**MEMORIA FINAL DE PROYECTO**

**Icono

El contenido generado por IA puede ser incorrecto.**

**CICLO FORMATIVO DE GRADO SUPERIOR**

**DESARROLLO DE APLICACIONES WEB**

**AUTOR**

JORDI RUIZ ROMÁN

**TUTOR**

JOSÉ JUAN LÓPEZ VÉLEZ

**COORDINADOR**

LORENA ISABEL PALACIO LATORRE

ÍNDICE

[1. INTRODUCCIÓN 5](#_Toc183456229)

[2. INTRODUCCIÓN EN INGLÉS 5](#_Toc183456230)

[3. OBJETIVOS 5](#_Toc183456231)

[3.1. Objetivos fase actual 5](#_Toc183456232)

[3.2. Objetivos fases futuras 5](#_Toc183456233)

[4. PLANIFICACIÓN 5](#_Toc183456234)

[4.1. Tabla de hitos 5](#_Toc183456235)

[4.2. Diagrama de Gantt 6](#_Toc183456236)

[5. ANÁLISIS 6](#_Toc183456237)

[5.1. Estado del arte 6](#_Toc183456238)

[5.2. Funcionalidades 6](#_Toc183456239)

[6. DISEÑO 6](#_Toc183456240)

[6.1. Requisitos técnicos 6](#_Toc183456241)

[6.2. Arquitectura web 7](#_Toc183456242)

[6.3. Diseño back-end 7](#_Toc183456243)

[6.3.1. Modelo de datos 7](#_Toc183456244)

[6.3.2. Servicios REST 7](#_Toc183456245)

[6.3.3. Paquetes adicionales 7](#_Toc183456246)

[6.4. Diseño front-end 7](#_Toc183456247)

[6.4.1. Mock-ups 7](#_Toc183456248)

[6.4.2. Guía de estilos 7](#_Toc183456249)

[6.4.3. Paquetes adicionales 7](#_Toc183456250)

[7. IMPLEMENTACIÓN 8](#_Toc183456251)

[7.1. Servidor 8](#_Toc183456252)

[7.2. Cliente 8](#_Toc183456253)

[8. DESPLIEGUE 8](#_Toc183456254)

[8.1. Modelo de despliegue utilizado 8](#_Toc183456255)

[8.2. Datos iniciales y configuración 8](#_Toc183456256)

[8.3. Pasos para el despliegue 8](#_Toc183456257)

[8.4. Proveedores y servicios utilizados 8](#_Toc183456258)

[9. HERRAMIENTAS UTILIZADAS 8](#_Toc183456259)

# INTRODUCCIÓN

Este documento presenta el trabajo realizado para el módulo de Proyecto del Ciclo Formativo de Grado Superior en Desarrollo de Aplicaciones Web. El objetivo principal es aplicar de forma práctica los conocimientos adquiridos durante el ciclo, desarrollando una aplicación web completa desde su planificación hasta su implementación.

El proyecto consiste en una página web para un gimnasio, que ofrece información sobre sus servicios, centros, guías de entrenamientos y herramientas para el seguimiento del progreso físico de los usuarios. Se ha utilizado una combinación de tecnologías modernas para ofrecer una solución útil, atractiva y funcional.

# INTRODUCCIÓN EN INGLÉS

This document presents the work carried out for the Project Module of the Higher Level Training Cycle in Web Application Development. The main objective is to practically apply the knowledge acquired during the cycle by developing a complete web application, from its planning to its implementation.

The project consists of a website for a gym, offering information about its services, locations, training guides, and tools for tracking users' physical progress. A combination of modern technologies has been used to provide a useful, attractive, and functional solution.

# OBJETIVOS

El propósito de este proyecto es desarrollar una aplicación web funcional y estructurada para un gimnasio, que sirva como plataforma centralizada para informar, asistir y fidelizar a los usuarios. El objetivo general es implementar una solución que combine servicios informativos, rutinas personalizadas y funcionalidades básicas de comercio electrónico, aplicando las tecnologías y conocimientos adquiridos a lo largo del ciclo formativo.

## Objetivos fase actual

Durante la fase correspondiente al módulo de Proyecto, se plantea alcanzar una solución básica pero completamente funcional, con los siguientes objetivos:

* Crear una estructura web clara, usable y responsive.
* Mostrar información detallada sobre centros, horarios y tarifas.
* Permitir el registro y autenticación de usuarios.
* Habilitar una zona privada con acceso a contenidos personalizados.
* Implementar un sistema de visualización de ejercicios mediante un cuerpo humano interactivo.
* Diseñar e integrar un sistema de progreso con seguimiento en gráficos y estadísticas.
* Conectar la aplicación con una base de datos para gestionar usuarios, ejercicios y centros.
* Simular el sistema de pago de Redsys para las diferentes tarifas del gimnasio.

## Objetivos fases futuras

Los siguientes objetivos se consideran ampliaciones que podrían incorporarse en futuras fases del proyecto, una vez consolidada la base funcional:

* Integración de un sistema de gamificación (puntos, logros, recompensas).
* Venta de productos de suplementación deportiva.
* Valoraciones y comentarios en productos y rutinas.
* Mejora del sistema de búsqueda con filtros avanzados.
* Implementación de métodos de pago reales y notificaciones de pedidos.
* Optimización de la seguridad y protección de datos personales.

# PLANIFICACIÓN

Para garantizar el éxito del desarrollo durante el módulo de Proyecto, se ha definido una planificación que divide el trabajo en fases concretas. Cada fase incluye tareas específicas con una duración estimada, lo que permite organizar el tiempo de manera eficiente y detectar posibles desviaciones.

## Tabla de hitos

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Fase |  |  | | --- | |  | | Tareas asociadas | Duración estimada | Fecha inicio | Fecha fin |
| |  | | --- | | **1. Análisis y diseño** |  |  | | --- | |  | | - Definición de requisitos - Diseño de estructura y base de datos | |  | | --- | | 7 días |  |  | | --- | |  | | |  | | --- | | 14/04/2025 |  |  | | --- | |  | | |  | | --- | | 21/05/2025 |  |  | | --- | |  | |
| **2. Maquetación web** | - Diseño de la interfaz principal (HTML, CSS, JS)  - Zona pública básica | 10 días | 22/04/2025 | |  | | --- | | 1/05/2025 |  |  | | --- | |  | |
| |  | | --- | | **3. Backend básico** |  |  | | --- | |  | | - Configuración de Django - Modelado de datos - Registro/login | 7 días | |  | | --- | | 1/05/2025 |  |  | | --- | |  | | |  | | --- | | 7/05/2025 |  |  | | --- | |  | |
| |  | | --- | | **4. Zona privada** |  |  | | --- | |  | | - Zona del staff - Acceso personalizado por usuario | |  | | --- | | 5 días |  |  | | --- | |  | | |  | | --- | | 8/05/2025 |  |  | | --- | |  | | |  | | --- | | 12/05/2025 |  |  | | --- | |  | |
| Cuerpo interactivo | -Visualizar videos por músculo | 7 días | 13/05/2025 | 20/05/2025 |
| Ingresar progreso | -Insertar pesos y repeticiones para cada usuario | 5 días | 21/05/2025 | 26/05/2025 |
| Seguimiento del progreso | -Seguimiento mediante gráficos  -Estadísticas de los pesos y ejercicios | 4 días | 27/05/2025 | 30/05/2025 |
| Despliegue del proyecto | -Despliegue en render | 2 días | 31/05/2025 | 01/06/2025 |
| Tiempo de mejorar el proyecto y solucionar errores | -Mejora del diseño  -Mejora del apartado responsive | 8 días | 02/06/2025 | 09/06/2025 |

## Diagrama de Gantt

Gráfico, Gráfico en cascada

El contenido generado por IA puede ser incorrecto.

# 5.ANÁLISIS

## Estado del arte

Actualmente existen numerosas aplicaciones web y móviles orientadas al mundo del fitness, como **MyFitnessPal**, **Fitbit**, **Freeletics**, **EntrenApp** o **Strong**, que permiten a los usuarios realizar un seguimiento de sus entrenamientos, controlar su nutrición, registrar su progreso físico y acceder a rutinas predefinidas o personalizadas.

Estas plataformas suelen destacar por una experiencia de usuario muy pulida y, en muchos casos, incluyen integración con dispositivos inteligentes, recomendaciones personalizadas basadas en inteligencia artificial y comunidades activas.

Sin embargo, muchas de estas soluciones son **cerradas, de pago o poco personalizables**, lo que puede limitar su uso en entornos educativos, centros deportivos específicos o proyectos personales.

La propuesta desarrollada en este proyecto, **FitCore**, se enfoca en cubrir funcionalidades clave de seguimiento de entrenamientos y progresos físicos de forma **gratuita, simple y autogestionable**. Además, destaca por permitir una **vinculación directa con centros deportivos** y una estructura de datos pensada para ser **extendida fácilmente en fases futuras** (como incluir rutinas, tienda o visualización del cuerpo humano interactivo).

En resumen, aunque existen soluciones comerciales avanzadas, FitCore se diferencia por su enfoque accesible, educativo y modular, siendo una base sólida para futuras ampliaciones sin depender de servicios externos.

## Funcionalidades

A continuación, se enumeran las funcionalidades que formarán parte de esta primera fase del desarrollo del proyecto, garantizando una base sólida y operativa para la aplicación web del gimnasio:

1. Página informativa del gimnasio

Visualización de los diferentes centros disponibles.

Descripción de instalaciones y horarios.

Visualización de tarifas actualizadas.

2. Gestión de usuarios

Registro de nuevos usuarios con validación de datos.

Inicio de sesión seguro con autenticación.

Zona privada para el usuario con acceso personalizado a funcionalidades.

3. Cuerpo humano interactivo (versión inicial)

Visualización de figura anatómica con zonas musculares destacadas.

Selección de músculo y muestra de ejercicios recomendados asociados.

# 4.Seguimiento del progreso (versión inicial)

Registro manual del progreso: peso levantado, repeticiones y fechas.

Visualización de registros mediante gráficos y estadísticas.

5. Sistema de pago por tarifas basado en Redsys, de forma solamente visual y no funcional.

# 6 DISEÑO

## Requisitos técnicos

1. Gestión de usuarios (registro/login)

Requisito técnico:

Uso del sistema de autenticación de Django (django.contrib.auth) para gestión de usuarios y sesiones.

Cifrado de contraseñas mediante el sistema de hashing seguro incluido en Django.

Middleware de autenticación y control de accesos a zonas privadas.

2. Página informativa del gimnasio

Requisito técnico:

Sistema de plantillas de Django (Django Templates).

Estructura modular con archivos HTML5, CSS3 y JavaScript para el frontend.

Almacenamiento de la información en modelos de base de datos gestionados por Django ORM.

3. Cuerpo humano interactivo

Requisito técnico:

Implementación inicial con JavaScript y archivo SVG seccionado por músculos.

Uso de videos de YouTube a modo de tutorial.

4. Seguimiento del progreso

Requisito técnico:

Modelos personalizados en Django para almacenar métricas del usuario (peso, repeticiones, fechas).

Uso de bibliotecas de gráficos como Chart.js para visualizar la evolución del usuario.

5. Arquitectura y despliegue

Requisito técnico:

Uso de contenedores Docker para el entorno de desarrollo y despliegue.

Separación de servicios (Django, Postgres) en un docker-compose.yml.

Configuración de entorno .env para gestión de credenciales y parámetros sensibles.

## Arquitectura web

La arquitectura web implementada en este proyecto se basa en el modelo MVC (Modelo-Vista-Controlador) proporcionado por el framework Django.

Además, se emplea tecnología basada en contenedores mediante Docker, lo que permite un entorno de desarrollo y despliegue más limpio, reproducible y aislado.

Elementos principales de la arquitectura

Frontend (Cliente)

HTML5, CSS3, JavaScript.

Django Templates para renderizado de vistas dinámicas.

Chart.js para la visualización de progreso del usuario.

Backend (Servidor)

Django (Python) actuando como servidor principal y controlador de lógica de negocio.

Sistema de rutas y vistas gestionadas con el patrón MVC.

Seguridad y control de acceso a través del sistema de autenticación integrado.

Base de datos

Postgres, gestionada por Django ORM para la persistencia de datos de usuarios, rutinas, productos, progreso, etc.

Servicios y tareas asincrónicas

Contenedores Docker

Contenedor para Django (app).

Contenedor para MySQL (base de datos).

Orquestación mediante docker-compose.

## Diseño back-end

El desarrollo del back-end se ha realizado utilizando el framework Django, basado en Python. Django ha sido elegido por su robustez, seguridad integrada y su enfoque en el patrón arquitectónico MVC (Modelo-Vista-Controlador), que permite mantener una estructura limpia y escalable.

Características destacadas del back-end:

Framework: Django 4.2.19

Lenguaje: Python 3.11

Base de datos: Postgres, gestionada mediante el ORM (Object-Relational Mapper) de Django, que facilita la interacción con la base de datos sin necesidad de escribir consultas SQL manuales.

Autenticación: Se ha utilizado el sistema de autenticación integrado de Django, permitiendo el registro, login y gestión de sesiones de usuarios.

Gestión de usuarios: Uso de UserModel extendido para almacenar información adicional del usuario (como progreso o rutinas).

Panel de administración: Se ha creado un panel de administración personalizado.

API REST (en fase futura): Se contempla la implementación de una API REST utilizando Django REST Framework, para posibilitar la conexión con futuras apps móviles o integraciones externas.

Seguridad: Protección contra ataques CSRF y XSS proporcionada por defecto. También se han implementado buenas prácticas como validación de formularios y control de permisos por rol.

Esta estructura permite una fácil ampliación, manteniendo la coherencia del código, reutilización de componentes y un mantenimiento más eficiente del sistema.

### Modelo de datos

La aplicación utiliza el ORM de Django sobre una base de datos relacional para estructurar y gestionar los datos. A continuación, se detallan los modelos principales definidos en el sistema, que permiten gestionar usuarios, entrenamientos y centros deportivos.

* **Modelos principales**
* **MyUser**
  + email (EmailField, único) — identificador principal del usuario.
  + username (CharField)
  + birthdate (DateField)
  + create\_date / update\_date (DateTimeField)
  + is\_staff, is\_superuser (BooleanField) — permisos administrativos.
* **Centro**
  + nombre (CharField)
  + direccion (TextField)
  + telefono (CharField)
  + horario (CharField)
  + imagen (ImageField) — imagen del centro.
* **Musculo**
  + nombre (CharField) — representa un músculo trabajado.
* **Ejercicio**
  + nombre (CharField)
  + musculo (ForeignKey a Musculo) — cada ejercicio se asocia a un grupo muscular.
* **Entrenamiento**
  + usuario (ForeignKey a MyUser) — quién realiza el entrenamiento.
  + ejercicio (ForeignKey a Ejercicio) — qué ejercicio se entrena.
  + fecha (DateField, auto) — fecha de realización.
  + Relación uno a muchos con Serie.
* **Serie**
  + entrenamiento (ForeignKey a Entrenamiento)
  + repeticiones (PositiveIntegerField)
  + peso\_levantado (FloatField)
* Relaciones clave
* Cada usuario puede registrar múltiples entrenamientos.
* Un entrenamiento contiene varias series.
* Cada ejercicio se asocia a un único músculo.
* Se puede registrar información de diferentes centros deportivos donde se pueden desarrollar las rutinas.

### Servicios REST

En este proyecto no se han implementado servicios REST. Toda la lógica de interacción entre el cliente y el servidor se gestiona mediante vistas tradicionales de Django y el renderizado de plantillas HTML en el servidor. Esto ha permitido mantener una estructura sencilla y coherente con los objetivos actuales del proyecto.

### Paquetes adicionales

Durante el desarrollo del proyecto se han utilizado varios paquetes adicionales de Python y Django, de los cuales destacan los siguientes por su impacto en la funcionalidad y experiencia del usuario:

* **django-allauth**: Facilita la autenticación de usuarios, incluyendo el registro, login y gestión de sesiones. Este paquete ha sido clave para implementar un sistema de usuarios robusto.
* **django-crispy-forms + crispy-bootstrap5**: Permiten una mejor presentación y personalización de los formularios utilizando Bootstrap 5, mejorando la experiencia visual y usabilidad del sitio sin necesidad de escribir mucho HTML adicional.
* **Pillow**: Biblioteca esencial para el tratamiento de imágenes, necesaria para la gestión de imágenes subidas por los usuarios (por ejemplo, imágenes de centros deportivos).
* **psycopg2-binary**: Driver utilizado para la conexión entre Django y la base de datos PostgreSQL, esencial para el funcionamiento de la aplicación.

## Diseño front-end

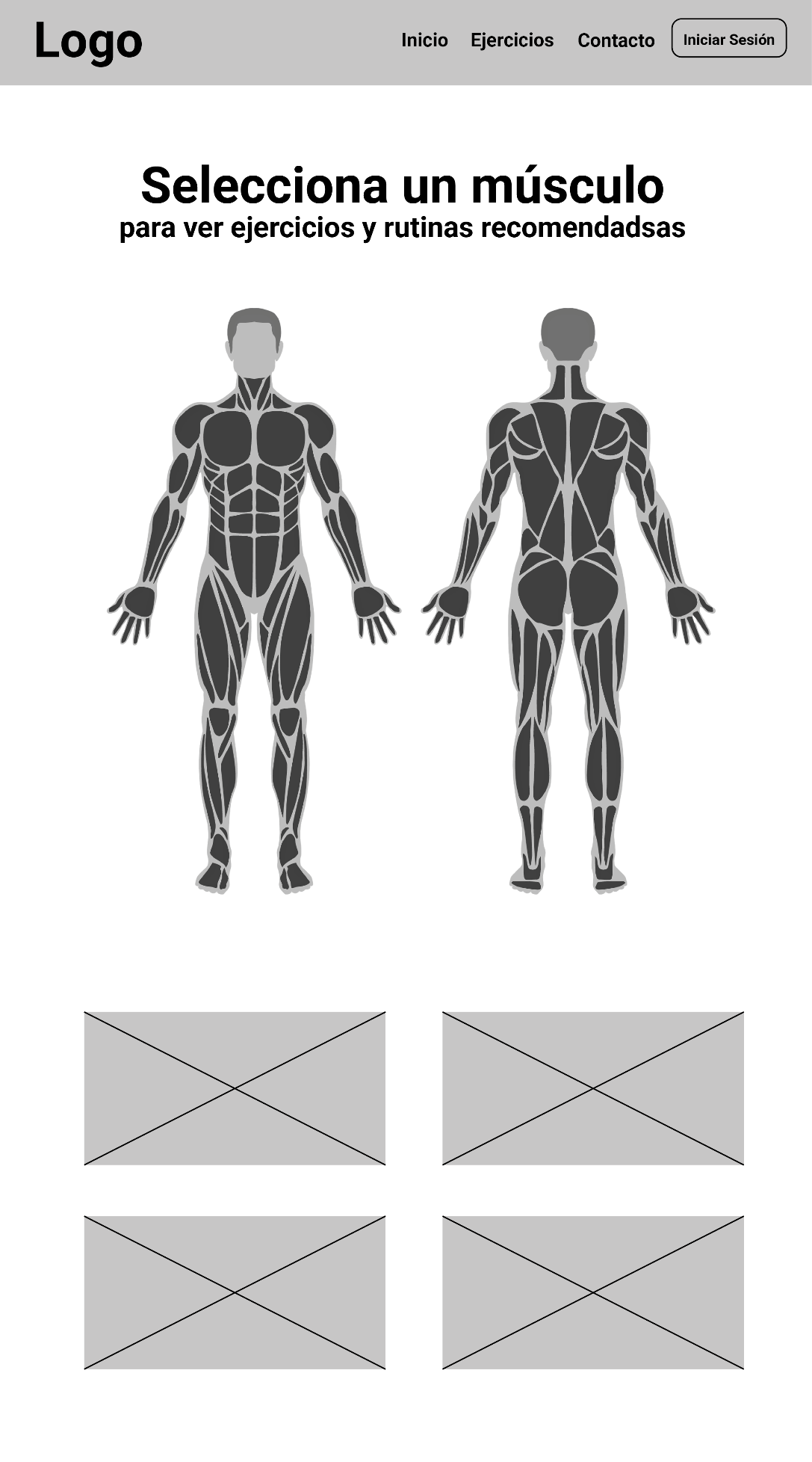
El diseño del frontend se ha realizado mediante plantillas HTML tradicionales de Django con el uso de CSS personalizado, Bootstrap y js. La presentación se ha centrado en lograr una interfaz clara y funcional, sin el uso de frameworks modernos de JavaScript como React o Vue.

Se han usado formularios estándar de Django para la entrada de datos, y se ha procurado una organización lógica de las páginas para facilitar la navegación por parte del usuario.

### Mock-ups

Diagrama

El contenido generado por IA puede ser incorrecto.



Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

### Guía de estilos

Gráfico, Gráfico en cascada

El contenido generado por IA puede ser incorrecto.

### Paquetes adicionales

6.4.4. Capturas de la aplicación

Capturas de las principales pantallas de la aplicación final en versión móvil, tablet y escritorio.

# IMPLEMENTACIÓN

## Servidor

La parte del servidor ha sido implementada utilizando **Django 4.2.19**, un framework robusto en Python que facilita el desarrollo de aplicaciones web de forma estructurada. Se ha definido una arquitectura basada en el modelo **MVC (Modelo-Vista-Controlador)**, adaptado a la estructura propia de Django: **Model-Template-View**.

Se han creado varios modelos clave como Centro, Musculo, Ejercicio, Entrenamiento, Serie y un modelo personalizado de usuario MyUser. Estos modelos permiten gestionar los datos principales de la aplicación, como los ejercicios realizados por cada usuario, sus progresos y la organización de los gimnasios o centros.

El sistema gestiona autenticación de usuarios con django-allauth, y permite registrar, iniciar sesión y utilizar funcionalidades de la plataforma de manera individual. Además, se han habilitado formularios personalizados para el registro de entrenamientos y series.

La base de datos utilizada es **PostgreSQL**, integrada mediante el conector psycopg2-binary, y todo el entorno se ejecuta bajo un servidor Gunicorn, preparado para despliegue en servicios como Render.

## Cliente

La parte cliente se ha implementado utilizando las **plantillas HTML de Django** junto con CSS personalizado y Bootstrap 5, a través del paquete crispy-bootstrap5, lo que ha permitido generar formularios y componentes visuales de forma sencilla y adaptativa.

No se ha utilizado un framework frontend (como React o Vue), ya que se ha optado por una solución ligera, con lógica principalmente en el servidor y una interfaz clara y accesible. El sistema permite navegar entre las distintas secciones (perfil, entrenamientos, historial, centros, etc.) y realizar acciones básicas como añadir y visualizar entrenamientos o gestionar el perfil de usuario.

# DESPLIEGUE

## Modelo de despliegue utilizado

Para el despliegue del proyecto se ha utilizado un modelo de despliegue en la nube, concretamente mediante la plataforma Render.com, que permite alojar aplicaciones web con backend en Python/Django de forma sencilla y eficiente.

El proyecto se ha configurado para funcionar en un entorno Linux con servidor de aplicaciones Gunicorn, y sirve los archivos estáticos con WhiteNoise, optimizando así el rendimiento de la aplicación sin necesidad de un servidor externo como Nginx.

La base de datos utilizada es PostgreSQL, proporcionada también por Render como servicio gestionado, lo que garantiza persistencia y escalabilidad en la gestión de datos.

Durante el despliegue se han tenido en cuenta aspectos como:

* Uso de entorno virtual y archivo requirements.txt para la instalación de dependencias.
* Configuración de variables de entorno (por ejemplo, DEBUG=False, SECRET\_KEY, conexión a base de datos, etc.).
* Uso de dj\_database\_url para parsear las credenciales de la base de datos desde el entorno.

Este modelo de despliegue permite escalar la aplicación fácilmente en caso de aumento de tráfico, mantener actualizaciones de forma sencilla y asegurar una alta disponibilidad sin necesidad de gestionar infraestructura propia.

## Datos iniciales y configuración

Para facilitar el uso y pruebas iniciales, se han cargado algunos datos básicos en el sistema:

* Centros de entrenamiento ficticios.
* Listado básico de músculos y ejercicios asociados.
* Cuentas de prueba para usuarios.

Esta carga se ha realizado utilizando scripts personalizados cuando ha sido necesario.

En el entorno de producción, se han creado también configuraciones relacionadas con el almacenamiento de imágenes, así como el uso de rutas seguras para archivos estáticos y media.

## Pasos para el despliegue

Para desplegar la aplicación de forma funcional en un entorno de producción, se han seguido los siguientes pasos:

1. **Preparación del repositorio**  
   Se ha organizado el proyecto en un repositorio Git, asegurando que el código esté limpio, actualizado y con un .gitignore correctamente configurado.
2. **Creación de cuenta y proyecto en Render**  
   Se ha utilizado [Render.com](https://render.com) como plataforma de despliegue. Se ha creado un nuevo servicio web y se ha vinculado con el repositorio del proyecto en GitHub.
3. **Configuración de entorno en Render**
   * Se han añadido las variables de entorno necesarias (SECRET\_KEY, DEBUG=False, DATABASE\_URL, etc.).
   * Se ha configurado el archivo Dockerfile para definir el build command y el start command:
     + **Build command**: pip install -r requirements.txt
     + **Start command**: gunicorn nombre\_proyecto.wsgi
4. **Configuración de la base de datos**
   * Se ha añadido una base de datos PostgreSQL desde Render.
   * Se ha copiado la DATABASE\_URL proporcionada por Render en las variables de entorno.
   * Se han aplicado las migraciones con: python manage.py migrate
5. **Gestión de archivos estáticos y media**
   * Se ha usado WhiteNoise para servir archivos estáticos en producción.
   * Se han ejecutado los comandos: python manage.py collectstatic
6. **Comprobaciones finales y despliegue**  
   Una vez cargado todo, se ha accedido al dominio proporcionado por Render para comprobar el correcto funcionamiento de la app.

## Proveedores y servicios utilizados

Durante el desarrollo y despliegue del proyecto, se han utilizado los siguientes proveedores y servicios:

* **Render.com**  
  Plataforma utilizada para el despliegue de la aplicación web y la base de datos PostgreSQL. Render permite gestionar el entorno de producción de forma sencilla, integrando directamente el repositorio GitHub y configurando variables de entorno, comandos de build y ejecución.
* **GitHub**  
  Servicio de control de versiones donde se aloja el repositorio del proyecto. Permite la integración continua con Render para desplegar automáticamente los cambios realizados en la rama principal.
* **PostgreSQL (Render)**  
  Motor de base de datos relacional usado en producción. Render proporciona una instancia gestionada de PostgreSQL con conexión segura mediante DATABASE\_URL.
* **Gunicorn**  
  Servidor WSGI utilizado para ejecutar la aplicación Django en entorno de producción.
* **WhiteNoise**  
  Middleware utilizado para servir archivos estáticos directamente desde el servidor, sin necesidad de un servidor adicional como NGINX.
* **Pillow**  
  Biblioteca utilizada para el manejo de imágenes (por ejemplo, al subir imágenes de centros deportivos).

# HERRAMIENTAS UTILIZADAS

Durante el desarrollo del proyecto se han utilizado diversas herramientas, tanto para la programación como para el control de versiones, diseño y despliegue:

* **Visual Studio Code**  
  Editor de código principal utilizado para el desarrollo del proyecto. Se ha utilizado junto a extensiones como Python, Django y Prettier para facilitar la escritura y organización del código.
* **Git y GitHub**  
  Git se ha utilizado para el control de versiones y GitHub como repositorio remoto. Permite llevar un seguimiento del desarrollo y facilitar el trabajo colaborativo.
* **Render**  
  Plataforma utilizada para el despliegue de la aplicación web y la base de datos PostgreSQL. Permite automatizar el despliegue con cada nuevo cambio en GitHub.
* **PostgreSQL**  
  Sistema de gestión de bases de datos relacional utilizado tanto en desarrollo como en producción.
* **Django**  
  Framework principal de desarrollo del backend. Se ha elegido por su rapidez de desarrollo, estructura basada en el patrón MVC y su comunidad activa.
* **WhiteNoise**  
  Herramienta integrada con Django para la gestión de archivos estáticos en producción sin necesidad de servidores externos como Nginx.
* **Gunicorn**  
  Servidor WSGI utilizado para ejecutar la aplicación en entorno de producción.
* **Pillow**  
  Librería de Python usada para el tratamiento y almacenamiento de imágenes.
* **Crispy Forms + Crispy Bootstrap5**  
  Paquetes adicionales para mejorar el diseño de los formularios y adaptar su estilo a Bootstrap 5.
* **Adobe Illustrator**  
  Herramienta utilizada para el diseño de la identidad visual del proyecto, incluyendo logotipos, isologos e iconografía. Facilita la creación de gráficos vectoriales con acabado profesional.