



FITLAND

Gimnasio FitLand

Desarrollo de Aplicaciones Web

IES El Cañaveral

Brandon Muzo Muzo

Alejandro Prado Flórez

Fecha de entrega: 15/06/2025

IES EL CAÑAVERAL

Departamento de Informática



Memoria del proyecto

Documento técnico

Índice

1.- Introducción	2
1.1.- Descripción y contexto del proyecto	2
1.2.- Motivación del proyecto	3
1.3.- Beneficios esperados.....	3
2.- Objetivo/s generales del proyecto	4
3.- Objetivos específicos	4
4.- Contexto actual	4
5.- Análisis de requisitos	6
5.1.- Diagrama de casos de uso.	6
5.2.- Requisitos funcionales principales:.....	7
5.3.- Requisitos no funcionales:	7
5.4.- Descripción de los usuarios y sus necesidades.	8
6.- Diseño de la aplicación	9
6.1.- Wireframe de la interfaz gráfica de usuario.	9
6.2.- Arquitectura del sistema.	9
6.3.- Diagrama de entidad-relación (E/R).	12
6.4.- Diseño de la base de datos: esquemas y tablas.	13
7.- Desarrollo de la aplicación.....	14
7.1.- Tecnologías y herramientas utilizadas (lenguajes de programación, frameworks, bases de datos, etc.).	14
7.2.- Descripción de las principales funcionalidades implementadas ilustrándolo con fragmentos de código relevantes.	15
8.- Planificación del proyecto.....	18
8.1.- Acciones.....	18
8.2.- Temporalización y secuenciación	19
9.- Pruebas y validación	20
9.1.1- Prueba unitaria: Guardado correcto del modelo Clase	20
9.1.2- Prueba unitaria: Guardado correcto del modelo Producto	20
9.2.1- Prueba de Integración: Relación entre Compra y DetalleCompra	21
9.2.2- Prueba de Integración: Relación entre Usuario y Suscripción	22
9.3.1.- Prueba de integración: Método suscripcionActiva() del modelo Usuario	23
10.- Relación del proyecto con los módulos del ciclo.....	24
11.- Conclusiones	26
12.- Proyectos futuros	27
13.- Bibliografía/Webgrafía	28
14.- Anexos.....	29
14.1.- Capturas de pantalla	29
14.2.- Diagrama de Gantt.....	33
14.3.- Enlace al repositorio y datos.....	34

1.- Introducción

En los últimos años, el interés por el ejercicio físico y el estilo de vida saludable ha aumentado notablemente, en gran parte gracias a las redes sociales. Plataformas como Instagram o TikTok han contribuido a que más personas se acerquen al mundo del fitness, ya sea por salud, estética o por bienestar personal. Esta tendencia se intensificó tras la pandemia, cuando mucha gente tomó conciencia de lo poco que se movía en su día a día.

A partir de esta realidad, surge la idea de desarrollar Fitland, una aplicación web para la gestión de un gimnasio, pensada para adaptarse a necesidades actuales de los usuarios. El objetivo del proyecto es ofrecer una plataforma en la que cualquier persona pueda informarse sobre el gimnasio, apuntarse, ver las clases disponibles e inscribirse, consultar horarios y acceder a otras funciones de forma sencilla y desde cualquier lugar.

Con esta página web queremos facilitar el primer paso a aquellas personas que todavía no se han animado a entrenar, y también mejorar la experiencia de quienes ya lo hacen. Sabemos que para muchos el inicio en un gimnasio puede generar dudas o inseguridad, por lo que una herramienta digital que explique bien los servicios, los precios, el tipo de clases y otras ventajas puede marcar la diferencia a la hora de motivarse a entrenar.

Este proyecto también busca modernizar el concepto tradicional de gimnasio, integrando la tecnología para hacer más eficientes tanto la gestión interna como la comunicación con los usuarios.

La idea es que FitLand sea una ayuda real para acercar el ejercicio físico a más personas y contribuir así a una vida más activa y saludable.

1.1.- Descripción y contexto del proyecto

El trabajo se centra en el desarrollo de una aplicación web orientada a la gestión integral de un gimnasio, Fitland. El objetivo principal del proyecto es ofrecer una solución moderna, accesible y funcional para los clientes y para el personal encargado de administrar el gimnasio.

En los últimos años, el interés por la actividad física ha crecido de forma significativa, especialmente tras la pandemia, momento en el que muchas personas comenzaron a preocuparse más por su salud y su bienestar. Esta tendencia, unida a la normalización del uso de plataformas digitales para realizar gestiones cotidianas, ha creado la oportunidad perfecta para unir tecnología y deporte en una misma herramienta.

Este proyecto se centra especialmente en la construcción de una plataforma que permita a los usuarios y al personal administrativo interactuar con el sistema desde el frontend, mediante una interfaz clara e intuitiva, desde un panel de gestión completo. El desarrollo se apoya en tecnologías modernas como Laravel, React +Vite, Inertia, Tailwind CSS y Stripe para los pagos, lo cual permite una experiencia fluida tanto para los desarrolladores como para los usuarios. Además, la aplicación está preparada para su despliegue en Render.

La plataforma incluye funcionalidades esenciales para la experiencia del usuario, como el registro e inicio de sesión, una página de inicio, un módulo de tienda con carrito de compra, sistema de suscripciones,

inscripciones a clases, visualización de horarios y un panel de administración (backend), desde el cual se puede gestionar todo el contenido de la aplicación web a nivel de datos.

Con este proyecto no solo buscamos cubrir las necesidades básicas de un gimnasio moderno, sino también facilitar la interacción digital entre el centro y sus usuarios, permitiendo una gestión más cómoda, rápida y eficaz para ambas partes.

1.2.- Motivación del proyecto

La idea de este proyecto surgió de la observación y recopilación de información de muchos pequeños centros deportivos, que se ven obligados a depender de soluciones comerciales costosas o de hojas de cálculo manuales para organizar sus actividades. Esta situación genera errores, falta de trazabilidad, pérdida de tiempo y una experiencia deficiente para los clientes. FitLand busca ofrecer una alternativa viable, personalizable y fácilmente desplegable para centros que necesiten digitalizarse sin tener que asumir costes elevados ni conocimientos técnicos avanzados.

Además, este tipo de software puede escalar fácilmente hacia otros entornos como academias de formación, centros culturales o cualquier espacio que necesite gestionar clases, usuarios y pagos. El enfoque modular del sistema permite su adaptación según las necesidades específicas del entorno.

1.3.- Beneficios esperados

- Disminuir el trabajo administrativo manual mediante un panel de control centralizado.
- Mejorar la experiencia de los clientes gracias a la posibilidad de autogestionar sus inscripciones y pagos.
- Automatización de procesos críticos como la validación de pagos o el control de aforos.
- Reducir errores humanos en la gestión de clases, productos y compras.
- Aumento de la trazabilidad y el control sobre las operaciones diarias.
- Facilidad de mantenimiento y escalabilidad por el uso de herramientas modernas como Laravel, React+Vite e Inertia.

2.- Objetivo/s generales del proyecto

El objetivo principal del proyecto ha sido desarrollar una plataforma web completa para la gestión de un gimnasio, donde se puedan centralizar y automatizar tareas como la inscripción a clases, el control de pagos y suscripciones, la compra de productos, la gestión de usuarios y la generación de facturas. Todo esto se ha planteado con la idea de ofrecer una experiencia sencilla y cómoda tanto para los usuarios como para los administradores. Además, la aplicación ha sido preparada para poder desplegarla en producción utilizando la plataforma Render.

3.- Objetivos específicos

- Identificar los requisitos funcionales y no funcionales que debe cumplir el sistema.
- Diseñar una arquitectura web eficiente basada en el patrón cliente-servidor con backend en Laravel y frontend con React+Vite e Inertia.
- Desarrollar una interfaz gráfica moderna, accesible y adaptada a móviles mediante Tailwind CSS y componentes React.
- Implementar la lógica de gestión de productos, usuarios, suscripciones, clases, compras y pagos.
- Integrar la pasarela de pagos Stripe para permitir transacciones seguras online.
- Registrar cada compra en la base de datos, incluyendo detalles e historial.
- Crear un sistema de roles que distinga entre clientes y personal administrativo.
- Validar el sistema mediante pruebas unitarias, funcionales y de integración.
- Documentar todo el proceso de diseño, desarrollo y pruebas para futuras ampliaciones.
- Generar la factura de la compra del carrito o la suscripción.
- Desplegar la web en producción utilizando la plataforma Render

4.- Contexto actual

Existen herramientas conocidas en el mercado como Sporttia, Mindbody o Zen Planner que permiten digitalizar centros deportivos. Sin embargo, muchas de estas plataformas están pensadas para grandes empresas, tienen costes de suscripción elevados o presentan interfaces complejas. FitLand propone una solución adaptada a la realidad de centros más modestos que buscan una herramienta ágil, completa y personalizable. Además, al estar desarrollada sobre tecnologías web modernas y open source, se puede mantener y extender fácilmente.

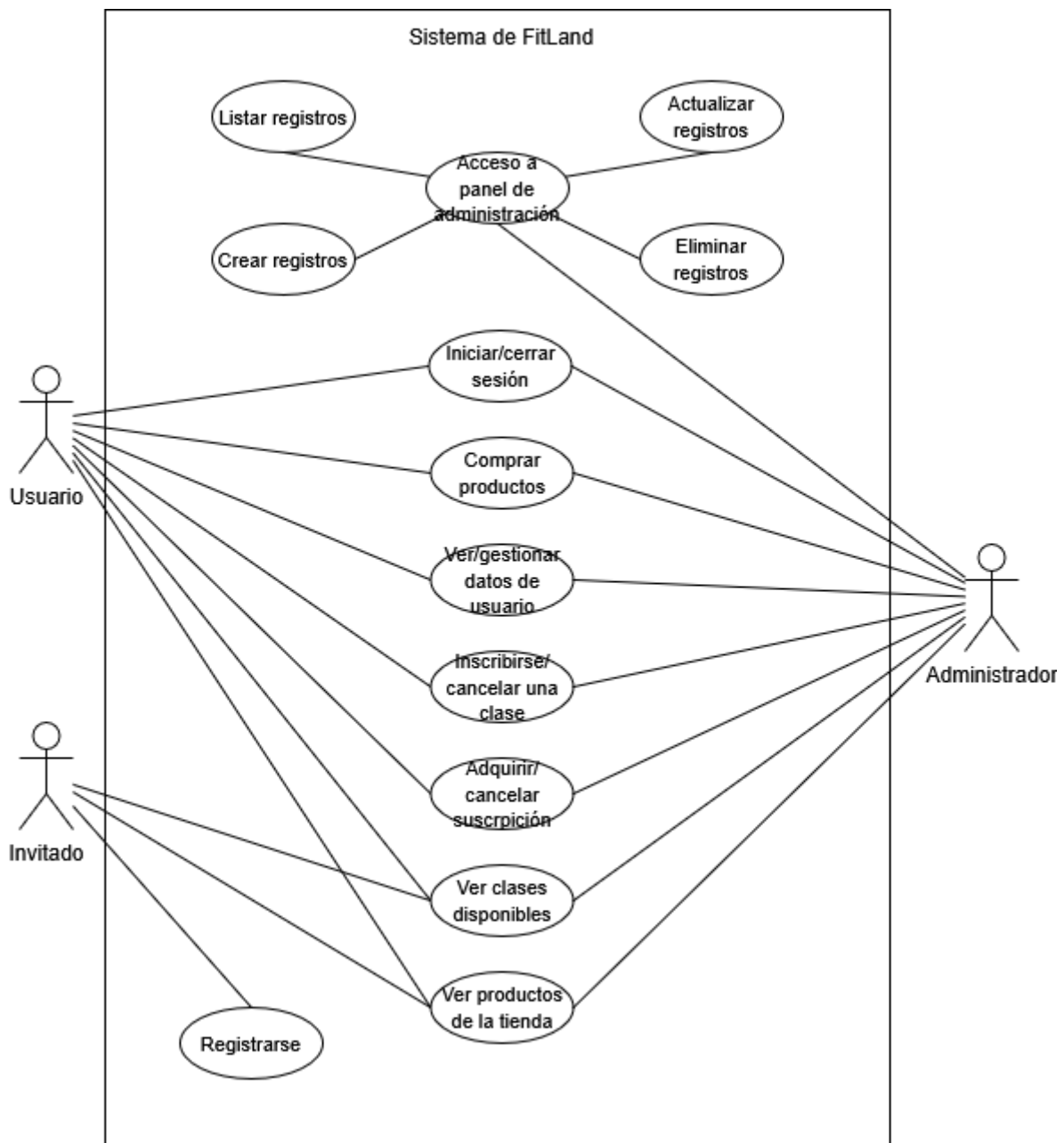
Conceptos clave:

- **Clase:** Evento con horario fijo, aforo y posibilidad de inscripción.
- **Usuario:** Persona registrada en el sistema, puede ser cliente o administrador.
- **Suscripción:** Plan mensual o anual que permite el acceso a clases.
- **Compra:** Operación que incluye productos adquiridos por un usuario.
- **Carrito:** Contenedor de productos pendientes de comprar.
- **Pago:** Transacción económica, asociada a compras o suscripciones.
- **Panel de Administrador:** Interfaz exclusiva para personal autorizado con funcionalidades CRUD por módulo.
- **Stripe:** Pasarela de pagos utilizada para transacciones seguras.
- **PostgreSQL:** Motor de base de datos utilizado por su robustez y capacidad de escalado.
- **Render:** Plataforma utilizada para desplegar la aplicación en producción, elegida por su facilidad de uso, integración con GitHub y soporte para entornos modernos como Laravel y PostgreSQL.

5.- Análisis de requisitos

La fase de análisis ha sido clave para definir el alcance del proyecto y establecer la base para su desarrollo. Antes de comenzar a programar, se identificaron los actores implicados, los casos de uso del sistema y los requisitos funcionales y no funcionales que debía cubrir la plataforma. Esto permitió estructurar el diseño del sistema y evitar errores o modificaciones innecesarias en etapas posteriores.

5.1.- Diagrama de casos de uso.



5.2.- Requisitos funcionales principales:

- Permitir el registro y login de usuarios.
- Gestión de la sesión y logout.
- Inscripción del usuario a clases disponibles, con control de aforo y validación según su tipo de suscripción.
- Visualización de las clases en un calendario semanal.
- Añadir productos al carrito, visualizarlo en tiempo real y realizar la compra.
- Integración con Stripe para generar automáticamente pagos al realizar compras o suscripciones.
- Restricción de inscripción a clases según el plan de suscripción (solo pueden inscribirse los usuarios con planes “Gold” o “Diamond”).
- Visualización y edición del perfil de usuario.
- Navegación intuitiva mediante barra de navegación: Mi Cuenta, Clases, Inscripciones, Tienda, Suscripciones y Carrito.
- Panel de administración completo con formularios para crear, editar y borrar cualquier entidad del sistema.
- Validación de formularios y protección contra accesos no autorizados.

5.3.- Requisitos no funcionales:

- **Rendimiento:** El sistema debe ofrecer un tiempo medio de respuesta inferior a 1 segundo para la mayoría de acciones de usuario (navegación, inscripciones, etc.).
- **Escalabilidad:** El sistema ha sido diseñado en arquitectura modular, permitiendo futuras ampliaciones (como gestión de entrenadores, app móvil, etc.).
- **Seguridad:**
 - Autenticación obligatoria para acceder a áreas restringidas.
 - Acceso a rutas administrativas limitado al rol de administrador.
 - Protección de rutas mediante middleware de Laravel.
 - Uso de HTTPS en despliegue final .
 - Integridad y trazabilidad garantizadas por registros de compras, pagos e inscripciones.

- **Usabilidad:**

- Interfaz clara e intuitiva, desarrollada con Tailwind CSS y React+Vite.
- Navegación accesible desde dispositivos móviles.
- Acciones agrupadas lógicamente en pestañas y secciones según el rol.

- **Trazabilidad:**

- Cada compra, pago e inscripción se guarda con su correspondiente usuario y fecha.
- Posibilidad de consultar registros históricos desde el panel admin.

5.4.- Descripción de los usuarios y sus necesidades.

- **Usuario registrado:**

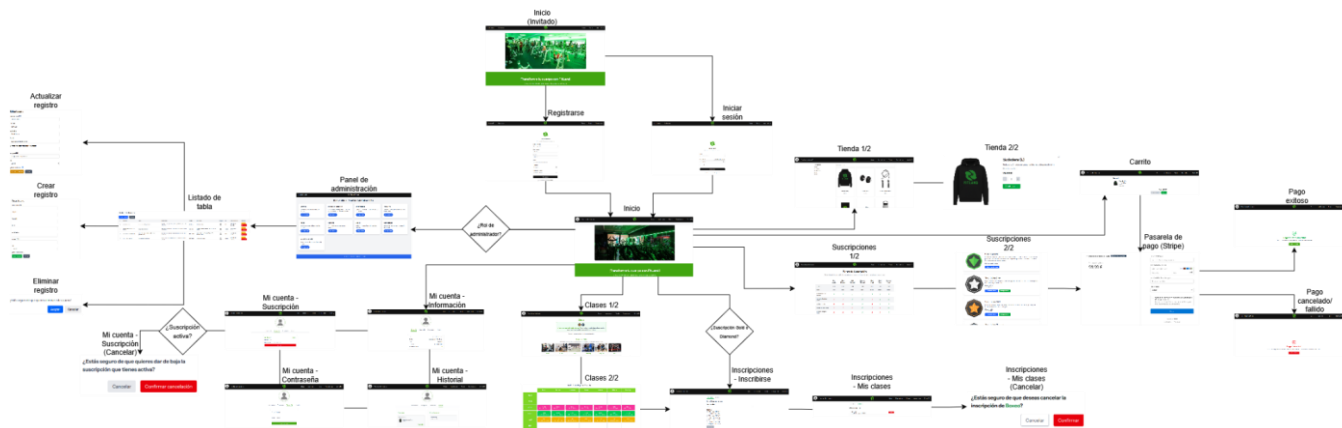
- Consultar clases disponibles y apuntarse según su suscripción.
- Comprar productos, revisar su carrito y pagar de forma segura.
- Necesita controlar su suscripción y tener acceso a su perfil.
- Uso de una interfaz clara y que funcione correctamente en la web.

- **Administrador:**

- Requiere controlar todas las entidades del sistema mediante un panel de administración.
- Necesita poder crear, editar y eliminar usuarios, productos, clases, compras, etc.
- Espera acceder fácilmente al historial de operaciones y pagos.
- Valora una interfaz organizada, rápida y con filtros útiles para trabajar con muchos registros.

6.- Diseño de la aplicación

6.1.- Wireframe de la interfaz gráfica de usuario.



6.2.- Arquitectura del sistema.

El sistema desarrollado para FitLand se basa en una arquitectura cliente-servidor con patrón MVC (Modelo-Vista-Controlador) implementado principalmente mediante el uso combinado de Laravel (PHP) para el backend y React (JavaScript) en el frontend, comunicándose entre ambos mediante Inertia.js. Para la persistencia de datos, se ha utilizado el sistema gestor de bases de datos PostgreSQL, una solución robusta, segura y ampliamente adoptada en entornos de producción.

Arquitectura Cliente-Servidor:

El sistema se divide en dos partes:

- **Cliente (Frontend):**
 - Desarrollado en React.
 - Renderiza las páginas que el usuario ve e interactúa con ellas (inicio, catálogo de productos, clases, suscripciones, inicio de sesión/registro, cuenta personal, etc.).
 - Usamos Inertia.js para facilitar la comunicación directa con Laravel sin necesidad de una API REST tradicional, manteniendo la experiencia de una SPA (Single Page Application).

- **Servidor (Backend):**

- Implementado en Laravel.
- Se encarga de la lógica del gimnasio, acceso a base de datos, validaciones, autenticación y control de permiso.
- A través de Inertia, responde a las acciones del usuario enviando datos directamente a los componentes de React.
- Laravel interactúa con PostgreSQL mediante Eloquent ORM, lo que permite mapear las entidades de dominio a tablas del sistema relacional de forma transparente.

Patrón de diseño MVC (Modelo - Vista - Controlador):

Laravel implementa de forma nativa este patrón en el que:

- **Modelo:**

- Usamos las entidades del sistema: Usuario, Producto, Suscripción, Clase, Compra, Inscripción, etc.

- **Vista:**

- En lugar de usar Blade como motor de vistas nativo de Laravel, optamos por React. Laravel, con Inertia.js, envía los datos directamente a los componentes React que representan las vistas de usuario.

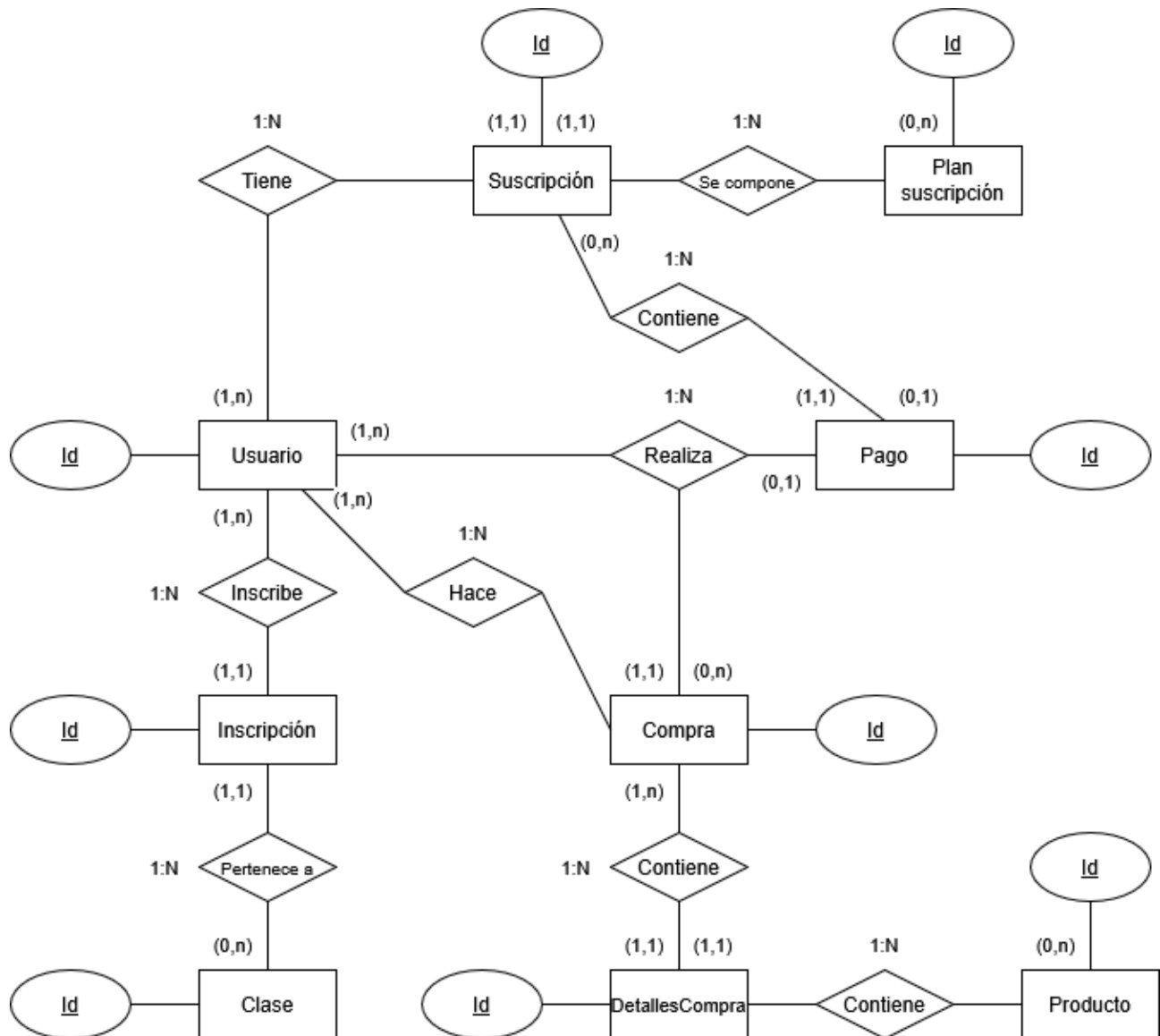
- **Controlador:**

- Gestiona la lógica de cada función: registrar un usuario, comprar productos, suscribirse, ver historial, inscribirse en clases, etc.
- Valida peticiones, coordina las acciones con los modelos y decide qué datos devolver al usuario.

Resumen tecnológico:

Capa	Tecnología	Rol principal
Cliente (UI)	React + Tailwind CSS	Interfaz gráfica y experiencia de usuario
Comunicación	Inertia.js	Enlace entre Laravel y React (SPA)
Lógica de negocio	Laravel (PHP)	Procesamiento de datos y reglas del negocio
Base de datos	PostgreSQL	Almacenamiento persistente y escalabilidad
Autenticación	Laravel Breeze	Registro, login, middleware y protección de rutas
Despliegue	Render	Servidor en la nube donde se aloja la app

6.3.- Diagrama de entidad-relación (E/R).



6.4.- Diseño de la base de datos: esquemas y tablas.



7.- Desarrollo de la aplicación

7.1.- Tecnologías y herramientas utilizadas (lenguajes de programación, frameworks, bases de datos, etc.).

Para llevar a cabo el desarrollo de FitLand hemos utilizado un conjunto de tecnologías modernas, cada una elegida por su eficiencia, facilidad de integración y la posibilidad de construir una aplicación escalable y profesional.

- **VSCode:** Ha sido nuestro editor principal. Lo elegimos por su ligereza, personalización, compatibilidad con plugins como Prettier y ESLint, y su integración con terminales, Git y Laravel.
- **Git y GitHub:** Hemos usado control de versiones de forma constante. Creamos un repositorio con el correo fitlandtfg@gmail.com y gestionamos los cambios de código usando comandos como git add . , git commit, git push y git pull. Esto nos permitió colaborar sin estorbarnos en el trabajo y mantener siempre una copia segura y actualizada del código.
- **Laravel 12:** Lo usamos como framework backend por su arquitectura MVC, claridad en la sintaxis, sistema de rutas, migraciones y facilidad para trabajar con bases de datos mediante Eloquent.
- **React + Vite:** React nos permitió crear componentes reutilizables e interfaces dinámicas. Vite, por su parte, facilitó un entorno de desarrollo moderno, con recarga en caliente y tiempos de compilación reducidos.
- **Inertia.js:** Elegimos Inertia porque simplifica mucho el desarrollo full stack, permitiéndonos conectar Laravel (backend) y React (frontend) sin necesidad de montar una API REST. Esto agiliza el flujo de trabajo y la integración de datos entre ambas partes.
- **Tailwind CSS:** Usamos este framework de estilos por su flexibilidad, utilidad en las clases HTML y gran capacidad de personalización responsive. Aunque al principio nos costó adaptarnos a él, al final conseguimos entenderlo y nos permitió construir una interfaz limpia y bien organizada.
- **PostgreSQL:** Usamos PostgreSQL como base de datos por su potencia, estabilidad y por ser ampliamente utilizada en entornos profesionales actuales.
- **Stripe:** Se integró para gestionar pagos online de forma segura, moderna y fiable.
- **DomPDF:** Decidimos implementarlo para generar facturas en PDF automáticamente tras una compra o suscripción, una funcionalidad inspirada en la experiencia en prácticas FCT con herramientas como Snappy.
- **Render:** Plataforma de despliegue en la nube que utilizamos para poner la web en producción. Su integración sencilla con GitHub y el soporte para entornos Laravel y PostgreSQL nos permitió publicar la aplicación de forma rápida.

7.2.- Descripción de las principales funcionalidades implementadas ilustrándolo con fragmentos de código relevantes.

A lo largo del proyecto hemos desarrollado múltiples funcionalidades clave. A continuación se presentan algunas de las más destacadas, ilustradas con fragmentos de código reales:

Inscripción a clases con validación de suscripción y aforo

Esta funcionalidad permite que solo usuarios con una suscripción activa tipo "Gold" o "Diamond" puedan inscribirse, y además valida que no se exceda el aforo máximo de la clase:

```
if (!$suscripcion || !in_array($suscripcion->plan->tipo, ['Gold', 'Diamond'])) {  
    return back()->withErrors([  
        'general' => 'Necesitas una suscripción Gold o Diamond para inscribirte en las clases.',  
    ]);  
}  
  
$clase = Clase::withCount('inscripciones')->findOrFail($request->clase_id);  
  
// Validar aforo disponible (aforo - inscritos)  
if ($clase->inscripciones_count >= $clase->aforo) {  
    return back()->withErrors(['general' => 'No puedes inscribirte a esta clase porque ya no hay plazas disponibles.']);  
}  
  
// Validar que la fecha no sea en el pasado  
if (now()->gt($request->fecha_inscripcion)) {  
    return back()->withErrors(['fecha_inscripcion' => 'No puedes inscribirte a una clase en el pasado.']);  
}  
  
// Guardar inscripción sin modificar aforo  
Inscripcion::create([  
    'usuario_id' => $request->usuario_id,  
    'clase_id' => $request->clase_id,  
    'fecha_inscripcion' => date('Y-m-d H:i:s', strtotime($request->fecha_inscripcion)),  
]);
```


Generación automática de factura en PDF

Cuando un usuario realiza una compra, se genera una factura en formato PDF usando DomPDF:

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Pago;
use Barryvdh\DomPDF\Facade\Pdf;
use Illuminate\Support\Facades\Auth;

class FacturaController extends Controller
{

    public function generarPDF()
    {
        $usuario = Auth::user();

        // Traer el último pago completado del usuario
        $pago = Pago::with(['compra.productos', 'suscripcion.plan'])
            ->where('usuario_id', $usuario->id)
            ->where('estado', 'completado')
            ->latest()
            ->first();

        if (!$pago) {
            abort(404, 'No se encontró un pago reciente');
        }

        $pdf = Pdf::loadView('facturas.factura', ['pago' => $pago]);

        return $pdf->stream('factura.pdf');
    }
}
```

Gestión del perfil de usuario

El usuario puede actualizar su imagen de perfil mediante un formulario. El backend valida y guarda la imagen:

```
public function actualizarImagen(Request $request)
{
    $request->validate(['imagen' => 'required|image|max:2048']);

    $user = $request->user();

    // Ruta física absoluta al directorio donde quieres guardar
    $carpetaDestino = public_path('/images/Perfil');

    // Crear la carpeta si no existe
    if (!file_exists($carpetaDestino)) {
        mkdir($carpetaDestino, 0755, true);
    }

    // Solo borramos si la imagen NO es la imagen por defecto
    if (
        $user->imagen &&
        $user->imagen !== '/images/defaults/avatar.jpg' &&
        file_exists(public_path($user->imagen))
    ) {
        unlink(public_path($user->imagen));
    }

    // Guardar con un nombre personalizado
    $extension = $request->file('imagen')->getClientOriginalExtension();
    $nombreArchivo = 'usuario_' . $user->id . '.' . $extension;

    $request->file('imagen')->move($carpetaDestino, $nombreArchivo);

    $user->imagen = '/images/Perfil/' . $nombreArchivo;
    $user->save();

    return response()->json([
        'foto_perfil_url' => asset($user->imagen),
        'success' => true,
    ]);
}
```

Visualización del horario de clases (frontend)

Desde React se listan las clases semanales con su horario para que los usuarios puedan inscribirse fácilmente:

```
{Object.keys(diasSemana).map((diaIng) => { alexp2004, last week * He hecho algunos cambios en la decoracion de la...
  const clase = clases.find((c) => {
    const [horaClase] = c.hora.split(':').map(Number);
    return c.dia === diaIng && horaClase === parseInt(hora);
  });
  return (
    <div key={diaIng + hora} className="bg-white h-20 p-1 border border-gray-300">
      {clase && (
        <div
          onClick={() => {
            window.location.href = estaLogueado ? '/inscribirse' : '/login';
          }}
          className={`
            ${coloresClase[clase.nombre] || 'bg-gray-500'}
            text-white rounded-md h-full w-full flex flex-col justify-center items-center text-center px-2 py-2
            shadow-sm hover:shadow-lg hover:brightness-110 transition duration-200 cursor-pointer
          `}
        >
          <div className="font-semibold text-sm leading-snug">{clase.nombre}</div>
          <div className="text-xs mt-1">
            {clase.hora} - {clase.aforo - clase.inscritos} de {clase.aforo} plazas
          </div>
        </div>
      )}
    </div>
  );
});
```

8.- Planificación del proyecto

8.1.- Acciones

Para desarrollar correctamente la plataforma de gestión del gimnasio Fitland, dividimos el trabajo en varias fases principales :

Toma de requisitos

- Definir funcionalidades básicas: registro, login, inscripción a clases, carrito, compras, pagos, facturas PDF.
- Recoger ideas de la empresa de prácticas sobre qué suele incluir una web real de gimnasio.
- Decidimos qué módulos necesitaría el panel de administración.

Análisis de requisitos

- Identificar relaciones entre los datos: qué información necesitaba cada tabla.
- Detectar qué restricciones de acceso habría según el rol del usuario, admin o cliente.
- Determinar qué acciones dependían de tener una suscripción activa.

Diseño

- Crear la estructura de la base de datos en PostgreSQL.
- Diseñar la navegación general de la aplicación inicio, mi cuenta, clases, tienda, suscripciones, inscripciones, login logout.
- Preparar vistas con Tailwind e Inertia React para que sea una experiencia limpia y rápida.

Codificación

- Configurar Laravel con Inertia.js, React, Tailwind y Vite.
- Programar el backend , las rutas, controladores y modelos Eloquent.
- Implementar el frontend con componentes reutilizables.
- Integrar Stripe para pagos y DomPDF para generar facturas en PDF.
- Desarrollar el sistema de control de suscripciones y su visualización en la página de Mi cuenta.

Pruebas

- Verificar la funcionalidad de cada módulo CRUD desde el panel de admin.
- Comprobar el flujo completo de compra e inscripción.
- Simular errores de usuario ,por ejemplo, contraseña incorrecta, pago fallido.
- Probar el comportamiento de la aplicación con y sin suscripción activa.

Documentación

- Escribir toda la memoria del proyecto detallando tecnologías usadas y decisiones tomadas y cambios.
- Añadir ejemplos de código representativo y capturas de pantalla si fuese necesario.
- Recoger la bibliografía y enlaces de todas las herramientas empleadas.
- Desplegar la aplicación con Render y documentar el proceso.

8.2.- Temporalización y secuenciación

Diagrama de Gantt

Temporalización y secuenciación del proyecto											
Fecha estimada de inicio: 31/03/2025											
Fecha estimada de finalización: 16/06/2025											
ACTIVIDADES	SEMANAS										
	1	2	3	4	5	6	7	8	9	10	11
Creación del proyecto y enlazado											
Creación del backend											
Creación del FrontEnd											
CORS (Cross-Origin Resource Sharing)											
Creación de la base de datos											
Creación y configuración del CRUD											
Creación de la páginas de FitLand (vista de usuario)											
Integración de interfaz de usuario											
Edición del apartado visual											
Despliegue de la API y pruebas											

9.- Pruebas y validación

9.1.1- Prueba unitaria: Guardado correcto del modelo Clase

Objetivo

Verificar que una instancia del modelo Clase puede guardarse correctamente en la base de datos cuando se le proporcionan los campos obligatorios.

Descripción

Esta prueba unitaria comprueba el funcionamiento básico del modelo Clase:

- Se crea una nueva instancia del modelo.
- Se le asignan valores válidos a los campos nombre, horario y aforo.
- Se utiliza el método `save()` para guardar el objeto.
- Finalmente, se verifica que el guardado ha sido exitoso mediante `assertTrue()`.

Resultado

La prueba se ejecutó correctamente, lo que confirma que el modelo Clase está correctamente configurado y puede persistir en la base de datos cuando se completan los datos requeridos.

9.1.2- Prueba unitaria: Guardado correcto del modelo Producto

Objetivo

Verificar que una instancia del modelo Producto puede guardarse correctamente en la base de datos y que sus datos quedan registrados conforme a lo esperado.

Descripción

Esta prueba unitaria valida la funcionalidad del modelo Producto de la siguiente manera:

- Se crea un nuevo producto usando `Producto::create()` con valores válidos para todos los campos obligatorios (nombre, descripción, precio, tipo, stock, imagen).
- A continuación, se utiliza el método `assertDatabaseHas()` para confirmar que el producto ha sido correctamente guardado en la tabla productos de la base de datos.
- Se comprueba específicamente que el nombre y el precio coinciden con los insertados.

Resultado

La prueba se ejecutó correctamente, confirmando que el modelo Producto permite guardar productos nuevos sin errores y que los datos se almacenan en la base de datos.

9.2.1- Prueba de Integración: Relación entre Compra y DetalleCompra

Objetivo

Esta prueba tiene como finalidad verificar que el modelo Compra puede relacionarse correctamente con uno o varios registros del modelo DetalleCompra. Se comprueba que la relación entre ambos modelos está bien definida y que permite acceder a los detalles desde una compra, y viceversa.

Descripción

En el sistema, cada usuario puede realizar compras que contienen productos. Estas compras se representan mediante el modelo Compra, mientras que los productos individuales adquiridos se gestionan mediante el modelo DetalleCompra.

Esta prueba simula una situación real donde:

1. Se crea un usuario con todos los campos obligatorios requeridos por la base de datos.
2. Se crea un producto válido con su nombre, tipo, precio y una imagen.
3. Se genera una compra asociada a ese usuario.
4. Se crea un detalle de compra que vincula dicho producto a la compra, indicando también la cantidad y el precio unitario.
5. Se verifica que:
 - o Desde la compra se puede acceder al detalle (\$compra->detalles).
 - o Desde el detalle se puede acceder a la compra (\$detalle->compra).

Resultado

La prueba se ejecutó correctamente, mostrando en consola que las relaciones entre los modelos son funcionales y devuelven los valores esperados.

9.2.2- Prueba de Integración: Relación entre Usuario y Suscripción

Objetivo

Verificar que un usuario puede tener correctamente asociada una suscripción activa, se ha desarrollado una prueba de integración que simula el proceso completo: creación de usuario, plan de suscripción y registro de la suscripción.

Descripción

La prueba se estructura en tres fases:

1. Creación de un usuario con todos los campos requeridos por la base de datos.
2. Creación de un plan de suscripción válido.
3. Registro de la suscripción asociando el plan y el usuario, con estado activo, y fechas de inicio y fecha fin válidas.

Finalmente, se verifican dos relaciones clave:

- Que el usuario tiene una suscripción activa asociada (\$usuario->suscripcionActiva).
- Que la suscripción accede correctamente al usuario (\$suscripcion->usuario).

Se utilizó el método refresh() para asegurar que el modelo de usuario incluya la relación recién creada antes de hacer las comprobaciones.

Esta prueba garantiza que:

- La base de datos acepta correctamente los registros relacionados.
- Las relaciones Eloquent están bien configuradas.
- El sistema es capaz de gestionar suscripciones activas correctamente en su lógica de negocio.

Resultado

La prueba se ejecutó correctamente, confirmando que un usuario puede tener correctamente asociada una suscripción activa.

9.3.1.- Prueba de integración: Método suscripcionActiva() del modelo Usuario

Objetivo

Validar que el método suscripcionActiva() del modelo Usuario devuelve correctamente la suscripción activa de un usuario, siempre que exista y esté dentro de las fechas establecidas.

Descripción

Esta prueba simula una situación real en la que:

- Se crea un usuario con todos los campos obligatorios.
- Se genera un plan de suscripción válido.
- Se crea una suscripción activa con fecha de inicio y fecha fin válida.
- Se comprueba que la función suscripcionActiva() devuelve correctamente esa suscripción activa.

Resultado

La prueba se ejecutó correctamente, confirmando que la relación entre Usuario y su suscripción activa está bien implementada y funcional.

10.- Relación del proyecto con los módulos del ciclo

Este proyecto está muy relacionado con lo aprendido durante el ciclo de Grado Superior de Desarrollo de Aplicaciones Web que hemos cursado en el IES el cañaveral. Desarrollando la web de FitLand, fuimos aplicando muchos de los conocimientos adquiridos en clase, y nos dimos cuenta de cómo cada módulo del ciclo tiene su reflejo en distintas partes del proyecto.

• Desarrollo Web en Entorno Cliente

En la parte del cliente trabajamos con React, que es fundamental para construir la interfaz de la aplicación. Con esta librería pudimos organizar la interfaz por componentes y gestionar el estado, los eventos y la interacción con los datos. También usamos Vite como entorno de desarrollo moderno, y añadimos librerías como swiper, clsx o @radix-ui/react para hacer la experiencia del usuario más dinámica y rápida.

• Desarrollo Web en Entorno Servidor

Para el backend utilizamos Laravel 12, aplicando la estructura MVC, el sistema de rutas, middleware, validaciones y controladores. También trabajamos con migraciones, modelos y relaciones entre tablas usando Eloquent. Esto nos ayudó a reforzar lo aprendido en este módulo, sobre todo en temas de lógica del servidor y control de acceso por roles.

• Despliegue de Aplicaciones Web

La aplicación fue preparada para funcionar en un entorno real, utilizamos Render como plataforma de despliegue. Desde ahí conectamos directamente con el repositorio de GitHub y automatizamos la publicación del proyecto. Durante este proceso configuramos las variables de entorno necesarias, integramos Stripe para gestionar pagos, conectamos la base de datos en PostgreSQL y habilitamos servicios como el correo electrónico y las colas de tareas. Todo esto nos sirvió para entender bien lo que implica pasar de una aplicación en desarrollo a una versión completamente funcional y disponible online.

• Bases de Datos

Usamos PostgreSQL como gestor de base de datos. Diseñamos tablas relacionadas como usuarios, productos, clases, compras, pagos, etc. Usamos relaciones del tipo hasMany, belongsTo, y aseguramos la integridad referencial con claves primarias y foráneas. Este módulo nos ayudó a estructurar bien la base de datos desde el inicio del proyecto.

• Entornos de Desarrollo

El desarrollo se hizo principalmente en local, usando Laravel, Inertia y PostgreSQL. Usamos VSCode como editor de código, y herramientas como ESLint y Prettier para mantener el código limpio y uniforme. También trabajamos con Git como sistema de control de versiones. Con esto logramos mantener una buena organización y flujo de trabajo.

- **Diseño de Interfaces Web y lenguaje de marcas**

En cuanto a la parte visual usamos Tailwind CSS, que nos permitió crear un diseño adaptable y limpio. Incorporamos componentes accesibles como @headlessui/react y lucide-react, y verificamos que todo funcionara bien tanto en el ordenador. Siempre intentamos que la experiencia del usuario fuera clara, rápida y fácil de usar.

- **Formación en Centros de Trabajo**

Uno de nosotros, durante sus prácticas en una empresa, trabajó con una herramienta llamada Snappy para generar PDFs. Investigando un poco más, dedujimos que DomPDF encajaba mejor con Laravel, así que decidimos integrarlo en el proyecto. Esto nos inspiró, y ahora el sistema genera automáticamente una factura en PDF cada vez que se realiza una compra o pago. Esta funcionalidad está inspirada directamente en la experiencia adquirida en el entorno profesional.

11.- Conclusiones

La verdad es que hacer este proyecto nos ha servido muchísimo. Para poner en práctica lo que hemos aprendido en clase, y también para ver de verdad cómo es desarrollar una aplicación web completa desde cero. Empezamos con una idea general y con algunos cambios fuimos dando forma a la web de FitLand, que hoy por hoy es una página funcional, intuitiva y que cumple con la mayoría de requisitos nos propusimos.

Una de las partes que más nos costó fue entender cómo usar Tailwind CSS. No lo habíamos tocado antes y al principio fue un poco lioso organizar los estilos. También tuvimos algún que otro lío con los formularios, sobre todo al darnos cuenta de que necesitábamos más campos de los que pensábamos, y eso nos obligó a ajustar varias cosas en la lógica de las tablas y en la base de datos.

Si hay algo que realmente nos ha aportado este proyecto es haber descubierto herramientas como Inertia, que no conocíamos y que nos sorprendió mucho, ya que permite trabajar frontend y backend juntos sin tener que montar una API. Usar Laravel 12, React y Vite también ha sido un reto muy interesante, porque en clase habíamos trabajado solo con Symfony o Angular, y esto ha sido una experiencia nueva que nos ha enseñado mucho. Además, hemos aprendido cómo se gestiona un proyecto en pareja: repartir el trabajo, organizar los tiempos, ayudarnos cuando uno se atascaba, y eso ha sido clave para llegar hasta el final.

También cambiamos bastantes cosas sobre la marcha. Por ejemplo, al principio no pensábamos generar PDFs, pero tras las prácticas en una empresa donde uno de nosotros trabajó con Snappy, decidimos buscar algo parecido y más simple que encajara con Laravel y fue cuando encontramos DomPDF, que al final usamos para generar las facturas automáticamente tras cada compra o pago, después de que funcionara todo utilizamos Render para desplegar la web.

Trabajar en equipo ha funcionado muy bien. Nos hemos entendido, repartido bien las tareas y nos hemos apoyado en todo momento. Si uno no podía avanzar un día, el otro se encargaba, y siempre estuvimos en contacto por si surgía algún imprevisto. Seguimos una planificación bastante clara, como se ve en el diagrama de Gantt, y eso nos ayudó a ir avanzando sin agobiarnos.

En resumen, ha sido un proyecto que hemos disfrutado mucho. Ver cómo iba tomando forma y cómo cada nueva funcionalidad mejoraba la web nos motivaba a seguir. Es de estos proyectos en los que empiezas con una cosa y con una idea clara y acabas queriendo añadirle mucho más. Nos sentimos orgullosos del trabajo que hemos hecho y, sobre todo, de haber construido algo útil, bien hecho y con potencial para seguir creciendo.

12.- Proyectos futuros

Aunque estamos muy contentos con cómo ha quedado FitLand, durante el desarrollo nos han ido surgiendo muchas ideas que por cuestión de tiempo no pudimos llevar a cabo, pero que nos encantaría implementar en un futuro.

Una de las cosas que teníamos en mente era generar un código QR que permitiera a los usuarios invitar fácilmente a un amigo. Habíamos pensado que esa invitación estuviera vinculada a la suscripción Diamond, para hacerla más atractiva. También nos habría gustado preparar una versión móvil de la aplicación, ya que hoy en día mucha gente accede desde el teléfono y eso mejoraría mucho la experiencia.

Además, vimos que sería muy útil contar con un sistema de reservas, para salas y para entrenadores personales. Esto daría más flexibilidad al usuario y permitiría una mejor organización por parte del gimnasio. También queríamos incluir un sistema de notificaciones, para que el usuario reciba un aviso automático cuando vaya a empezar una clase o cuando pase algo importante relacionado con su cuenta o su suscripción.

Por otro lado, este proyecto podría servir de base para otros más adelante. Por ejemplo, podría adaptarse fácilmente para academias, centros de formación o cualquier otro negocio que trabaje con suscripciones, clases y pagos. Si lo seguimos desarrollando, podríamos añadir funciones más avanzadas, como pagos recurrentes o un chat entre usuarios y entrenadores. Podríamos llevarlo a móvil con React Native, ahora que ya tenemos la base web funcionando bien.

En resumen, aunque FitLand ya está completo, sabemos que todavía tiene mucho potencial de crecimiento y nos satisface y motiva el poder seguir mejorándolo poco a poco.

13.- Bibliografía/Webgrafía

Tailwind CSS (2025), "Tailwind CSS Documentation", <https://tailwindcss.com/docs>

ReactJS (2025), "React – A JavaScript library for building user interfaces", <https://react.dev>

Laravel (2025), "Laravel - The PHP Framework for Web Artisans", <https://laravel.com/docs>

Inertia.js (2025), "Inertia.js Documentation", <https://inertiajs.com>

Vite (2025), "Vite - Next Generation Frontend Tooling", <https://vitejs.dev>

Stripe (2025), "Stripe Documentation - Payments Integration", <https://stripe.com/docs>

DomPDF (2025), "dompdf/dompdf - GitHub repository", <https://github.com/dompdf/dompdf>

PostgreSQL (2025), "PostgreSQL: Administration", <https://neon.com/postgresql/postgresql-administration>

Neider Ruiz (2024), "CRUD Full stack con Laravel, Inertia y React",
<https://www.youtube.com/watch?v=G4pUX8aVGGQ>

The Net Ninja (2023), "Tailwind CSS Crash Course", <https://www.youtube.com/watch?v=ft30zcMIFao>

W3Schools (2025), "Learn Web Development", <https://www.w3schools.com>

Lucide Icons (2025), "Lucide – Beautiful & consistent icon toolkit", <https://lucide.dev>

Headless UI (2025), "Accessible UI components built with Tailwind CSS", <https://headlessui.com>

Git (2025), "Git - Distributed Version Control System", <https://git-scm.com>

GitHub (2025), "GitHub Docs – Collaborate and build better software", <https://docs.github.com>

Stack Overflow (2025), "Respuestas y soluciones a errores comunes", <https://stackoverflow.com>

Dev.to (2025), "Community articles on Laravel, React and full stack development", <https://dev.to>

Everyday.codes (2023), "Guías prácticas sobre Inertia.js y Laravel", <https://everyday.codes/inertia-js-with-laravel/>

LogRocket Blog (2025), "Understanding SSR in React with Inertia.js", <https://blog.logrocket.com>

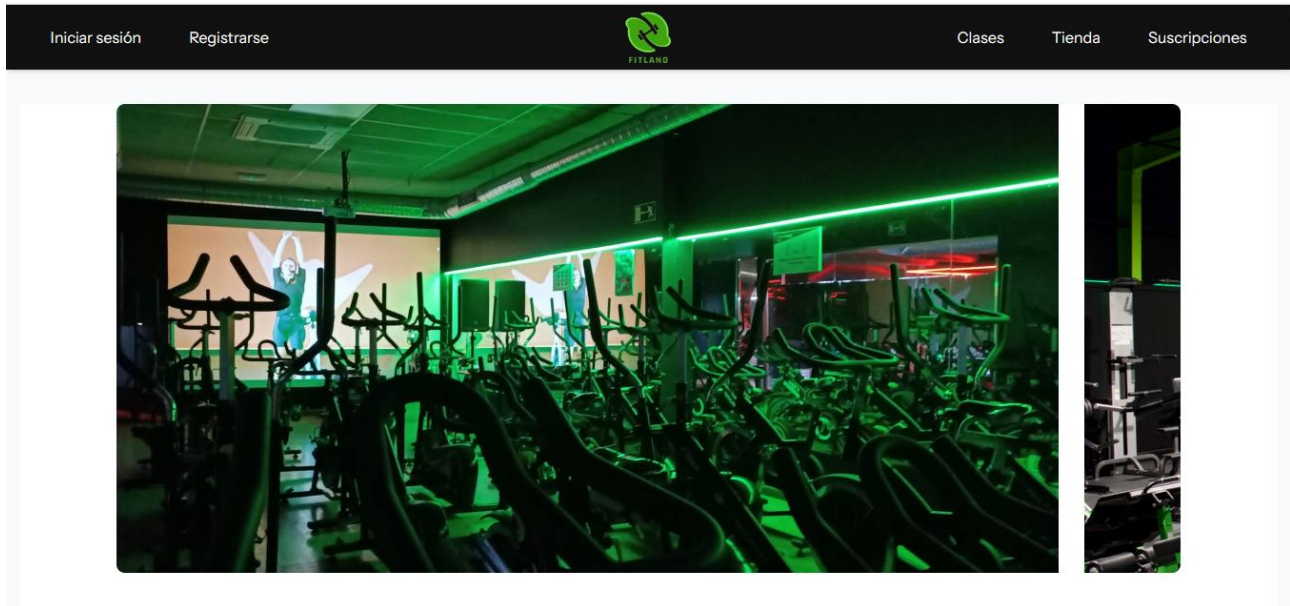
Render (2025), "Render - The modern cloud for developers", <https://render.com/docs>

Swiper.js (2025), "Swiper - The Most Modern Mobile Touch Slider": <https://swiperjs.com/demos>

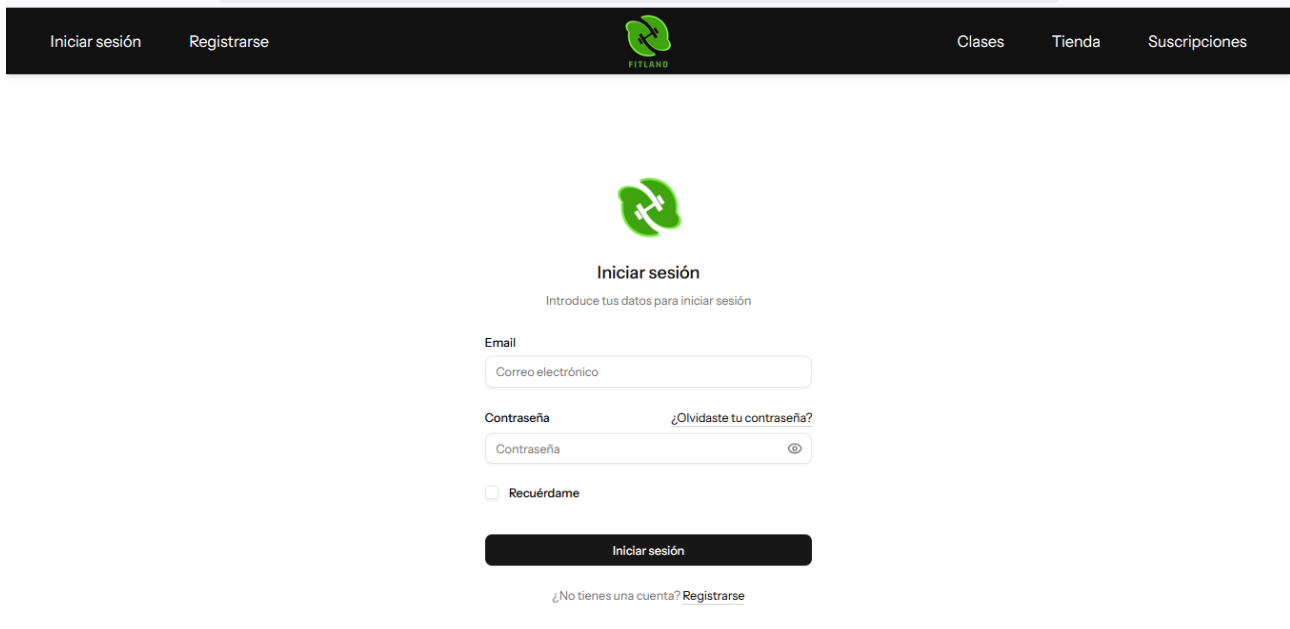
14.- Anexos

14.1.- Capturas de pantalla


Inicio:




Login:



Registro:

[Iniciar sesión](#)
[Registrarse](#)

[Clases](#)
[Tienda](#)
[Suscripciones](#)



Crear una cuenta

Introduce tus datos para crear una cuenta

Nombre y apellidos

Documentación

Domicilio

Email

Contraseña

Confirmar contraseña

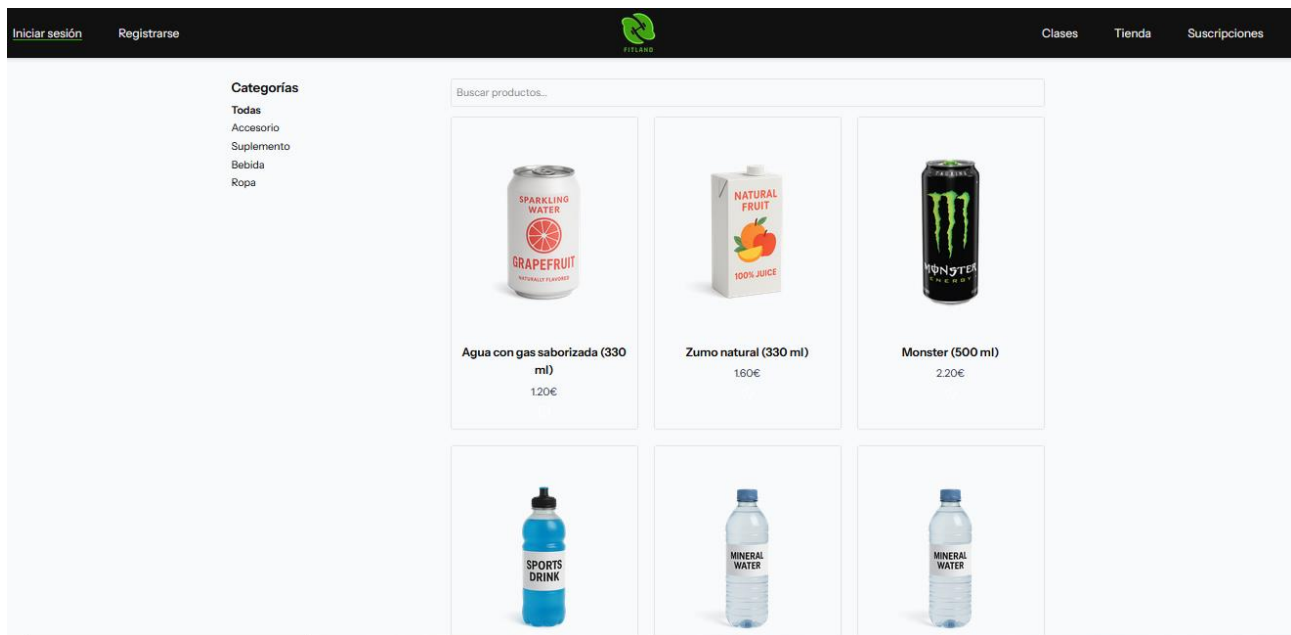
[Crear cuenta](#)

[¿Ya tienes una cuenta? Iniciar sesión](#)


Clases:

Del 16 al 22 de junio de 2025							
	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
06:00							
07:00							
08:00	Zumba 08:00 - 0 de 20 plazas	Zumba 08:00 - 20 de 21 plazas	Zumba 08:00 - 20 de 20 plazas	Zumba 08:00 - 20 de 20 plazas	Zumba 08:00 - 20 de 20 plazas	Zumba 08:00 - 20 de 20 plazas	Zumba 08:00 - 20 de 20 plazas
09:00	Pilates 09:20 - 19 de 18 plazas	Pilates 09:20 - 18 de 18 plazas	Pilates 09:20 - 18 de 18 plazas	Pilates 09:20 - 18 de 18 plazas	Pilates 09:20 - 18 de 18 plazas	Pilates 09:20 - 18 de 18 plazas	Pilates 09:20 - 18 de 18 plazas
10:00	GAP 10:40 - 16 de 16 plazas	GAP 10:40 - 15 de 15 plazas	GAP 10:40 - 15 de 15 plazas	GAP 10:40 - 15 de 15 plazas	GAP 10:40 - 15 de 15 plazas	GAP 10:40 - 15 de 15 plazas	GAP 10:40 - 15 de 15 plazas
11:00							
12:00	HIIT 12:00 - 16 de 16 plazas	HIIT 12:00 - 16 de 16 plazas	HIIT 12:00 - 16 de 16 plazas	HIIT 12:00 - 16 de 16 plazas	HIIT 12:00 - 16 de 16 plazas	HIIT 12:00 - 16 de 16 plazas	HIIT 12:00 - 16 de 16 plazas
13:00	Full Body Workout 13:20 - 18 de 18 plazas	Full Body Workout 13:20 - 18 de 18 plazas	Full Body Workout 13:20 - 18 de 18 plazas	Full Body Workout 13:20 - 18 de 18 plazas	Full Body Workout 13:20 - 18 de 18 plazas	Full Body Workout 13:20 - 18 de 18 plazas	Full Body Workout 13:20 - 18 de 18 plazas
14:00	Cardio Dance 14:40 - 22 de 22 plazas	Cardio Dance 14:40 - 22 de 22 plazas	Cardio Dance 14:40 - 22 de 22 plazas	Cardio Dance 14:40 - 22 de 22 plazas	Cardio Dance 14:40 - 22 de 22 plazas	Cardio Dance 14:40 - 22 de 22 plazas	Cardio Dance 14:40 - 22 de 22 plazas

Tienda:



Suscripciones:




Plan de prueba

Con este plan de prueba gratuito, podrás acceder al gimnasio sin límites durante 30 días. Es ideal para probar nuestras instalaciones antes de contratar un plan completo. Solo puede usarse una vez y sin haber contratado otra suscripción antes.

30 días de prueba gratuitos

[Obtener plan de prueba](#)




Suscripción Silver

El plan más económico. Incluye acceso ilimitado al gimnasio durante el periodo contratado. No permite inscribirse a ninguna de las clases.

Mensual: 9.99 €
Anual: 99.99 €

[Suscripción mensual](#) [Suscripción anual](#)




Suscripción Gold


Incluye todo lo del plan Silver, además de inscripción ilimitada a cualquier clase del gimnasio y una mochila de deportiva de FitLand de regalo.

Mensual: 14.99 €
Anual: 149.99 €

[Suscripción mensual](#) [Suscripción anual](#)

Inscripciones:

 Panel de administración



ClasesInscripcionesTiendaSuscripcionesCarrito (0)

[Inscribirse](#) [Mis clases](#)

Inscribirse a una clase


Nombre de clase


Selecciona una clase

Inscribirse

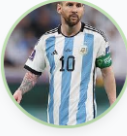
© 2025 FitLand. Todos los derechos reservados.

Mi cuenta:

 Panel de administración




ClasesInscripcionesTiendaSuscripcionesCarrito (0)




[Información](#) [Suscripción](#) [Contraseña](#) [Historial](#)

Nombre:	Brandon Muzo
Email:	cuentawamp@gmail.com
DNI:	59876789K
Domicilio:	Calle Libertad 21
Suscripción activa:	Sí


Carrito:

 Panel de administración



ClasesInscripcionesTiendaSuscripcionesCarrito (1)

Tu carrito



Mancuerna
Precio: €10.99
Stock: 34
- 1 +

Total: €10.99
Descuento (10%): -€1.10
Total a pagar: €9.89

Vaciar carrito **Pagar**

Panel administración:

Volver al inicio

Administrador

Bienvenido al Panel de Administración

Usuarios

Gestiona los usuarios registrados en FitLand.

Ver opciones

Planes de suscripción

Gestiona los distintos planes a los que se podrán suscribir los usuarios registrados.

Ver opciones

Suscripciones

Gestiona las suscripciones disponibles para los usuarios.

Ver opciones

Productos

Gestiona los productos disponibles en la tienda online.

Ver opciones

Pagos

Gestiona los pagos realizados por los usuarios.

Ver opciones

Compras

Gestiona las compras realizadas por los usuarios.

Ver opciones

Clases

Gestiona las clases disponibles para los usuarios.

Ver opciones

Inscripciones

Gestiona las inscripciones a las clases de los usuarios.

Ver opciones

Detalle de Compra

Gestiona los productos asociados a cada compra.

Ver opciones

© 2025 FitLand. Todos los derechos reservados.

Usuarios:

Gestión de Usuarios

Crear Usuario

Volver

ID	Nombre	Email	Contraseña	DNI/NIF	Domicilio	Imagen	Rol	Email Verificado	Acciones
35	Test User	test684f56db08dc2@user.com	\$2y\$04\$9l8tu34GgKd8GQvWXXtTKGsd4stBBASl68z2DyaJXZ7gD oKJVV	12345678A	Calle Falsa 123	Imagen	User	No verificado	<div>Editar</div> <div>Eliminar</div>
34	Test User	test684f54fc16882@user.com	\$2y\$04\$01MB67n7mKd2cg8qjHlQY9InFskMgIO38CohnGWAIG/3 qgBOW	12345678A	Calle Falsa 123	Imagen	User	No verificado	<div>Editar</div> <div>Eliminar</div>
33	Test User	test684f545ed04cb@user.com	\$2y\$04\$4PjDpPhJXyYkTWuFQdeQFwYzWuvYXsR9kPDID80 yShuS	12345678A	Calle Falsa 123	Imagen	User	No verificado	<div>Editar</div> <div>Eliminar</div>
32	Test User	test684f540c5a0c2@user.com	\$2y\$04\$0tceqVBAD60LSvHbPQwWu5veun4ZG48TjZgQCeqyIu7m c0Q2U3	12345678A	Calle Falsa 123	Imagen	User	No verificado	<div>Editar</div> <div>Eliminar</div>
31	Test User	test684f53e0332a8@user.com	\$2y\$04\$5vew2ZRW8huu4fZ2cTxUgpyVwXmm8bcX/W2MFn/1HkQ.4 yge	12345678A	Calle Falsa 123	Imagen	User	No verificado	<div>Editar</div> <div>Eliminar</div>
30	Test User	test684f53616377@user.com	\$2y\$04\$beIDlkoXlbgCRbgS1wpIqO5uQvFvDOnwSP9VkuANhp0 jf0a95K	12345678A	Calle Falsa 123	Imagen	User	No verificado	<div>Editar</div> <div>Eliminar</div>
29	Test User	test684f532166232@user.com	\$2y\$04\$AdHlEEVbWqTBCNbv5vY0exuJDer6ZB7PBD4/kvOQZ6A/3 FXR20	12345678A	Calle Falsa 123	Imagen	User	No verificado	<div>Editar</div> <div>Eliminar</div>

14.2.- Diagrama de Gantt



14.3.- Enlace al repositorio y datos

Repositorio GitHub: <https://github.com/FitLandTFG/FitLand.git>

Correo del equipo: fitlandtfg@gmail.com