

---

# **MEMORIA TÉCNICA**

## **FitZone Gym**

---

*Sistema de Gestión de Gimnasio*

• GRUPO G28 •

**Darío Suárez Domenech**

**Pablo Tormo Martínez**

**Marcos Seva Berenguer**

**Curso Académico 2025-2026**

Asignatura: Ingeniería Web

# Introducción

---

Este documento resume la visión general del sistema y su descripción técnica. La aplicación implementa un portal para un gimnasio con una parte pública y zonas privadas para socios, monitores y webmaster. En la zona de socios se centraliza la gestión de actividades, reservas, saldo y planes, los monitores disponen de su propio área para consultar su calendario y clases asignadas y el webmaster gestiona el ciclo de vida de los usuarios (aceptar, rechazar, bloquear y activar), valida actividades propuestas y consulta informes de uso del sistema. El objetivo es ofrecer una experiencia sencilla para el usuario final y, a la vez, mantener una base técnica clara y mantenible.

## Descripción técnica del sistema

---

La solución se construye sobre Laravel 11 y PHP 8.2, siguiendo una arquitectura MVC clásica. Las rutas se declaran en `routes/web.php` y derivan las peticiones hacia controladores o closures. La capa de presentación utiliza Blade para renderizar vistas en `resources/views`, y el frontend se apoya en Tailwind CSS y Vite para el empaquetado de assets. Para peticiones HTTP y utilidades, el proyecto incluye Axios y el cliente HTTP de Laravel.

En la capa de datos se usa el Query Builder de Laravel (`DB::table`) para consultas directas sobre una base de datos relacional. Esto permite obtener listados de actividades, planes, reservas y transacciones sin añadir complejidad extra cuando no es necesario un modelo completo.

La seguridad de acceso se resuelve con el sistema de autenticación nativo de Laravel (`Auth`) y middlewares de rol y estado, que limitan el acceso a las rutas privadas y garantizan que sólo los usuarios con un rol específico puedan operar en sus respectivas páginas. La lógica de negocio más sensible (reservas, saldo y renovaciones) se agrupa en controladores para mantener la trazabilidad del flujo.

En cuanto a interoperación, el sistema integra servicios externos mediante APIs HTTP. Destacan el TPV para recargas de saldo y la tienda externa para catálogo y compras. Estas integraciones se configuran en `config/services.php` y se consumen con el cliente HTTP de Laravel, manejando errores y estados de respuesta para mantener la experiencia estable.

# Mockups

## Inicio



## Instalaciones

A grid of four images showing different gym facilities: "Sala de Musculación" (weights), "Zona de Spinning" (stationary bikes), "Box de CrossFit" (functional training area), and "Sala de Yoga y Pilates" (yoga/pilates studio). Each image has a caption and a small description below it.

A light blue section titled "Servicios Adicionales" (Additional Services) listing amenities such as locker rooms, cardio zones, WiFi, showers, parking, and a sports shop.



## Tarifas

**Plan Básico**  
Acceso General  
30€/mes  
• Acceso a todas las instalaciones  
• Incluye 2 clases extra gratuitas al mes

**Plan Premium**  
Acceso Total  
45€/mes  
• Acceso completo a instalaciones  
• Incluye 6 clases extra gratuitas al mes

**Información Adicional**  
Actividades con Coste Extra:  
Spinning (por clase)  
CrossFit (por clase)  
Yoga (por clase)

**Condiciones**  
5.50€ • Sin permanencia  
6.00€ • Sin permanencia  
6.00€ • Cancelación hasta 24h antes  
• Saldo recargable para actividades

**Contacto**  
Tu centro deportivo de confianza, alcanza tus objetivos con nosotros.  
Calle Principal 123, Madrid  
+34 900 123 456  
+34 900 123 457  
info@fitzone.com  
soporte@fitzone.com

**Horarios**  
Lunes - Viernes: 6:00 - 23:00  
Sábados: 8:00 - 21:00  
Domingos: 9:00 - 19:00

## Contacto

**Contacto**  
¿Tienes alguna pregunta? Estamos aquí para ayudarte.

**Información de Contacto**

**Dirección**  
Calle Principal 123  
28000 Madrid, España

**Teléfono**  
+34 900 123 456  
+34 900 123 457

**Email**  
info@fitzone.com  
soporte@fitzone.com

**Horario**  
Lun - Vie: 6:00 - 23:00  
Sáb: 8:00 - 21:00  
Domingos: 9:00 - 19:00

**Envíenos un Mensaje**

Nombre: \_\_\_\_\_  
Email: \_\_\_\_\_  
Teléfono: \_\_\_\_\_  
Mensaje: \_\_\_\_\_

**Enviar Mensaje**

**Contacto**  
Tu centro deportivo de confianza, alcanza tus objetivos con nosotros.  
Calle Principal 123, Madrid  
+34 900 123 456

**Horarios**  
Lunes - Viernes: 6:00 - 23:00  
Sábados: 8:00 - 21:00  
Domingos: 9:00 - 19:00

## Hazte socio

**Hazte Socio**  
Completa el formulario y nos pondremos en contacto contigo

Nombre \* \_\_\_\_\_ Apellidos \* \_\_\_\_\_  
Email \* \_\_\_\_\_ Teléfono \* \_\_\_\_\_  
DNI/NIE \* \_\_\_\_\_ Fecha de Nacimiento \* dd/mm/aaaa  
Dirección \* \_\_\_\_\_  
Ciudad \* \_\_\_\_\_ Código Postal \* \_\_\_\_\_

Nota: Tu solicitud será revisada por nuestro equipo. Te contactaremos por email en un plazo máximo de 48 horas para confirmar tu alta como socio.

**Enviar Solicitud**

## Actividades disponibles (perfil de socio)

The screenshot shows the 'Actividades Disponibles' section. It lists several classes with details like day, time, location, and monitor. Buttons for 'Reservar Plaza' (Reserve Spot) or 'Completo' (Full) are shown. The 'Reserva' button for 'Yoga Relax' is highlighted in green.

Actividad	Día	Hora	Lugar	Monitor	Precio	Estado Reserva
Spinning Intenso	Lunes	18:00 - 19:00	Sala Spinning	Monitor: Ana García	5.00€	Reservar Plaza
CrossFit Avanzado	Martes	18:00 - 19:00	Box CrossFit	Monitor: Carlos Ruiz	8.00€	Reservar Plaza
Yoga para Principiantes	Miércoles	17:00 - 18:00	Sala Yoga	Monitor: Laura Martín	6.00€	Reservar Plaza
Spinning Morning	Jueves	07:00 - 08:00	Sala Spinning	Monitor: Ana García	5.00€	Reservar Plaza
CrossFit Principiantes	Viernes	18:30 - 19:30	Box CrossFit	Monitor: Miguel Torres	8.00€	Completo
Yoga Relax	Sábado	10:00 - 11:00	Sala Yoga	Monitor: Laura Martín	6.00€	✓ Ya reservada

## Mis Reservas (perfil de socio)

The screenshot shows the 'Mis Reservas' section. It lists a single reservation for 'Yoga Relax' on Saturday at 10:00 AM. The status is 'Pagado' (Paid) and there is a 'Cancelar' (Cancel) button.

## Gestión de saldo (perfil de socio)

The screenshot shows the 'Gestión de Saldo' section. It displays the available balance (25.50€), information about automatic deduction for reservations, and a history of movements. There are buttons for top-up amounts (10€, 20€, 50€) and a 'Recargar' (Top-up) button. A note says 'El saldo se descuenta automáticamente al reservar actividades'.

**Saldo Disponible:** 25.50€

**Información:**

- El saldo se descuenta automáticamente al reservar actividades.
- Si cancelas una reserva, el saldo se reembolsa.
- Las recargas se procesan mediante TPV seguro.
- El saldo no tiene fecha de caducidad.

**Historial de Movimientos:**

Movimiento	Fecha	Saldo
Recarga TPV	28 Nov 2023	+20.00€
Yoga Relax	27 Nov 2023	-6.00€
Recarga TPV	28 Nov 2023	+50.00€

## Mi perfil (perfil de socio)

FitzZone Gym - Socio

Saldo: 25.50€ Juan Pérez Salir

Actividades Mis Reservas Gestión de Saldo Mi Perfil Tienda Plan

**Mi Perfil**  
Información de tu cuenta de socio

Datos Personales:

Nombre: Juan Pérez	Email: juan.perez@email.com
Teléfono: +34 600 123 456	DNI: 12345678A
Dirección: Calle Ejemplo 123, 28001 Madrid	

**Guardar Cambios**

Cambiar Contraseña:

Contraseña Actual:

Nueva Contraseña:

Confirmar Nueva Contraseña:

**Actualizar Contraseña**

**Mi membresía**

Número de Socio: FZ-2025-0123
Fecha de Alta: 15 Enero 2025
Próximo Cobro: 5 Marzo 2025
Estado: Activo

**Acceso a instalaciones**

- ✓ Sala de musculación
- ✓ Zona de cardio
- ✓ Vestuarios y duchas
- ✓ Todas las instalaciones

## Página de tienda (perfil de socio)

FitzZone Gym - Socio

Saldo: 25.50€ Juan Pérez Salir

Actividades Mis Reservas Gestión de Saldo Mi Perfil Tienda Plan

**Tienda FitzZone**  
Productos y accesorios para mejorar tu entrenamiento

Buscar productos: Categoría: Todas

Mostrando 8 productos

<b>Proteína Whey Premium</b> Proteína de alta calidad para recuperación muscular 35.99€ Stock: 15	<b>Bola Deportiva</b> Bola de 750ml con diseño ergonómico 12.50€ Stock: 30	<b>Toalla Premium</b> Toalla de microfibra absorbente 18.00€ Stock: 20	<b>Bandas de Resistencia</b> Set de 3 bandas con diferentes resistencias 24.99€ Stock: 12
<b>Estera Yoga</b> Estera antideslizante de 6mm. 29.99€ Stock: 8	<b>Guantes Training</b> Guantes con agarre reforzado 16.50€ Stock: 18	<b>Mochila Deportiva</b> Mochila con compartimentos especiales 45.00€ Stock: 10	<b>Reloj Fitness</b> Monitor de actividad con GPS 69.99€ Stock: 5

**Añadir al carrito**

## Página de planes (perfil de socio)

FitzZone Gym - Socio

Saldo: 25.50€ Juan Pérez Salir

Actividades Mis Reservas Gestión de Saldo Mi Perfil Tienda Plan

**Plan**  
Gestiona tu plan de membresía

Plan Actual: Premium

Información:  
• Precios: 20€/mes  
• Clases gratis: 6  
• Clases disponibles: 4

**Cambiar Plan**

Básico: 10€/mes	Premium: 20€/mes
-----------------	------------------

**Cambio de plan seguro**

## Solicitudes de Membresía (perfil de webmaster)

The screenshot shows the 'Solicitudes' (Applications) section of the FitZone Gym administration interface. It displays two pending membership applications:

Nombre	Estado	Email	Teléfono	Tarifa Solicitud	Fecha de Solicitud
Maria González	Pendiente	maria.gonzalez@email.com	+34 600 111 222	Premium	29 Nov 2025
Ana Martín	Aceptada	ana.martin@email.com	+34 600 555 666	Elite	27 Nov 2025

Buttons for 'Aceptar' (Accept) and 'Rechazar' (Reject) are visible next to each application row.

## Gestión de Socios (perfil de webmaster)

The screenshot shows the 'Gestión de Socios' (Member Management) section of the FitZone Gym administration interface. It displays three members:

Total Socios	Socios Activos	Socios Bloqueados
3	2	1

Below this, a table lists the active members:

Nombre	Email	Fecha Alta	Tarifa	Saldo	Estado	Acciones
Juan Pérez	juan.perez@email.com	15 Ene 2025	Premium	25.50€	Activo	Bloquear
Laura Fernández	laura.fernandez@email.com	10 Feb 2025	Elite	45.00€	Activo	Bloquear
Carlos López	carlos.lopez@email.com	05 Mar 2025	Básica	0.00€	Bloqueado	Activar

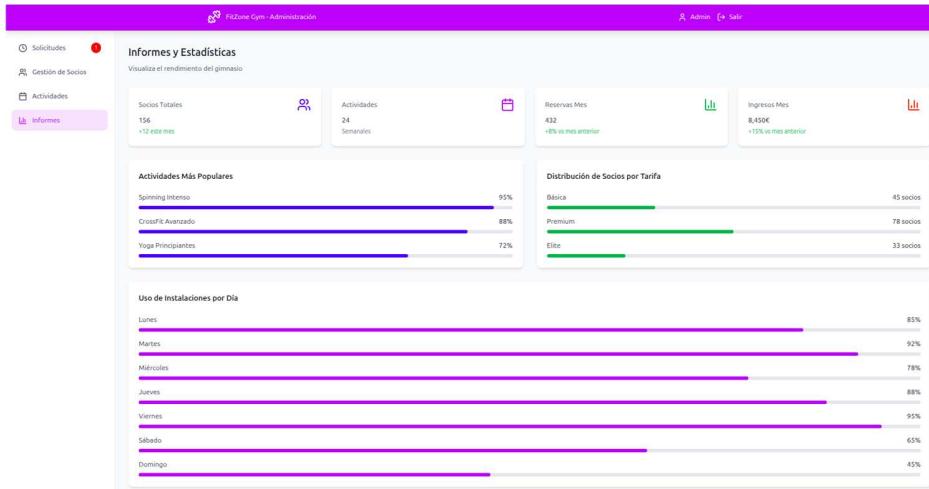
## Gestión de Actividades (perfil de webmaster)

Nombre	Tipo	Día	Horario	Sala	Monitor	Plazas	Coste	Acciones
Spinning Intenso	Spinning	Lunes	18:00 - 19:00	Sala Spinning	Ana García	20	5.00€	<a href="#">Eliminar</a>
CrossFit Avanzado	CrossFit	Martes	19:00 - 20:00	Box CrossFit	Carlos Ruiz	15	8.00€	<a href="#">Eliminar</a>
Yoga para Principiantes	Yoga	Miércoles	17:00 - 18:00	Sala Yoga	Laura Martín	25	6.00€	<a href="#">Eliminar</a>

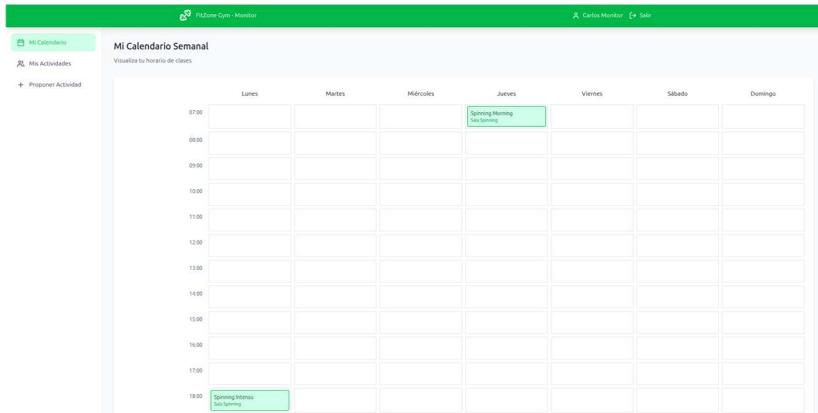
Nombre	Tipo	Día	Horario	Sala	Monitor	Plazas	Coste	Acciones
Spinning Intenso	Spinning	Lunes	18:00 - 19:00	Sala Spinning	Ana García	20	5.00€	<a href="#">Eliminar</a>
CrossFit Avanzado	CrossFit	Martes	19:00 - 20:00	Box CrossFit	Carlos Ruiz	15	8.00€	<a href="#">Eliminar</a>
Yoga para Principiantes	Yoga	Miércoles	17:00 - 18:00	Sala Yoga	Laura Martín	25	6.00€	<a href="#">Eliminar</a>

\* Si se elige la periodicidad como semanal el backend registrará en base de datos todas las clases que queden en el horario elegido en el mes actual.

## Informes y Estadísticas (perfil de webmaster)



## Mi Calendario Semanal (perfil de monitor)



## Mis Actividades (perfil de monitor)

**Mis Actividades**  
Gestiona las clases que imparten

**Spinning Intenso**  
Spinning  
Lunes 18:00-19:00  
Sala Spinning

**Spinning Morning**  
Spinning  
Jueves 07:00-08:00  
Sala Spinning

**Detalles de la Actividad**

Nombre	Spinning Intenso
Tipo	Spinning
Día	Lunes
Sala	Sala Spinning
Ocupación	15 / 20 plazas

**Horario**  
18:00 - 19:00

**Participantes Inscritos**

- Juan Pérez
- Maria González
- Pedro Sánchez

+ 12 participante(s) más...

## Histórico de Actividades (perfil de monitor)

**Historico de Actividades**  
Revisa el historico de tus clases impartidas

**Spinning Intenso**  
Spinning  
2024-12-13 18:00 - 19:00  
Sala Spinning

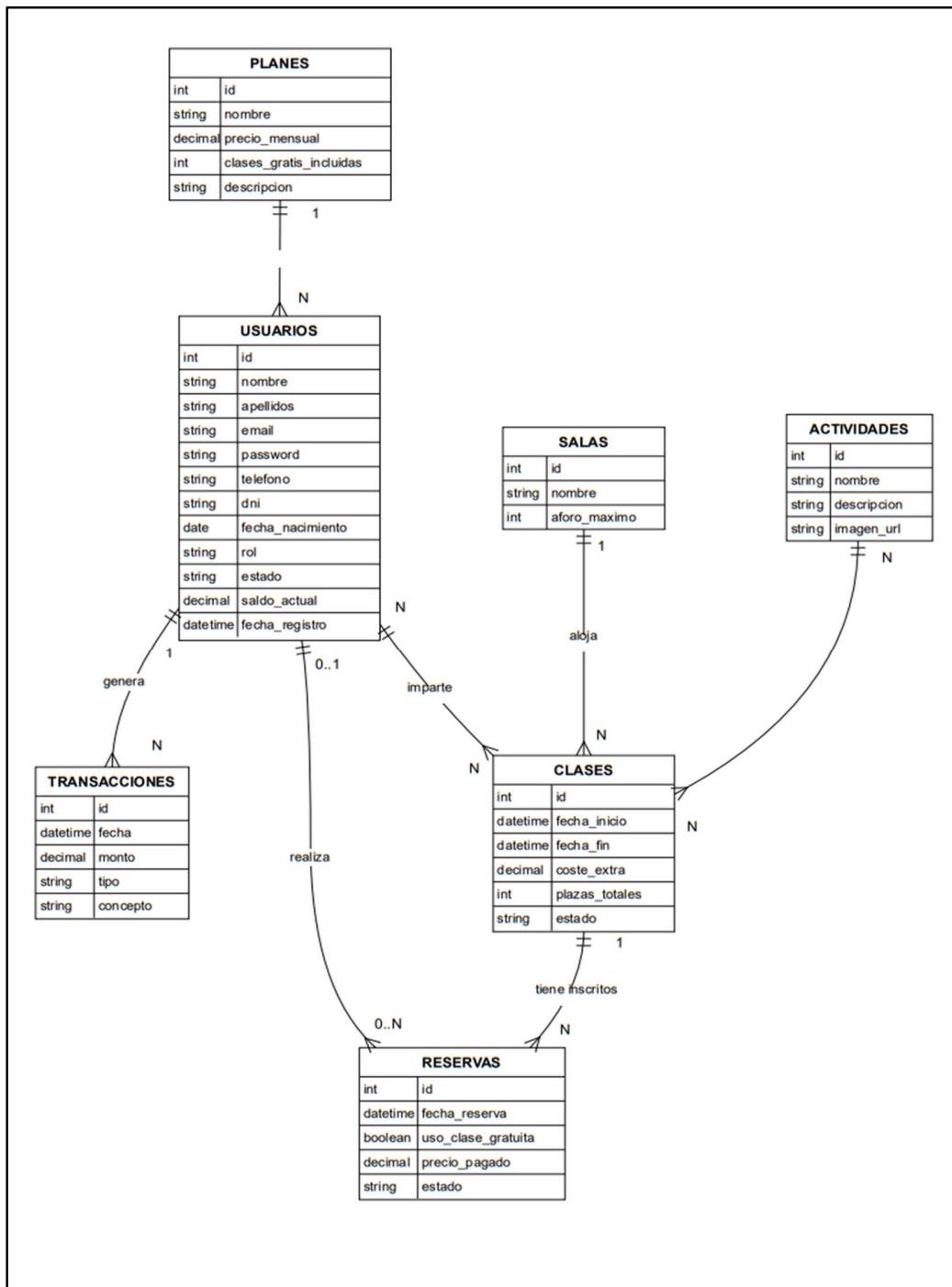
**Spinning Morning**  
Spinning  
2024-12-12 07:00 - 08:00  
Sala Spinning

**Yoga Flow**  
Yoga  
2024-12-11 19:00 - 20:00  
Sala Yoga

Selecciona una actividad para ver los detalles

# Diagramas

## Diagrama de clases



## Metodologías y planificación

---

El trabajo se organizó con GitHub Flow. Cada funcionalidad se desarrolló en una rama independiente, y la integración se realizó mediante pull request hacia la rama `develop`. Este flujo permite revisar cambios, resolver conflictos y mantener un historial claro de decisiones.

La planificación se repartió por perfiles de usuario. El equipo se dividió en tres personas: una responsable del área de socios, otra del área de webmaster y otra del área de monitores. Como el área de monitores era menos trabajo, para igualar la carga equitativamente, el encargado de los monitores también implementó la API y la publicó para acceso externo en <https://iw-practica2-production.up.railway.app/>. Las rutas expuestas son las definidas en `routes/api.php`.

## Descripción de la implementación

---

### Parte pública

La parte pública de la aplicación se encuentra en el archivo `routes/web.php` y las vistas dentro de la carpeta `resources/views/public`.

El enrutado se realiza mediante funciones anónimas para las páginas de `inicio`, `actividades`, `tarifas`, `contacto`, `login` y `register`.

Para estas pantallas, no se necesitan controladores especiales. En lugar de eso, se utiliza `DB::table(...)` para acceder a los datos. Esto hace que la lectura sea más fácil cuando solo necesitas listados básicos de datos.

La página de inicio, que se encuentra en `resources/views/public/inicio.blade.php`, combina un elemento destacado con llamadas a la acción hacia el login y el registro.

La vista de actividades se encuentra en `resources/views/public/actividades.blade.php`. Esta vista muestra la información de la tabla `actividades`. La información que se muestra incluye la imagen y la descripción de cada actividad. También hay una sección para servicios adicionales que ayuda a completar el contenido de las actividades.

La página de tarifas (`resources/views/public/tarifas.blade.php`) presenta los planes disponibles y resalta el plan premium con estilos distintos, pero el contenido se mantiene en base a los datos de la tabla `planes`.

El contacto (`resources/views/public/contacto.blade.php`) se resuelve con un formulario que hace POST a `/contacto`. En `routes/web.php` se valida el payload. Además, si hay sesión iniciada se fuerza logout para mantener la experiencia pública limpia.

El login usa `Auth::attempt` y, tras autenticar, se comprueba si el socio necesita renovar su plan. Si la fecha de renovación ya venció, se intenta la renovación automática descontando saldo o se redirige al flujo de renovación.

El registro de usuario en `/register` revisa los datos, crea un nuevo usuario como `socio`, lo pone en estado `pendiente` y le da un plan inicial. También le asigna una contraseña por defecto utilizando bcrypt.

## Parte de socios

Para acceder a la zona privada, se utilizan dos middlewares: uno para el rol y otro para el estado. Estos middlewares están en `app/Http/Middleware/RoleMiddleware.php` y `app/Http/Middleware/EstadoMiddleware.php`.

Los socios que están activos pueden acceder a las rutas que comienzan con `/socio/*`. Todas estas rutas están definidas en el archivo `routes/web.php` y son gestionadas por el archivo `app/Http/Controllers/SocioController.php`.

En las actividades y reservas es donde se encuentra la mayor parte de la lógica. Antes de mostrar las actividades, se llama a la función `limpiarReservasPendientesExpiradas` para liberar las plazas de las reservas pendientes que han caducado. Las reservas pendientes que han caducado son aquellas que tienen más de 10 minutos. Esto también ajusta la asistencia actual de las actividades y reservas.

En la función `getActividades()`, se juntan datos de las clases, las actividades, las salas y el usuario monitor. Esto ayuda a seleccionar las clases que serán en el futuro y que tengan plazas libres. Si el plan ha expirado, se redirige a la pantalla de bloqueo del plan.

El flujo de reserva en `reservarActividad()` verifica varios aspectos. Controla si ya hay una reserva similar, la ventana de 10 minutos para reservas pendientes, el uso de clases gratuitas que incluye el plan cada mes y el pago con el saldo disponible. Al confirmar, se registra la reserva, se actualiza el aforo y se inserta una transacción en el historial del socio. El histórico (`getReservas()`) ordena por fecha de clase y actualiza estados vencidos. La cancelación revierte el aforo y devuelve el importe si se pagó, dejando constancia en transacciones.

La gestión de saldo usa la api de un TPV externo. `setSaldo()` crea una transaccion pendiente, llama a `https://tpv-backend-cbbg.onrender.com/api/v1/payments/init` con `Http::post` y redirige al `paymentUrl`. La configuración sale de `config/services.php`. En el callback (`handleSaldoCallback()`), se valida el token, se verifica el pago y se actualizan transacciones y el

**saldo\_actual** del socio. Si hay renovación pendiente y saldo suficiente, se ejecuta la renovación automática y se registra el cargo.

El perfil es una herramienta que ayuda a mantener los datos del socio actualizados. Con el perfil, puedes hacer varias cosas. Puedes obtener el perfil, actualizar el perfil y cambiar la contraseña. Todas estas acciones se realizan de manera segura, validando la información y utilizando un hash para proteger los datos del socio. Esto nos permite tener mayor seguridad al momento de realizar estos cambios. También permite visualizar el número de socio asignado y la fecha del próximo cobro del plan.

La tienda es una parte muy importante. En la tienda, los usuarios pueden obtener diferentes productos y realizar compras. La tienda utiliza una API externa para funcionar. Si la configuración de la tienda no está correcta o si hay problemas con la conexión, el sistema está diseñado para manejar estos errores y evitar problemas mayores.

Cuando un usuario realiza una compra en la tienda, el sistema utiliza un mecanismo de seguridad llamado **lockForUpdate**. Esto ayuda a asegurar que todo salga bien durante el proceso de compra y evita lo que se conoce como condiciones de carrera, que podrían causar problemas con las transacciones. La tienda y su función de compra están diseñadas para ser seguras y fiables.

La gestión del plan (**getPlan**, **setPlan**, **renovarPlan**) muestra el plan actual y los disponibles, calcula clases gratuitas restantes del mes, permite programar cambios para la siguiente renovación y bloquea acceso cuando la renovación está vencida.

## Área de monitor

La arquitectura de este módulo se centraliza en el MonitorController y protege su acceso mediante el middleware role:monitor, aplicado a un grupo de rutas específico definido en routes/web.php.

La implementación se divide en tres componentes funcionales principales:

### 1. Dashboard y Calendario Interactivo

La funcionalidad central es el Dashboard (/monitor/dashboard), que despliega un calendario semanal dinámico.

Lógica Temporal: Se hace un uso intensivo de la librería Carbon. El controlador procesa un parámetro de desplazamiento (offset) para calcular, mediante startOfWeek() y addWeeks(), el rango de fechas exacto a visualizar.

Estructura de Datos: La vista recibe una matriz estructurada, lo que permite renderizar una rejilla HTML precisa utilizando clases de Tailwind CSS.

Interactividad: Para optimizar la experiencia de usuario, se han implementado ventanas modales temporales con Javascript. Esto permite consultar detalles clave (horario, sala, aforo) sin incurrir en la latencia de una recarga de página completa.

## **2. Gestión de Actividades (Lógica Híbrida)**

La sección de Mis Actividades (/monitor/actividades) implementa una lógica de filtrado para manejar las actividades que tiene un monitor:

Filtrado por Defecto: el sistema muestra las clases desde el inicio del día actual (`startOfDay()`) en adelante.

Navegación Contextual: Se implementó una lógica condicional que permite acceder a clases pasadas desde el calendario. Si el controlador detecta un `clase_id` específico en la petición, fuerza una búsqueda directa en la base de datos, ignorando el filtro de hoy.

Feedback Visual: Al visualizar una sesión pasada, el diseño cambia, advirtiendo visualmente al monitor de que está consultando un registro histórico.

## **3. Histórico y Consultas Optimizadas**

El módulo de Histórico (/monitor/historico) permite la auditoría de todas las clases finalizadas.

Optimización SQL: Para evitar el problema de N+1 queries, se optimizó la consulta utilizando joins explícitos con las tablas de reservas y users.

Funcionalidad: Esto permite recuperar en una sola carga el listado completo de participantes (nombre y email) de cualquier sesión anterior, visualizable mediante modales emergentes para una revisión rápida de la asistencia.

Parte de webmaster

## **Área de webmaster**

El área de webmaster centraliza la gestión administrativa del gimnasio mediante el controlador `WebmasterController`, protegido por el middleware `role:webmaster`. Esta área implementa el backoffice completo con cuatro módulos principales: solicitudes de membresía, gestión de socios, gestión de actividades y clases, e informes estadísticos.

### **1. Dashboard y Estadísticas en Tiempo Real**

El dashboard (/webmaster/dashboard) proporciona una vista general del estado del sistema mediante consultas optimizadas:

- Métricas principales: Utiliza DB::table con count() para calcular solicitudes pendientes, socios activos/bloqueados y clases programadas para el día actual.
- Próximas clases: Emplea joins entre las tablas clases, actividades, salas y users para mostrar las siguientes 10 sesiones con información completa (actividad, sala, monitor, ocupación).
- Accesos rápidos: Proporciona enlaces directos a las funcionalidades más utilizadas (crear clase, crear actividad, ver informes).

## **2. Gestión del Ciclo de Vida de Usuarios**

El módulo de solicitudes (/webmaster/solicitudes) implementa un workflow de aprobación con transacciones de base de datos:

- Aprobación (aprobarSolicitud): Genera un número de socio único con formato SOC-{hash}, actualiza el estado a 'activo', establece la fecha de próxima renovación (Carbon::now()->addMonth()) y dispara el envío de email de bienvenida. Todo el proceso se encapsula en DB::beginTransaction para garantizar atomicidad.
- Rechazo (rechazarSolicitud): Cambia el estado a 'bloqueado' y notifica al solicitante mediante email. La transacción asegura que no queden estados inconsistentes.
- Generación de número de socio: Utiliza un algoritmo que verifica unicidad mediante bucle do-while con exists() en base de datos.

La gestión de socios (/webmaster/socios) permite consultar, filtrar y modificar el estado de los miembros activos:

- Filtros dinámicos: Implementa búsqueda por nombre/email/número de socio y filtrado por estado (activo/bloqueado/pendiente) mediante \$request->filled().
- Cambio de estado (cambiarEstadoSocio): Valida el nuevo estado contra un whitelist y actualiza usando update() de Eloquent.
- Paginación: Utiliza paginate(20) para mantener el rendimiento con grandes

volúmenes de usuarios.

### **3. CRUD de Actividades y Gestión de Clases**

El CRUD de actividades (/webmaster/actividades/\*) implementa las operaciones completas:

- Creación/edición: Valida campos required (nombre, descripción) y opcional (imagen\_url) utilizando Request validation.
- Eliminación protegida (eliminarActividad): Verifica mediante exists() si hay clases asociadas antes de permitir el borrado, previniendo violaciones de integridad referencial.

La gestión de clases (/webmaster/clases/\*) es el módulo más complejo:

- Filtrado multicriteria: Combina actividad\_id, monitor\_id y fecha mediante encadenamiento de where() condicionales basados en \$request->filled().
- Creación de clases periódicas (crearClasesPeriodicas): Implementa lógica de generación automática que calcula fechas futuras según periodicidad (diaria/semanal) utilizando Carbon::addDay() y addWeek() en bucle, insertando múltiples registros en una sola transacción.
- Validación de capacidad: Verifica que plazas\_totales no exceda aforo\_maximo de la sala relacionada.
- Estados de clase: Permite transición entre 'programada', 'finalizada' y 'cancelada' con validación de estados válidos.

### **4. Informes y Análisis de Uso**

El módulo de informes (/webmaster/informes/installaciones) agrega datos de múltiples fuentes:

- Uso de instalaciones: Calcula clases por sala mediante groupBy() y ocupación promedio con avg() de la relación asistencia\_actual/plazas\_totales.
- Actividades populares: Ordena actividades por total de reservas mediante count() sobre la tabla de reservas con join a clases y actividades.

- Socios más activos: Identifica top 10 usuarios con más participación usando `groupBy('users.id')` y `orderBy('total_reservas', 'desc')->limit(10)`.
- Ingresos por actividad: Suma el campo `coste_extra` filtrando clases con cargo adicional, agrupando por actividad.

Toda la información se calcula dinámicamente en tiempo real consultando las tablas relacionales, sin datos precalculados, garantizando información actualizada.

## **5. Consideraciones de Seguridad y Validación**

- Protección de mass assignment: Los modelos User, Actividad y Clase definen `$fillable` explícitamente para prevenir inyección de campos no autorizados.
- Validación de entrada: Todos los formularios utilizan Request validation con reglas específicas (`required`, `email`, `unique`, `exists`).
- Transacciones: Operaciones críticas (aprobar solicitud, crear clases periódicas) se envuelven en try-catch con `DB::beginTransaction/commit/rollBack`.
- Confirmaciones JavaScript: Acciones destructivas (eliminar, bloquear) requieren confirmación del usuario mediante `confirm()` antes del submit.

## Puesta en producción con Railway

---

<https://iw-practica2-production.up.railway.app/>

El despliegue de la aplicación se realizó en la plataforma PaaS Railway, lo que presentó desafíos específicos respecto al entorno de desarrollo local, principalmente en la gestión de assets y la persistencia de datos.

**Compilación de Assets (Vite):** En local, Laravel utiliza `npm run dev` para servir estilos en tiempo real. En producción, esto no es viable. Se configuró el Build Command de Railway para ejecutar `npm install && npm run build`. Esto genera los archivos estáticos de CSS y JS en la carpeta `public/build`. Para que Laravel localice estos archivos correctamente, fue crítico configurar la variable de entorno `APP_URL` con el dominio real proporcionado por Railway.

**Ciclo de vida de la Base de Datos:** Dado que es un proyecto académico y se requiere probar funcionalidades con datos limpios y estructurados, se configuró el Start Command para reiniciar la base de datos en cada despliegue: `php artisan migrate:fresh --seed --force && php artisan serve --host=0.0.0.0 --port=$PORT`

Se utiliza `--force` para saltar la protección de producción de Laravel. Esto ejecuta los seeders desarrollados, que generan automáticamente una planificación de clases realista para las próximas 4 semanas y usuarios con diferentes roles para pruebas.

**Configuración de Entorno:** Las claves de las APIs externas (TPV y Tienda) y la configuración de base de datos (MySQL gestionado por Railway) se inyectaron a través de las variables de entorno del servicio, asegurando que no se expongan credenciales en el repositorio de código.

## Problemas encontrados y su solución

---

Uno de los problemas que encontramos fue que la gente bloqueaba plazas con reservas que no habían confirmado. Para solucionar esto, creamos un sistema que elimina automáticamente las reservas pendientes que han expirado y también ajustamos el número de plazas disponibles.

También era importante evitar que la gente hiciera reservas duplicadas. Así que ahora, antes de hacer una reserva, el sistema verifica si ya hay una reserva confirmada o pendiente para ese mismo evento o producto.

Otro problema común era que la gente no tenía suficiente saldo para hacer una reserva o una compra en la tienda. Ahora, antes de confirmar cualquier reserva o compra, el sistema verifica el saldo disponible y, si no es suficiente, le manda un mensaje claro al usuario para que sepa qué pasa. La renovación pendiente con saldo insuficiente se detecta cuando el usuario inicia sesión y también

cuando se realiza el callback de recarga. En ambos casos, el sistema redirige al usuario al flujo de renovación.

Por último, si los servicios externos como TPVV o tienda no están configurados, el sistema controla el error y muestra un mensaje de información sin interrumpir la navegación del usuario.

## Mejoras y ampliaciones

---

Para mejorar, sería buena idea cambiar el uso directo de `DB::table` a Eloquent. Esto nos permitiría aprovechar las relaciones y reducir el código repetido. También sería útil unificar los estados de reserva en constantes para evitar errores al cambiar de estado. De esta manera, el código sería más fácil de mantener y entender. Al utilizar Eloquent, podemos aprovechar sus características para simplificar el código y hacerlo más eficiente. Además, al tener los estados de reserva en constantes, podemos evitar errores y hacer que el código sea más consistente. Esto nos ayudará a mejorar la calidad del código y a reducir los errores.

La renovación automática se utiliza en varias rutas y podría estar en un servicio o trabajo para mantener una sola fuente de verdad. Faltan pruebas de integración para los flujos de reserva, renovación y recarga de saldo.

Si el proyecto crece, sería útil mover los callbacks de TPVV y las respuestas de la tienda a colas con registros estructurados para depurar en caso de errores.

## Referencias

---

- Laravel Routing: <https://laravel.com/docs/routing>
- Laravel Authentication: <https://laravel.com/docs/authentication>
- Laravel HTTP Client: <https://laravel.com/docs/http-client>
- Laravel Database Transactions: <https://laravel.com/docs/database#database-transactions>
- Carbon: <https://carbon.nesbot.com/docs/>