UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

TerapiTrack: plataforma web de rehabilitación con registro de ejercicios en vídeo

Presentado por Alberto Lanchares Diez
en Universidad de Burgos — 13 de febrero de 2026
Tutores: José Luis Garrido Labrador y José Miguel Ramírez Sanz

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. José Luis Garrido Labrador y D. José Miguel Ramírez Sanz, profesores del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Exponen:

Que el alumno D. Alberto Lanchares Diez, con DNI 71362969H, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado TerapiTrack: plataforma web de rehabilitación con registro de ejercicios en vídeo.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 13 de febrero de 2026

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. José Luis Garrido Labrador	D. José Miguel Ramírez Sanz

Resumen

Este Trabajo de Fin de Grado presenta el diseño e implementación de TerapiTrack, una plataforma web para la gestión de sesiones de telerehabilitación orientadas a pacientes con enfermedad de Parkinson que residen en zonas rurales o alejadas de los centros médicos de referencia.

El sistema permite a profesionales sanitarios crear ejercicios terapéuticos en vídeo, programar sesiones personalizadas, guiarlas seleccionando los ejercicios a realizar y evaluar de forma remota el desempeño de los pacientes; mientras que estos pueden realizar las sesiones en su domicilio, grabando sus ejercicios desde el navegador, con la cámara del dispositivo, repetir los ejercicios asignados cuando lo deseen para practicar y consultar su progreso a lo largo del tratamiento. Además, el administrador puede gestionar usuarios y vinculaciones entre profesionales y pacientes.

TerapiTrack se ha desarrollado con Flask y Jinja, utiliza SQLAlchemy sobre SQLite y PostgreSQL para la gestión de datos e integra Cloudinary para el almacenamiento escalable de vídeos. Además, sigue criterios de accesibilidad web para facilitar su uso por parte de personas con limitaciones motoras o con poca experiencia tecnológica, incluyendo la utilización de un mando SNES para el paciente.

El proyecto incluye una arquitectura modular basada en blueprints, un modelo de datos normalizado, pruebas unitarias con cobertura del 99 % sobre los módulos críticos y despliegue en Heroku.

Descriptores

Telerehabilitación, enfermedad de Parkinson, aplicación web sanitaria, registro de ejercicios en vídeo, Flask, SQLAlchemy, Cloudinary.

Abstract

This Final Degree Project presents the design and implementation of TerapiTrack, a web platform for managing telerehabilitation sessions aimed at patients with Parkinson's disease who live in rural areas or far from specialized medical centers.

The system allows healthcare professionals to create therapeutic video exercises, schedule personalized sessions, guide them by selecting the exercises to be performed, and remotely evaluate patient performance; while patients can carry out their sessions at home, recording their exercises directly in the browser with the device's camera, repeating the assigned exercises whenever they want for additional practice, and viewing their progress throughout the treatment. In addition, an administrator can manage users and the links between professionals and patients.

TerapiTrack has been developed using Flask and Jinja, employs SQLAlchemy with SQLite and PostgreSQL for data management and integrates Cloudinary for scalable video storage. Furthermore, it follows web accessibility guidelines to ensure usability for individuals with motor limitations or limited technological experience, including support for a SNES gamepad as an alternative input device for patients.

The project features a modular architecture based on blueprints, a normalized data model, unit tests achieving 99 % coverage on critical modules, and deployment on Heroku.

Keywords

Telerehabilitation, Parkinson's disease, healthcare web application, video exercise recording, Flask, SQLAlchemy, Cloudinary.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

A medida que la población envejece y los servicios sanitarios se concentran principalmente en las ciudades, muchas personas que residen en zonas rurales encuentran dificultades para acceder a terapias de rehabilitación. Esta situación afecta de manera especial a quienes padecen determinados problemas de salud crónicos, entre ellos la enfermedad de Parkinson, un trastorno neurodegenerativo progresivo que provoca rigidez muscular, temblores y dificultades en el movimiento, y para el que actualmente no existe cura. No obstante, la realización continuada de ejercicios de rehabilitación física y cognitiva permite ralentizar en cierto grado la evolución de la enfermedad, mejorar la movilidad y mantener durante más tiempo la independencia y la calidad de vida de las personas afectadas [12, 15, 34].

Las limitaciones de movilidad y la lejanía de los centros sanitarios de referencia complican aún más el acceso a estas terapias, repercutiendo en el bienestar de los pacientes y en el esfuerzo de sus familias [12]. Ante esta problemática, surge TerapiTrack, una plataforma web diseñada para facilitar el acompañamiento y el seguimiento de terapias, principalmente para personas diagnosticadas con la enfermedad de Parkinson, aunque su enfoque flexible y modular permite también su uso en otros casos que requieran terapias a distancia, como personas con otras enfermedades crónicas. TerapiTrack busca eliminar barreras geográficas y de accesibilidad, adaptándose a las necesidades y capacidades de distintos tipos de usuarios [30].

1.1. Contexto del problema

En España, las dificultades de acceso regular a terapias de rehabilitación se acentúan especialmente en la conocida como «España vaciada», donde una parte muy importante de la población está formada por personas de edad

avanzada que aún viven en zonas rurales con medios sanitarios limitados. Este es un colectivo más propenso a padecer trastornos neurodegenerativos como la enfermedad de Parkinson y que, además, suele depender de familiares o cuidadores para desplazarse a los hospitales y centros sanitarios, donde realizan sus rehabilitaciones. Para muchos de estos pacientes, viajar hasta la ciudad únicamente para realizar sesiones de rehabilitación supone invertir varias horas en transporte, afrontar costes económicos adicionales y organizar la agenda de toda la familia, lo que dificulta mantener la regularidad necesaria en las terapias [12, 15, 34].

El proyecto en el que se enmarca este Trabajo de Fin de Grado forma parte de una colaboración más amplia con el Servicio de Neurología del Hospital Universitario de Burgos, liderada por la doctora Esther Cubo. El objetivo general de esta colaboración es mejorar la calidad de vida de las personas con enfermedad de Parkinson mediante el uso de herramientas de inteligencia artificial y de telemedicina, trabajando en tres líneas principales: la detección temprana de la enfermedad a partir de pruebas que miden la bradicinesia, la detección de caídas y la aplicación de programas de telerehabilitación con retroalimentación automática [2, 5, 13, 29, 36]. Dentro de este conjunto de proyectos, este TFG se sitúa en esta última línea y se centra en el desarrollo de una aplicación web que permita a los pacientes realizar en su domicilio ejercicios de rehabilitación, guiados por vídeos preparados por neurólogos y terapeutas del hospital.

1.2. Propuesta de solución

El objetivo principal de TerapiTrack es acercar la rehabilitación y el control terapéutico al entorno cotidiano del paciente, ofreciendo una aplicación web que organice de forma sencilla las sesiones de ejercicios y permita registrar su realización en el domicilio. El desarrollo de la aplicación se ha apoyado en una fase de diseño especialmente extensa, en la que durante las primeras semanas del proyecto se han definido con detalle los requisitos y las entidades clave del dominio (usuarios, pacientes, profesionales, ejercicios, sesiones, evaluaciones y vídeos de respuesta), así como los actores implicados y los casos de uso más relevantes del proceso de telerehabilitación. A partir de este análisis se han construido el modelo entidad-relación y su correspondiente esquema relacional normalizado, que sustentan la base de datos de TerapiTrack, garantizando la coherencia de la información clínica manejada. Estos modelos, junto con el diagrama de casos de uso y la arquitectura modular del sistema, han guiado el diseño de los distintos paneles de usuario

y de los flujos de interacción que se describen en los capítulos posteriores y en la documentación técnica de los anexos.

Sobre esta base de diseño, se ha desarrollado la base funcional de la aplicación, estructurada en tres paneles diferenciados: administrador, profesional sanitario y paciente. El administrador puede gestionar usuarios, roles y vinculaciones entre pacientes y profesionales, así como configurar parámetros generales del sistema (como por ejemplo, las políticas de retención de vídeos o ciertos límites de almacenamiento). El profesional dispone de herramientas para mantener su lista de pacientes, crear y catalogar ejercicios a partir de vídeos, programar sesiones terapéuticas personalizadas y revisar posteriormente las grabaciones y evaluarlas. Por su parte, el paciente accede a un panel simplificado desde el que puede consultar sus próximas sesiones, realizarlas guiados por el profesional, repetir los ejercicios asignados en las sesiones y visualizar su evolución a partir de las evaluaciones registradas.

De este modo, TerapiTrack cubre de extremo a extremo el flujo terapéutico, desde la creación de ejercicios y la programación de las sesiones hasta la evaluación remota del desempeño del paciente.

1.3. Materiales entregados

Para facilitar la validación, el uso y la posible evolución del sistema, se entrega junto con la memoria un conjunto de materiales adicionales:

- El código fuente completo del proyecto, junto con su historial de cambios en el repositorio de control de versiones (incluida la rama `Pruebas` utilizada durante el desarrollo), disponible en: https://github.com/lanchares/TerapiTrack.

- La definición de la base de datos y un juego de datos de prueba que permite ejecutar los principales casos de uso sin necesidad de configuración adicional. En el USB entregado se incluye una base de datos de «juguete», precargada con usuarios, pacientes, sesiones y vídeos de ejercicios, para que el tribunal pueda trabajar con la herramienta sin tener que preparar los datos.

- La documentación técnica incluida en los anexos, donde se detallan la arquitectura del sistema, los modelos de datos y los principales diagramas de diseño.

- Un manual de instalación y de usuario, con instrucciones paso a paso para desplegar la aplicación y una guía básica para cada tipo de usuario.
- Un resumen de las pruebas realizadas sobre el sistema, tanto automatizadas con *pytest* como manuales sobre el modelo de datos y las operaciones habituales, junto con los resultados obtenidos.
- Varios vídeos breves que muestran el funcionamiento de las distintas partes de TerapiTrack (panel de administrador, profesional y paciente), de manera que el tribunal pueda hacerse una idea clara de la aplicación sin tener que levantar todo el entorno.
- Un breve documento con las credenciales de acceso de prueba para los distintos perfiles, incluido en el dispositivo USB bajo el nombre `leeme.txt`.

El objetivo de este material adicional es que cualquier persona interesada pueda entender con rapidez el alcance del proyecto, comprobar su funcionamiento y disponer de una base sobre la que seguir trabajando.

Además, la versión desplegada de TerapiTrack está disponible en: `https://terapitrack-tfg-dccabad31cfb.herokuapp.com`, accesible mediante las credenciales de prueba incluidas en el fichero `leeme.txt` del USB entregado.

2. Objetivos del proyecto

Este capítulo recoge los objetivos que se persiguen con la realización del proyecto. En primer lugar se presentan los objetivos funcionales, que describen qué se espera que ofrezca la aplicación a sus usuarios. A continuación, se detallan los objetivos técnicos, relacionados con la forma de construir y desplegar el sistema, y por último, se recogen los objetivos personales planteados para el desarrollo del trabajo.

2.1. Objetivos funcionales

- Gestionar diferentes perfiles de usuario (administrador, paciente y profesionales sanitarios) con permisos diferenciados según su rol.
- Permitir la vinculación de pacientes con los profesionales responsables de su seguimiento terapéutico.
- Ofrecer una biblioteca de ejercicios y recursos terapéuticos en formato audiovisual, organizada y filtrable según las necesidades de cada paciente, con una serie de vídeos de ejemplo incluidos en el proyecto y la posibilidad de que cada profesional incorpore sus propios vídeos demostrativos adaptados a cada caso.
- Facilitar la creación y programación de sesiones de rehabilitación personalizadas, formadas por combinaciones de ejercicios.
- Registrar la realización de las sesiones y asociar las grabaciones de los ejercicios al historial de cada paciente, para su posterior revisión y evaluación.

- Proporcionar a los pacientes un entorno sencillo donde puedan consultar y ejecutar sus sesiones programadas, realizar los ejercicios guiados por vídeo y revisar la evolución de sus evaluaciones.
- Asegurar la trazabilidad básica de la evolución terapéutica, ofreciendo a los profesionales una visión estructurada de las sesiones realizadas y de las puntuaciones registradas.

En conjunto, estos objetivos funcionales recogen las necesidades detectadas en el contexto del proyecto y concretan las capacidades que debe ofrecer TerapiTrack a sus distintos tipos de usuarios.

2.2. Objetivos técnicos

- Construir una aplicación web modular que facilite el mantenimiento y la incorporación de nuevas funcionalidades en el futuro.
- Diseñar e implementar una base de datos relacional que modele usuarios, pacientes, profesionales, ejercicios, sesiones y evaluaciones, garantizando la integridad de los datos clínicos manejados.
- Desarrollar una interfaz web accesible y clara, diseñada para pacientes con posibles dificultades motoras y/o cognitivas.
- Incorporar mecanismos de autenticación, control de acceso por roles y cifrado de contraseñas, que contribuyan a la protección de la información sensible.
- Preparar la aplicación para su despliegue en un entorno en la nube, permitiendo el acceso remoto al sistema desde distintos dispositivos.
- Definir y ejecutar una batería de pruebas sobre los componentes principales del sistema (modelos, controladores y flujos de usuario) que ayude a detectar fallos, tanto de diseño como de programación, y a comprobar su correcto funcionamiento.

Por su parte, los objetivos técnicos se apoyan en principios clásicos de ingeniería del software, como el diseño modular y la separación de responsabilidades, el uso de bases de datos relacionales bien estructuradas, y en la aplicación de buenas prácticas de accesibilidad y de despliegue de aplicaciones web en la nube [6, 30, 23, 24, 27, 35].

2.3. Objetivos personales

- Aplicar de forma integrada los conocimientos adquiridos a lo largo del Grado en un proyecto completo, desde el análisis de requisitos hasta el despliegue de la aplicación.
- Profundizar en el desarrollo de aplicaciones web orientadas al ámbito sanitario y a la telerehabilitación.
- Mejorar la capacidad de trabajo autónomo, planificación y gestión del tiempo en un proyecto de larga duración.
- Adquirir experiencia práctica en el uso de herramientas profesionales de desarrollo, control de versiones y documentación técnica.
- Contribuir, en la medida de lo posible, a mejorar el día a día de las personas que necesitan rehabilitación, desarrollando una herramienta que facilite el seguimiento de las terapias en su propio domicilio.

Finalmente, los objetivos personales reflejan el interés por el ámbito sanitario y la telerehabilitación, así como la voluntad de consolidar habilidades profesionales y aportar una solución útil dentro de la colaboración con el Hospital Universitario de Burgos [1, 5].

3. Conceptos teóricos

Este capítulo revisa los conceptos teóricos básicos que sirven de soporte a TerapiTrack, tanto en su diseño como en su implementación. Primero se abordan nociones relacionadas con la enfermedad de Parkinson y la rehabilitación y, a continuación, se tratan aspectos de accesibilidad en aplicaciones sanitarias, los patrones de diseño, los modelos de datos y los fundamentos de validación y pruebas de software que se han tenido en cuenta durante el desarrollo.

3.1. Enfermedades degenerativas y enfermedad de Parkinson

Las enfermedades degenerativas son trastornos en los que determinadas estructuras del organismo se deterioran de forma progresiva, lo que provoca una pérdida lenta pero continua de funciones. Cuando el sistema nervioso central se ve afectado, este deterioro suele traducirse en problemas de movimiento, equilibrio, memoria o comunicación, que requieren tratamientos prolongados y un seguimiento intensivo [15, 34].

La enfermedad de Parkinson es un trastorno neurodegenerativo crónico que afecta principalmente al control del movimiento. Se caracteriza por temblor en reposo, rigidez muscular, lentitud en la iniciación de los movimientos y alteraciones del equilibrio, y actualmente no tiene cura. Aunque puede aparecer en adultos jóvenes, afecta sobre todo a personas de edad avanzada; se estima que aproximadamente una de cada cien personas mayores de 60 años y en torno a un 2 % de la población española mayor de 65 años padecen la enfermedad, lo que implica que muchos pacientes presentan limitaciones de

movilidad y dependen de terceras personas para desplazarse, especialmente cuando viven en ámbitos rurales [16, 32, 34].

La combinación de tratamiento farmacológico y programas de rehabilitación específicos permite reducir la intensidad de los síntomas, mantener la autonomía durante más tiempo y mejorar la calidad de vida de los pacientes [3, 34].

3.2. Terapia ocupacional y rehabilitación

La terapia ocupacional es una disciplina sanitaria que ayuda a las personas a mantener o mejorar su autonomía en las actividades significativas de su vida diaria, adaptando tanto las tareas como el entorno cuando existe una enfermedad, lesión o discapacidad [11, 14]. En pacientes con enfermedad de Parkinson, la terapia ocupacional trabaja aspectos como vestirse, la higiene personal, la movilidad dentro del domicilio o la organización de rutinas, con el objetivo de prolongar la autonomía en las actividades básicas y sociales [11].

La rehabilitación de trastornos del movimiento suele combinar fisioterapia, terapia ocupacional, ejercicio físico estructurado y, en algunos casos, logopedia. Este tipo de intervención requiere sesiones frecuentes y un seguimiento continuado, por lo que acceder de forma regular a los servicios especializados resulta especialmente complicado para pacientes que viven lejos de los centros de referencia o que tienen dificultades para desplazarse [12].

3.3. Telemedicina y telerehabilitación

La telemedicina hace referencia al uso de las tecnologías de la información y la comunicación para prestar servicios sanitarios a distancia, permitiendo que paciente y profesional interactúen sin encontrarse en el mismo lugar físico. Dentro de este ámbito, la telerehabilitación se centra en el diseño, la supervisión y la evaluación de programas de rehabilitación mediante herramientas remotas, como la videoconferencia o plataformas web específicas [26].

En el caso de la enfermedad de Parkinson, la telerehabilitación ofrece la posibilidad de mantener la frecuencia de las terapias aunque el paciente resida en la «España vaciada» o tenga movilidad reducida. Permite reducir desplazamientos, facilita la adherencia al tratamiento y hace posible que el profesional revise de forma diferida los ejercicios realizados en casa [12, 31].

3.4. Accesibilidad y usabilidad en aplicaciones sanitarias

En el contexto de la telerehabilitación para personas con enfermedad de Parkinson, la accesibilidad de una aplicación no es un añadido opcional, sino una condición para que la herramienta pueda utilizarse en la práctica diaria. Si los menús son complejos, los textos difíciles de leer o se requieren demasiados pasos para completar una tarea, es probable que muchos pacientes abandonen el uso de la plataforma aunque el contenido terapéutico sea adecuado [30].

Por ello, este tipo de aplicaciones se diseña pensando especialmente en usuarios con posibles limitaciones motoras y menor experiencia con dispositivos digitales, como ocurre con muchos pacientes de Parkinson. En TerapiTrack las decisiones de accesibilidad se han centrado principalmente en la interfaz dirigida al paciente, mientras que las vistas destinadas al profesional sanitario mantienen una estructura más cercana a la de una aplicación web convencional, asumiendo un mayor grado de familiaridad con estas herramientas.

Las pautas Web Content Accessibility Guidelines (WCAG) ofrecen recomendaciones sobre contraste, tipografía, estructura y compatibilidad con tecnologías de apoyo, que sirven como referencia para adaptar la interfaz a personas mayores o con dificultades motoras [35]. En la parte de paciente de TerapiTrack estas ideas se han traducido en un menú sencillo, textos breves y claros, pocos elementos interactivos por pantalla y compatibilidad con dispositivos alternativos, como el mando SNES integrado únicamente en las vistas de paciente.

3.5. Patrones de diseño y arquitectura Modelo-Vista-Controlador

En el desarrollo de aplicaciones de software es habitual apoyarse en patrones de diseño que separan responsabilidades y facilitan la evolución del sistema; en el caso de TerapiTrack, esto se traduce en el uso de una arquitectura de estilo Modelo-Vista-Controlador (MVC) [9]. Este patrón organiza la aplicación en tres componentes principales:

- **Modelo**: representa los datos y las reglas de negocio asociadas a cada entidad del dominio, incluyendo la definición de estructuras de almacenamiento y restricciones de integridad.
- **Vista**: define cómo se presenta la información al usuario, normalmente mediante páginas web o plantillas que muestran formularios, listados y mensajes.
- **Controlador**: actúa como intermediario entre el modelo y la vista, recibiendo las peticiones del usuario, invocando la lógica necesaria y seleccionando la respuesta adecuada.

Este tipo de arquitectura favorece que el código relacionado con la persistencia de datos, la lógica de aplicación y la interfaz de usuario puedan evolucionar de manera relativamente independiente, algo especialmente útil en proyectos que pueden ampliarse en el futuro [9].

3.6. Bases de datos relacionales

Las bases de datos relacionales permiten organizar la información en tablas relacionadas entre sí mediante claves primarias y foráneas, garantizando la integridad de los datos mediante restricciones y reglas de consistencia [6]. Este modelo es el que se ha aplicado en TerapiTrack para organizar usuarios, pacientes, profesionales sanitarios, ejercicios, sesiones, evaluaciones y vídeos de respuesta.

En este contexto aparecen relaciones de distinto tipo: **uno a muchos**, como la que se da entre un paciente y las sesiones que tiene programadas, o entre un profesional sanitario y las sesiones que coordina; **uno a uno**, como la que existe entre la tabla de usuarios y las tablas de paciente o profesional, donde cada registro de usuario se especializa en un único tipo de perfil; y **muchos a muchos**, como ocurre al vincular profesionales sanitarios con varios pacientes o al asociar ejercicios concretos a distintas sesiones, que se representan mediante tablas intermedias específicas [6].

La **normalización** es el proceso mediante el cual se descompone la información en tablas más pequeñas para reducir la redundancia y evitar anomalías en las operaciones de inserción, actualización y borrado. En términos prácticos, suele implicar diseñar el esquema de forma que cumpla las principales formas normales: primera forma normal (1FN), que exige atributos con valores atómicos; segunda forma normal (2FN), que elimina dependencias parciales respecto a claves compuestas; y tercera forma normal

(3FN), que evita dependencias transitivas entre atributos que no forman parte de la clave [6].

En el caso de TerapiTrack, estas formas normales se han tenido en cuenta al distribuir los atributos en tablas especializadas y al introducir tablas intermedias como *Paciente_Profesional* o *Ejercicio_Sesion*, lo que facilita mantener la coherencia y la trazabilidad del modelo de datos [6].

3.7. Validación y pruebas de software

La validación de un sistema de software busca comprobar que la solución desarrollada cumple los requisitos definidos y que se comporta de forma correcta en los escenarios de uso previstos. Para ello se utilizan diferentes tipos de pruebas, que pueden combinarse según las necesidades del proyecto [23].

Entre las más habituales se encuentran las **pruebas unitarias**, centradas en módulos o funciones concretas; las **pruebas de integración**, que comprueban el funcionamiento conjunto de varios componentes; y las **pruebas funcionales** o de sistema, orientadas a verificar flujos de usuario o casos de uso completos. Automatizar parte de estas pruebas permite detectar errores de diseño o programación de manera temprana y reduce el riesgo de fallos en fases avanzadas del desarrollo [23].

Este conjunto de conceptos proporciona el contexto teórico y tecnológico que sustenta la memoria y ayuda a entender las decisiones que se detallan en los capítulos posteriores.

4. Técnicas y herramientas

Este capítulo presenta las metodologías, tecnologías y utilidades empleadas en el desarrollo. El propósito es ofrecer una visión ordenada del entorno de trabajo y de la base tecnológica sobre la que se construye la aplicación, de manera que sirva de referencia para entender las decisiones de diseño.

4.1. Metodología de desarrollo

Metodologías ágiles y Jira

Las metodologías ágiles plantean una forma de organizar los proyectos de software en iteraciones cortas, con entregas frecuentes y una revisión continua de las tareas pendientes [23]. En lugar de planificar todo el desarrollo al detalle desde el principio, se trabaja con ciclos de duración limitada en los que se van cerrando bloques pequeños de funcionalidad.

En TerapiTrack se ha utilizado **Jira** como herramienta de apoyo a esta forma de trabajo [1]. Jira permite crear tareas, agruparlas en sprints, registrar incidencias y visualizar el estado del proyecto en tableros, lo que facilita ver de un vistazo qué está hecho, qué está pendiente y qué se ha replanificado durante el desarrollo.

4.2. Tecnologías de backend

Python y Flask

El *backend* de TerapiTrack está desarrollado en **Python**, un lenguaje de programación de propósito general muy extendido en el ámbito del desarrollo

web, la ciencia de datos y la automatización [23]. Entre sus ventajas destacan una sintaxis relativamente sencilla y una comunidad amplia que mantiene librerías para la mayoría de necesidades habituales en un proyecto.

Sobre Python se ha utilizado el *framework* web **Flask**, que proporciona las piezas básicas para construir una aplicación web: definición de rutas, manejo de peticiones y respuestas HTTP, integración con sistemas de plantillas y soporte para extensiones que cubren necesidades como la autenticación o el acceso a bases de datos [18, 19].

En el proyecto la aplicación principal se define en `app.py` y se organiza en varios módulos de controladores que agrupan las rutas según el rol del usuario (`admin_controlador.py`, `profesional_controlador.py`, `auth_controlador.py`, etc.), siguiendo el patrón de *blueprints* descrito en la documentación oficial y en guías prácticas específicas [20, 25]. Además, se han definido decoradores propios en `decoradores.py` para comprobar el rol y otras condiciones de acceso antes de ejecutar determinadas vistas.

SQLAlchemy

Para la interacción con la base de datos se emplea **SQLAlchemy**, una biblioteca de mapeo objeto–relacional (ORM, *Object–Relational Mapping*) ampliamente utilizada en el ecosistema Python [33]. Un ORM permite trabajar con las tablas de la base de datos a través de clases y objetos, de forma que las consultas y actualizaciones se expresan en código Python y la biblioteca se encarga de traducirlas a SQL [21].

En TerapiTrack los modelos se encuentran separados en ficheros como `usuario.py`, `paciente.py`, `profesional.py`, `ejercicio.py`, `sesion.py` o `evaluacion.py`, y se complementan con un módulo de asociaciones para las relaciones de muchos a muchos. SQLAlchemy se inicializa en el módulo de extensiones (`extensiones.py`), lo que permite reutilizar la misma instancia en toda la aplicación [33].

4.3. Gestión de datos

Bases de datos en desarrollo y producción

Durante el desarrollo local se ha utilizado **SQLite** como motor de base de datos principal [6]. SQLite es un sistema de base de datos relacional embebido que almacena toda la información en un único fichero en disco y

no requiere un servidor independiente, lo que simplifica la configuración del entorno de trabajo.

Para el despliegue en la nube se ha configurado la aplicación para trabajar con **PostgreSQL**, un gestor de bases de datos relacional de código abierto ampliamente usado en entornos de producción. En Heroku se emplea el complemento de base de datos PostgreSQL, al que la aplicación se conecta mediante la cadena de conexión definida en las variables de entorno [27, 28].

La selección del motor de base de datos en cada entorno se realiza a través del fichero `config.py`, donde se define la `SQLALCHEMY_DATABASE_URI`. Además, se dispone del script `poblar_bd.py`, que permite inicializar la base de datos con datos de ejemplo (usuarios, pacientes, ejercicios, sesiones y evaluaciones) para facilitar las pruebas.

Herramienta complementaria: DB Browser for SQLite

Como apoyo al trabajo, con la base de datos local se ha utilizado **DB Browser for SQLite**, una herramienta gráfica que permite inspeccionar tablas, ejecutar consultas y modificar registros de forma visual [7]. Su uso ha sido especialmente útil en las primeras iteraciones del diseño del esquema y durante la depuración de datos de prueba.

4.4. Tecnologías de frontend

HTML5, JavaScript y Jinja

La capa de presentación de TerapiTrack se construye sobre varias tecnologías web:

- **HTML5** (HyperText Markup Language) es el estándar de marcado utilizado para definir la estructura de las páginas web, incluyendo encabezados, párrafos, formularios o tablas.
- **JavaScript** es un lenguaje de programación orientado al desarrollo en el lado del cliente, que permite añadir interactividad a las páginas, reaccionar a eventos del usuario y realizar peticiones asíncronas al servidor.
- **Jinja** es un motor de plantillas para Python que se integra con Flask y permite generar HTML de forma dinámica a partir de plantillas y datos, reutilizando estructuras comunes en distintas vistas [19].

Las plantillas principales se organizan en carpetas por rol (`admin`), (`profesional`), (`paciente`). Todas ellas heredan de una plantilla base (`base.html`) que define la estructura común de menús, cabeceras, mensajes y algun estilo común estre vistas. Entre las vistas más relevantes se encuentran los distintos paneles de control (`dashboard.html`) y las páginas de ejecución de sesiones (`ejecutar_sesion.html`), donde se combinan los vídeos de ejemplo con la cámara del paciente y los controles necesarios para completar la sesión.

Bootstrap y Bootswatch

Para el diseño visual se ha empleado **Bootstrap**, un *framework* CSS que ofrece componentes predefinidos (botones, menús, rejillas, formularios) y un sistema de diseño adaptable a diferentes tamaños de pantalla [23]. Sobre Bootstrap se han aplicado temas de **Bootswatch**, que proporcionan estilos alternativos listos para usar sin modificar el marcado HTML [22].

El uso de estos componentes facilita mantener una apariencia coherente en toda la aplicación y aplicar criterios básicos de accesibilidad, como tamaños de fuente adecuados, contraste suficiente y distribución ordenada de los elementos en la interfaz [30, 35].

Chart.js

Para la representación gráfica de la evolución de los pacientes se ha utilizado **Chart.js**, una biblioteca JavaScript para crear gráficos sobre *canvas* HTML5 a partir de datos y opciones de configuración sencillas [8]. En TerapiTrack se utiliza para generar las gráficas temporales de puntuaciones en las vistas de profesional y paciente, ofreciendo una visualización clara de la evolución del tratamiento.

Soporte para mando SNES

Además del uso con ratón y teclado, TerapiTrack incorpora soporte para un mando SNES conectado por USB mediante la *Gamepad API* del navegador. En el área de paciente, la mayoria de acciones de navegación (mover el foco entre elementos, seleccionar opciones o volver atrás) pueden realizarse con los botones del mando, lo que facilita el uso de la aplicación a personas con dificultades motoras finas o menor familiaridad con el ratón [30].

Esta funcionalidad se ha implementado mediante un módulo JavaScript que escucha los eventos del mando, traduce las pulsaciones a acciones de la

interfaz y actualiza el foco de forma visual para indicar qué elemento está seleccionado en cada momento.

4.5. Testing y garantía de calidad

Pytest

Las pruebas automáticas se han realizado con **Pytest**, un marco de testing para Python que permite definir casos de prueba y agruparlos en módulos dentro de la carpeta `tests` [24]. Pytest ofrece utilidades para preparar datos de prueba mediante *fixtures*, parametrizar pruebas y generar informes con el resultado de la ejecución.

En el contexto de TerapiTrack, se ha utilizado para comprobar el funcionamiento de los modelos de datos y algunos de los flujos principales definidos en los controladores (gestión de usuarios, sesiones y evaluaciones), reduciendo el riesgo de introducir errores al modificar el código.

4.6. Despliegue e infraestructura

Heroku

Para ejecutar la aplicación en un entorno accesible desde internet se ha utilizado **Heroku**, una plataforma *Platform as a Service* (PaaS) [28]. Heroku permite desplegar aplicaciones a partir de un repositorio Git y definir el proceso de arranque mediante un fichero `Procfile`, en el que se indica el comando que debe ejecutarse para iniciar la aplicación Flask con un servidor compatible con WSGI [27]. WSGI (Web Server Gateway Interface) es una especificación que define cómo se comunican las aplicaciones web en Python con los servidores que las ejecutan, de manera que distintos servidores y *frameworks* puedan interoperar entre sí [18].

La configuración concreta del entorno (por ejemplo, la URL de la base de datos PostgreSQL o la clave secreta de Flask) se realiza mediante variables de entorno, que la aplicación lee en tiempo de ejecución a través del módulo de configuración.

Para el almacenamiento de archivos multimedia se ha integrado **Cloudinary**, un servicio externo que permite gestionar y servir de forma eficiente los vídeos y otros recursos generados por la aplicación [4].

4.7. Control de versiones y colaboración

Git y GitHub

El código del proyecto se gestiona con **Git**, un sistema de control de versiones distribuido que mantiene un historial de cambios y permite trabajar con ramas para desarrollar nuevas funcionalidades de forma aislada [10]. Además, como repositorio remoto se ha empleado **GitHub**, que ofrece alojamiento para el código, sistema de *issues* para registrar tareas e incidencias y herramientas para revisar cambios antes de integrarlos en la rama principal.

Para el trabajo diario se ha utilizado **Git Bash** como interfaz de línea de comandos, lo que facilita ejecutar los comandos de Git desde el propio entorno de desarrollo.

4.8. Otras herramientas de desarrollo

Durante el desarrollo también se han utilizado varias herramientas de apoyo:

- **Visual Studio Code**: es un entorno de desarrollo integrado utilizado como editor principal para el código Python, las plantillas y los ficheros de configuración, con extensiones específicas para Flask, Git y trabajo con SQLite.
- **LaTeX Workshop y Overleaf**: son herramientas para la edición y compilación de la memoria y los anexos en LaTeX, combinando el trabajo local en Visual Studio Code con la plataforma colaborativa Overleaf [17].
- **Draw.io**: es una aplicación para la creación de diagramas, utilizada para representar la arquitectura general del sistema, el modelo de datos y los principales flujos de uso.

4.9. Resumen de herramientas utilizadas

El conjunto de técnicas y herramientas descritas en este capítulo constituye la base tecnológica de TerapiTrack y sirve como referencia para comprender los aspectos de diseño e implementación que se desarrollan en los capítulos de aspectos relevantes y en los anexos técnicos.

Herramienta	Ámbito	Descripción
Python	Backend	Lenguaje principal del servidor y scripts.
Flask	Backend	Framework web para rutas y vistas.
SQLAlchemy	Base de datos	ORM para modelos y consultas.
SQLite / PostgreSQL	Base de datos	Motores relacionales en desarrollo y producción.
DB Browser for SQLite	Base de datos	Cliente gráfico para la BD local.
Jinja	Plantillas	Generación de HTML dinámico.
HTML5 / JavaScript	Frontend	Estructura y lógica en el navegador.
Bootstrap / Bootswatch	Frontend	CSS y temas para la interfaz.
Chart.js	Frontend	Gráficos de evolución de pacientes.
Gamepad API	Frontend	Soporte del mando SNES.
Cloudinary	Storage	Servicio para vídeos y recursos.
Pytest	Testing	Pruebas automatizadas del sistema.
Jira	Gestión	Planificación y seguimiento de tareas.
Git	Versionado	Control de versiones del código.
Git Bash	Versionado	Uso de Git desde la consola.
GitHub	Repositorio	Alojamiento remoto e *issues*.
VS Code	Desarrollo	Editor principal del proyecto.
LaTeX Workshop / Overleaf	Doc.	Edición y compilación en LaTeX.
Draw.io	Doc.	Diagramas de arquitectura y flujos.
Heroku	Despliegue	Ejecución de la aplicación en la nube.

Tabla 4.1: Herramientas utilizadas en el proyecto

Como se resume en la tabla 4.1, estas herramientas cubren desde el desarrollo y la gestión de datos hasta el despliegue y la documentación del proyecto, proporcionando un entorno de trabajo completo para TerapiTrack.

5. Aspectos relevantes del desarrollo del proyecto

Este capítulo describe los aspectos más significativos del desarrollo de TerapiTrack desde un punto de vista práctico. Se presentan las decisiones adoptadas en cuanto al ciclo de vida del proyecto, la arquitectura de la aplicación, el diseño del modelo de datos y la implementación de los flujos de uso más relevantes, así como algunos problemas encontrados y las soluciones que se han aplicado en cada caso.

5.1. Ciclo de vida y organización del trabajo

El proyecto no se ha desarrollado como un bloque único, sino en varias iteraciones en las que se iba ampliando la funcionalidad y ajustando el diseño en función de lo que se iba aprendiendo. En la práctica, esto ha supuesto combinar una planificación inicial con una forma de trabajar más flexible, en la que se han ido revisando prioridades y corrigiendo decisiones técnicas cuando era necesario [23].

Desde el principio se utilizó Jira para organizar el trabajo en tareas manejables. Al comienzo había simplemente una lista de tareas generales (por ejemplo, «crear modelo de datos básico» o «pantallas de login y registro»), pero a medida que el proyecto fue creciendo se reorganizó el tablero en secciones por rol (paciente, profesional, administrador) y por tipos de actividad (desarrollo, pruebas, documentación). En lugar de mantener la lista plana inicial, se crearon tareas épicas que agrupaban funcionalidades relacionadas, se definieron historias de usuario más claras y se desglosaron en tareas pequeñas que resultaban más fáciles de abordar y de dar por cerra-

das [1]. Este cambio ayudó a localizar más rápido qué parte del sistema se estaba tocando en cada momento y a evitar que se quedaran funcionalidades «a medias».

Git y GitHub se han utilizado de forma continuada para gestionar el código fuente, aunque con una organización de ramas poco convencional. El repositorio remoto de GitHub ha servido como copia de seguridad y como registro de la evolución del proyecto, pero en lugar de trabajar sobre la rama `main` y crear ramas auxiliares para funcionalidades concretas, casi todo el desarrollo real se ha llevado a cabo en una rama llamada `Pruebas`. La idea inicial era experimentar con cambios y funcionalidades en esta rama auxiliar antes de integrarlos en `main`, manteniendo la rama principal estable y con un historial limpio.

Durante un periodo de varios meses en verano se trabajó de forma intensiva en local, sin realizar *commits* ni subir el código a GitHub, lo que hizo que después se tuvieran que subir bloques grandes de cambios de una sola vez y que algunas tareas marcadas como completadas en Jira no tengan asociado un *commit* concreto [1]. Una vez que se retomó la rutina de *commits* más frecuentes, toda esa actividad quedó registrada en la rama `Pruebas`, desde la que se ha desarrollado la mayor parte de las funcionalidades (gestión de ejercicios, sesiones, vídeos, evaluaciones, tests unitarios con alta cobertura de código (en torno al 99 %), integración de Cloudinary, mejoras de documentación en el código, etc.).

Aunque no siempre se ha logrado, en la última fase se intentó corregir esta situación manteniendo una rutina más disciplinada: cuando se completaba una tarea se procuraba hacer un *commit* con un mensaje descriptivo y, en la medida de lo posible, realizar el *push* y la actualización del estado de la tarea en Jira de forma casi simultánea. A raíz de esta experiencia, se ha tomado conciencia de la importancia de mantener una cierta disciplina en el uso de las herramientas de control de versiones: realizar *commits* con cierta frecuencia, escribir mensajes claros y mantener sincronizado el estado de las tareas en Jira con los cambios reales en el código. La rama `Pruebas` se integró en `main` mediante un merge final realizado justo antes de la entrega del proyecto, de modo que el historial completo de desarrollo quedó reflejado en la rama principal y visible en el repositorio para futuras consultas o ampliaciones del sistema [1].

5.2. Arquitectura y organización del código

La aplicación sigue una arquitectura basada en el patrón MVC (Modelo-Vista-Controlador) adaptado a Flask, de modo que la lógica de presentación, los datos y el control de flujo se mantienen separados [9]. Esta separación facilita localizar rápidamente dónde está definida cada parte del sistema y permite modificar una capa sin afectar a las demás.

Estructura de carpetas

El código fuente de TerapiTrack se organiza en torno a la carpeta `src`, que contiene los elementos principales de la aplicación:

- `controladores`: agrupa las rutas y la lógica de negocio por rol (autenticación, administración, profesional y paciente), e incluye las funciones de vista y los decoradores que verifican el rol del usuario antes de permitir el acceso.
- `modelos`: define las entidades del dominio (usuarios, pacientes, profesionales, ejercicios, sesiones, evaluaciones y vídeos de respuesta) y las tablas de asociación para las relaciones de muchos a muchos.
- `vistas`: contiene las plantillas Jinja organizadas en subcarpetas por rol (administrador, profesional, paciente), todas ellas heredando de una plantilla base común para mantener una estructura y un estilo coherentes.
- `static`: almacena los recursos estáticos de la interfaz web (hojas de estilo, imágenes y vídeos de ejemplo).
- Otros módulos de soporte, dedicados a la configuración de la aplicación, la inicialización de extensiones como SQLAlchemy o Flask-Login y la definición de formularios web.

Además, en la raíz del proyecto se encuentran el módulo principal de arranque de la aplicación, el script para inicializar la base de datos con datos de prueba, la carpeta `tests` con las pruebas automatizadas y los ficheros de configuración necesarios para el despliegue y la ejecución en distintos entornos.

Esta organización ha resultado clave para mantener el proyecto manejable a medida que crecía el número de funcionalidades.

Uso de *blueprints*

Flask permite organizar las rutas en *blueprints*, que son módulos independientes que después se registran en la aplicación principal. En TerapiTrack se ha seguido este patrón para separar la lógica según el rol del usuario (administrador, profesional y paciente), agrupando en cada *blueprint* las vistas y flujos de trabajo correspondientes a ese rol [20, 25].

De esta forma, cada conjunto de rutas puede evolucionar de manera relativamente independiente sin mezclar código de distintos roles en un mismo módulo.

Decoradores para control de acceso

Uno de los aspectos que ha requerido más atención ha sido asegurar que cada usuario solo pueda acceder a las funcionalidades que le corresponden. Para ello se han definido decoradores personalizados que comprueban el rol del usuario antes de ejecutar una vista.

Por ejemplo, un decorador específico verifica que el usuario autenticado tenga rol de administrador; si no es así, redirige a la página principal con un mensaje de error. De forma similar, otros decoradores se encargan de que las rutas destinadas a profesionales o pacientes solo sean accesibles para el tipo de usuario adecuado.

Esta solución ha simplificado la lógica de las vistas, evitando tener que repetir las mismas comprobaciones de seguridad en cada ruta.

5.3. Diseño del modelo de datos

El modelo de datos se ha diseñado aplicando principios de normalización para evitar redundancias y mantener la coherencia de la información [6]. El diseño se ha llevado al menos hasta tercera forma normal (3FN), evitando atributos multivaluados o repetidos y separando en tablas diferentes la información que pertenece a entidades conceptualmente distintas. De esta forma se reduce la redundancia, se facilita el mantenimiento de la base de datos y se minimiza el riesgo de inconsistencias cuando se actualizan los datos clínicos o las asignaciones terapéuticas.

Las entidades principales y sus relaciones se describen a continuación.

Entidades principales

- **Usuario**: representa a cualquier persona que accede al sistema, con campos básicos (nombre, correo, contraseña cifrada) y un campo de rol que indica si es administrador, profesional o paciente.
- **Paciente**: extiende la información de un usuario de tipo paciente, incluyendo datos clínicos y de contacto.
- **Profesional**: extiende la información de un usuario de tipo profesional, con datos de especialidad y contacto.
- **Ejercicio**: define un ejercicio de rehabilitación, con su nombre, descripción, duración estimada y la ruta al vídeo de ejemplo.
- **Sesion**: representa una sesión terapéutica asignada a un paciente en una fecha concreta, con un estado que indica si está pendiente, en curso o completada.
- **VideoRespuesta**: almacena la grabación del paciente al realizar un ejercicio dentro de una sesión, junto con la fecha de grabación y la URL del vídeo.
- **Evaluacion**: contiene la calificación y los comentarios del profesional sobre el desempeño del paciente en un ejercicio concreto de una sesión.

Relaciones de muchos a muchos

Algunas relaciones del sistema requieren tablas intermedias para conectar entidades de forma flexible:

- **Paciente-Profesional**: un paciente puede estar atendido por varios profesionales y un profesional puede llevar el seguimiento de varios pacientes. Esta relación se modela mediante una tabla de asociación específica, que permite modificar las vinculaciones sin duplicar información ni alterar los datos básicos de pacientes y profesionales.
- **Ejercicio-Sesión**: una sesión puede incluir varios ejercicios y un mismo ejercicio puede aparecer en distintas sesiones. Para representar esta relación se utiliza la tabla intermedia `Ejercicio_Sesion`, que enlaza cada sesión con los ejercicios que la componen y sirve de punto de unión con las entidades `VideoRespuesta` y `Evaluacion`, de modo que cada combinación de sesión y ejercicio pueda tener su propia grabación y su propia evaluación.

La decisión de normalizar estas relaciones evita duplicar información y facilita modificar las asignaciones sin afectar a los datos de base de las entidades.

Claves e índices

Todas las tablas tienen una clave primaria autogenerada (`id`) y claves foráneas que garantizan la integridad referencial. En algunos casos se han añadido índices adicionales (por ejemplo, sobre el campo `email` en Usuario o sobre la fecha de la sesión) para mejorar el rendimiento de las consultas más frecuentes.

Decisiones y problemas de diseño

Más allá de la descripción estática de las entidades, el diseño del modelo de datos requirió varias iteraciones hasta encontrar una solución que reflejara bien el dominio del problema. Uno de los primeros debates fue cómo representar a los diferentes tipos de usuario. En las primeras versiones se partió de dos tablas separadas, una para `Usuario` y otra para `Paciente`, sin una estructura clara para incorporar después a los profesionales. Sin embargo, a medida que se fue incorporando la figura del profesional sanitario resultaba poco natural mezclar en la misma entidad tanto las credenciales como los datos clínicos del paciente y los datos de perfil del profesional, y empezaron a aparecer campos que solo tenían sentido para algunos tipos de usuario. A partir de esta experiencia se decidió separar la entidad genérica `Usuario` de las entidades específicas `Paciente` y `Profesional`, manteniendo en la primera la autenticación y el rol, y trasladando a las segundas la información propia de cada perfil. Esta decisión simplificó la lógica de la aplicación y permitió mantener el modelo preparado para añadir nuevos roles en el futuro sin romper la estructura existente.

También fue necesario reflexionar sobre cómo modelar las relaciones entre profesionales y pacientes, y entre ejercicios y sesiones. Inicialmente se barajó la posibilidad de restringir a un único profesional por paciente o de asociar los ejercicios directamente a las sesiones, pero pronto se vio que esto no se ajustaba a la realidad clínica ni a los requisitos de flexibilidad del sistema. Finalmente se optó por modelar ambas relaciones como muchos a muchos mediante tablas de asociación: una para vincular pacientes y profesionales, y otra para enlazar ejercicios y sesiones. Esta última tabla actúa como punto de unión entre el plan terapéutico (la sesión) y el contenido concreto

(los ejercicios), y permitió reutilizar un mismo ejercicio en varias sesiones distintas sin duplicar información.

La parte más delicada llegó al integrar en el modelo la información del desempeño del paciente a lo largo del tiempo. En los primeros bocetos solo existía una entidad de `Progreso`, que concentraba tanto la referencia al ejercicio y al paciente como la puntuación, los comentarios del profesional y la ruta del vídeo de respuesta, lo que dificultaba separar el hecho de haber realizado un ejercicio de la valoración que se hacía sobre él.

Tras varias pruebas se reorganizó esta parte introduciendo las entidades `VideoRespuesta` y `Evaluacion`, ambas asociadas a la tabla intermedia `Ejercicio_Sesion`. De este modo, cada ejercicio de cada sesión puede tener su propia grabación y, opcionalmente, una evaluación asociada, lo que refleja mejor el flujo real del tratamiento y evita inconsistencias cuando se añaden, modifican o repiten sesiones.

5.4. Flujos de uso más relevantes

Esta sección describe algunos de los flujos más importantes del sistema, explicando las decisiones técnicas que se han tomado y los problemas que han surgido durante su implementación.

Autenticación y gestión de roles

El flujo de autenticación se basa en Flask-Login, que proporciona sesiones gestionadas automáticamente y decoradores para proteger rutas que requieren autenticación previa. Cuando un usuario inicia sesión, el sistema verifica su correo y contraseña (cifrada con Bcrypt) y, si son correctos, almacena su identidad en la sesión.

Figura 5.1: Pantalla de acceso a TerapiTrack con formulario de autenticación.

En la Figura 5.1 se muestra el formulario de acceso común a todos los roles del sistema.

Una vez autenticado, el rol del usuario determina a qué parte de la aplicación puede acceder. Los decoradores personalizados (`@admin_required`, `@profesional_required`, `@paciente_required`) comprueban este rol antes de ejecutar cada vista.

Al principio del proyecto, las comprobaciones de rol se hacían directamente en cada ruta, lo que generaba código repetitivo y propenso a errores. La introducción de decoradores centralizó esta lógica y simplificó el mantenimiento del sistema.

Creación y asignación de sesiones terapéuticas

Un profesional puede crear una sesión para un paciente seleccionando los ejercicios que debe realizar y la fecha en que debe completarla. Este proceso se implementa mediante un formulario en el que se elige al paciente, se añaden ejercicios desde una lista disponible y se establece la fecha y hora en que debe ejecutarse.

Al guardar la sesión, el sistema crea un registro en la tabla `Sesion` y registros asociados en la tabla intermedia `Ejercicio_Sesion` para cada ejercicio incluido.

Uno de los problemas que surgió al implementar este flujo fue evitar que se pudieran añadir ejercicios duplicados a la misma sesión. La solución consistió

en incluir una restricción de unicidad sobre la pareja (sesión, ejercicio) en la tabla intermedia y validar en el controlador que no se intente insertar el mismo ejercicio dos veces.

Ejecución de sesiones por parte del paciente

Cuando un paciente accede a su panel, puede ver las sesiones que tiene asignadas en las proximas tres semanas. Al seleccionar una sesión, se muestra una página en la que el paciente sigue en todo momento las indicaciones del profesional: es este quien controla qué ejercicio se realiza en cada momento y cuándo se pasa al siguiente, mientras que en la pantalla del paciente solo se muestra el vídeo de ejemplo del ejercicio activo y se registra la grabación del ejercicio.

Este flujo ha sido uno de los más complejos de implementar, porque requiere coordinar la reproducción de vídeos, la captura desde la cámara del navegador y el almacenamiento de las grabaciones. Inicialmente se almacenaban los vídeos en una carpeta local del servidor (`uploads`), pero esto resultaba poco escalable y complicaba el despliegue en Heroku. La solución fue integrar Cloudinary como servicio externo de almacenamiento, de forma que los vídeos grabados por el paciente se suben directamente desde el navegador y el sistema solo guarda la URL en la base de datos [4].

Evaluación del desempeño

Una vez que el paciente ha completado una sesión, el profesional puede acceder al listado de sesiones completadas y revisar los vídeos grabados por el paciente. Para cada ejercicio de la sesión, el profesional puede asignar una calificación entre 1 y 5, y añadir comentarios sobre qué aspectos se han realizado correctamente y cuáles necesitan mejorar.

Esta información se almacena en la tabla `Evaluacion`, vinculada al registro concreto de `Ejercicio_Sesion`, de forma que cada ejercicio dentro de una sesión tiene su propia evaluación independiente. Esto permite al paciente consultar más adelante qué ejercicios ha realizado bien y en cuáles debe centrar su atención en futuras sesiones.

Resumen de flujos principales

La tabla 5.1 resume los flujos de uso más relevantes de TerapiTrack, indicando el actor principal, una descripción breve y las vistas implicadas en cada caso.

Flujo	Actor	Descripción	Vistas implicadas
Autenticación y acceso	Usuario (cualquier rol)	Introduce credenciales, accede al panel correspondiente.	Login, panel de administración, profesional y paciente.
Creación de sesión	Profesional	Selecciona paciente, elige ejercicios, fija la fecha de realización.	Sesiones profesional, creación de sesión y detalle de sesión.
Ejecución de sesión	Paciente	Consulta sesiones, sigue indicaciones profesional, graba ejercicios.	Mis sesiones del paciente y ejecución de sesión.
Evaluación de ejercicios	Profesional	Revisa grabaciones, añade puntuación y comentarios a ejercicios.	Sesiones, evaluación sesión, evaluación ejercicio.
Consulta historial y progreso	Paciente / profesional	Consultar historial sesiones, evolución de puntuaciones.	Progreso paciente (paciente), progreso paciente (profesional).

Tabla 5.1: Flujos de uso principales.

Visualización del progreso del paciente

Además de almacenar las evaluaciones individuales de cada ejercicio, TerapiTrack genera gráficos de evolución temporal que muestran de forma agregada las puntuaciones obtenidas por el paciente en sus sesiones. Estas visualizaciones permiten identificar tendencias de mejora o empeoramiento y facilitan que paciente y profesional comprendan de un vistazo cómo ha evolucionado el tratamiento a lo largo del tiempo.

5.5. Interfaz de usuario

Antes de implementar las plantillas HTML se realizaron bocetos en papel de las principales pantallas (paneles, listados y formularios), que sirvieron como guía inicial, aunque varias vistas se ampliaron o remodelaron durante el desarrollo, dando lugar a la interfaz web actual de TerapiTrack para cada rol del sistema.

Interfaz del administrador

El administrador dispone de un panel de control que le permite consultar de un vistazo el estado global del sistema y acceder rápidamente a las principales acciones de gestión (usuarios, vinculaciones y parámetros generales).

En la Figura 5.2 se muestran los indicadores básicos y los accesos directos más relevantes, mientras que desde la pantalla de configuración (Figura 5.3) se pueden ajustar aspectos como la política de retención de vídeos o el límite de almacenamiento.

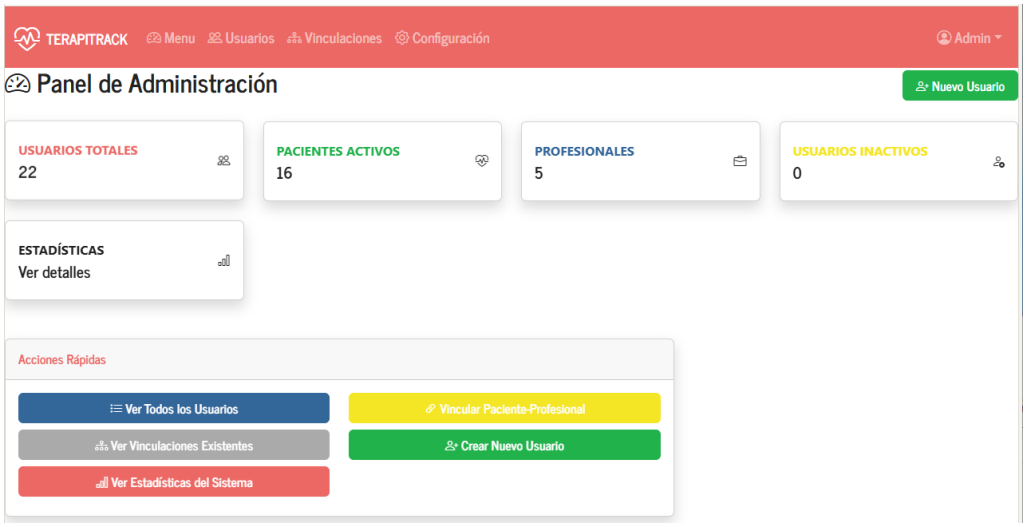

Figura 5.2: Panel principal de administración con métricas y accesos rápidos.

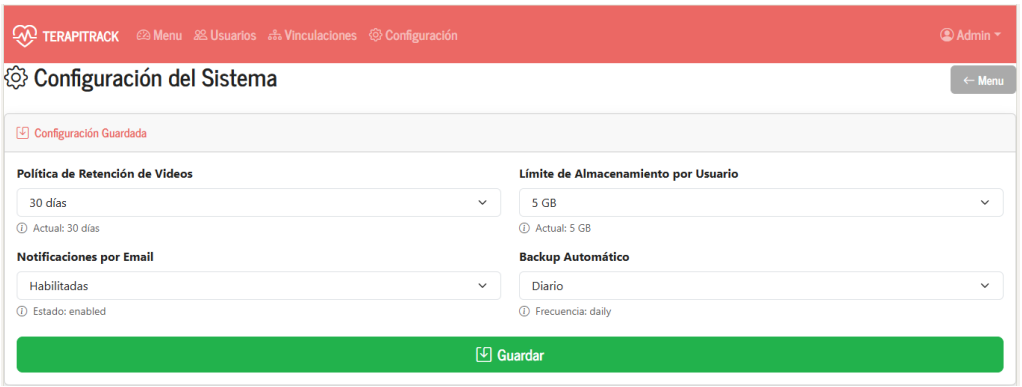

Figura 5.3: Pantalla de configuración del sistema y política de retención de vídeos.

Interfaz del profesional

El profesional dispone de un panel principal donde visualiza el número de pacientes asignados, sesiones pendientes y accesos a las acciones más

habituales. Desde este panel puede gestionar sus sesiones programadas (Figura 5.4), crear nuevas combinando ejercicios y fechas, y consultar el progreso de cada paciente mediante una gráfica temporal de puntuaciones generada con Chart.js (Figura 5.5).

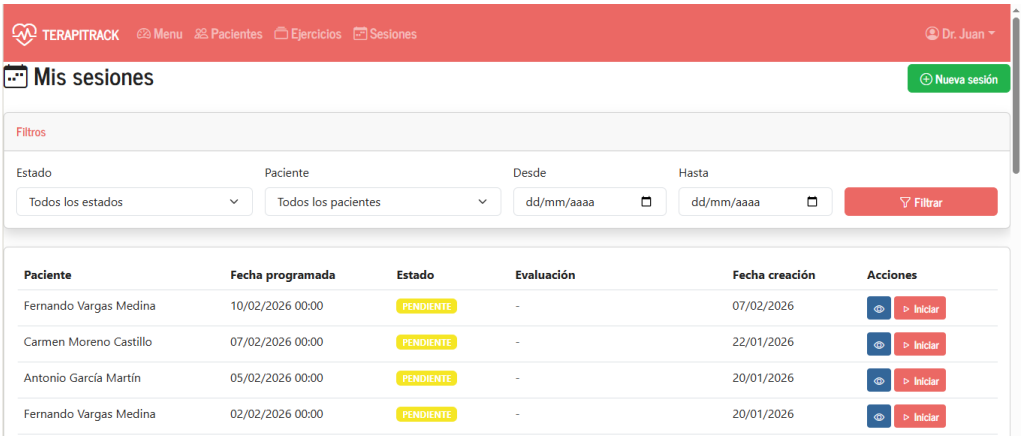

Figura 5.4: Listado de sesiones programadas con filtros básicos.

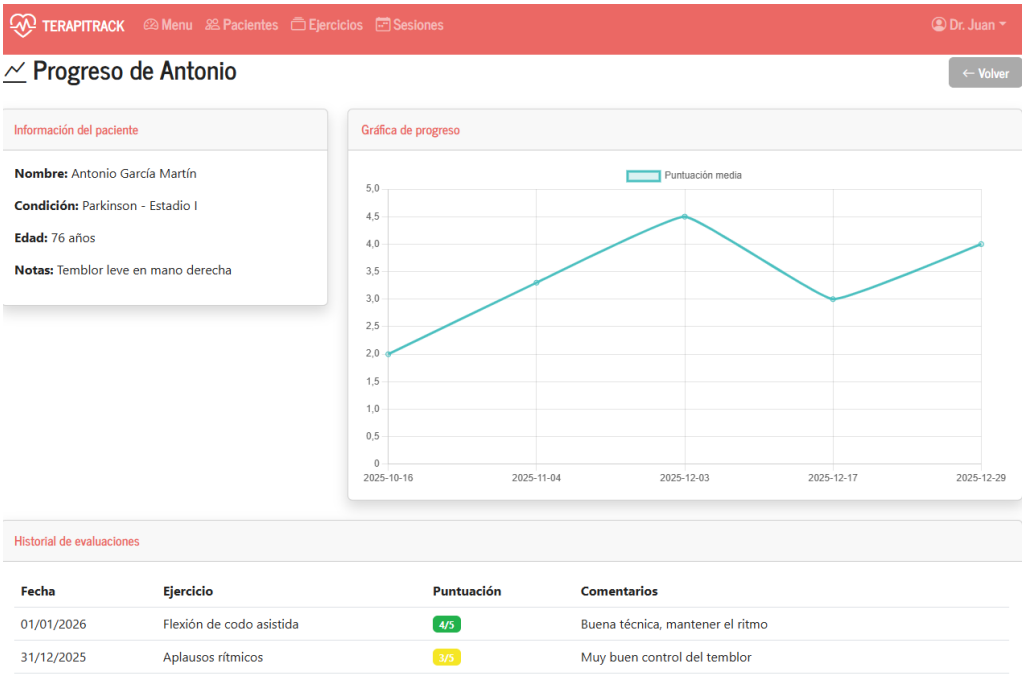

Figura 5.5: Pantalla de progreso del paciente con gráfica temporal de puntuaciones.

Finalmente, la evaluación detallada de cada ejercicio se realiza en una vista que coloca lado a lado el vídeo demostrativo y la grabación del paciente, junto con los controles para asignar una puntuación y registrar comentarios. Esta pantalla (Figura 5.6) constituye el último paso del flujo de trabajo del profesional.

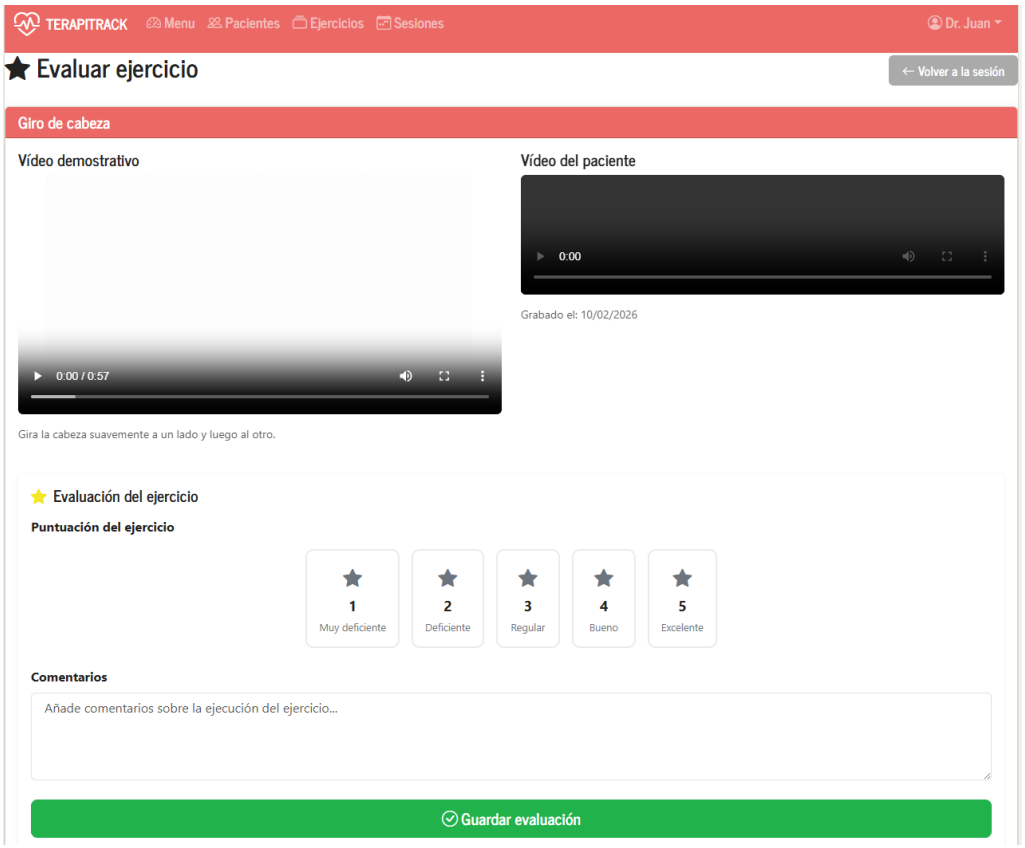

Figura 5.6: Evaluación de un ejercicio con vídeo demostrativo y grabación del paciente.

Interfaz del paciente

El paciente accede a un panel principal simplificado, con cuatro accesos directos a *Ejercicios*, *Sesiones*, *Progreso* y *Ayuda*, dispuesto en forma de mandos circulares fáciles de seleccionar. Desde esta pantalla puede desplazarse entre las opciones con las flechas del mando SNES y confirmar con los botones de acción, lo que reduce la necesidad de movimientos de precisión con el ratón y facilita el uso a pacientes con temblor o rigidez.

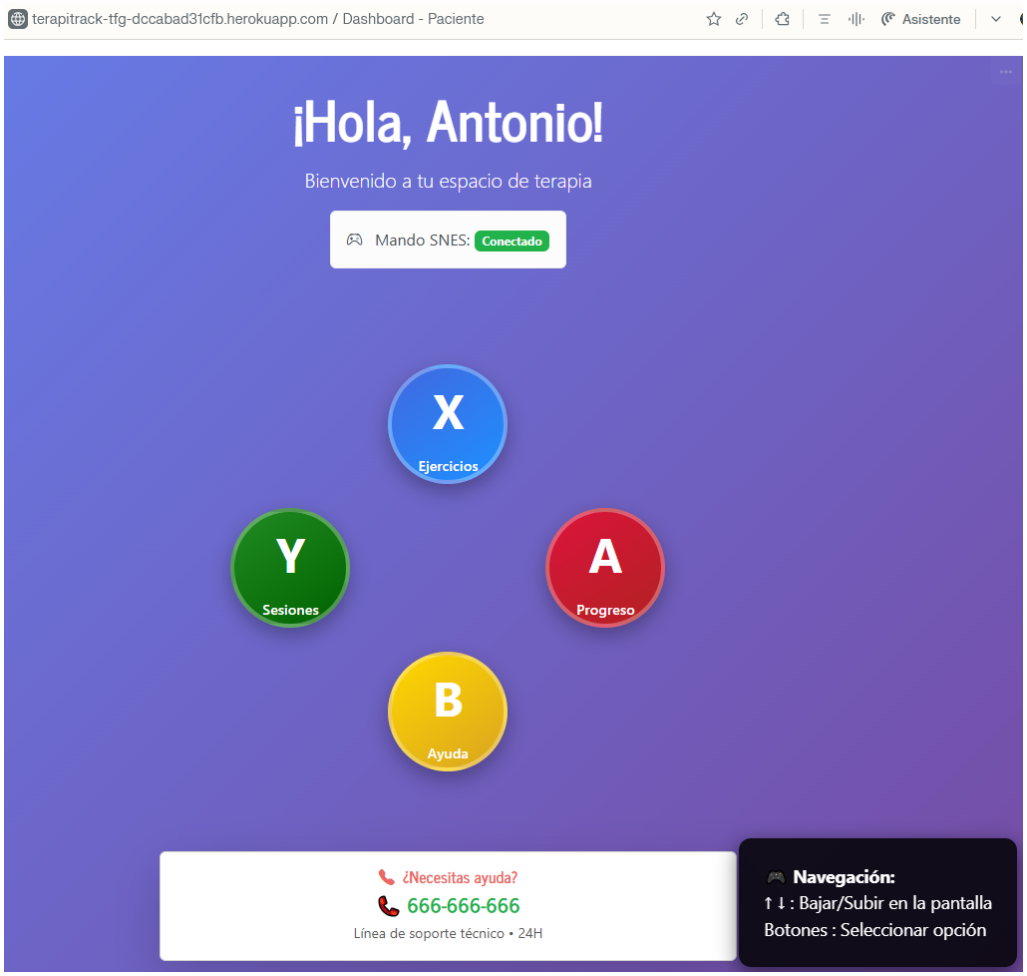

Figura 5.7: Panel principal del paciente con accesos simplificados.

La sección *Sesiones* muestra un calendario con las próximas semanas y un resumen de la sesión seleccionada, indicando profesional, especialidad y número de ejercicios asignados. En *Ejercicios*, el paciente visualiza tarjetas con el vídeo de ejemplo, una breve descripción y la duración estimada, pudiendo reproducir cada ejercicio y conocer cuántas veces se le ha asignado.

En *Progreso* se presentan indicadores numéricos (evaluaciones totales, mejor nota, media) y una gráfica temporal de las puntuaciones obtenidas a partir de las evaluaciones almacenadas, acompañadas de las tarjetas de ejercicios evaluados con los comentarios del profesional. Por último, la vista *Ayuda* actúa como guía rápida del mando SNES, explicando qué hace cada botón dentro de la aplicación para que el paciente pueda recordar en cualquier momento cómo desplazarse y seleccionar opciones.

5.6. Decisiones de diseño en la interfaz

La interfaz de TerapiTrack se ha diseñado teniendo en cuenta que muchos de los usuarios finales (pacientes) pueden ser personas mayores con poca experiencia en el uso de aplicaciones web, y que algunos de ellos presentan limitaciones motoras derivadas de la enfermedad de Parkinson [26, 35].

Navegación simplificada

Cada rol dispone de un menú de navegación adaptado a sus necesidades, con un número reducido de opciones claramente etiquetadas. Por ejemplo, el menú del paciente incluye únicamente «Sesiones», «Ejercicios», «Ayuda» y «Progreso», evitando sobrecargar la interfaz con opciones que no va a utilizar.

La plantilla base (`base.html`) define la estructura común de todas las páginas (cabecera, menú, pie de página, estilos) y se encarga de mostrar mensajes de error o confirmación de forma coherente en todo el sistema mediante *flash messages* de Flask.

Controles grandes y contraste adecuado

En las pantallas de ejecución de sesiones se han utilizado botones grandes, con un tamaño de fuente generoso y un espaciado suficiente entre elementos para facilitar la interacción con ratón, teclado o mando SNES. Los colores de fondo y de texto se han elegido siguiendo las recomendaciones de WCAG para asegurar un contraste suficiente, especialmente en mensajes de error (rojo) y de éxito (verde) [35].

Feedback visual claro

Durante la ejecución de una sesión, el sistema muestra de forma clara en qué ejercicio se encuentra el paciente. Cuando se graba un vídeo, aparece un indicador de «grabando» que cambia de color, y una vez finalizada la grabación se muestra un mensaje de confirmación antes de pasar al siguiente ejercicio.

Este tipo de feedback se consideró fundamental durante las pruebas, porque ayuda al paciente a sentir que tiene el control del proceso y reduce la incertidumbre sobre si el sistema está funcionando correctamente. Además, en el panel del paciente se ha habilitado el uso de un mando SNES como

dispositivo de entrada alternativo, de forma que los ejercicios puedan ejecutarse con menos movimientos de precisión y con un patrón de interacción más simple que el ratón tradicional [30].

5.7. Pruebas y validación

El sistema se ha probado de forma continua durante el desarrollo, combinando pruebas automáticas sobre los modelos y controladores con pruebas manuales sobre los flujos de usuario más complejos.

Pruebas unitarias con Pytest

Se ha implementado un conjunto de pruebas unitarias en la carpeta `tests`, que cubre las operaciones básicas de los modelos (creación, actualización, borrado y consultas) y varios de los flujos principales de los controladores. Para ejecutarlas se utiliza el comando `pytest tests/ -v -cov=src -cov-report=html`, que genera un informe de cobertura por fichero en formato HTML [24].

En la versión final del proyecto se han definido 190 tests y se ha alcanzado una cobertura global del 99 %, con todos los modelos y las funciones auxiliares al 100 % y los controladores principales (administrador, profesional, paciente y autenticación) en torno al 99 % de líneas cubiertas. Aunque esta cobertura no garantiza que el código esté libre de errores, sí proporciona una red de seguridad que facilita realizar cambios sin romper funcionalidades ya existentes [24]. En la práctica, estas pruebas permitieron detectar regresiones al refactorizar controladores y modelos, de modo que se corrigieron varios fallos de validación y de control de acceso antes de desplegar nuevas versiones.

En la práctica, la batería de pruebas permitió comprobar la cobertura global del código, que en la versión final del proyecto alcanza en torno al 99 % en los módulos principales.

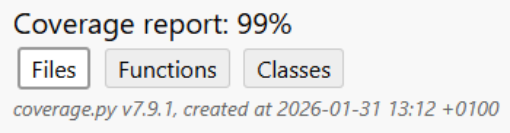

Figura 5.8: Resumen de cobertura generado con `pytest`.

Pruebas manuales en entornos reales

Además de las pruebas automáticas, se han realizado pruebas manuales tanto en el entorno de desarrollo como en el despliegue de Heroku, simulando distintos perfiles de usuario (administrador, profesional, paciente) y comprobando el comportamiento del sistema con datos reales y con conexiones a Cloudinary [4]. En paralelo, se han ejecutado pruebas sobre la base de datos desde `flask shell` y mediante consultas SQL en DB Browser for SQLite para verificar la creación de entidades, las relaciones de muchos a muchos y operaciones de actualización habituales (cambio de estado de sesiones, modificación de evaluaciones, etc.), que se describen con más detalle en el anexo de documentación técnica. Estas pruebas han permitido detectar problemas que no eran evidentes en el entorno local, como tiempos de respuesta más largos al subir vídeos grandes o errores de permisos en rutas específicas, cuya corrección se ha ido registrando en Jira y consolidando en la rama `Pruebas`, siguiendo el ciclo iterativo descrito en la primera sección de este capítulo [1].

Análisis estático con SonarQube

Además de las pruebas automatizadas con `pytest`, se realizó un análisis estático del código utilizando SonarQube, como se puede ver en la figura 5.8, con el objetivo de detectar *code smells*, posibles errores y problemas de mantenibilidad. Para ello se configuró un proyecto específico para TerapiTrack, se lanzó el escaneo sobre el código de `src` y se revisaron los informes generados en la interfaz web de la herramienta.

El resultado del análisis indicó que no se detectaban vulnerabilidades de seguridad ni errores críticos, y que la mayor parte de las incidencias correspondían a recomendaciones de estilo (nombres de variables, longitud de funciones o duplicidad leve de código). Estas observaciones se utilizaron para refactorizar algunos controladores y modelos, mejorando la legibilidad y reduciendo el acoplamiento entre módulos, aunque se mantuvieron ciertos *smells* considerados aceptables para el alcance de un prototipo académico.

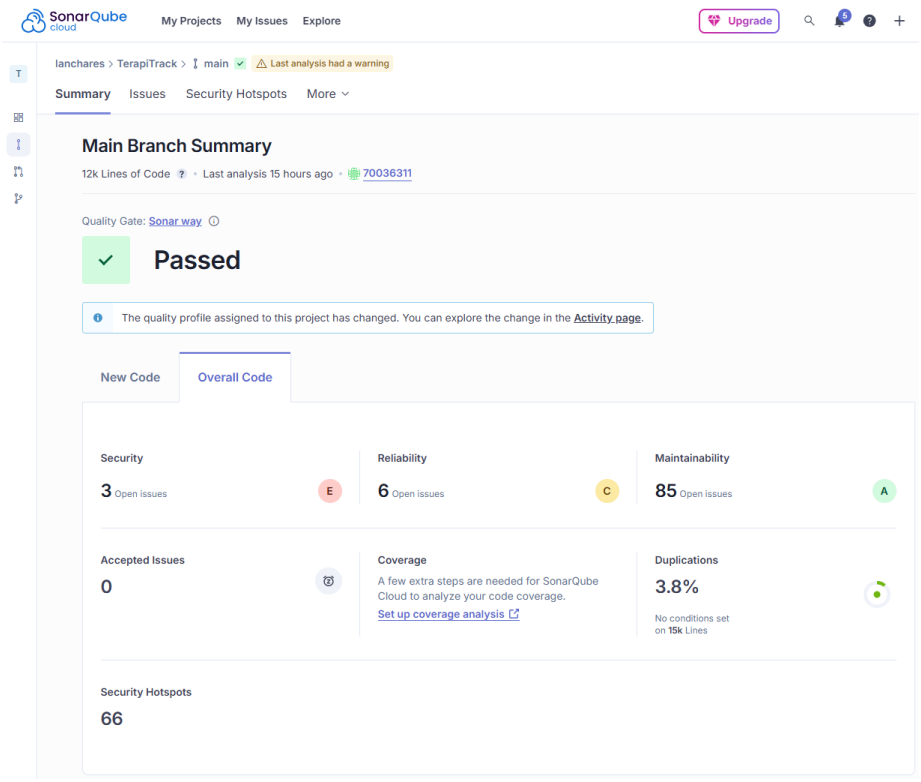

Figura 5.9: Resumen del análisis estático de TerapiTrack en SonarQube.

5.8. Retos técnicos y soluciones adoptadas

Durante el proyecto surgieron muchos desafios técnicos, los mas significativos fueron:

1. **Persistencia efímera en Heroku:** el sistema de archivos local de Heroku se reinicia diariamente, eliminando vídeos subidos. Se solucionó integrando Cloudinary para almacenamiento persistente en la nube.

2. **Conflictos en Jinja2:** en vistas complejas, la duplicidad de etiquetas `endblock` rompía el diseño sin generar errores explícitos. Se solucionó mediante una auditoría y mostrando *logs* por pantalla.

3. **Captura de eventos de hardware:** la implementación inicial del mando con `pygame` (backend) falló en el despliegue web. La solución fue migrar la lógica al frontend usando la Gamepad API de HTML5.

6. Trabajos relacionados

El desarrollo de TerapiTrack se enmarca en un ámbito de investigación activo en el que se han realizado diversos trabajos centrados en la telerehabilitación, el análisis de ejercicios mediante vídeo y la aplicación de técnicas de inteligencia artificial para evaluar el desempeño de pacientes [5]. Este capítulo presenta un breve resumen de los trabajos más relevantes que han servido como referencia o que comparten objetivos similares con el proyecto actual [2, 36].

6.1. Trabajos previos del grupo de investigación

Varios de los tutores y colaboradores del proyecto han participado en trabajos anteriores relacionados con la telerehabilitación y el análisis automático de ejercicios, lo que ha proporcionado una base sólida para el diseño de TerapiTrack [13, 29].

Evaluación de ejercicios de rehabilitación en vídeo

El Trabajo de Fin de Grado de Lucía Núñez Calvo abordó la evaluación automática de ejercicios de rehabilitación a partir de vídeos grabados por los pacientes. En ese proyecto se exploraron técnicas de extracción de características y comparación de poses para determinar si un paciente estaba realizando correctamente un ejercicio frente a un vídeo de referencia [2].

TerapiTrack comparte el enfoque de trabajar con vídeos como fuente principal de información, aunque en este caso la evaluación del desempeño la realiza el profesional de forma manual en lugar de aplicar análisis automático.

La experiencia acumulada en ese trabajo ha servido para diseñar la estructura de almacenamiento de vídeos y para prever una posible ampliación futura del sistema que integre técnicas de inteligencia artificial [2].

Detección de ejercicios en vídeos de rehabilitación

El Trabajo de Fin de Grado de Luis Ángel Espinosa Lafuente se centró en la detección automática del tipo de ejercicio que un paciente estaba realizando a partir de la secuencia de poses capturadas en un vídeo. Este proyecto permitió validar que es posible clasificar ejercicios de forma fiable utilizando modelos entrenados sobre conjuntos de datos etiquetados [36].

En TerapiTrack cada ejercicio forma parte de una sesión planificada y queda identificado de forma explícita en el modelo de datos, de modo que la aplicación siempre sabe qué ejercicio está realizando el paciente cuando graba su respuesta. Aunque no se utiliza clasificación automática, el diseño actual facilitaría incorporar en el futuro un módulo de detección que complemente esta identificación y permita asociar métricas adicionales de calidad a cada ejercicio grabado [36].

Infraestructura para telerehabilitación y análisis online

El Trabajo de Fin de Máster de José Luis Garrido Labrador desarrolló una infraestructura completa para gestionar programas de telerehabilitación, incluyendo el almacenamiento de vídeos, la asignación de ejercicios y el seguimiento de la evolución de los pacientes. Este trabajo sentó las bases arquitectónicas que se han reutilizado parcialmente en TerapiTrack, especialmente en lo relativo a la organización del modelo de datos y la separación de roles [13].

Detección de poses con Detectron2

El Trabajo de Fin de Máster de José Miguel Ramírez Sanz exploró el uso de Detectron2, un *framework* de detección de objetos y poses, para extraer información sobre la posición de las articulaciones de un paciente a partir de vídeos de ejercicios. Aunque TerapiTrack no incorpora análisis automático de poses en su versión actual, los resultados de ese trabajo han servido para entender las limitaciones técnicas y los requisitos de calidad de vídeo que serían necesarios si se decidiera añadir esta funcionalidad en el futuro [29].

6.2. Sistemas de telerehabilitación para Parkinson

Existen en la literatura varios sistemas orientados específicamente a pacientes con enfermedad de Parkinson que combinan telerehabilitación con técnicas de análisis automático [12].

Cubo y colaboradores presentaron un sistema de telerehabilitación basado en técnicas de *deep learning* para evaluar el desempeño de pacientes con Parkinson durante la realización de ejercicios en su domicilio. El sistema utiliza redes neuronales convolucionales para analizar vídeos y clasificar la calidad de los movimientos, proporcionando *feedback* automático al paciente y al profesional [5].

TerapiTrack comparte el objetivo de facilitar el seguimiento remoto de pacientes con Parkinson, pero adopta un enfoque más centrado en la gestión del flujo de trabajo clínico (asignación de sesiones, almacenamiento de vídeos, evaluación manual por parte del profesional) que en el análisis automático. Esta decisión se debe a que la evaluación manual permite al profesional tener en cuenta aspectos cualitativos que son difíciles de capturar mediante algoritmos, y porque la integración de técnicas de inteligencia artificial requiere conjuntos de datos etiquetados y validados que no estaban disponibles al inicio del proyecto [31].

6.3. Diferencias y aportaciones de TerapiTrack

Frente a los trabajos relacionados mencionados, TerapiTrack aporta las siguientes características diferenciadoras:

- **Gestión completa del ciclo de trabajo clínico**: TerapiTrack no se centra únicamente en el análisis de vídeos, sino en todo el proceso que va desde la asignación de ejercicios por parte del profesional hasta la evaluación del desempeño del paciente y la consulta del historial de sesiones [13].
- **Interfaz accesible y adaptada a los tres roles**: el sistema proporciona vistas específicas para administradores, profesionales y pacientes, con navegación simplificada y criterios de accesibilidad pensados para usuarios con poca experiencia tecnológica o con limitaciones motoras [26, 35].

- **Almacenamiento externo de vídeos**: la integración con Cloudinary resuelve los problemas de escalabilidad y despliegue que surgen al almacenar vídeos en el propio servidor, facilitando el uso del sistema en entornos de producción reales [4].
- **Evaluación cualitativa por parte del profesional**: a diferencia de los sistemas basados en análisis automático, TerapiTrack permite al profesional registrar comentarios y observaciones detalladas sobre cada ejercicio, algo especialmente valioso en el contexto clínico donde el juicio experto sigue siendo fundamental [12].
- **Preparado para integrar IA más adelante**: aunque la versión actual no realiza análisis automático de poses ni clasificación de ejercicios, el modelo de datos y la forma de almacenar los vídeos están pensados para poder incorporar módulos de visión artificial y *deep learning* en futuras versiones sin rediseñar por completo la aplicación [2, 29, 36].

Estos aspectos hacen de TerapiTrack un sistema complementario a los trabajos previos, que puede servir como base para futuras ampliaciones que integren técnicas de inteligencia artificial sin renunciar a la supervisión humana del proceso terapéutico [5].

7. Conclusiones y líneas de trabajo futuras

7.1. Conclusiones

El desarrollo de TerapiTrack ha permitido pasar de una idea inicial de plataforma de telerehabilitación a una herramienta web funcional que gestiona usuarios, ejercicios, sesiones terapéuticas y evaluaciones en un entorno realista. A lo largo del proyecto se han cubierto los principales objetivos funcionales y técnicos planteados, dejando una base razonablemente sólida sobre la que se pueden apoyar futuras extensiones del sistema.

Desde el punto de vista funcional, la aplicación ofrece paneles diferenciados para administradores, profesionales sanitarios y pacientes, de forma que cada tipo de usuario dispone de un entorno adaptado a sus tareas habituales. El sistema permite gestionar la vinculación entre pacientes y profesionales, crear y catalogar ejercicios en vídeo, programar sesiones personalizadas y registrar su realización, incluyendo la subida y almacenamiento de las grabaciones generadas en el domicilio del paciente. Además, se ha incorporado una trazabilidad básica de la evolución terapéutica a través del historial de sesiones y evaluaciones asociadas a cada paciente [27].

En el ámbito técnico, se ha diseñado una arquitectura modular basada en Flask y *blueprints*, apoyada en un modelo de datos relacional normalizado que separa claramente las entidades de usuario, paciente, profesional, ejercicio, sesión, vídeo de respuesta y evaluación [33, 18]. La integración de SQLAlchemy como ORM ha facilitado la evolución del esquema sin perder integridad referencial, mientras que el uso de Cloudinary ha permitido delegar el almacenamiento de vídeos en un servicio específico, evitando pro-

blemas de espacio y de ancho de banda en el propio servidor. El despliegue en Heroku y la configuración diferenciada de los entornos de desarrollo y producción han demostrado que la aplicación puede ejecutarse en la nube con una configuración razonable de recursos [4, 27, 28].

Otro aspecto destacable es la incorporación de pruebas automatizadas con *pytest* sobre los módulos más críticos, alcanzando una cobertura de casi el 100 % en modelos, decoradores y componentes de configuración [24]. Estas pruebas, combinadas con pruebas manuales sobre los flujos de uso principales en el entorno desplegado, han ayudado a detectar errores de diseño e implementación en fases tempranas y han dado una mayor tranquilidad a la hora de refactorizar código o ajustar funcionalidades. La decisión de seguir una metodología de trabajo iterativa apoyada en Jira también ha sido útil para planificar, seguir el avance real del proyecto y reordenar prioridades cuando ha sido necesario [1].

Desde una perspectiva más personal, el proyecto ha servido para integrar conocimientos de desarrollo web, bases de datos, control de versiones, despliegue en la nube y documentación técnica en un caso de uso centrado en la mejora de la calidad de vida de personas con enfermedad de Parkinson.

Entre las lecciones aprendidas destacan la utilidad de organizar el código en *blueprints* y emplear decoradores para el control de acceso, la importancia de diseñar un modelo de datos normalizado con tablas intermedias para las relaciones de muchos a muchos y las ventajas de integrar servicios externos como Cloudinary para resolver problemas de escalabilidad en el almacenamiento de vídeos. El trabajo ha dejado claro, además, lo importante que es mantener cierta disciplina en el uso de herramientas como Git y Jira, así como diseñar pensando en la accesibilidad y en las limitaciones de los usuarios finales desde el principio [35, 30]. Aunque han aparecido dificultades técnicas y de organización en distintos momentos, la experiencia global ha sido muy enriquecedora y ha permitido entregar una primera versión de TerapiTrack que cumple los objetivos planteados y puede utilizarse en un contexto de telerehabilitación supervisada.

7.2. Líneas de trabajo futuras

La versión actual de TerapiTrack cubre el ciclo básico de planificación, realización y evaluación de sesiones terapéuticas, pero deja abiertas varias líneas de mejora que podrían abordarse en trabajos posteriores. Algunas de estas líneas están relacionadas con la ampliación de la funcionalidad y otras con la incorporación de técnicas de análisis de vídeo más avanzadas o con

la adaptación del sistema a un entorno clínico real [5]. En particular, las mejoras descritas en este apartado forman parte de la evolución del proyecto de telerehabilitación en el que se enmarca TerapiTrack y no están incluidas en el alcance de la implementación realizada en este TFG.

En primer lugar, una evolución natural sería integrar módulos de análisis automático de las grabaciones utilizando técnicas de visión artificial y aprendizaje automático, aprovechando los trabajos previos del grupo en detección de poses y evaluación de ejercicios a partir de vídeo [2, 29, 36]. Esto permitiría complementar la evaluación manual del profesional con métricas objetivas sobre la ejecución de los movimientos, generando indicadores de calidad de los ejercicios o alertas cuando se detecten patrones de empeoramiento [12, 31]. La integración de estos módulos de procesamiento automático de vídeo se considera parte de la evolución futura del sistema y no se ha abordado en la versión desarrollada en este TFG.

En segundo lugar, sería interesante ampliar las funcionalidades de explotación de datos clínicos y de visualización del progreso del paciente. Actualmente el sistema ofrece un historial estructurado de sesiones y evaluaciones, pero se podrían incorporar gráficos más completos, filtros por tipo de ejercicio o por profesional y resúmenes periódicos tanto para los profesionales como para los propios pacientes. Este tipo de información agregada ayudaría a tomar decisiones sobre la adaptación de las terapias y a comunicar de forma más clara los avances o dificultades encontrados durante el tratamiento [5, 12, 15, 34].

Además, quedan pendientes algunas mejoras funcionales identificadas durante el desarrollo y registradas en Jira, como la incorporación de notificaciones por correo electrónico (por ejemplo, para recordar sesiones próximas o informar de nuevas evaluaciones) y la posibilidad de exportar las evaluaciones de un paciente a un informe en formato PDF. Estas tareas no afectan al funcionamiento básico de TerapiTrack, pero su implementación contribuiría a mejorar la comunicación con los usuarios y a facilitar la documentación de los resultados terapéuticos en futuras versiones del sistema.

Otra línea de mejora tiene que ver con la accesibilidad y la experiencia de usuario [35]. Aunque la interfaz ya sigue criterios básicos de accesibilidad y se ha pensado para personas mayores con posibles limitaciones motoras, sería recomendable realizar pruebas de usabilidad con pacientes reales y profesionales para identificar barreras concretas y ajustar textos, colores, tamaño de los elementos y flujos de navegación [26]. También podría valorarse la adaptación de la interfaz a dispositivos móviles o tabletas, que en muchos casos resultan más cómodos en el entorno doméstico [30].

Otra línea de trabajo, ligada al proyecto global de telerehabilitación, sería la incorporación de un sistema de videollamada en tiempo real entre paciente y profesional, de manera que algunas sesiones puedan realizarse de forma síncrona y supervisada [5]. Este módulo complementaría el modelo actual basado en vídeos grabados y evaluación diferida, pero su diseño e implementación quedan fuera del alcance de este TFG.

Desde el punto de vista de la integración con el ecosistema sanitario, una evolución relevante sería conectar TerapiTrack con otros sistemas clínicos, como historias clínicas electrónicas o plataformas de citación y teleconsulta. Esta integración permitiría evitar la duplicación de información, mejorar la coordinación entre distintos profesionales y facilitar que TerapiTrack se incorpore como una herramienta más dentro de los procesos asistenciales habituales. Para ello sería necesario estudiar los requisitos legales y de interoperabilidad, así como los estándares de intercambio de datos empleados en cada entorno [5].

Por último, cabría explorar la posibilidad de extender el uso de TerapiTrack más allá de la enfermedad de Parkinson, adaptando los catálogos de ejercicios y los criterios de evaluación a otras patologías crónicas que se puedan beneficiar de programas de rehabilitación remota [11, 14]. El diseño modular del sistema, tanto en el modelo de datos como en la interfaz, facilita esta reutilización siempre que se definan de forma adecuada los nuevos perfiles de usuario, los tipos de ejercicio y las necesidades específicas de cada grupo de pacientes [6, 23]. De este modo, el trabajo realizado podría servir como base para una plataforma más generalista de telerehabilitación, manteniendo el foco en la calidad, la accesibilidad y la seguridad de la información sanitaria [5].

Bibliografía

[1] Atlassian. Jira software: proyecto terapitrack. https://terapitrack.atlassian.net/, 2025. [Internet; descargado 8-enero-2026].

[2] Lucía Núñez Calvo. Evaluación de ejercicios de rehabilitación en vídeo. https://github.com/lnc1002/TFG-Evaluacion-Ejercicios-Rehabilitacion, 2021. Trabajo Fin de Grado, Universidad de Burgos. [Internet; descargado 8-enero-2026].

[3] Cleveland Clinic. Parkinson's disease: What it is, causes, symptoms & treatment. https://my.clevelandclinic.org/health/diseases/8525-parkinsons-disease-an-overview, 2025. [Internet; descargado 8-enero-2026].

[4] Cloudinary. Cloudinary console. https://console.cloudinary.com/, 2025. [Internet; descargado 8-enero-2026].

[5] Esther Cubo, Álvaro García-Bustillo, and otros. A low-cost system using a big-data deep-learning framework for telerehabilitation in parkinson's disease. *Healthcare*, 2023.

[6] C. J. Date. *An Introduction to Database Systems*. Pearson, 2004.

[7] DB Browser for SQLite. Db browser for sqlite. https://sqlitebrowser.org/, 2025. [Internet; descargado 8-enero-2026].

[8] Nick Downie. Chart.js documentation. https://www.chartjs.org/docs/latest/, 2025. [Internet; descargado 8-enero-2026].

[9] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[10] Git SCM. Git documentation. https://git-scm.com/doc, 2025. [Internet; descargado 8-enero-2026].

[11] Hospital for Special Surgery. About rehabilitation: Physical, occupational and speech therapy. https://www.hss.edu/health-library/conditions-and-treatments/list/rehabilitation, 2025. [Internet; descargado 8-enero-2026].

[12] S. Khera. Effect of telerehabilitation in parkinson disease: A systematic review and meta-analysis. *Physical Therapy*, 2025.

[13] José Luis Garrido Labrador. Infraestructura para telerehabilitación y análisis online. https://github.com/jlgarridol/FIS-FBIS, 2021. Trabajo Fin de Máster. [Internet; descargado 8-enero-2026].

[14] National Board for Certification in Occupational Therapy. Occupational therapy. https://www.nbcot.org/occupational-therapy, 2015. [Internet; descargado 8-enero-2026].

[15] National Institute of Neurological Disorders and Stroke. Parkinson's disease. https://www.ninds.nih.gov/health-information/disorders/parkinsons-disease, 2024. [Internet; descargado 8-enero-2026].

[16] Organización Médica Colegial de España. Una de cada cien personas mayores de 60 años padece parkinson en todo el mundo. https://www.medicosypacientes.com/articulo/una-de-cada-cien-personas-mayores-de-60-anos-padece-parkinson-en-todo-el-mundo/, 2023. [Internet; descargado 14-enero-2026].

[17] Overleaf. Overleaf online latex editor. https://www.overleaf.com/, 2025. [Internet; descargado 8-enero-2026].

[18] Pallets Projects. Documentación de flask. https://flask.palletsprojects.com/en/stable/, 2025. [Internet; descargado 8-enero-2026].

[19] Pallets Projects. Documentación de jinja. https://jinja.palletsprojects.com/en/stable/, 2025. [Internet; descargado 8-enero-2026].

[20] Pallets Projects. Flask blueprints. https://flask.palletsprojects.com/en/stable/blueprints/, 2025. [Internet; descargado 8-enero-2026].

[21] Pallets Projects. Flask patterns: Sqlalchemy. https://flask.palletsprojects.com/en/stable/patterns/sqlalchemy/, 2025. [Internet; descargado 8-enero-2026].

[22] Thomas Park. Bootswatch. https://bootswatch.com/, 2025. [Internet; descargado 8-enero-2026].

[23] Roger S. Pressman. *Ingeniería del software: un enfoque práctico*. McGraw-Hill, 2010.

[24] pytest Dev Team. pytest documentation: Capturing warnings. https://docs.pytest.org/en/stable/how-to/capture-warnings.html, 2025. [Internet; descargado 8-enero-2026].

[25] Real Python. Using flask blueprints to organize your application. https://realpython.com/flask-blueprint/, 2024. [Internet; descargado 8-enero-2026].

[26] Michael J. Rosen. Telerehabilitation technologies: Accessibility and usability. *International Journal of Telerehabilitation*, 1(1):7–22, 2009.

[27] Salesforce. Aplicación heroku terapitrack-tfg. https://terapitrack-tfg-dccabad31cfb.herokuapp.com/, 2025. [Internet; descargado 8-enero-2026].

[28] Salesforce. Heroku dashboard. https://dashboard.heroku.com/, 2025. [Internet; descargado 8-enero-2026].

[29] José Miguel Ramírez Sanz. Detección de poses con detectron2 para programas de telerehabilitación. https://github.com/Josemi/TFM-FIS-IA, 2021. Trabajo Fin de Máster. [Internet; descargado 8-enero-2026].

[30] Secure Medical. Disability and virtual healthcare: Accessibility features that matter. https://securemedical.com/telemedicine/disability-and-virtual-healthcare-accessibility-features-that-matter/, 2025. [Internet; descargado 8-enero-2026].

[31] A. Silva. Telerehabilitation for people with parkinson's disease: protocol of a randomized trial. *BMJ Open*, 2025.

[32] Sociedad Española de Neurología. Un 2 de parkinson. https://www.sen.es/saladeprensa/pdf/Link268.pdf, 2015. [Internet; descargado 14-enero-2026].

[33] SQLAlchemy Project. Documentación de sqlalchemy. https://www.sqlalchemy.org/, 2025. [Internet; descargado 8-enero-2026].

[34] World Health Organization. Parkinson disease. https://www.who.int/news-room/fact-sheets/detail/parkinson-disease, 2023. [Internet; descargado 8-enero-2026].

[35] World Wide Web Consortium. Web content accessibility guidelines (wcag) 2.2. https://www.w3.org/TR/WCAG22/, 2024. [Internet; descargado 8-enero-2026].

[36] Luis Ángel Espinosa Lafuente. Detección de ejercicios en vídeos de rehabilitación. https://github.com/fravian99/Deteccion-de-ejercicios-en-videos-de-rehabilitacion, 2022. Trabajo Fin de Grado, Universidad de Burgos. [Internet; descargado 8-enero-2026].