iLERNA.

SISTEMAS DE GESTIÓN EMPRESARIAL

PROYECTO PRIMER TRIMESTRE

GymFortMoment

ALUMNO:

DAVID GUTIÉRREZ ORTIZ

PROFESOR:

MARTIN RIVERO

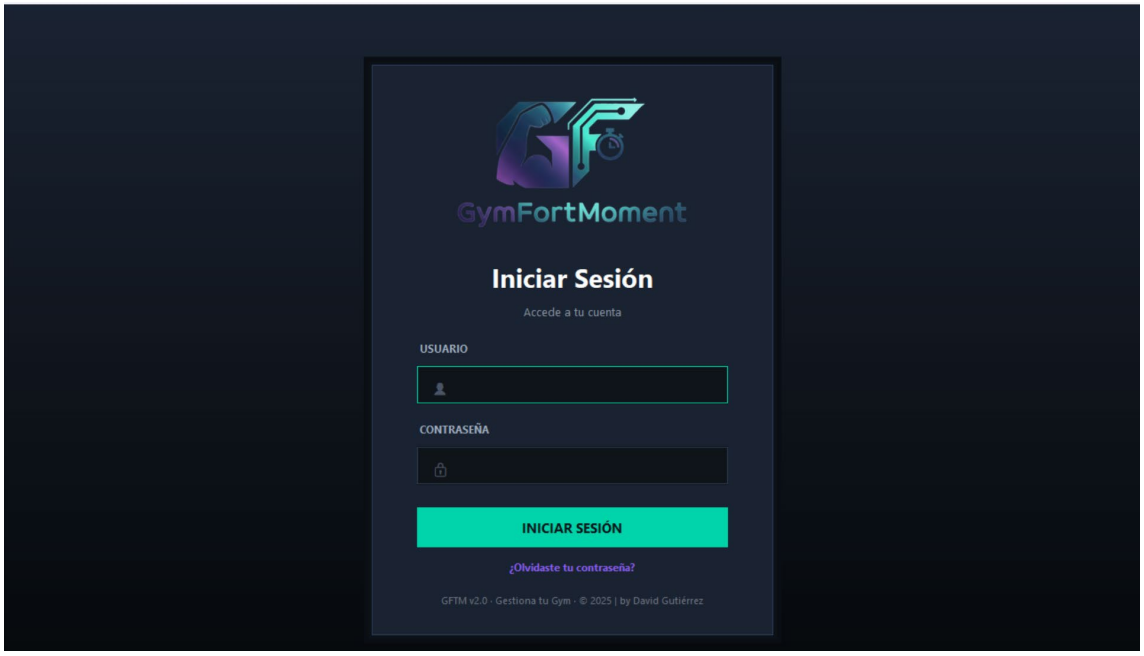
SISTEMA “**GYMFORTMOMENT ”**

INTRODUCCIÓN

La aplicación **GymForTMoment** se desarrolla con el objetivo de proporcionar una herramienta sencilla y eficaz para la gestión integral de un gimnasio. El sistema centraliza la información de clientes, pagos, reservas y aparatos, permitiendo un control organizado del día a día.

El gimnasio ofrece un servicio 24/5, y su actividad requiere conocer en todo momento qué aparatos están libres, qué sesiones están asignadas y qué clientes tienen pagos pendientes.

Este aplicativo reúne todas esas necesidades en un entorno unificado, intuitivo y fácilmente ampliable, sirviendo como base para futuras mejoras o nuevas funcionalidades.



REQUISITOS FUNCIONALES

En este apartado detallo los requisitos funcionales que, a mi juicio, son necesarios para que la aplicación *GymForTMoment* cubra correctamente las necesidades del gimnasio. El objetivo es identificar de forma clara qué debe permitir el sistema en relación con la gestión de clientes, aparatos, reservas y pagos mensuales. Estos requisitos servirán como base para definir los casos de uso y el resto del modelado del proyecto.

1. GESTIÓN DE CLIENTES

RF-01 — Registro de nuevos clientes

El sistema permitirá dar de alta a un cliente introduciendo sus datos básicos. Esta acción es esencial para que pueda realizar reservas y asociarle pagos.

RF-02 — Modificación de datos de un cliente

Será posible actualizar la información de un cliente si se detecta algún cambio o error en sus datos.

RF-03 — Eliminación de clientes

El sistema permitirá dar de baja a un cliente cuando ya no forme parte del gimnasio.

RF-04 — Búsqueda de clientes

Se podrá localizar un cliente mediante su ID, nombre u otro dato relevante, facilitando su consulta o edición.

RF-05 — Listado general de clientes

La aplicación mostrará todos los clientes registrados, permitiendo una visión rápida del censo actual del gimnasio.

2. GESTIÓN DE APARATOS

RF-06 — Registro de aparatos de entrenamiento

El sistema permitirá añadir nuevos aparatos al inventario. Aunque existan varios aparatos iguales, cada máquina se tratará como un elemento individual.

RF-07 — Consulta del estado de los aparatos

Se podrá consultar si un aparato está disponible, reservado o inactivo según su uso reciente.

RF-08 — Listado de aparatos

La aplicación mostrará un listado completo con todos los aparatos registrados en el sistema.

3. GESTIÓN DE RESERVAS

RF-09 — Crear una reserva de 30 minutos

El sistema permitirá asignar a un cliente una reserva sobre un aparato en un tramo fijo de 30 minutos.

RF-10 — Validación de disponibilidad antes de reservar

Antes de crear una reserva, el sistema comprobará que el aparato está libre en ese tramo horario y en ese día.

RF-11 — Cancelación de reservas

Será posible cancelar una reserva para liberar el tramo correspondiente y permitir que otro cliente pueda ocuparlo.

RF-12 — Listado de ocupación por aparato y día

El sistema generará un listado con las horas ocupadas y libres de un aparato en un día concreto (de lunes a viernes), indicando qué cliente tiene cada reserva realizada.

4. GESTIÓN DE PAGOS

RF-13 — Generación mensual de recibos

Una vez al mes, el sistema generará de forma automática los recibos correspondientes a todos los clientes activos.

RF-14 — Registro de pagos realizados

El sistema permitirá marcar si un cliente ha abonado su mensualidad.

RF-15 — Listado de clientes morosos

La aplicación mostrará los clientes que tienen pagos pendientes, facilitando el control de morosidad.

REQUISITOS FUNCIONALES OPCIONALES / EXTRAS

Los siguientes requisitos no forman parte estricta del núcleo del sistema, pero aparecen de forma implícita en el enunciado del proyecto o pueden aportar una mayor coherencia al funcionamiento general de la aplicación. Se incluyen como posibles ampliaciones si el tiempo y la implementación lo permiten.

RF-E01 — Validación de horario permitido

El sistema solo permitirá crear reservas dentro del horario operativo del gimnasio, establecido como 24 horas de lunes a viernes. De esta manera se evitan errores y se respeta el marco de funcionamiento del centro.

RF-E02 — Cuota mensual fija para todos los clientes

El sistema gestionará una cuota mensual única, ya que todos los clientes pagan la misma mensualidad independientemente de las actividades que realicen. Este requisito facilita la generación de recibos mensuales.

REQUISITOS NO FUNCIONALES

En este apartado describo los requisitos no funcionales que considero necesarios para garantizar que la aplicación funcione de manera fiable, coherente y cómoda para el usuario. Aunque no afectan directamente a las operaciones principales del sistema, sí buscan influir en su calidad, mantenimiento y estabilidad.

1. CALIDAD Y FIABILIDAD

RNF-01 — Integridad de los datos

El sistema debe garantizar que la información almacenada en la base de datos sea coherente y no se generen duplicidades en clientes, aparatos ni reservas. Esto ayuda a evitar errores durante la gestión diaria.

RNF-02 — Persistencia estable con SQLite

La aplicación utilizará SQLite como motor de almacenamiento, asegurando que los datos permanecen disponibles tras cada ejecución y que las operaciones CRUD funcionen de forma consistente.

2. RENDIMIENTO Y EFICIENCIA

RNF-03 — Respuesta adecuada en operaciones comunes

Las acciones habituales (listar clientes, crear reservas, consultar disponibilidad, etc.) deben ejecutarse con fluidez, sin tiempos de espera perceptibles para el usuario.

RNF-04 — Consultas optimizadas para reservas

El sistema debe ser capaz de generar listados de ocupación por día sin ralentizarse, incluso si la cantidad de clientes y reservas aumenta.

3. USABILIDAD Y EXPERIENCIA DE USUARIO

RNF-05 — Interfaz cuidada y visualmente coherente

La aplicación contará con una interfaz desarrollada en Tkinter, diseñada de forma clara y agradable, siguiendo una estructura visual coherente. Se priorizará la legibilidad, la organización y una experiencia de usuario fluida, evitando una apariencia básica o poco cuidada. La interfaz incluirá elementos visuales (colores, distribución, tamaños, paneles...) que faciliten el uso del sistema y transmitan profesionalidad.

RNF-06 — Validación básica de entradas

La aplicación debe manejar errores comunes del usuario (campos vacíos, formatos incorrectos, IDs inexistentes...) evitando bloqueos o cierres inesperados.

4. MANTENIBILIDAD Y ESTRUCTURA DEL CÓDIGO

RNF-07 — Arquitectura modular (MVC)

El proyecto debe mantener una separación clara entre modelo, controladores y vistas para facilitar su comprensión, mantenimiento y ampliación futura.

RNF-08 — Código legible y comentado de forma natural

El código debe incluir comentarios breves cuando sea necesario, evitando excesos. Su objetivo es facilitar que cualquier lector entienda qué hace cada módulo sin necesidad de explicaciones externas.

5. SEGURIDAD BÁSICA

RNF-09 — Control mínimo sobre modificaciones

Solo las operaciones permitidas por la lógica del sistema podrán ejecutarse: por ejemplo, no se debe permitir reservar en tramos inválidos o eliminar un cliente si tiene reservas activas sin confirmación.

RNF-10 — Evitar fallos críticos

La aplicación debe manejar excepciones esperables (errores de conexión, entradas inválidas, índices fuera de rango...) para evitar que el programa se cierre inesperadamente.

DIAGRAMA DE CASOS DE USO | LISTA DE CASOS DE USO

En este apartado recojo los casos de uso que definen cómo se espera que interactúe el actor principal con la aplicación *GYMFORTMOMENT* . Estos casos de uso describen, a un nivel general, las acciones que el sistema debe permitir realizar para cubrir las necesidades de gestión del gimnasio. A partir de esta lista se construirá el diagrama correspondiente.

CASOS DE USO DEL SISTEMA

Gestión de Clientes

- **CU-01 — Registrar cliente**
- **CU-02 — Modificar datos de un cliente**
- **CU-03 — Eliminar cliente**
- **CU-04 — Consultar datos de un cliente**
- **CU-05 — Listar clientes**

Gestión de Aparatos

- **CU-06 — Registrar aparato de entrenamiento**
- **CU-07 — Listar aparatos**

Gestión de Reservas

- **CU-08 — Crear reserva de 30 minutos**
- **CU-09 — Cancelar reserva**
- **CU-10 — Consultar disponibilidad de un aparato por día**

Gestión de Pagos

- **CU-11 — Generar recibos mensuales**
- **CU-12 — Registrar pago de un cliente**
- **CU-13 — Listar clientes morosos**

A continuación se presenta el diagrama de Casos de Uso del sistema GYMFORTMOMENT . Para hacerlo más accesible, se ha optado por una representación visual más clara y organizada y que se sale quizá de la representación más estándar. Aun así, se mantienen los elementos oficiales del lenguaje UML (actor, casos de uso y relaciones).

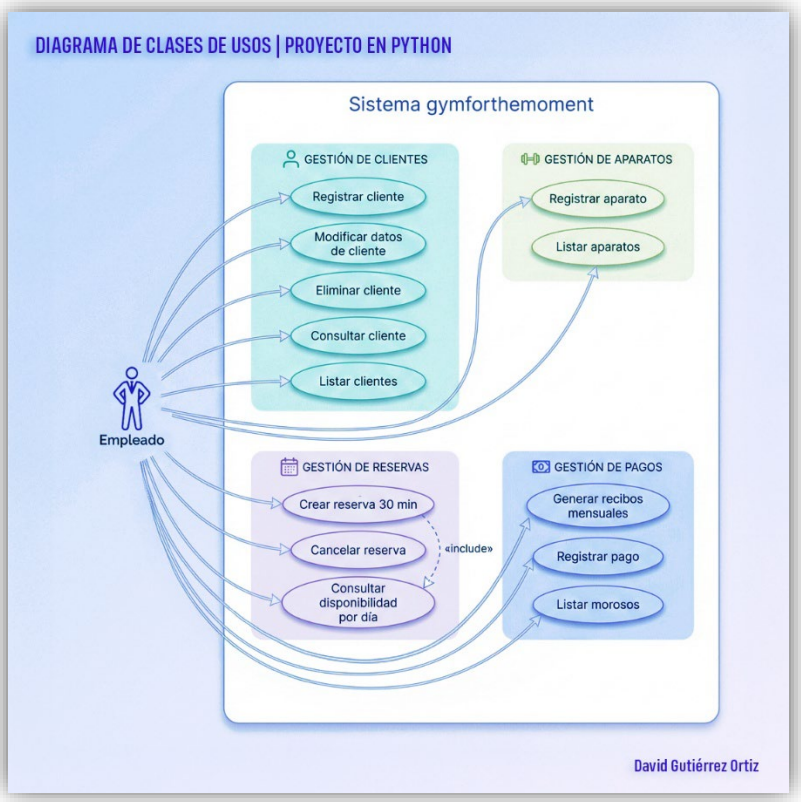
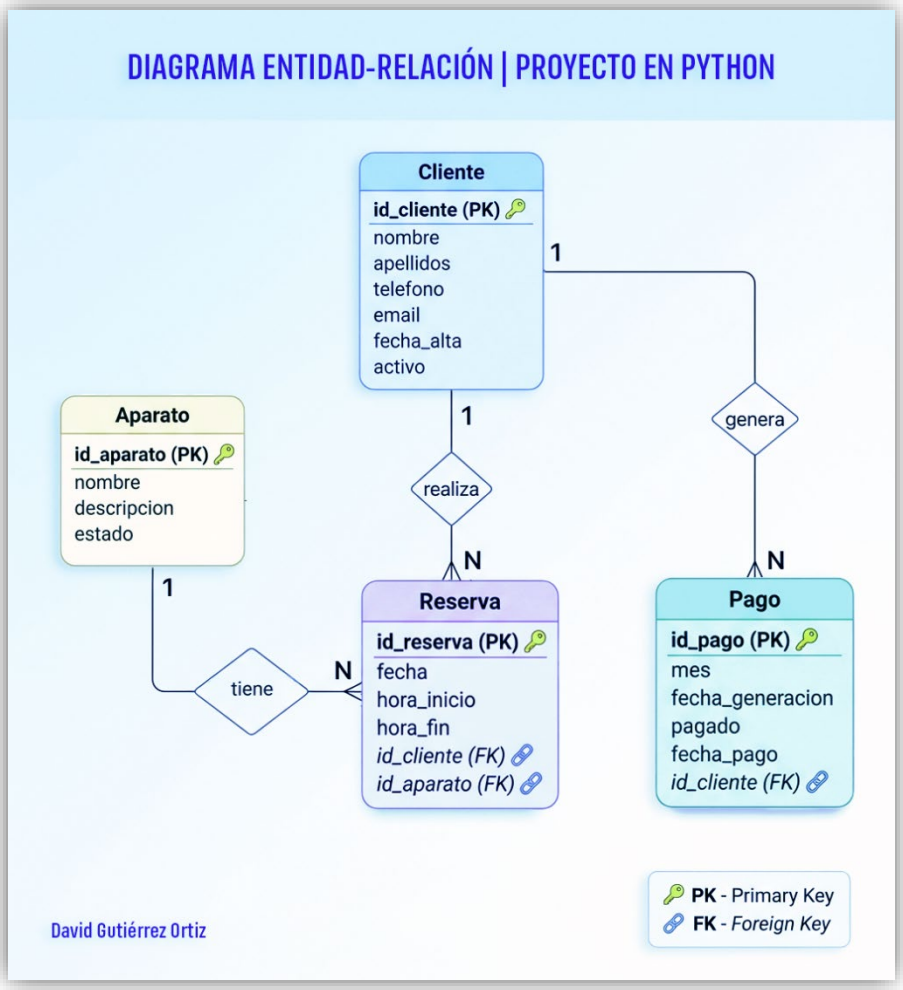


DIAGRAMA ENTIDAD-RELACIÓN

En este apartado presento el Diagrama Entidad–Relación del sistema *GYMFORTMOMENT* . Después de definir los requisitos funcionales y los casos de uso, este diagrama sirve como base conceptual que refleja cómo se estructuran los datos, qué entidades componen el sistema y cómo se relacionan entre sí.

Su objetivo es representar, de forma clara y visual, la información esencial que manejan las distintas áreas del gimnasio: clientes, aparatos, reservas y pagos mensuales. A través del modelo E-R, se establecen las cardinalidades, claves primarias, claves foráneas y las dependencias necesarias para garantizar un diseño sólido antes de construir la base de datos en SQLite.



MODELO RELACIONAL NORMALIZADO

Tras el diagrama conceptual, a continuación, desarrollo el modelo relacional que se implementará posteriormente en SQLite. Este modelo representa las tablas definitivas del sistema, con sus claves primarias, claves foráneas, tipos de dato y restricciones.

El diseño ha sido normalizado hasta Tercera Forma Normal (3FN) para asegurar integridad, evitar redundancia y facilitar el mantenimiento de la base de datos.

🟦 Tabla: **Cliente**

Atributo	Tipo	Restricciones
id_cliente	INTEGER	PK, AUTOINCREMENT, NOT NULL
nombre	VARCHAR(100)	NOT NULL
apellidos	VARCHAR(100)	NOT NULL
telefono	VARCHAR(15)	UNIQUE
email	VARCHAR(100)	UNIQUE, NOT NULL
fecha_alta	DATE	NOT NULL
activo	BOOLEAN	DEFAULT 1

🟩 Tabla: **Aparato**

Atributo	Tipo	Restricciones
id_aparato	INTEGER	PK, AUTOINCREMENT, NOT NULL
nombre	VARCHAR(100)	NOT NULL, UNIQUE
descripcion	TEXT	
estado	VARCHAR(50)	NOT NULL

Tabla: **Reserva**

Atributo	Tipo	Restricciones
id_reserva	INTEGER	PK, AUTOINCREMENT, NOT NULL
fecha	DATE	NOT NULL
hora_inicio	TIME	NOT NULL
hora_fin	TIME	NOT NULL
id_cliente	INTEGER	FK → Cliente(id_cliente), NOT NULL
id_aparato	INTEGER	FK → Aparato(id_aparato), NOT NULL

Tabla: Pago

Atributo	Tipo	Restricciones
id_pago	INTEGER	PK, AUTOINCREMENT, NOT NULL
mes	VARCHAR(7)	NOT NULL
fecha_generacion	DATE	NOT NULL
pagado	BOOLEAN	DEFAULT 0
fecha_pago	DATE	NULL
id_cliente	INTEGER	FK → Cliente(id_cliente), NOT NULL

NORMALIZACIÓN (1FN - 3FN)

1- Primera Forma Normal (1FN)

El modelo cumple 1FN porque:

- Todas las tablas tienen **clave primaria única**.
- No hay atributos multivaluados ni divididos.
- Cada celda contiene un único valor atómico.
- No existen grupos repetitivos.

2 - Segunda Forma Normal (2FN)

Se cumple automáticamente porque:

- Todas las claves primarias son **simples** (un solo atributo).
- No existen dependencias parciales.
- Cada atributo depende completamente de su clave primaria.

3 - Tercera Forma Normal (3FN)

El modelo también cumple 3FN porque:

- No existen dependencias transitivas.
- No hay atributos que dependan de otros atributos no clave.
- Cada campo depende **solo** de la clave primaria de su tabla.

Modelo completamente normalizado.

CONCLUSIÓN

El desarrollo de este proyecto ha sido una experiencia realmente completa, en la que he podido aplicar y conectar todo lo aprendido en Python dentro de un sistema funcional. Aunque el resultado cumple con los objetivos planteados, no puedo evitar sentir que aún existe margen para mejorar, especialmente en detalles visuales que me quedaron por pulir y en la depuración de algunas interacciones de la interfaz.

Este proceso me ha dejado claro que todavía tengo mucho camino por recorrer en cuanto a confianza y soltura técnica, pero también he comprobado que, con trabajo y constancia, soy capaz de enfrentar un proyecto de este tamaño. Me llevo la sensación de que podría haber llegado un poco más lejos, de que debería saber bastante más de lo que sé (...) pero también la certeza de que este proyecto es un buen punto de partida para seguir creciendo y mejorando en lo posible como desarrollador.