0€

Presupuesto Total

- **Mano de obra:** 48,000€
- **Infraestructura y servicios:** $1,200€ + 240€ = 1,440€$
- **Licencias de software:** 596€
- **Total:** $48,000€ + 1,440€ + 596€ = 50,036€$

El presupuesto podría abarataarse, ya que se propone un equipo de 4 desarrolladores que podría ser más reducido. También se ha contado con que todos los integrantes del equipo usarían un IDE de pago (IntelliJ), es decir, todos tocarían alguna parte de la API, pero realmente una o dos personas se podrían hacer cargo de generar y mantener la API, por lo que se podría prescindir de alguna licencia de software.

Todo el presupuesto se basa en una estimación ficticia.

6 Documentación Técnica: análisis, diseño, implementación y pruebas.

6.1 Especificación de requisitos

Requisitos de Software

- **Sistema Operativo:**

- Windows 10/11
- **Docker:**
 - *Docker Engine* versión 20.10 o superior
 - *Docker Compose* versión 1.29.2 o superior
- **Java:**
 - *JDK 17* o superior
- **MySQL:**
 - MySQL versión 8.0 o superior
- **Herramientas de Desarrollo:**
 - *IDE* de desarrollo (*IntelliJ, Eclipse, VS Code, etc.*)
 - *Git* para control de versiones
 - *Postman* para pruebas de *API*
- **Dependencias del Proyecto:**
 - *Spring Boot* versión 2.5 o superior
 - *Hibernate ORM*
 - *JPA* (Java Persistence API)
 - *Swagger* para documentación de *API*
 - *Lombok* para simplificación del código

Requisitos de Hardware

- **Servidor (En local -> localhost):**
 - CPU: Procesador con al menos 4 núcleos
 - Memoria RAM: Mínimo 8 GB
 - Almacenamiento: Mínimo 50 GB de espacio libre en disco
- **Cliente (Móvil/Emulador):**

- Android 8.0 (Oreo) o superior
- Espacio de almacenamiento: Mínimo 500 MB para la instalación de la APK y datos de la aplicación

6.2 *Análisis del sistema*

6.3 *Diseño del sistema:*

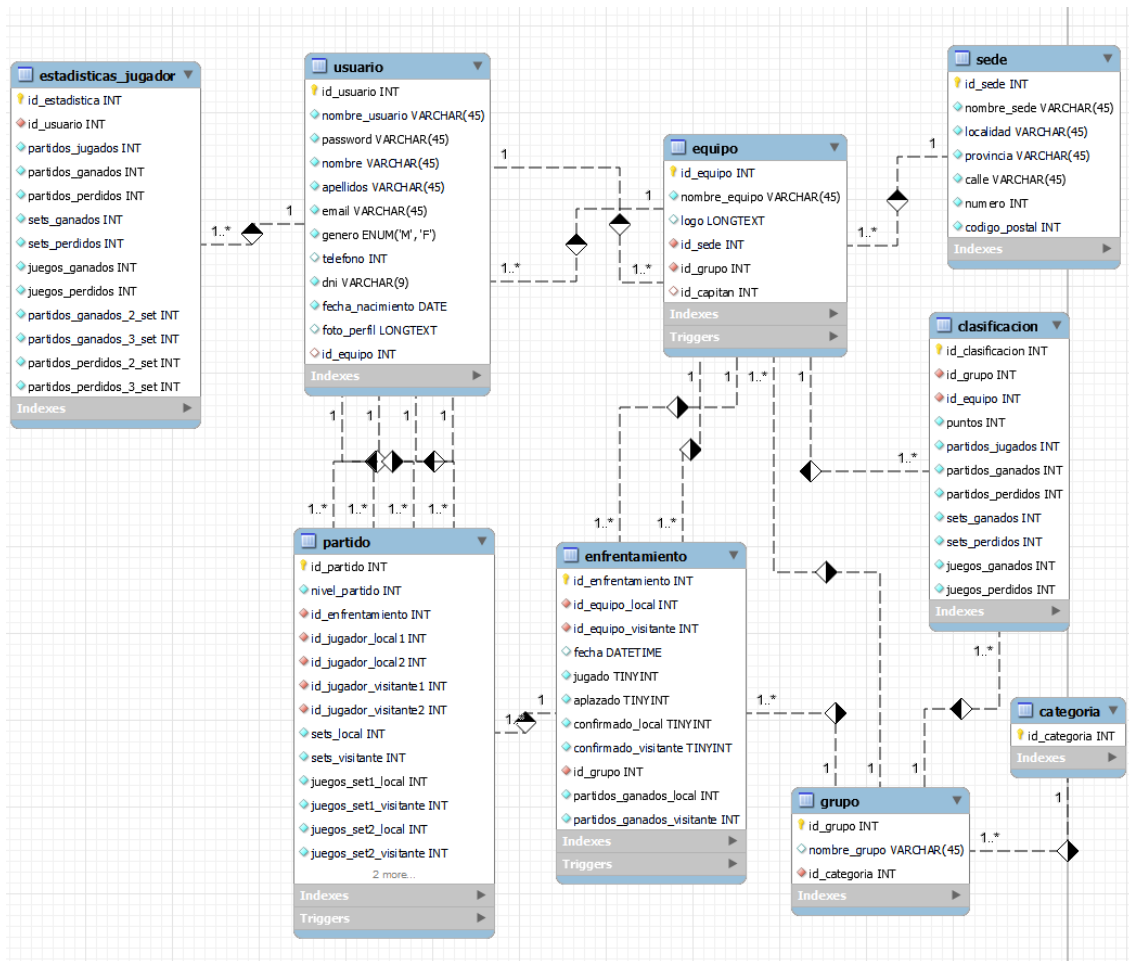
6.3.1 *Diseño de la Base de Datos*

La base de datos para la aplicación se ha estructurado de manera eficiente para soportar todas las funcionalidades requeridas. A continuación se detallan las tablas que componen la base de datos, sus campos, tipos de datos y relaciones entre ellas.

Cada una de estas tablas está diseñada para almacenar información específica que se interrelaciona mediante claves foráneas (FK), permitiendo así un sistema robusto y escalable para la gestión de competiciones deportivas de pádel.

Diagrama de la base de datos:

En este diagrama se pueden ver todas las relaciones entre las tablas de la base de datos.



Nomenclaturas a conocer:

- **PK:** Primary Key, clave primaria.
- **FK:** Foreign Key, clave foránea.
- **AI:** AutoIncrement, autoincremento.
- **TINYINT:** boolean, true = 1 | false = 0.
- **INT:** integer, dato numérico.
- **VARCHAR:** dato en forma de string.
- **LONGTEXT:** string largo.
- **DATE:** dato con formato de fecha.
- **ENUM:** dato fijo a elegir dentro de una enumeración de posibilidades.
- **(X):** contiene la longitud que deberá tener el dato ingresado en el campo.

6.3.1.1 Tabla Sede

Esta tabla almacena la información de las sedes donde se realizan las competiciones.

Campo	Tipo de Dato	Descripción
idSede	INT, PK, AI	Identificador único de la sede
nombreSede	VARCHAR(45)	Nombre de la sede
localidad	VARCHAR(45)	Localidad de la sede
provincia	VARCHAR(45)	Provincia de la sede
calle	VARCHAR(45)	Calle de la sede
numero	INT(3)	Número de la calle
codigoPostal	INT(5)	Código postal de la sede

6.3.1.2 Tabla Categoría

Esta tabla almacena las categorías de los equipos a través de un identificador único.

Se utilizarán las categorías con identificador 1-5, siendo 1 la categoría más alta.

Campo	Tipo de Dato	Descripción
idCategoría	INT, PK, AI	Identificador único de la categoría

6.3.1.3 Tabla Grupo

Esta tabla almacena los grupos que existan dentro de cada categoría.

Campo	Tipo de Dato	Descripción
idGrupo	INT, PK, AI	Identificador único del grupo
nombreGrupo	VARCHAR(45)	Nombre del grupo
idCategoría	INT, FK	Identificador de la categoría

6.3.1.4 Tabla Clasificación

Esta tabla almacena la clasificación de los equipos en los grupos.

Campo	Tipo de Dato	Descripción
idClasificacion	INT, PK, AI	Identificador único de la clasificación
idGrupo	INT, FK	Identificador del grupo

idEquipo	INT, FK	Identificador del equipo
puntos	INT	Puntos del equipo
partidosJugados	INT	Número de partidos jugados
partidosGanados	INT	Número de partidos ganados
partidosPerdidos	INT	Número de partidos perdidos
setsGanados	INT	Número de sets ganados
setsPerdidos	INT	Número de sets perdidos
juegosGanados	INT	Número de juegos ganados
juegosPerdidos	INT	Número de juegos perdidos

6.3.1.5 Tabla Equipo

Esta tabla almacena la información de los equipos.

Campo	Tipo de Dato	Descripción
idEquipo	INT, PK, AI	Identificador único del equipo
nombreEquipo	VARCHAR(45)	Nombre del equipo
logo	LONGTEXT	Logo del equipo
idSede	INT, FK	Identificador de la sede
idGrupo	INT, FK	Identificador del grupo
idCapitan	INT, FK	Identificador del capitán del equipo

6.3.1.6 Tabla Usuario

Esta tabla almacena la información de los usuarios.

Campo	Tipo de Dato	Descripción
idUsuario	INT, PK, AI	Identificador único del usuario
nombreUsuario	VARCHAR(45)	Nombre de usuario
password	VARCHAR(45)	Contraseña del usuario
nombre	VARCHAR(45)	Nombre del usuario
apellidos	VARCHAR(45)	Apellidos del usuario
email	VARCHAR(45)	Correo electrónico del usuario
genero	ENUM("M", "F")	Género del usuario
telefono	INT(9)	Teléfono del usuario
dni	VARCHAR(9)	DNI del usuario
fechaNacimiento	DATE	Fecha de nacimiento del usuario
fotoPerfil	LONGTEXT	Foto de perfil del usuario
idEquipo	INT, FK, NULL	Identificador del equipo al que pertenece

6.3.1.7 Tabla Enfrentamiento

Esta tabla almacena la información de los enfrentamientos entre equipos.

Campo	Tipo de Dato	Descripción
idEnfrentamiento	INT, PK, AI	Identificador único del enfrentamiento
idEquipoLocal	INT, FK	Identificador del equipo local
idEquipoVisitante	INT, FK	Identificador del equipo visitante
fecha	DATETIME	Fecha del enfrentamiento
jugado	TINYINT	Indica si el enfrentamiento se ha jugado
aplazado	TINYINT	Indica si el enfrentamiento se ha aplazado
confirmadoLocal	TINYINT	Confirmación del resultado del equipo local
confirmadoVisitante	TINYINT	Confirmación del resultado del equipo visitante
idGrupo	INT, FK	Identificador del grupo al que pertenece
partidosGanadosLocal	INT(1)	Partidos ganados por el equipo local
partidosGanadosVisitante	INT(1)	Partidos ganados por el equipo visitante

6.3.1.8 Tabla Partido

Esta tabla almacena la información de los partidos dentro de un enfrentamiento.

Campo	Tipo de Dato	Descripción
idPartido	INT, PK, AI	Identificador único del partido
nivelPartido	INT(1)	Nivel del partido
idEnfrentamiento	INT, FK	Identificador del enfrentamiento al que pertenece
idJugadorLocal1	INT, FK	Identificador del primer jugador local
idJugadorLocal2	INT, FK	Identificador del segundo jugador local
idJugadorVisitante1	INT, FK	Identificador del primer jugador visitante
idJugadorVisitante2	INT, FK	Identificador del segundo jugador visitante
setsLocal	INT(1)	Sets ganados por el equipo local
setsVisitante	INT(1)	Sets ganados por el equipo visitante
juegosSet1Local	INT(1)	Juegos ganados por el equipo local en el primer set
juegosSet1Visitante	INT(1)	Juegos ganados por el equipo visitante en el primer set
juegosSet2Local	INT(1)	Juegos ganados por el equipo local en el segundo set
juegosSet2Visitante	INT(1)	Juegos ganados por el equipo visitante en el segundo set
juegosSet3Local	INT(1)	Juegos ganados por el equipo local en el tercer set
juegosSet3Visitante	INT(1)	Juegos ganados por el equipo visitante en el tercer set

6.3.1.9 Tabla EstadísticasJugador

Esta tabla almacena las estadísticas de cada jugador.

Campo	Tipo de Dato	Descripción
idEstadística	INT, PK, AI	Identificador único de la estadística
idUsuario	INT, FK	Identificador del usuario
partidosJugados	INT	Número de partidos jugados
partidosGanados	INT	Número de partidos ganados
partidosPerdidos	INT	Número de partidos perdidos
setsGanados	INT	Número de sets ganados
setsPerdidos	INT	Número de sets perdidos
juegosGanados	INT	Número de juegos ganados
juegosPerdidos	INT	Número de juegos perdidos
partidosGanados2Set	INT	Número de partidos ganados en 2 sets
partidosGanados3Set	INT	Número de partidos ganados en 3 sets
partidosPerdidos2Set	INT	Número de partidos perdidos en 2 sets
partidosPerdidos3Set	INT	Número de partidos perdidos en 3 sets

6.3.2 Diseño de la Interfaz de usuario.

6.3.3 Estado del arte de la Heurística de la Interacción del Usuario con la Interfaz

Investigación de la Heurística

Para investigar la heurística de la interacción del usuario con la interfaz, se han seguido los principios de diseño de interacción establecidos por Jakob Nielsen y Rolf Molich.

Estos son los principios, conocidos como las "10 Heurísticas de Usabilidad de Nielsen":

1. **Visibilidad del estado del sistema:** El sistema mantiene a los usuarios informados sobre lo que está ocurriendo mediante una retroalimentación adecuada en un tiempo razonable.

2. **Relación entre el sistema y el mundo real:** La interfaz utiliza un lenguaje familiar para el usuario, con palabras, frases y conceptos conocidos, en lugar de terminología técnica.
3. **Control y libertad del usuario:** Los usuarios pueden deshacer y rehacer acciones fácilmente, permitiéndoles salir de situaciones no deseadas sin complicaciones.
4. **Consistencia y estándares:** Los usuarios no deben preguntarse si diferentes palabras, situaciones o acciones significan lo mismo. Se siguen las convenciones de la plataforma.
5. **Prevención de errores:** La interfaz está diseñada para prevenir errores y, cuando esto no es posible, se proporciona una forma de corregirlos fácilmente.
6. **Reconocimiento antes que recuerdo:** Las opciones y características son visibles y fáciles de encontrar, reduciendo la carga de memoria del usuario.
7. **Flexibilidad y eficiencia de uso:** La interfaz permite a los usuarios expertos realizar tareas más rápidamente sin interferir con la experiencia de los principiantes.
8. **Estética y diseño minimalista:** La interfaz contiene solo la información esencial y evita el desorden, permitiendo que los usuarios se concentren en sus tareas.
9. **Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de errores:** Los mensajes de error son expresados en lenguaje claro y ofrecen soluciones.

10. **Ayuda y documentación:** Se proporciona ayuda y documentación fácilmente accesible, aunque se prefiere que el sistema sea usable sin necesidad de consulta.

Aplicación de Heurísticas al Desarrollo Minimalista de la Aplicación

En el desarrollo de la aplicación de gestión de una liga de pádel, se aplicaron estos principios de la siguiente manera:

- **Visibilidad del estado del sistema:** La interfaz proporciona retroalimentación inmediata después de cada acción del usuario, como la actualización de resultados de partidos, con mensajes de confirmación y actualización visual del estado.
- **Relación entre el sistema y el mundo real:** Se utiliza terminología familiar para los usuarios de pádel, como "partidos", "clasificación", "equipos", y "jugadores", en lugar de términos técnicos.
- **Consistencia y estándares:** Se siguen las convenciones estándar de diseño de aplicaciones móviles y web, como el uso de iconos reconocibles para navegación y acciones comunes (ej. icono de casa para la pantalla principal).
- **Prevención de errores:** Se utilizan formularios con validación en tiempo real para evitar la entrada de datos incorrectos. Por ejemplo, al registrar un usuario se deben rellenar todos los campos, si el DNI o teléfono no tienen una longitud de 9 dígitos se mostrará un error informativo en el campo correspondiente.
- **Reconocimiento antes que recuerdo:** Los menús y opciones están siempre visibles o fácilmente accesibles mediante menús desplegables, minimizando la necesidad de recordar rutas complejas.

- **Estética y diseño minimalista:** Se ha adoptado un enfoque minimalista, eliminando elementos innecesarios y destacando solo lo esencial. Esto facilita a los usuarios concentrarse en la gestión de la liga sin distracciones.

Por ejemplo: en la ficha de equipo, se muestran los datos principales del equipo junto a cuatro listas desplegables, la primera lista se encuentra desplegada por defecto, pero las demás listas deberán ser desplegadas por el usuario. De esta forma se da control al usuario sobre la información que quiere visualizar, y permite que la interfaz no se sobrecargue visualmente.

6.3.4 Diseño de la Navegación entre Pantallas

Principios de Diseño de Navegación

Para que la navegación entre pantallas sea fluida, intuitiva y eficiente, se han considerado los siguientes principios:

1. **Simplicidad:** La estructura de la navegación es simple y plana, minimizando la cantidad de niveles jerárquicos para que los usuarios lleguen a su destino con el menor número de clics posible.
2. **Consistencia:** Los patrones de navegación son consistentes. Por ejemplo, el menú principal siempre está accesible desde el mismo lugar.
3. **Predictibilidad:** La navegación es predecible e intuitiva; los usuarios siempre saben dónde están y cómo volver a la pantalla anterior o a la pantalla principal, ya sea a través de un botón 'Home', o de un botón 'Back'.

Implementación del Diseño de Navegación

- **Pantalla Principal:** Actúa como el hub central con accesos directos a las funcionalidades principales como "Clasificación", "Partidos jugados", "Próximos partidos".
- **Menú de Navegación:** Un menú de navegación persistente, accesible desde la parte inferior de la pantalla, proporciona acceso a todas las secciones de la aplicación. En algunas pantallas desaparecerá.
- **Navegación mediante desplegables:** En pantallas con múltiples secciones de información (como la pantalla ficha de equipo), se utiliza una navegación mediante listas desplegables para permitir a los usuarios cambiar rápidamente entre diferentes vistas de datos, sin rellenar toda la pantalla.
- **Navegación Contextual:** En las vistas de detalle (por ejemplo, detalles de un equipo o jugador), se proporcionan enlaces y botones para realizar acciones contextuales directamente desde esa vista, como editar información o ver estadísticas relacionadas.

6.3.5 Diseño de la Aplicación.

El diseño de la aplicación se ha enfocado de una forma modular y escalable, integrando el backend con la aplicación móvil mediante una API REST. La elección de tecnologías actuales como Spring Boot, MySQL y Flutter ha permitido crear una solución eficiente que garantice una experiencia de usuario óptima y una fácil mantenibilidad del sistema para los desarrolladores.

Arquitectura de la Base de Datos

La base de datos es el corazón de la aplicación de gestión de liga de pádel, diseñada para almacenar información relevante sobre usuarios, equipos, partidos y resultados. Se ha utilizado MySQL como sistema de gestión de bases de datos relacional.

- **Tablas Principales**
- **Relaciones entre tablas**
- **Triggers:**
 - Triggers para actualizar automáticamente las estadísticas del equipo después de la inserción de resultados.
 - Triggers para mantener la integridad referencial entre tablas.

Arquitectura de la API

La API REST se ha diseñado utilizando Spring Boot con JPA e Hibernate para facilitar la interacción con la base de datos y gestionar las operaciones CRUD.

- **Controladores (Controllers):**
 - Gestionan las solicitudes HTTP entrantes y llaman a los servicios adecuados para procesar la lógica de negocio.
 - Exponen endpoints RESTful para las operaciones CRUD en usuarios, equipos, enfrentamientos, partidos, estadísticas de jugadores, clasificación, grupos y categorías .
- **Servicios (Services):**
 - Contienen la lógica de negocio de la aplicación.
 - Interactúan con los repositorios para realizar operaciones en la base de datos.
- **Repositorios (Repositories):**

- Utilizan JPA e Hibernate para interactuar con la base de datos.
- Permiten realizar operaciones CRUD de manera eficiente y sencilla con una mínima cantidad de código.

Estructura de la aplicación en Flutter

- **Estructura del Proyecto:**
 - **Pantallas (Screens):** Contienen las interfaces de usuario para las diferentes funcionalidades de la aplicación.
 - **Funciones:** Componentes con funciones reutilizables que se utilizan a lo largo de toda la aplicación.
 - Providers
 - Provider de tema (Esquema de colores)
 - Provider de usuario (Identifica al usuario que tiene la sesión iniciada).
 - BottomBar
 - Widgets propios reutilizables (tarjetas y listas)
 - **Modelos:** Definen las estructuras de datos que va a manejar la aplicación, tanto para mostrarlos en las diferentes pantallas, como para interactuar con la *API*. Además manejan la comunicación con la *API* REST para obtener y enviar datos.
 - Usuario
 - Equipo
 - Sede
 - Grupo
 - Categoría

- Enfrentamiento
 - Categoría
 - Estadísticas Jugadores
 - Clasificación
- **Assets:** Contiene imágenes y recursos que se utilizan en la aplicación.

Flujo entre pantallas

- **Inicio de Sesión y Registro:**
 - Pantallas de inicio de sesión y registro que interactúan con la API para autenticar y crear nuevos usuarios.
- **Pantalla de Inicio (Dashboard):**
 - Pantalla de bienvenida para un jugador que se ha registrado y aún no tiene un equipo asignado.
 - Pantalla de inicio en la cual se muestran datos del jugador como el nombre y el equipo. Además muestra los partidos jugados y los próximos partidos por jugar del equipo correspondiente.
- **Gestión y visualización de Equipo:**
 - Funcionalidades para la edición (solo para el capitán) y visualización de información de los equipos.
 - Interfaz intuitiva para agregar y gestionar miembros del equipo (solo para el capitán).
- **Información de jugador:** Muestra información sobre un jugador en concreto.
- **Perfil:** permite ver la información del perfil de usuario, así como modificarla fácilmente.

- **Visualización de enfrentamiento:** Permite ver la información de un enfrentamiento, así como su resultado una vez jugado.
- **Insertar resultados de Partidos:**
 - Calendario de partidos donde los usuarios pueden v
 - Funcionalidad para registrar y visualizar resultados de los partidos jugados.

Seguridad

- **Autenticación y Autorización:**
 - Implementación de JWT (JSON Web Tokens) para la autenticación de usuarios.
 - Control de acceso a diferentes funcionalidades basado en roles de usuario.
- **Protección de Datos:**
 - Encriptación de contraseñas y otros datos sensibles (Funcionalidad por implementar, pero que está previamente investigada, a falta de pruebas para que funcione correctamente). Actualmente se guardan los datos en texto plano, pero en una futura fase de desarrollo pasarán a estar encriptados mediante *Bcrypt*, el cual actualmente es el estándar de encriptado debido a su resistencia a los ataques de fuerza bruta y otras técnicas de cracking.

Éste método de encriptación es fácilmente configurable a través de Spring Security (Incluido en el código de la *API*, pero no implementado).

6.4 Implementación:

6.4.1 Entorno de desarrollo.

Al tratarse de un proyecto que aún no se ha llevado a producción, el entorno de desarrollo ha sido totalmente local.

Se ha implementado un servidor localhost para la base de datos *MySQL*, y otro servidor, también local (tomcat) para la *API rest* a través de *SpringBoot*.

- *MySQL*: localhost:3306
- *Tomcat*: localhost:8080

En cuanto al entorno de desarrollo del frontend, se han utilizado tanto un dispositivo Android físico, como un emulador de este mismo sistema operativo para desarrollar y realizar las pruebas de la aplicación. La capa de frontend, conecta con el backend a través del servidor *tomcat* mencionado anteriormente, y utiliza los *endpoints* de la *API Rest* para enviar y recibir datos al servidor *MySQL*.

6.4.2 Estructura del código.

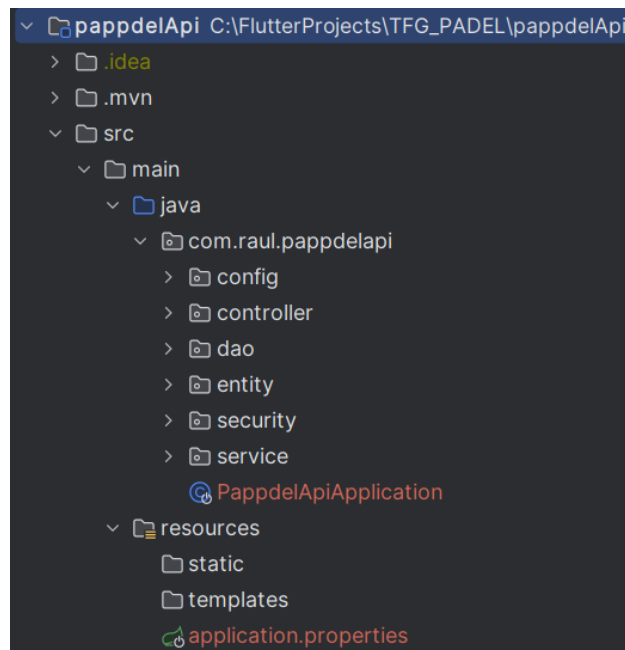
A continuación detallo la estructura del código en las diferentes capas.

6.4.2.1 Código en base de datos

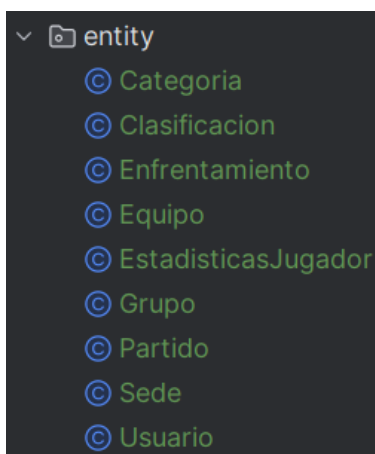
- **Script de creación de BBDD:** Se ha generado un script de base de datos utilizando MySQL Workbench.
- **Generación de datos ficticios mediante IA:** Se ha empleado inteligencia artificial para la generación de datos ficticios a través de un script *SQL*.
- **Script de Triggers para estadísticas automáticas:** Se han implementado triggers en la base de datos para la generación automática de estadísticas al insertar datos de partidos.

6.4.2.2 Código en API Rest – Spring Boot

Se ha desarrollado una API REST estructurada en una arquitectura de tres capas utilizando Spring Boot. Esta arquitectura sigue un diseño modular que mejora la mantenibilidad y escalabilidad de la aplicación.

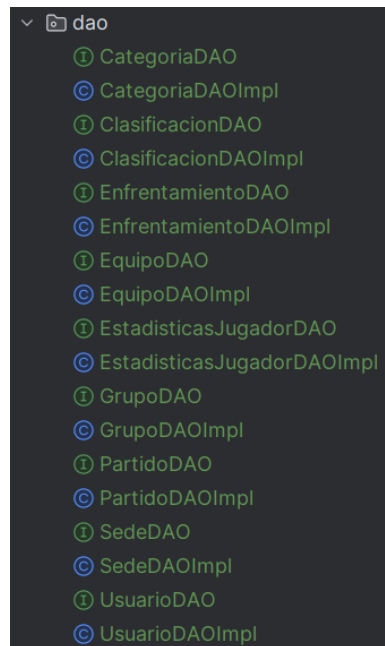


Entidades de la API REST



Capas de la API REST

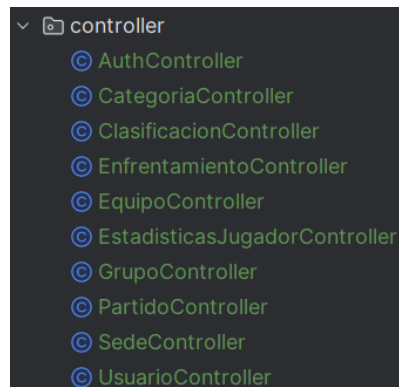
- Capa de Acceso a Datos (DAO/Repository)



- **Capa de Servicio (Service)**

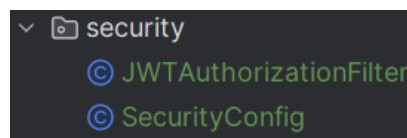


- **Capa de Controlador (Controller)**



Configuración de Seguridad y Gestión de Tokens

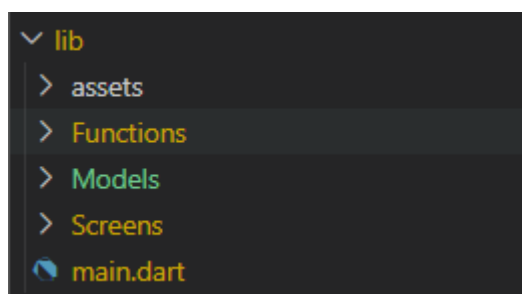
- Seguridad
- Generación de Tokens



6.4.2.3 Código en Flutter

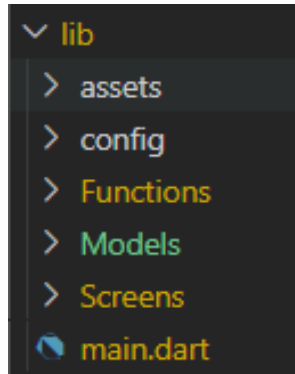
Estructura general

Estructura general de las carpetas del proyecto que contienen el código de flutter en archivos .dart



Estructura Funcions

Contiene diferentes funcionalidades que van a ser utilizadas dentro de las pantallas de la aplicación, como el `bottom_bar.dart`, `mis_widgets.dart`, o el provider del perfil de usuario.



Estructura Models

Se ha creado una estructura de modelos de entidades las cuales actúan como objetos para enviar o recibir datos a través de la *API Rest*. Estos modelos contienen los atributos correspondientes para almacenar la información recibida desde la base de datos, la deserializa convirtiéndola en objeto, y para enviar los datos realiza la operación inversa, serializa los datos convirtiéndolos en formato map/json para enviarlos a través de la *API*.



Ejemplo de código de una entidad:

```
class JugadorPadel {
    int? _idJugador;
    late String _nombreUsuario;
    // ignore: unused_field
    String? _password;
    late String _nombre;
    late String _apellidos;
    late String _email;
    late String _genero;
    late String _telefono;
    late String _dni;
    late String _fechaNacimiento;
    String? _fotoPerfil;
    int? _idEquipo;

    // Constructor por defecto
    JugadorPadel() {
        _idJugador = 0;
        _nombreUsuario = '';
        _nombre = '';
        _apellidos = '';
        _email = '';
        _genero = '';
        _telefono = '';
        _dni = '';
        _fechaNacimiento = '';
        _fotoPerfil = '';
        _idEquipo = 0;
    }

    // Constructor sin id
    JugadorPadel.withoutId(this._nombreUsuario, this._password, this._nombre, this._apellidos, this._email, this._genero, this._telefono, this._dni, this._fechaNacimiento, this._fotoPerfil, this._idEquipo);

    // Constructor con id
    JugadorPadel.withId(this._idJugador, this._nombreUsuario, this._password, this._nombre, this._apellidos, this._email, this._genero, this._telefono, this._dni, this._fechaNacimiento, this._fotoPerfil, this._idEquipo);
}
```

```
// Constructor desde map
JugadorPadel.fromMap(Map<String, dynamic> map) {
    _idJugador = (map['id_jugador'] != null) ? map['id_jugador'] : null;
    _nombreUsuario = (map['nombre_usuario'] != null) ? map['nombre_usuario'] : '';
    _nombre = (map['nombre'] != null) ? map['nombre'] : '';
    _apellidos = (map['apellidos'] != null) ? map['apellidos'] : '';
    _email = (map['email'] != null) ? map['email'] : '';
    _genero = (map['genero'] != null) ? map['genero'] : '';
    _telefono = (map['telefono'] != null) ? map['telefono'] : '';
    _dni = (map['dni'] != null) ? map['dni'] : '';
    _fechaNacimiento = (map['fecha_nacimiento'] != null) ? map['fecha_nacimiento'] : '';
    _fotoPerfil = (map['foto_perfil'] != null) ? map['foto_perfil'] : '';
    _idEquipo = (map['id_equipo'] != null) ? map['id_equipo'] : null;
}

// Getters y setters
int? get idJugador => _idJugador; // Unnecessary use of getter and setter to wrap a field.ðTry removing the getter and setter and renaming the field.
String get nombreUsuario => _nombreUsuario; // Unnecessary use of getter and setter to wrap a field.ðTry removing the getter and setter and renaming the field.
String get nombre => _nombre; // Unnecessary use of getter and setter to wrap a field.ðTry removing the getter and setter and renaming the field.
String get apellidos => _apellidos; // Unnecessary use of getter and setter to wrap a field.ðTry removing the getter and setter and renaming the field.
String get email => _email; // Unnecessary use of getter and setter to wrap a field.ðTry removing the getter and setter and renaming the field.
String get genero => _genero; // Unnecessary use of getter and setter to wrap a field.ðTry removing the getter and setter and renaming the field.
String get telefono => _telefono; // Unnecessary use of getter and setter to wrap a field.ðTry removing the getter and setter and renaming the field.
String get dni => _dni; // Unnecessary use of getter and setter to wrap a field.ðTry removing the getter and setter and renaming the field.
String get fechaNacimiento => _fechaNacimiento; // Unnecessary use of getter and setter to wrap a field.ðTry removing the getter and setter and renaming the field.
String get fotoPerfil => _fotoPerfil; // Unnecessary use of getter and setter to wrap a field.ðTry removing the getter and setter and renaming the field.
int? get idEquipo => _idEquipo; // Unnecessary use of getter and setter to wrap a field.ðTry removing the getter and setter and renaming the field.

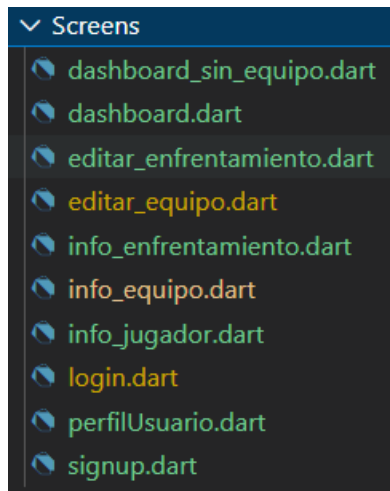
set idJugador(int? idJugador) {
    _idJugador = idJugador;
}

set nombreUsuario(String nombreUsuario) {
    _nombreUsuario = nombreUsuario;
}

set nombre(String nombre) {
    _nombre = nombre;
}
```

Estructura Screens

Se trata de los archivos que contienen la interfaz y funcionalidad de cada una de las pantallas de la aplicación.



Ejemplo de un archivo de interfaz:

```

23 @override
24 Widget build(BuildContext context) {
25   final temaActual = Provider.of<CargadorTema>(context).temaActual;
26
27   height = MediaQuery.of(context).size.height;
28   width = MediaQuery.of(context).size.width;
29
30   return Scaffold(
31     backgroundColor: temaActual.colorScheme.primary,
32     body: SafeArea(
33       child: SingleChildScrollView(
34         child: Column(
35           children: [
36
37             //Contenedor con la cabecera
38             Container(
39               padding: const EdgeInsets.symmetric(vertical: 15, horizontal: 20),
40               alignment: Alignment.bottomCenter,
41               height: 130,
42               child: Row(
43                 mainAxisAlignment: MainAxisAlignment.spaceBetween,
44                 crossAxisAlignment: CrossAxisAlignment.center,
45                 children: [
46
47                   Flexible(
48                     child: Column(
49                       mainAxisAlignment: MainAxisAlignment.min,
50                       mainAxisAlignment: MainAxisAlignment.spaceBetween,
51                       crossAxisAlignment: CrossAxisAlignment.start,
52                       children: [
53                         Row(
54                           children: [
55                             Flexible(
56                               child: Text(
57                                 nombreUsuario,
58                                 style: const TextStyle(
59                                   fontSize: 26,
60                                   fontWeight: FontWeight.bold,
61                                   color: Colors.white
62                                 ), // TextStyle
63                               // Text

```

6.5 Pruebas.

Las pruebas son necesarias para asegurar que la aplicación funcione correctamente, y ofrezca una buena experiencia de usuario.

A continuación se detallan las pruebas que se han llevado a cabo durante el desarrollo de la aplicación, tanto a nivel de backend como de frontend.

6.5.1.1 Verificación de Conexión a la Base de Datos

- **Conexión Inicial:**

1. Asegurarse de que el servidor de base de datos (*MySQL*) está activo, o en su defecto el contenedor de Docker correspondiente.

- **Pruebas de Integridad de Datos:**

1. Ejecutar el script *SQL* para crear la base de datos y las tablas, insertar datos de prueba y configurar los triggers.
2. Verificar que las tablas, datos y triggers se han creado correctamente y comprobando que funciona correctamente a través de consultas *SQL*, utilizando la herramienta *MySQL Workbench*.

6.5.1.2 Pruebas de API

Lo primero, se verifica que los servidores de base de datos (*MySQL* y *API REST*) están en funcionamiento, o en su defecto los contenedores Docker correspondientes.

1. **Herramienta de Pruebas (Postman):**

- Configurar Postman para apuntar a la URL del API REST (*http://localhost:8080*).
- Crear y ejecutar pruebas para cada endpoint de la API, verificando las respuestas y asegurándose de que el comportamiento es el esperado.

2. **Pruebas Funcionales:**

- **GET Requests:** Verificar que las solicitudes GET devuelvan los datos correctos.
- **POST Requests:** Probar la creación de nuevos recursos y verificar que los datos se guarden correctamente en la base de datos.
- **PUT Requests:** Probar la actualización de recursos existentes.
- **DELETE Requests:** Verificar la eliminación de recursos y la integridad referencial en la base de datos.

3. Pruebas de Validación y Errores:

- Verificar que la API maneje correctamente las solicitudes inválidas o mal formateadas.
- Comprobar que los mensajes de error sean claros y útiles para el usuario.

6.5.1.3 Pruebas de la Aplicación Móvil

1. Configuración Inicial:

- Instalar la APK en un dispositivo Android físico y en un emulador.
- Asegurarse de que el dispositivo físico esté conectado a la misma red que el ordenador que ejecuta los servidores localhost de base de datos y API, para garantizar la conectividad con la base de datos.

2. Pruebas de Conectividad:

- Verificar que la aplicación móvil puede conectarse al backend y que las funcionalidades básicas (inicio de sesión, registro, operaciones CRUD etc.) funcionan correctamente.

3. Pruebas de Usabilidad:

- Evaluar la interfaz de usuario para asegurar que es intuitiva y fácil de usar.

- Realizar pruebas de flujo de usuario para verificar que la navegación entre pantallas es fluida y eficiente.

4. Pruebas Funcionales:

- Probar todas las funcionalidades principales de la aplicación (gestión de equipos, visualización de información de equipos y jugadores, visualización de partidos, resultados y estadísticas, etc.).
- Asegurarse de que todas las interacciones con la API REST se realizan correctamente y que los datos se muestran y actualizan en tiempo real.

5. Hot Reload con Flutter:

- Durante el diseño de la interfaz, se utilizó un dispositivo móvil físico conectado al ordenador para aprovechar la característica de hot reload de Flutter.
- Esto me ha permitido una rápida evolución en el diseño y desarrollo, ya que cualquier cambio en el archivo .dart se actualiza automáticamente en la aplicación instalada en el dispositivo, sin necesidad de recompilar toda la aplicación, lo cual llevaría mucho tiempo.
- Esta característica facilita la identificación y corrección de errores en la interfaz de usuario de manera eficiente, ya que se ejecuta en modo debug/desarrollador y localiza los errores de manera visual en la interfaz, o directamente en el IDE al guardar el archivo.

6.5.1.4 Pruebas de Seguridad

Autenticación y Autorización:

- Verificar que los mecanismos de autenticación y autorización están correctamente implementados.

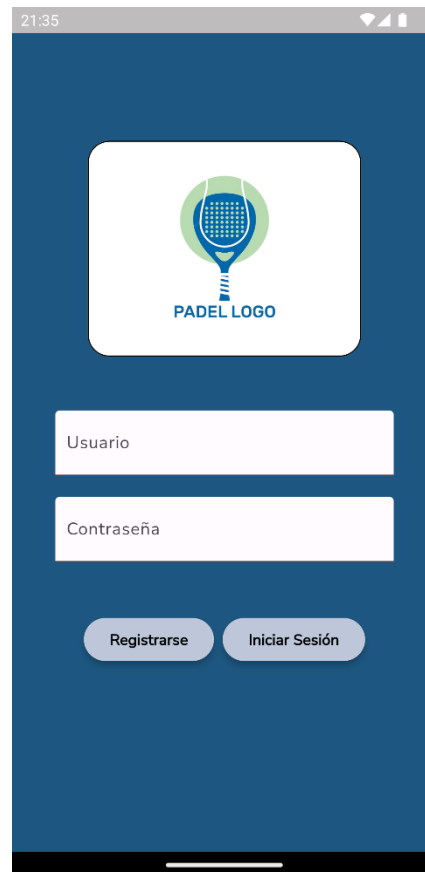
- Asegurarse de que solo los usuarios autenticados pueden acceder a las funcionalidades restringidas por la *API*, a través de *Spring Security*.

7 Manuales de usuario

7.1 Manual de usuario

Pantalla de login

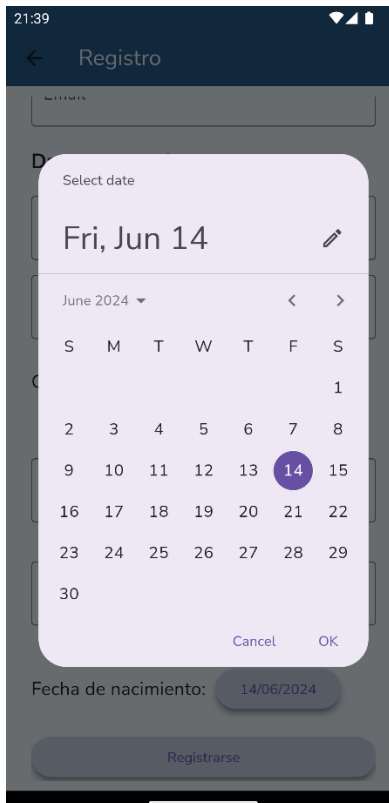


Se deberán introducir los datos del usuario para iniciar sesión, los campos están controlados por si se pulsa el botón iniciar sesión si los campos se encuentran vacíos



Registro de usuario

Si no se tiene una cuenta, se deberá efectuar un registro de usuario introduciendo los datos, todos los campos está controlados al validar el formulario, por lo que si no se

introducen unos datos válidos, dará error. Si se completa el registro se volverá a la pantalla de inicio de sesión.



Dashboard Bienvenida

Si no se tiene asignado un equipo, se mostrará el siguiente dashboard de bienvenida y se dará la opción de volver, ya que no hay información que mostrar al usuario.

Dashboard

El siguiente dashboard muestra información sobre el usuario y su equipo. Tiene un botón para cambiar entre modo oscuro o modo claro en la aplicación. Y una barra inferior para navegar por la aplicación.

Contiene dos listas scrolleables horizontalmente en la que podremos ver los próximos partidos y los partidos ya jugados, con los resultados de estos. Además contiene la clasificación del grupo al que pertenece el equipo.



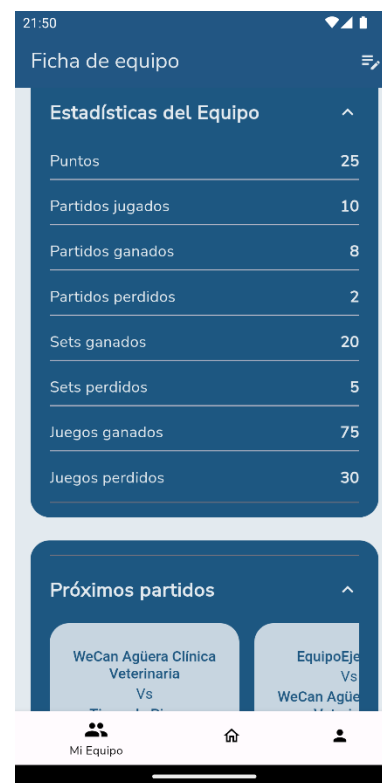
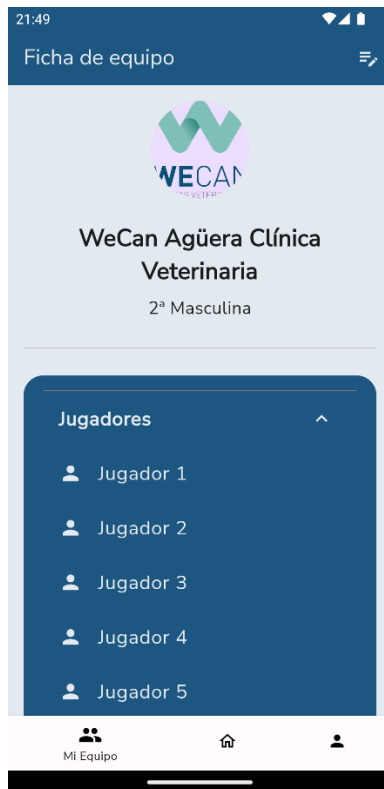
Información del perfil

Muestra la información confidencial del usuario actual, y permite modificarla. Todos los campos están validados.



Pantalla de ficha de equipo

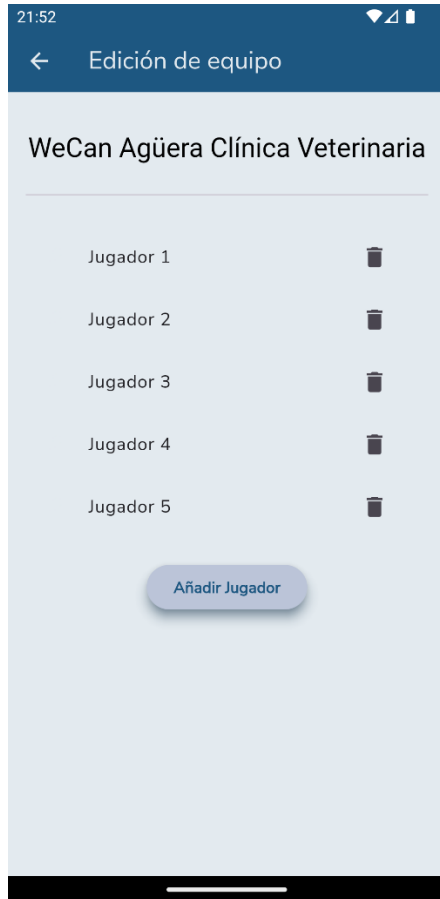
Muestra información sobre un equipo, en este caso se ha accedido desde el menú inferior, así que muestra la información de mi equipo, contiene listas desplegables donde ver la información correspondiente.



Al pulsar sobre un jugador se abrirá la ficha de dicho jugador



Si el usuario es capitán del equipo, podrá añadir y eliminar jugadores de su propio equipo mediante el botón superior derecho en la ficha de equipo.



7.2 Manual de instalación (para desarrollador, sin APK)

Requisitos Previos

Antes de comenzar con la instalación, hay que asegurarse de tener los siguientes componentes instalados y configurados:

- *Docker*.
- *Visual Studio* con el entorno de desarrollo para *Dart* y *Flutter* correctamente configurado.
- Conexión a Internet para descargar las imágenes y dependencias necesarias.

Paso 1: Despliegue de la Base de Datos y API

1. Tener el archivo *docker-compose.yml* que se encuentra en el repositorio.

2. **Ejecuta *Docker Compose***

En la terminal, sitúate en el directorio donde has guardado el archivo *docker-compose.yml* y ejecuta el siguiente comando:

```
$ docker-compose up -d
```

Esto descargará las imágenes de Docker necesarias y levantará los contenedores para la base de datos *MySQL* y la *API*.

3. **Ejecuta el script de base de datos**

A continuación, hay que conectarse al contenedor de MySQL y ejecutar el script *SQL* para crear la base de datos, insertar datos de prueba y crear los triggers correspondientes.

Usa el siguiente comando para ingresar al contenedor MySQL:

```
docker exec -it <container_id> bash  
mysql -u padeluser -p pappdelBBDD < /path/script_pappdel.sql
```

```
password: padelpassword
```

Paso 2: Instalación de la Aplicación en un Teléfono Móvil Físico o Emulador

1. **Configura el entorno de Dart y Flutter**
2. **Clona el repositorio de la aplicación**
3. **Actualiza la dirección de la API**

Abre el proyecto en Visual Studio y navega al archivo de configuración donde se define la URL de la API. Actualiza la variable global con la dirección IP de la máquina donde se están ejecutando los contenedores de Docker.

```
static const ip = "127.0.0.1" <- esta IP  
const String apiUrl = "http://%ip:8080";
```

4. Instala la aplicación en el dispositivo

Conecta tu teléfono móvil físico al mismo red Wi-Fi que tu máquina de desarrollo, o configura un emulador en Visual Studio. Asegúrate de que el dispositivo esté detectado por Flutter:

```
flutter devices
```

Instala la aplicación mediante el botón de run y eligiendo el dispositivo o ejecutando:

```
flutter run
```

Con estos pasos, se debería tener una instalación completa y funcional de la aplicación del proyecto.

8 Conclusiones y posibles ampliaciones

El desarrollo de este proyecto ha sido un proceso enriquecedor que me ha permitido alcanzar una serie de objetivos clave, además de proporcionar importantes aprendizajes y avances en diversas áreas, sobre todo en cuanto a despliegue en remoto.

A lo largo del proyecto, se ha logrado consolidar una aplicación funcional y eficiente que se adapta a las necesidades específicas de los usuarios de las ligas deportivas de pádel.

Posibles Ampliaciones

Aunque el proyecto ha alcanzado unas funcionalidades básicas, existen varias áreas en las que se pueden realizar mejoras para aumentar aún más su funcionalidad y usabilidad.

1. **Administración total por parte de los administradores de la liga.**
2. **Ampliación de ligas:** El poder adaptar la aplicación a cualquier empresa privada que requiera de un software para gestionar ligas y dar un valor añadido a su competición. Que cada liga pague una cuota por el uso de la app, y que se puedan crear torneos a partir de esa cuota, cobrando también un porcentaje del valor de la inscripción a dicha competición.
3. **Integración con Redes Sociales:** La capacidad de compartir resultados y eventos en redes sociales puede aumentar la visibilidad de las ligas y atraer a más participantes.
4. **Módulo de Análisis Avanzado:** Implementar un módulo de análisis que utilice técnicas de Big Data y aprendizaje automático para proporcionar estadísticas avanzadas y predicciones puede ofrecer un valor añadido a los usuarios.
5. **Sistema de Inscripción y Pago en Línea:** La incorporación de un sistema de inscripción y pago en línea puede facilitar la gestión de nuevas inscripciones y el manejo de cuotas.
6. **Mejoras en la Seguridad:** Implementar medidas de seguridad adicionales, como autenticación de dos factores y encriptación avanzada de datos.
7. **Reportes Personalizados:** Permitir a los usuarios generar reportes personalizados según sus necesidades y problemas.

9 Bibliografía

- **Nielsen, J. (1995).** *10 Usability Heuristics for User Interface Design*. Nielsen Norman Group. Enlace: <https://www.nngroup.com/articles/ten-usability-heuristics/>.
- **Nielsen Norman Group. (2018).** *The Definition of User Experience (UX)*. Nielsen Norman Group. Enlace: <https://www.nngroup.com/articles/definition-user-experience/>.
- **Deploy Java + Spring Boot + Mysql + Docker:**
<https://www.youtube.com/watch?v=cMXTd6PoFpo>
- **Autenticación de usuarios mediante Spring Security + JWT**
Authentication
https://www.youtube.com/watch?v=nwqQYCM4YT8&list=PLx89vzy-Ta0qIf_swgzVY4z2PcqGDduO&index=6
- **Widgets útiles:** <https://www.youtube.com/watch?v=hDVZykw113I>
- **Documentación oficial de Flutter:** <https://docs.flutter.dev/ui/widgets>
- **Iconos y fuentes de MaterialDesign:**
<https://fonts.google.com/icons?icon.size=24&icon.color=%23e8eae8&icon.plaform=web>
- **Documentación sobre API's recibida en clase**

10 Anexos

Enlace al proyecto en GitHub

- <https://github.com/rauliscar/pappdel.git>