Nombre: Manuel Palomeque Duarte

**Proyecto Final de Grado DAM**



Contenido

[1. Identificación de necesidades 2](#_Toc200748220)

[1.1. Contextualización 2](#_Toc200748221)

[1.2. Tipo de Empresa 2](#_Toc200748222)

[1.3. Necesidades el proyecto 3](#_Toc200748223)

[1.4. Análisis de competencia y diferenciación con nosotros 4](#_Toc200748224)

[1.5. ¿Por qué elegir este proyecto? 5](#_Toc200748225)

[1.6. Oportunidades de negocio en el sector 5](#_Toc200748226)

[1.6. Características específicas 6](#_Toc200748227)

[1.7. Obligaciones fiscales, laborales y de prevención de riesgos (PRL) 7](#_Toc200748228)

[1.8. Ayudas y subvenciones 8](#_Toc200748229)

[2. Diseño del proyecto 10](#_Toc200748230)

[2.1. Contenido y aspectos principales 10](#_Toc200748231)

[2.2. Viabilidad técnica 10](#_Toc200748232)

[2.3. Fases del proyecto 11](#_Toc200748233)

[2.4. Objetivos del proyecto 11](#_Toc200748234)

[2.5. Recursos de hardware y software 11](#_Toc200748235)

[2.6. Estudio económico 12](#_Toc200748236)

[2.7. Modelo de solución 13](#_Toc200748237)

[2.8. Gráficos y visualizaciones 14](#_Toc200748238)

[3. Ejecución del proyecto 16](#_Toc200748239)

[3.1. Planificación 16](#_Toc200748240)

[3.2. Riesgos 17](#_Toc200748241)

[3.3. Documentación 18](#_Toc200748242)

[3.3.1. Ficheros de configuración 18](#_Toc200748243)

[3.3.2. Características técnicas 22](#_Toc200748244)

[3.3.3. Código fuente 24](#_Toc200748245)

[3.3.4. Política de seguridad implementada 26](#_Toc200748246)

[4. Screenshots de la aplicación 28](#_Toc200748247)

[5. Incidencias 36](#_Toc200748248)

# 1. Identificación de necesidades

## 1.1. Contextualización

El proyecto desarrollado se sitúa en el marco del sector de la actividad física, el deporte y el bienestar personal, un sector en constante crecimiento y transformación. Dentro de este ámbito, las empresas de gestión deportiva —como gimnasios, centros de entrenamiento, academias de yoga, pilates, artes marciales o incluso centros culturales con actividades físicas programadas— han visto en la tecnología un aliado estratégico para mejorar sus procesos y su relación con los clientes.

En los últimos años, especialmente tras la pandemia de la COVID-19, se ha acelerado la necesidad de implementar sistemas digitales que permitan una gestión más eficiente del aforo, las reservas y la trazabilidad de usuarios. Esto no solo responde a una necesidad sanitaria o de control, sino también a una demanda creciente de los usuarios, quienes esperan contar con herramientas móviles rápidas, intuitivas y accesibles para interactuar con los servicios que consumen.

En este contexto, la mayoría de grandes cadenas de gimnasios ya cuentan con apps propias que permiten al usuario gestionar sus reservas, suscripciones y datos personales. Sin embargo, las pequeñas y medianas empresas del sector no disponen habitualmente de los recursos técnicos ni económicos para desarrollar o mantener una aplicación personalizada, lo que las deja en desventaja frente a sus competidores.

Este proyecto busca precisamente reducir esa brecha tecnológica, ofreciendo una solución adaptable, económica y funcional para gimnasios de tamaño pequeño o medio que desean ofrecer a sus usuarios una plataforma sencilla para consultar clases, reservar su plaza y gestionar su actividad desde una app Android, sin depender de software externo o costosas plataformas de terceros.

Además, el sistema está diseñado con tecnologías ampliamente utilizadas y de fácil mantenimiento (PHP, MySQL, Android Java), lo cual facilita su implementación por parte de personal técnico local, sin necesidad de depender de proveedores externos o licencias costosas. Esto convierte el proyecto en una propuesta accesible, escalable y con gran potencial de aplicación práctica en el sector.

La herramienta también permite a la empresa:

* Tener control en tiempo real sobre ocupación y reservas.
* Mejorar la planificación y distribución de clases.
* Obtener datos que pueden ser útiles para tomar decisiones de negocio (horarios con más demanda, clases más reservadas, etc.).

## 1.2. Tipo de Empresa

La empresa que desarrollará y comercializará este proyecto será una iniciativa emprendedora de base tecnológica, inicialmente constituida como autónomo o, en una segunda fase de consolidación y crecimiento, como Sociedad Limitada (SL).

El modelo de negocio está orientado a ofrecer soluciones digitales accesibles a pequeñas y medianas empresas del sector del deporte y bienestar, mediante un sistema de venta directa, modelo SaaS (Software as a Service) o marca blanca personalizada. La visión de crecimiento contempla:

* **Fase inicial (1-2 años):** desarrollo del producto mínimo viable (MVP), validación con clientes reales, primeras ventas y optimización del sistema.
* **Fase de expansión (3-5 años):** crecimiento comercial a nivel nacional, ampliación de servicios y funcionalidades, formación de equipo técnico y administrativo, y constitución como SL para acceder a financiación externa, subvenciones o inversores privados.

La empresa se enfocará no solo en el desarrollo técnico, sino también en la formación, acompañamiento e instalación del sistema, especialmente en clientes con escasos conocimientos digitales, brindando un enfoque integral que refuerce la fidelización y el valor percibido del producto.

Estas organizaciones suelen contar con una plantilla reducida, una estructura de gestión sencilla y recursos técnicos y financieros limitados, lo que les dificulta acceder a soluciones de software costosas o de gran escala como las que utilizan las grandes cadenas de fitness. A menudo, la gestión de sus reservas, clases y horarios se realiza de forma manual, mediante hojas impresas, llamadas telefónicas o mensajes de WhatsApp, lo cual genera errores, pérdidas de información, overbooking o una experiencia poco profesional para el cliente.

## 1.3. Necesidades el proyecto

La necesidad principal que cubre este proyecto es la de automatizar y facilitar la gestión de reservas de clases o actividades con plazas limitadas, desde una plataforma accesible tanto para los usuarios como para los administradores del centro. Esta necesidad surge de varios factores identificados en el día a día de estas organizaciones:

1. **Control de aforo y plazas**: Muchos centros deben limitar la cantidad de personas por clase por cuestiones de espacio, calidad del servicio o normativa legal (especialmente post-COVID). Llevar este control manualmente genera conflictos, confusión o duplicidad de reservas.
2. **Acceso desde el móvil**: El perfil del cliente actual está acostumbrado a gestionar sus servicios desde el teléfono. Ofrecer una aplicación móvil para reservar clases mejora notablemente la experiencia del usuario y la imagen del negocio.
3. **Reducción de carga administrativa**: Las reservas por teléfono, correo o en papel consumen tiempo y recursos del personal. Automatizar este proceso reduce errores, mejora la organización y libera tiempo para tareas más productivas.
4. **Mejora del servicio al cliente**: El sistema permite al usuario ver en todo momento qué clases están disponibles, cuántas plazas quedan y cuáles ha reservado. Esto genera confianza, transparencia y fidelidad.
5. **Recogida de datos y toma de decisiones**: Con el sistema, se pueden registrar patrones de reserva, clases más demandadas, horarios con menos asistencia, etc., lo cual ofrece información muy útil para ajustar la oferta a las preferencias de los usuarios.
6. **Imagen moderna y profesional**: Contar con una app propia, aunque sea básica, transmite una imagen de modernidad, organización y profesionalismo, algo cada vez más valorado por los clientes, incluso en negocios pequeños.

La empresa ideal para este sistema es una que:

* Ofrezca clases o sesiones por horarios con cupos limitados.
* Cuente con una base de clientes activa y recurrente.
* Desee optimizar su proceso de reservas y control de asistencia.
* Busque una solución tecnológica sencilla, económica y autogestionable.
* Valore la digitalización de su negocio sin depender de plataformas externas.

El sistema está pensado para poder ser instalado en un servidor local (por ejemplo, con XAMPP) o en un hosting web básico, y gestionado por el propio personal administrativo o técnico del centro, sin necesidad de conocimientos avanzados. Esta flexibilidad lo convierte en una herramienta viable tanto para centros urbanos como para instalaciones deportivas en zonas rurales, donde la oferta tecnológica suele ser aún más limitada.

## 1.4. Análisis de competencia y diferenciación con nosotros

Actualmente existen varias plataformas y apps que ofrecen sistemas de gestión para gimnasios y centros deportivos. Algunas de las más conocidas en el mercado español e internacional son:

* **Trainingym**
* **Resasports**
* **Virtuagym**
* **BookyWay**
* **Gympass**

**Principales características de la competencia:**

* Suelen ofrecer amplias funcionalidades (gestión de pagos, CRM, seguimiento del cliente, marketing, etc.).
* Funcionan en modelo SaaS con suscripciones mensuales.
* Requieren conexión constante a internet y servidores externos.
* Pueden resultar complejas o costosas para pequeños negocios.

**Diferenciación del proyecto:**

* **Enfoque en centros pequeños/medianos** con recursos limitados.
* **Instalación local o en hosting propio**, sin necesidad de pagar licencias mensuales.
* **Autonomía técnica**: puede ser gestionado internamente sin necesidad de soporte externo constante.
* **Aplicación sencilla** y pensada desde una lógica de uso real en centros con plantilla reducida.
* **Posibilidad de personalización** a bajo coste.
* **Escalabilidad modular**, lo que permite añadir funciones según las necesidades del negocio sin cambiar de plataforma.

## 1.5. ¿Por qué elegir este proyecto?

El proyecto desarrollado responde de forma directa, concreta y eficaz a las necesidades identificadas en el apartado anterior. La implementación de un sistema de gestión de reservas para clases en un gimnasio permite automatizar procesos que tradicionalmente se realizaban de forma manual o mediante sistemas poco eficientes.

La necesidad de controlar el aforo se resuelve mediante la integración de un sistema de cupos por clase, que evita el overbooking y facilita al usuario conocer en tiempo real si hay plazas disponibles. La necesidad de acceso móvil se cubre a través de una aplicación Android desarrollada en Java, pensada para ser intuitiva y de fácil acceso desde cualquier dispositivo móvil.

Asimismo, se resuelve la necesidad de ahorrar tiempo y reducir errores administrativos, ya que las reservas se registran automáticamente en la base de datos, y el sistema impide duplicidad de reservas o errores humanos típicos de una gestión manual. El usuario final puede hacer todo desde su propio móvil, sin necesidad de contacto directo con el personal, lo que también se alinea con la tendencia actual de autoservicio digital.

Por otra parte, el sistema contempla una estructura escalable, que permite ampliar funcionalidades en el futuro como notificaciones, pagos online, gestión de cancelaciones o informes estadísticos, atendiendo a otras necesidades que puedan surgir conforme evolucione el negocio o cambien las circunstancias.

## 1.6. Oportunidades de negocio en el sector

El sector de la actividad física y la gestión deportiva ofrece un amplio abanico de oportunidades de negocio, especialmente en el ámbito de la digitalización de procesos. Muchas pequeñas empresas carecen de sistemas tecnológicos propios, por lo que este tipo de soluciones tienen un gran potencial para ser comercializadas.

Entre las oportunidades destacables:

* **Software como servicio (SaaS)**: El sistema podría ofrecerse mediante un modelo de suscripción mensual o anual, permitiendo a gimnasios usar la plataforma sin necesidad de grandes inversiones iniciales.
* **Modelo de personalización a medida**: Ofrecer la app con el branding y características propias de cada gimnasio (marca blanca).
* **Ampliación a otros sectores**: El mismo sistema podría adaptarse fácilmente para academias, escuelas de danza, centros culturales, clínicas de fisioterapia con citas por hora, etc.
* **Venta como producto cerrado**: Para centros que deseen tener el control total sobre el sistema, se podría comercializar como una solución "llave en mano".
* **Servicios complementarios**: A futuro, se podrían incorporar módulos de marketing (envío de recordatorios, promociones), informes estadísticos, pagos en línea o incluso integración con redes sociales.

La creciente conciencia sobre la salud, el auge del deporte y la demanda de experiencias tecnológicas por parte de los usuarios son factores que **favorecen notablemente la adopción de este tipo de sistemas**, por lo que el proyecto no solo tiene aplicabilidad práctica, sino también **potencial de rentabilidad comercial**.

## 1.6. Características específicas

Las principales características del proyecto son:

1. Aplicación Android en Java para el usuario final.
2. Sistema de login y registro de usuarios.
3. Visualización de clases disponibles y plazas.
4. Sistema de reserva y cancelación en tiempo real.
5. Base de datos en MySQL y backend en PHP bajo servidor local con XAMPP.
6. Panel de administración opcional para el control de usuarios y reservas.
7. Arquitectura modular que permite ampliación futura.

## 1.7. Obligaciones fiscales, laborales y de prevención de riesgos (PRL)

Al constituir la empresa como **autónomo** o **SL**, se deben contemplar diversas obligaciones legales y administrativas:

**Fiscalidad (Impuestos)**

* **Autónomo**:
  + Alta en Hacienda (modelo 036 o 037) y Seguridad Social.
  + Declaraciones trimestrales de IVA (modelo 303) e IRPF (modelo 130).
  + Declaración anual de operaciones con terceros (modelo 347, si aplica).
* **Sociedad Limitada**:
  + Impuesto de Sociedades (25% sobre beneficios).
  + IVA trimestral.
  + Declaración de retenciones de IRPF por trabajadores o profesionales externos.
  + Llevanza de contabilidad mercantil y presentación de cuentas anuales en el Registro Mercantil.

**Obligaciones laborales:**

* En caso de contratación de personal:
  + Registro en la Seguridad Social.
  + Contratos laborales conforme al Estatuto de los Trabajadores.
  + Pago de seguros sociales.
  + Prevención de Riesgos Laborales (PRL): evaluación de riesgos, formación, coordinación de actividades empresariales.

**Prevención de Riesgos Laborales (PRL):**

Aunque en el caso de empresas sin empleados no es obligatorio contratar un servicio de PRL, si se contrata personal:

* Es obligatorio contar con un plan de prevención.
* Se debe formar a los trabajadores en riesgos específicos de su puesto.
* Es obligatorio realizar reconocimientos médicos (según el puesto).
* Cumplimiento de la Ley de Prevención de Riesgos Laborales (Ley 31/1995).

**Protección de Datos (LOPD-GDD y RGPD):**

El sistema gestiona datos personales de usuarios (nombres, reservas, correos electrónicos), por lo que se deben cumplir las siguientes normativas:

* Registro de actividades de tratamiento.
* Avisos legales y políticas de privacidad.
* Consentimiento informado para el tratamiento de datos.
* Medidas de seguridad técnicas y organizativas (cifrado, contraseñas, etc.).

## 1.8. Ayudas y subvenciones

El desarrollo e implementación de un sistema de gestión de reservas para gimnasios puede beneficiarse de diversas ayudas y subvenciones públicas, tanto a nivel nacional como autonómico y europeo, especialmente si se plantea como un proyecto emprendedor o se orienta a la transformación digital de pequeñas empresas.

A continuación, se detallan las principales vías de financiación o apoyo económico que podrían aplicarse al proyecto, ya sea para su desarrollo, implementación o expansión comercial:

* **Kit Digital (España)**

El programa Kit Digital, impulsado por el Gobierno de España y financiado por los fondos Next Generation EU, está diseñado para ayudar a las pequeñas empresas, microempresas y autónomos a digitalizar sus negocios. Este sistema podría incluirse dentro de las categorías de “Gestión de procesos” o “Presencia en internet y digitalización de relaciones con clientes”.

* + **Beneficiarios**: Autónomos y pymes de hasta 49 empleados.
  + **Importe subvencionable**: Hasta 2.000 € en negocios pequeños.
  + **Aplicación al proyecto**: La app puede considerarse una herramienta de digitalización de procesos internos, y por tanto ser financiada con este bono.
* **Subvenciones autonómicas y locales**

Cada comunidad autónoma en España ofrece programas de fomento al emprendimiento tecnológico, innovación o digitalización, así como ayudas específicas para jóvenes emprendedores o empresas de base tecnológica.

Por ejemplo:

**Andalucía Emprende**, **Programa TICCámaras (Cámaras de Comercio)**.

Ayuntamientos o diputaciones también pueden disponer de ayudas puntuales para modernización de negocios o digitalización de comercios.

Estas subvenciones pueden cubrir gastos de desarrollo, adquisición de equipos, formación, o contratación de servicios profesionales.

* **Programa de Garantía Juvenil / SEPE**

El SEPE y otras entidades públicas impulsan programas dirigidos a jóvenes desempleados inscritos en Garantía Juvenil o en búsqueda de su primer empleo, que decidan emprender o poner en marcha una actividad tecnológica.

* Apoyo económico al emprendimiento juvenil.
* Programas de formación y mentoring.
* Bonificaciones en la cotización de la seguridad social si se constituye como autónomo.
* Ayudas europeas para innovación y TIC. Existen programas europeos como:
  + Erasmus+ for Young Entrepreneurs.
  + Digital Europe Programme.
  + EIC Accelerator (anteriormente SME Instrument).

Estos fondos son más complejos de solicitar, pero si el proyecto crece o se plantea a nivel internacional o colaborativo, podrían representar una fuente importante de financiación a medio plazo.

# 2. Diseño del proyecto

## 2.1. Contenido y aspectos principales

El proyecto se centra en el desarrollo e implementación de una **solución tecnológica integral para la gestión de reservas y asistencia a clases** en centros deportivos de tamaño pequeño o medio. La solución incluye:

* Aplicación móvil nativa para Android.
* Backend en PHP con conexión a base de datos MySQL.
* API REST para comunicación entre la app y el servidor.

Se abordan aspectos clave como:

* Gestión de usuarios y control de acceso.
* Consulta de actividades, plazas disponibles y horarios.
* Reservas y cancelaciones.
* Registro de la actividad del cliente.

## 2.2. Viabilidad técnica

**Viabilidad tecnológica:**

* Lenguajes utilizados: Java (Android), PHP (backend), SQL (MySQL).
* Infraestructura mínima requerida: servidor local o hosting con soporte PHP y MySQL.
* Compatibilidad: sistema compatible con Android 6.0 o superior, navegadores web modernos para el panel administrativo.
* Seguridad: implementación de validaciones, control de sesiones.

**Viabilidad operativa:**

* Sistema diseñado para ser autogestionado por el personal del centro sin conocimientos avanzados.
* Mantenimiento sencillo mediante tecnologías ampliamente documentadas.

## 2.3. Fases del proyecto

| **Fase** | **Descripción** | **Duración Estimada** |
| --- | --- | --- |
| **Análisis** | Recolección de requisitos, análisis del entorno y del usuario final. | 1 semana |
| **Diseño técnico** | Arquitectura del sistema, modelo de datos, diseño UI/UX y estructura del backend. | 1 semana |
| **Desarrollo** | Codificación de la app Android, backend PHP, base de datos y API. | 3 semanas |
| **Integración y pruebas** | Test unitarios, test de integración, detección de errores, pruebas funcionales. | 1 semana |
| **Documentación** | Elaboración de manuales de usuario, técnico, y documentación del código. | 4 días |
| **Despliegue** | Instalación en entorno real/local y pruebas finales con usuarios. | 3 días |
| **Evaluación final** | Recogida de feedback, mejoras, y cierre del proyecto. | 2 días |

## 2.4. Objetivos del proyecto

**Objetivo general:** Desarrollar e implementar una solución digital económica y funcional para la gestión de reservas en centros deportivos de tamaño pequeño o medio.

**Objetivos específicos:**

Desarrollar una aplicación móvil intuitiva para usuarios.

Diseñar un backend eficiente para gestionar datos y reservas.

Implementar una base de datos segura y escalable.

Garantizar una arquitectura modular para futuras ampliaciones.

## 2.5. Recursos de hardware y software

* **Hardware mínimo para entorno de desarrollo:**
  + Ordenador con mínimo 8 GB de RAM, procesador i5, SSD.
  + Dispositivo Android (móvil o emulador).
  + Red local para pruebas internas.
* **Hardware del entorno de producción (local o servidor web):**
  + Servidor con mínimo 2 GB de RAM, 1 vCPU.
  + Almacenamiento de 20 GB.
  + Conectividad con IP estática o dominio configurado.
* **Software:**
  + Android Studio.
  + XAMPP (Apache + MySQL + PHP).
  + Visual Studio Code.
  + Navegadores web modernos.
* **Materiales:**
  + Licencias open-source (XAMPP, Android SDK).
  + Internet
  + Ordenador
* **Recursos humanos:**
  + 1 desarrollador Android.
  + 1 desarrollador backend.
  + 1 diseñador UI/UX (freelance o polivalente).
  + 1 responsable de pruebas y documentación.
  + 1 administrador técnico del centro (cliente).

## 2.6. Estudio económico

|  |  |
| --- | --- |
| **Concepto** | **Coste estimado (€)** |
| Desarrollo app Android | 1.200 € |
| Desarrollo backend y BBDD | 800 € |
| Diseño UI/UX y prototipado | 300 € |
| Pruebas y documentación | 200 € |
| Servidor y hosting anual | 120 € |
| Publicación en Google Play (único pago) | 25 € |
| Mantenimiento técnico (anual) | 300 € |
| **TOTAL** | **2.945 €** |

El proyecto podría autofinanciarse inicialmente o presentarse a:

* Kit Digital (hasta 2.000 €).
* Ayudas al autoempleo tecnológico o emprendimiento.
* Subvenciones autonómicas para jóvenes emprendedores.
* Apoyo familiar o fondos propios.

## 2.7. Modelo de solución

**Descripción técnica**

El sistema consiste en una aplicación móvil Android conectada mediante API REST a un backend en PHP que gestiona una base de datos MySQL. El usuario se registra, consulta actividades disponibles y realiza reservas. El administrador puede modificar clases, visualizar asistencia, y descargar reportes.

**Tablas**

* **Dirección IP**

|  |  |  |
| --- | --- | --- |
| **Dispositivo** | **IP estimada** | **Función** |
| Servidor XAMPP | 192.168.1.1 | Backend y BBDD |
| Móviles | DHCP | Clientes Android |
|  |  |  |

* **Base de datos**

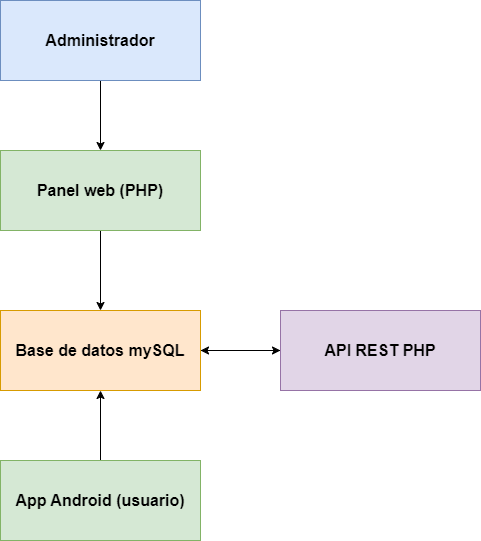
|  |  |
| --- | --- |
| **Tabla** | **Campos principales** |
| usuarios | id, nombre, email, contraseña |
| clases | id, nombre, fecha, hora, cupo\_actual, cupo\_maximo |
| reservas | id, id\_usuario, id\_clase, fecha\_reserva |

* **Usuarios y permisos:**

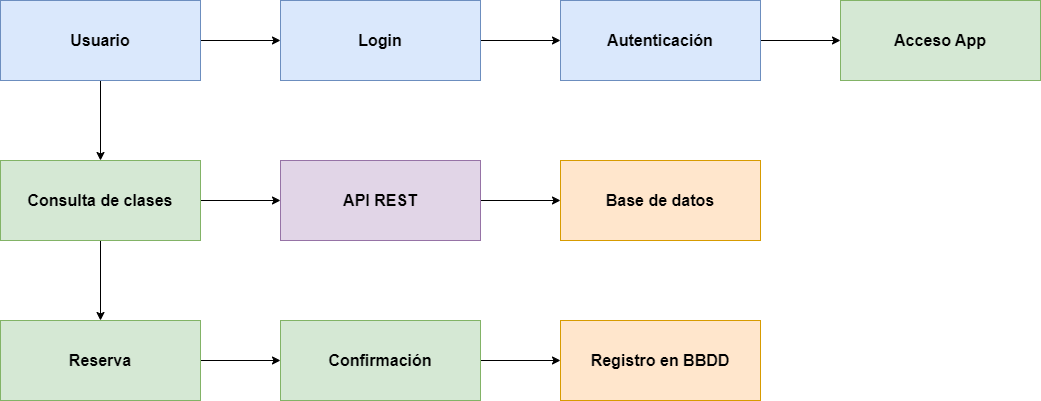
|  |  |
| --- | --- |
| **Rol** | **Permisos** |
| Usuario | Ver clases, reservar, cancelar, ver historial |
| Admin | Añadir/modificar clases, ver usuarios, estadísticas |

## 2.8. Gráficos y visualizaciones

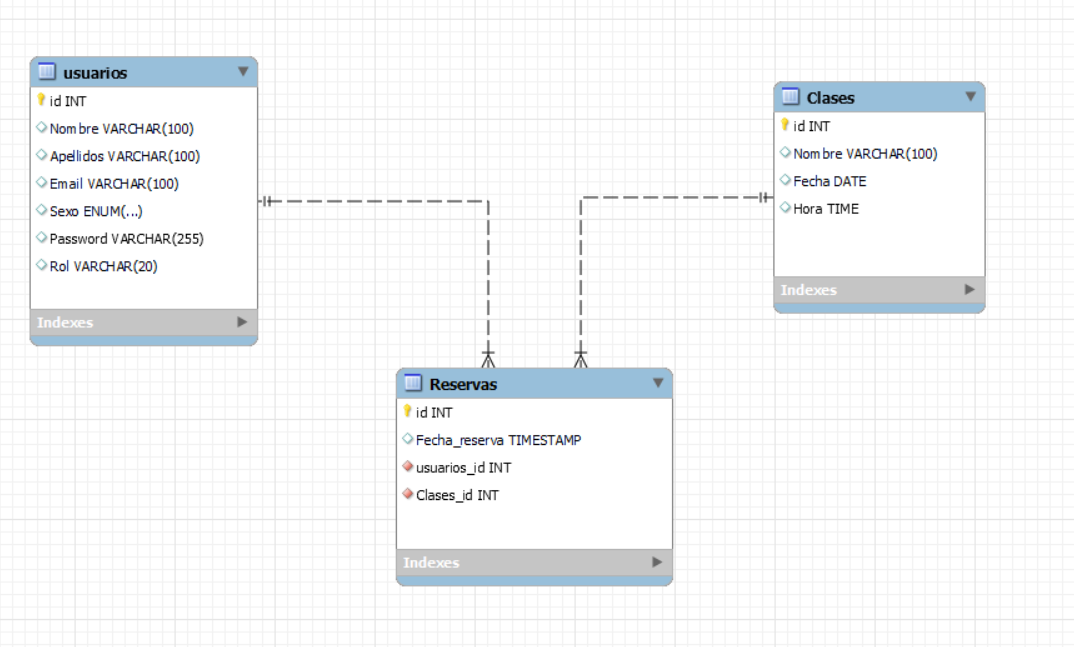
**Organigrama del sistema:**

****

**Diagrama de flujo de datos:**

****

**Modelo entidad-relación (E/R):**



# 3. Ejecución del proyecto

## 3.1. Planificación

Durante la ejecución del proyecto se estableció una estructura de desarrollo secuencial e interdependiente, organizada en fases progresivas y alineadas con los objetivos funcionales del sistema. El proceso comenzó con una etapa de análisis, donde se identificaron los requisitos esenciales para que la aplicación cumpliera con su propósito: ofrecer a los usuarios del gimnasio una herramienta móvil para gestionar reservas de clases deportivas de forma autónoma, eficiente y visualmente clara.

A partir de esa definición inicial, se diseñó el modelo de datos que sustenta toda la lógica del sistema. Este diseño se basó en entidades clave como usuarios, clases y reservas, estableciendo relaciones entre ellas que garantizaran la integridad referencial y permitieran operaciones como registrar nuevas clases, asignar cupos, verificar inscripciones y gestionar cancelaciones. La elaboración de este esquema no solo fue técnica, sino que también respondió a flujos de uso reales y supuestos de interacción por parte del usuario final. La base de datos fue implementada en **MySQL** y gestionada a través de **phpMyAdmin** bajo el entorno local proporcionado por **XAMPP**.

Con la estructura de datos definida, se procedió al desarrollo del backend. Aquí se implementaron una serie de endpoints PHP que componen una API RESTful. Cada ruta fue diseñada para cubrir funcionalidades específicas: inicio de sesión, registro de usuario, obtención de clases disponibles, inscripción a clases, cancelación de reservas, y actualización de perfil. Esta etapa dependía directamente del modelo de datos, ya que todas las operaciones requerían una base sólida para operar de manera coherente y segura.

La implementación del backend supuso la necesidad de recursos como un servidor local (Apache), el motor de base de datos (MySQL), y herramientas de prueba como **Postman** para validar el formato y comportamiento de las respuestas. También se requirió establecer una capa de seguridad básica: validaciones de entrada, respuestas controladas en formato JSON, y manejo de errores HTTP para facilitar su interpretación desde el cliente.

En paralelo, se desarrolló la aplicación Android en Java. Antes de escribir código funcional, se abordó el diseño visual mediante interfaces XML. Se crearon pantallas para todas las vistas necesarias: login, registro, menú principal, listado de clases, reservas activas, perfil de usuario y edición de datos. Este diseño visual no fue meramente estético, sino que sentó las bases para el desarrollo de la lógica interna. De este modo, el equipo de desarrollo pudo implementar la lógica orientada a eventos, las conexiones HTTP con la API, y la gestión del estado del usuario dentro de la aplicación.

Durante esta fase, se hizo uso de **Android Studio**, emuladores virtuales y dispositivos físicos con permisos de depuración activados. Se requería permiso de acceso a internet declarado en el AndroidManifest.xml, imprescindible para establecer comunicación con el backend. La funcionalidad del modo oscuro también se implementó en esta fase, con persistencia por usuario utilizando SharedPreferences y definición de temas en los archivos de recursos XML (themes.xml, colors.xml, values-night/).

Cada etapa del desarrollo estuvo sujeta a su propio protocolo de actuación. En el caso de la base de datos, se aplicaron buenas prácticas de modelado, normalización y pruebas con datos de ejemplo. En el backend, se estableció un flujo de validación con pruebas unitarias y manuales para cada endpoint. En el desarrollo Android, se verificó el flujo completo de navegación, la correcta visualización de los estados de inscripción, y la respuesta a eventos como errores de red o inputs inválidos. Todas las actividades fueron registradas, documentadas y respaldadas, tanto en copias locales como en versiones incrementales del proyecto.

Este proceso disciplinado y progresivo garantizó que cada parte del sistema estuviera alineada, funcionando sobre una base robusta, reutilizable y preparada para su mantenimiento o ampliación futura.

## 3.2. Riesgos

En todo proceso de desarrollo tecnológico, anticiparse a los posibles riesgos operativos, técnicos y organizativos es una práctica esencial para asegurar el éxito del proyecto. En el caso concreto del desarrollo de este proyecto, se identificaron desde el inicio una serie de amenazas potenciales que, de materializarse, podrían comprometer la estabilidad del sistema, la continuidad del desarrollo o la calidad del producto final. Estos riesgos no fueron concebidos como obstáculos inevitables, sino como situaciones prevenibles mediante un enfoque proactivo y estratégico.

Uno de los primeros riesgos detectados fue la dependencia de tecnologías locales, concretamente el uso de XAMPP para el backend. Aunque es una herramienta válida para entornos de desarrollo, su configuración local podría generar limitaciones en la escalabilidad y en el trabajo en equipo si se pretendiese distribuir el desarrollo entre varios entornos o colaboradores. Para mitigar este riesgo, se estableció una estructura clara de archivos y se realizaron respaldos periódicos de la base de datos y del directorio del servidor, de forma que cualquier problema técnico pudiera revertirse sin pérdidas significativas.

Otro punto crítico fue el riesgo asociado a la gestión de la persistencia de datos y sesiones en Android. En entornos móviles, una mala gestión de SharedPreferences o errores en la recuperación de estado pueden derivar en fallos de experiencia de usuario o incluso pérdida de datos personales. Para prevenirlo, se validó rigurosamente cada flujo de escritura y lectura de preferencias, asegurando que los datos quedaran correctamente almacenados y que se comportaran de forma coherente tras el cierre o reinicio de la aplicación.

También se contemplaron los riesgos de seguridad básica, propios de aplicaciones conectadas a redes. Aunque GymSpot no trata con información crítica como datos bancarios, sí gestiona credenciales de usuario y datos personales como nombre y correo electrónico. Para prevenir vulnerabilidades, se estableció un uso obligatorio de POST en operaciones sensibles, se evitaron exposiciones de datos innecesarias en respuestas JSON, y se validaron todos los campos recibidos en el servidor, evitando inyecciones SQL mediante el uso de consultas preparadas con PDO.

Desde una perspectiva operativa, se identificó el riesgo de bloqueos en el desarrollo por falta de pruebas progresivas. A menudo, en proyectos donde se retrasa la fase de validación, los errores se acumulan y su corrección se vuelve más costosa. Para mitigar este escenario, se adoptó un enfoque iterativo: cada módulo funcional fue validado de forma independiente antes de ser integrado con el resto del sistema, lo que permitió localizar errores más fácilmente y asegurar el correcto funcionamiento del conjunto.

Finalmente, se consideró el riesgo vinculado a la gestión de dependencias visuales y temáticas, como el modo oscuro. La implementación de un sistema de apariencia personalizable implicaba alterar el comportamiento visual de la interfaz en tiempo real, lo cual podía generar errores de consistencia o incompatibilidades entre versiones de Android. Para prevenirlo, se utilizaron librerías y componentes oficiales del ecosistema MaterialComponents, se definieron temas en themes.xml y colors.xml, y se probó el sistema en distintos dispositivos virtuales y reales para verificar su comportamiento.

En conclusión, el proceso de identificación y gestión de riesgos no se ha limitado a un apartado teórico, sino que se ha integrado en cada etapa del desarrollo. Esto permitió anticipar problemas, reforzar la estabilidad del sistema y garantizar una experiencia sólida para el usuario final. El resultado fue un proyecto no solo funcional, sino resiliente ante imprevistos técnicos y bien documentado para su evolución futura.

## 3.3. Documentación

### 3.3.1. Ficheros de configuración

Este proyecto está compuesto por dos grandes partes: una aplicación móvil Android desarrollada en Java, y un backend PHP que gestiona la lógica del servidor y las conexiones con la base de datos MySQL.

* **Configuración Android**
* ***AndroidManifest.xml:*** Define el comportamiento global de la aplicación. Entre los elementos configurados encontramos:
  + Permisos: Necesario para que la app pueda hacer peticiones HTTP al backed.

**<uses-permission android:name="android.permission.INTERNET" />**

* + Actividades (pantallas): Se registran las Activities del proyecto:

**<?php**

**<activity**

**android:name=".LoginActivity"**

**android:exported="true">**

**<intent-filter>**

**<action android:name="android.intent.action.MAIN" />**

**<category android:name="android.intent.category.LAUNCHER" />**

**</intent-filter>**

**</activity>**

**?>**

* + Configuración de tema y seguridad de red

**android:theme="@style/Theme.ReservaClasesApp"**

**android:networkSecurityConfig="@xml/network\_security\_config"**

* ***network\_security\_config.xml***

**<network-security-config>**

**<domain-config cleartextTrafficPermitted="true">**

**<domain includeSubdomains="true">10.0.2.2</domain>**

**</domain-config>**

**</network-security-config>**

10.0.2.2 es una IP especial en Android que hace referencia al localhost de la máquina host, útil al probar la app en emuladores con un servidor local.

* ***res/values/colors.xml y res/values-night/colors.xml***

Este xml incluye la configuración que tiene la aplicación tanto en modo claro como oscuro.

* ***data\_extraction\_rules.xml y backup\_rules.xml***

Estos archivos están incluidos para futuras configuraciones de backup y restauración de datos automáticos en Android 12 o superior.

* **Configuración del backend (PHP)**
* ***db.php***

***$pdo = new PDO("mysql:host=localhost;dbname=gimnasio;charset=utf8", "root", "");***

***$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);***

Este archivo centraliza la conexión a la base de datos gimnasio. Usa PDO y manejo de errores:

* ***functions.php***

Este archivo contiene funciones reutilizables que se incluyen en todos los scripts PHP del backend. Centraliza la lógica repetitiva como generar respuestas en JSON y validar usuarios.response($status, $message, $data = []): estructura estándar de respuesta en JSON.

***<?php***

***function response($status, $message, $data = []) {***

***echo json\_encode([***

***'status' => $status,***

***'message' => $message,***

***'data' => $data***

***]);***

***exit;***

***}***

***function authUser($pdo, $email, $password) {***

***$stmt = $pdo->prepare("SELECT \* FROM usuarios WHERE email = ?");***

***$stmt->execute([$email]);***

***$user = $stmt->fetch(PDO::FETCH\_ASSOC);***

***if ($user && password\_verify($password, $user['password'])) {***

***unset($user['password']);***

***return $user;***

***}***

***return false;***

***}***

***?>***

En la carpeta de api/ tenemos la lógica del servidor. Cada archivo actúa como un endpoint de la API que responde a peticiones desde la app móvil. Todos utilizan db.php para conectarse a la base de datos y functions.php para responder en formato JSON. Aquí hay algunos ejemplos:

* ***login.php:*** Permite a los usuarios iniciar sesión en la app
  + Conecta a la base de datos
  + Busca al usuario por email.
  + Verifica la contraseña usando *password\_verify().*
  + Si es válida, devuelve *id\_usuario, nombre, apellidos, edad, sexo, email, rol.*
  + Contraseña cifrada con hash (password\_hash y verify).
  + No se expone la contraseña
* ***registro.php:*** Permite crear cuentas de usuario en la app.
  + Recibe: nombre, apellidos, edad, sexo, email, password.
  + Valida seguridad de la contraseña: 8+ caracteres, 1 mayúscula, 1 número, 1 símbolo.
  + Verifica si el correo ya existe.
  + Cifra la contraseña con password\_hash().
  + Inserta el nuevo usuario con rol "usuario".
  + Devuelve mensaje de éxito o error.
* ***clases.php:*** Obtiene el listado de clases.
  + Recupera todas las clases ordenadas por fecha y hora.
  + Por cada clase, comprueba si el usuario está inscrito.
  + Devuelve la lista con datos como: nombre, fecha, hora, cupo y estado de
* ***editar\_usuario.php:***  Actualiza datos del perfil (nombre, email, y opcionalmente la contraseña).
  + Verifica que no haya otro usuario con el mismo correo.
  + Si hay cambio de contraseña:
    - Verifica la actual
    - Cifra la nueva
  + Actualiza nombre y email siempre.
* ***mis\_clases.php:***  Lista las clases en las que el usuario ya está inscrito.
  + Consulta con JOIN entre *reservas* y *clases*.
  + Devuelve ID, nombre, fecha y hora de cada clase registrada.
  + Todas las contraseñas se almacenan cifradas con password\_hash.
  + Se valida la identidad del usuario antes de operaciones sensibles (editar, cancelar).
  + Se impide registrarse si no hay cupo.

### 3.3.2. Características técnicas

* **Arquitectura general:**

|  |  |  |
| --- | --- | --- |
| **Componente** | **Tecnología principal** | **Descripción** |
| App móvil | Android (Java) | App Android nativo para gestión de clases |
| Backend | PHP + MySQL | API REST para manejar usuarios, clases y reservas |

* **Herramientas de desarrollo:**

|  |  |  |
| --- | --- | --- |
| **Área** | **Herramienta** | **Versión** |
| IDE móvil | Android Studio | Hedgehog o superior |
| Servidor PHP | XAMPP | PHP 7.4+ |
| Base de datos | MySQL | 5.7 o superior |
| Emulador Android | API 30-34 | Con acceso a red local |

* **Lenguaje y tecnologías utilizadas:**

|  |  |
| --- | --- |
| **Parte** | **Lenguaje/Tecnología** |
| App Android | Java, XML |
| Backend API | PHP (PDO, MySQL) |
| Base de datos | SQL (MySQL) |
| Comunicación | HTTP, JSON |
| Estilos UI | Material Design, recursos XML Android |
| Seguridad | password\_hash, password\_verify, Shared\_Preferences, validaciones manuales |

* **Requisitos del dispositivo Android**
  + Versión mínima: Android 5.0 (API 21)
  + Red: Acceso local a la IP del servidor (10.0.2.2) en emulador, o red local real)
  + Permisos necesarios: Internet
  + Modo claro/oscuro: configurable por el usuario
* **Requisitos del dispositivo Android**
  + Protocolo: HTTP
  + Formato de intercambio: JSON
  + Peticiones:
    - POST para login, registro, reservas, edición de usuario.
    - GET para consultar clases y reservas.
  + Seguridad:
    - No se expone la contraseña
    - Uso de SharedPreferences para persistir la sesión local
    - Verificación de cupos y registros en backend.

### 3.3.3. Código fuente

* **Activities JAVA**

**LoginActivity:** Pantalla de inicio de sesión. Verifica credenciales con el backend.

**RegisterActivity**: Registro de nuevos usuarios. Envía datos al backed.

**MainMenuActivity**: Pantalla principal tras iniciar sesión. Acceso a clases y perfil.

**ClasesActivity**: Muestra listado de clases. Permite registrarse en ellas.

**MisClasesActivity**: Muestra las clases a las que el usuario se ha inscrito.

**PerfilActivity**: Muestra los datos del usuario.

**EditarPerfilActivity**: Permite modificar el perfil.

**BaseActivity**: Clase base que incluye el menú del usuario (ver perfil, cerrar sesión).

* **Adaptadores y modelo**

**ClaseAdapter:** Adaptador para el RecyclerView de clases. Permite mostrar y gestionar reservas.

**Clase:** Modelo de datos que representa una clase.

* **Layouts XML**

|  |  |  |
| --- | --- | --- |
| **Archivo XML** | **Asociación** | **Descripción** |
| activity\_login.xml | LoginActivity | Diseño del formulario de acceso. |
| activity\_register.xml | RegisterActivity | Formulario de registro de usuarios. |
| activity\_main\_menu.xml | MainMenuActivity | Pantalla de bienvenida con botones. |
| activity\_clases.xml | ClasesActivity | Lista de clases (RecyclerView) y botón de volver. |
| activity\_mis\_clases.xml | MisClasesActivity | Lista de clases registradas por el usuario. |
| activity\_perfil.xml | PerfilActivity | Vista de datos del usuario. |
| activity\_editar\_perfil.xml | EditarPerfilActivity | Campos editables para nombre, correo y contraseña. |
| item\_clase.xml | ClaseAdapter | Tarjeta para mostrar clase disponible. |
| item\_mis\_clases.xml | MisClasesActivity | Vista de clase reservada con botón para cancelar. |

* **Tablas de la Base de Datos**
  + **Usuarios**
    - Campo email tiene restricción UNIQUE para evitar registros duplicados.
    - Las contraseñas se almacena con password\_hash.

|  |  |  |
| --- | --- | --- |
| **Campo** | **Tipo** | **Descripción** |
| Id | Int(11) Auto\_Increment | Clave primaria, única por usuario |
| Nombre | Varchar(100) | Nombre usuario |
| Apellidos | Varchar(100) | Apellidos usuario |
| Email | Varchar(100) | Email usuario |
| Sexo | Enum(‘hombre’, ‘mujer) | Sexo usuario |
| Password | Varchar(255) | Contraseña usuario |
| Rol | Varchar(20) | Rol |

* + **Clases**
    - El campo se incrementa/decrementa desde el backend al reservar o cancelar una clase.

|  |  |  |
| --- | --- | --- |
| **Campo** | **Tipo** | **Descripción** |
| Id | Int(11) Auto\_Increment | Clave primaria, única por clase |
| Nombre | Varchar(100) | Nombre clase |
| Fecha | Date | Fecha programada de la clase |
| Hora | Time | Hora inicio de la clase |
| Cupo\_maximo | Int | Máximo personas permitidas |
| Cupo\_actual | Int (Default 0) | Número personas inscritas |

* + **Reservas**
    - Las relaciones entre tablas nos permiten saber:
      * Que clase ha reservado un usuario
      * Cuántos usuarios están inscritos a una clase.

|  |  |  |
| --- | --- | --- |
| **Campo** | **Tipo** | **Descripción** |
| Id | INT(11) Auto\_Increment | Clave primaria de cada reserva |
| Id\_usuario | INT | Clave foránea a usuarios(id) |
| Id\_clase | INT | Clase foránea clases a clases(id) |
| Fecha\_reserva | TIMESTAMP | Fecha y hora de la reserva |

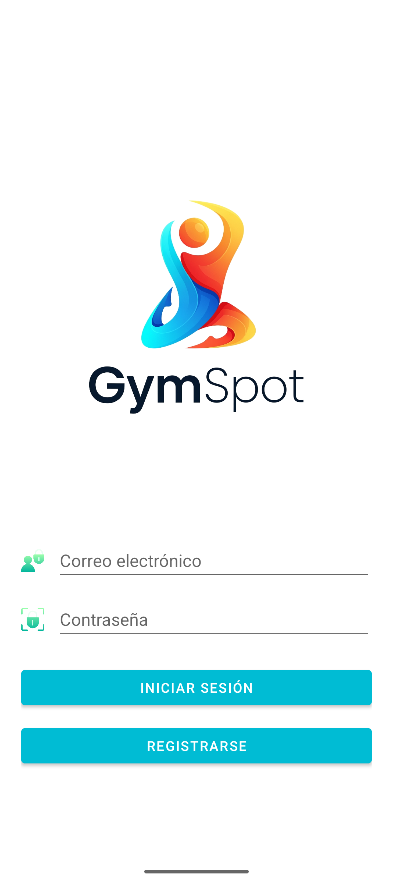
### 3.3.4. Política de seguridad implementada

La seguridad de este proyecto consta de:

* **Seguridad del backend:**
  + Cifrado de contraseñas: El backend utiliza funciones nativas de PHP para proteger contraseñas.
  + Validacion de datos en cada API: Antes de realizar cualquier operación en base de datos, se valida que los datos esperados estén presentes. Esto evita errores por llamadas incompletas o manipuladas.
  + Control de cupos y acceso:
    - Antes de inscribir a un usuario se verifica que la clase exista y que haya cupos disponibles.
    - Se valida que el usuario realmente esté inscrito antes de permitir cancelar.
    - Previene reservas duplicadas o cancelaciones inválidas.
* **Seguridad en el cliente (Android)**
  + Persistencia segura con SharedPreferences: Tras iniciar sesión, los datos del usuario se guardan localmente pero nunca se almacena la contraseña.
  + Destrucción de sesión: Cuando el usuario cierra sesión, los datos se eliminan completamente
  + Control UI basado en estado: el botón de reserva cambia su visibilidad según el estado del usuario, esto refuerza la seguridad a nivel de experiencia de usuario y evita errores visuales o duplicaciones:
    - Si está inscrito, el botón se oculta.
    - Si hay cupo, también se oculta.

# 4. Screenshots de la aplicación

* **Pantalla inicial**: Primera pantalla de la aplicación donde el usuario o el administrador iniciarán sesión.

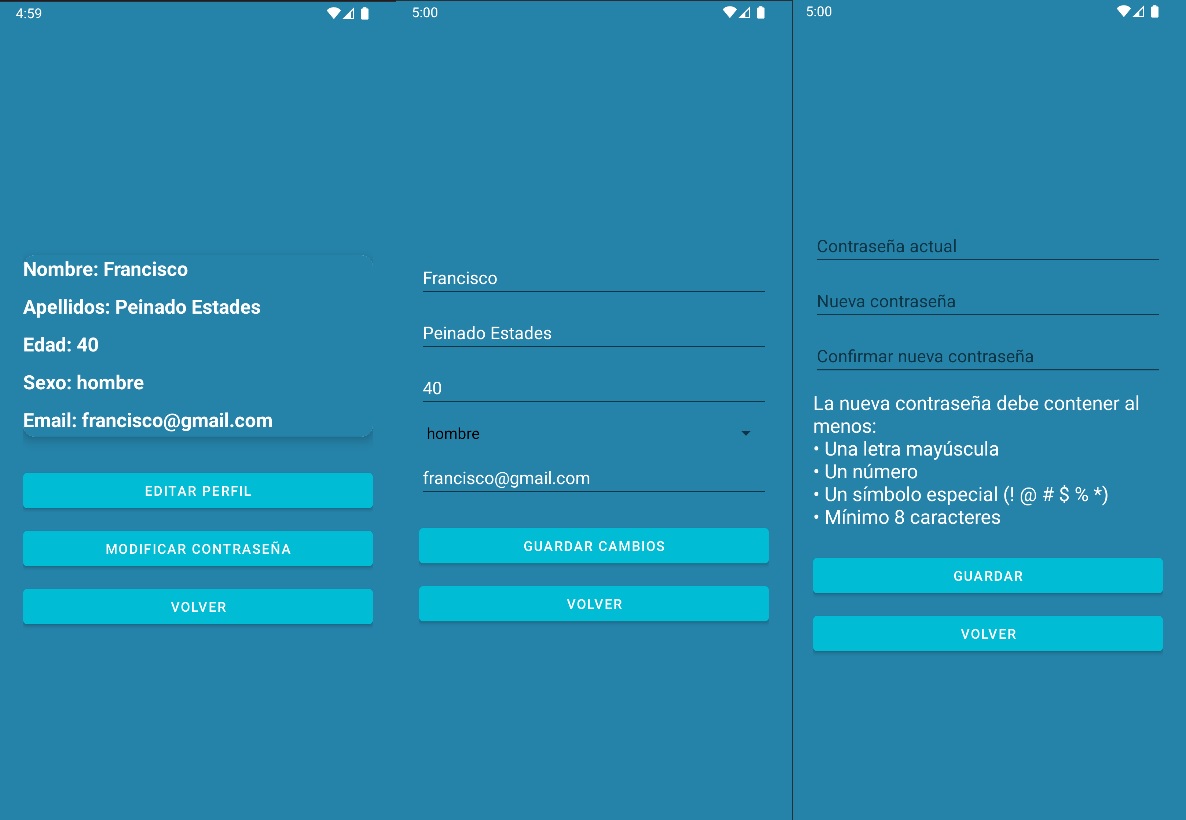


* **Pantalla principal**: Primera pantalla que se muestra al iniciar sesión, dependiendo si quién se conecta es usuario administrador o cliente y verá las diferentes opciones a realizar. En el header veremos las mismas opciones tanto para administrador como para cliente. De izquierda a derecha podemos observar un icono y el nombre de la persona conectada. Un botón para activar modo claro/oscuro y un botón de hamburguesa con dos opciones a elegir:
  + **Ver perfil**
  + **Cerrar sesión.**

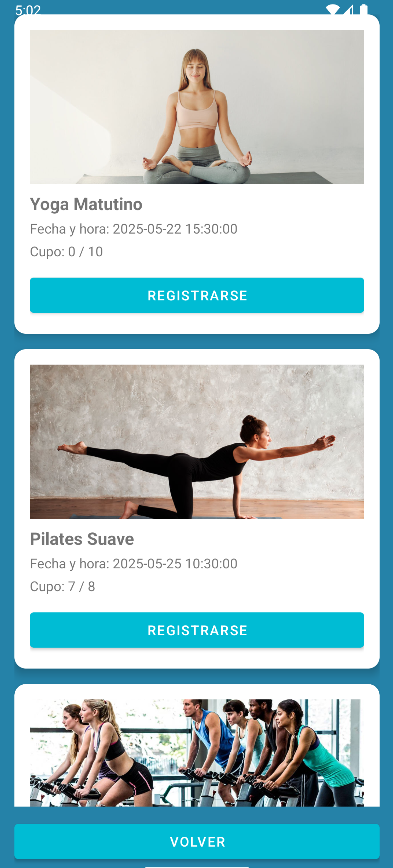
**Pantalla de cliente:** Un cliente podrá ver el listado de clases existentes y ver un listado de clases donde esté registrado.



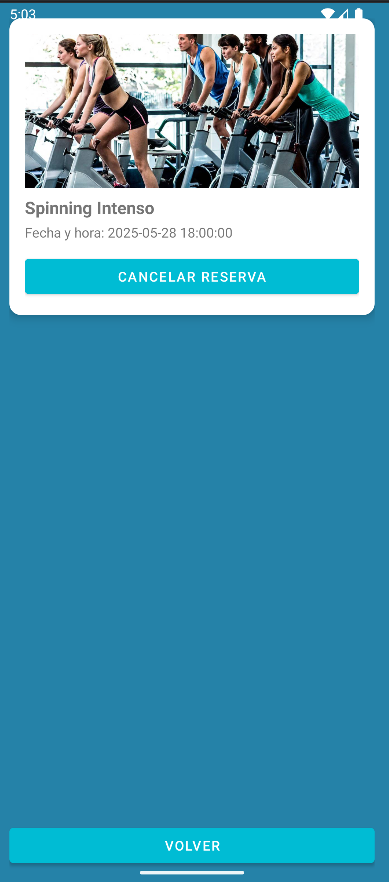
* **Pantalla Ver perfil:** Se podrá ver con detalle el perfil del usuario logueado. Aparte de sus datos dispondrá de los botones de:
  + **Editar perfil**: Donde podrá modificar sus datos.
  + **Modificar contraseña**: Donde podrá cambiar su contraseña de inicio de sesión



* **Pantalla listado de clases:** El usuario podrá observar las clases disponibles y la posibilidad de registrarse a ellas. Si ya se encuentra registrado se lo indicará la tarjeta de la clase al igual de si el aforo esta completo.

-

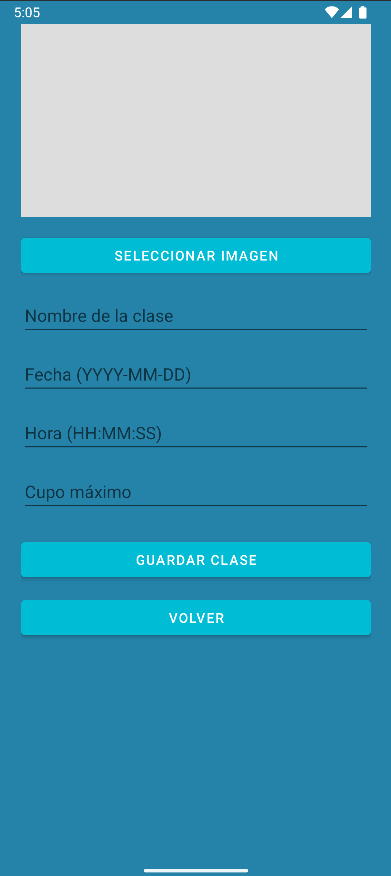
* **Pantalla Mis clases registradas**: El usuario podrá ver en clases se encuentra registrada con el detalle de las mismas y la posibilidad de cancelar la reserva.



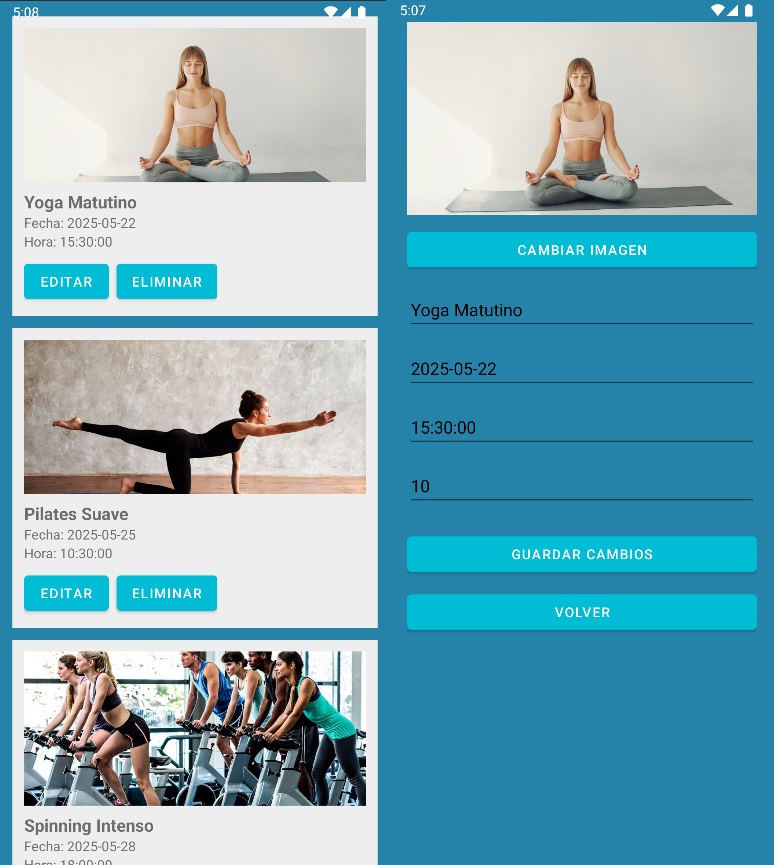
* **Pantalla de administrador:** Un administrador podrá:
  + Ver el listado de clases existentes.
  + Agregar una nueva clase.
  + Editar una clase existente.



* **Pantalla de Agregar Clase:** Un administrador podrá agregar una clase al listado con los datos necesarios para ello.



* **Pantalla de Editar Clase:** Un administrador podrá editar una clase existente e incluso eliminarla.



# 5. Incidencias

Para garantizar una gestión eficiente ante cualquier problema técnico o funcional en la aplicación, se establece el siguiente protocolo de actuación ante incidencias:

* **Recopilación de Información**: Antes de intervenir en una incidencia, se debe recoger toda la información relevante:
  + Descripción clara del problema: ¿Qué sucede? ¿Qué se esperaba que ocurriera?
  + Pasos para reproducir el error: Secuencia detallada de acciones.
  + Capturas de pantalla o mensajes de error: Para facilitar el diagnóstico.
  + Dispositivo o entorno afectado: Sistema operativo, navegador, versión de la app, etc.
* **Posible Solución**: Una vez evaluada la incidencia, se procede de la siguiente forma:
  + Clasificación del tipo de problema:
  + Error de lógica / funcionalidad
  + Problema de conexión o base de datos
  + Error de interfaz o usabilidad
  + Fallo del entorno (XAMPP, Android Studio, red)
* **Acciones correctivas:**
  + Revisar logs de servidor y consola Android
  + Corregir código afectado
  + Reiniciar servicios (Apache, MySQL)
  + Verificar configuraciones de IP, rutas, permisos
* **Pruebas posteriores**: Confirmar que el error no se repite tras aplicar la solución.
* **Registro de Incidencias:** Llevar un historial organizado de incidencias resueltas o pendientes:
  + Fecha y hora
  + Usuario que reporta
  + Descripción del fallo
  + Solución aplicada
  + Responsable técnico
  + Estado: Resuelto, Pendiente, En revisión

# 6. Manual de instalación

* **Software necesario:**
  + XAMPP
  + Android Studio (o un emulador Android / dispositivo físico)
  + Navegador web (Chrome, Firefox, etc.)
* **Instalación del Backend (XAMPP)**

1. **Instalar XAMPP**
2. **Copiar el backend:** C:\xampp\htdocs\gimnasio-backend
3. **Importar base de datos:** archivo .sql incluido en la carpeta del backend
4. **Configurar conexión a base de datos:** Abrir el archivo config.php o .env dentro del backend

*DB\_HOST=localhost*

*DB\_USER=root*

*DB\_PASSWORD=*

*DB\_NAME=gimnasio\_db*

* **Instalación del Proyecto Android**

1. **Abrir en Android Studio el proyecto.**
2. **Verificar conexión con backend.**
3. **Ejecutar app.**