UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

**TFG del Grado en Ingeniería Informática**

# Herramienta de gestión para el seguimiento de pacientes con video

Presentado por Alberto Lanchares Diez
en Universidad de Burgos — 8 de enero 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 Herramienta de gestión para el seguimiento de pacientes con video.

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, 8 de enero de 2026

Vº. Bº. del Tutor:

D. José Luis Garrido Labrador

Vº. Bº. del co-tutor:

D. José Miguel Ramírez Sanz

## Resumen

Este Trabajo Fin de Grado presenta el diseño e implementación de TerapiTrack, una herramienta 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 de referencia.

El sistema permite a profesionales sanitarios crear ejercicios terapéuticos en vídeo, programar sesiones personalizadas y evaluar de forma remota el desempeño de los pacientes, mientras que estos pueden realizar los ejercicios en su domicilio, grabar su ejecución mediante la cámara del navegador y consultar su progreso a lo largo del tratamiento.

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

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, Flask, Python, SQLAlchemy, accesibilidad web, vídeo terapéutico, Cloudinary, aplicación web sanitaria.

**Abstract**

This Bachelor's Thesis presents the design and implementation of TerapiTrack, a web-based tool for managing telerehabilitation sessions aimed at Parkinson's disease patients living in rural areas or far from reference centers.

The system allows healthcare professionals to create therapeutic video exercises, schedule personalized sessions, and remotely evaluate patient performance, while patients can perform exercises at home, record their execution using the browser camera, and track their progress throughout treatment.

TerapiTrack has been developed using Python and Flask, employs SQLAlchemy with SQLite and PostgreSQL for data management, integrates Cloudinary for scalable video storage, and follows web accessibility guidelines to facilitate use by people with motor limitations or limited technological experience.

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

**Keywords**

Telerehabilitation, Parkinson's disease, Flask, Python, SQLAlchemy, web accessibility, therapeutic video, Cloudinary, healthcare web application.

# 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.

La enfermedad de Parkinson es 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. Las limitaciones de movilidad y la lejanía de los centros de referencia complican aún más el acceso a estas terapias, repercutiendo en el bienestar de los pacientes y en el esfuerzo de sus familias.

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. 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.

## 1.1. Contexto del problema

En España, esta problemática se acentúa especialmente en la conocida como “España vaciada”, donde una parte muy importante de la población

está formada por personas de edad avanzada. Éste 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. 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 sin duda dificulta mantener la regularidad necesaria en las terapias.

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. [5, 2, 34, 15, 31]. Dentro de este conjunto de proyectos, el 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.

Se ha desarrollado la base funcional de dicha aplicación, estructurada en tres paneles diferenciados para 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 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 evaluaciones asociadas. Por su parte, el paciente accede a un panel simplificado desde el que puede consultar sus próximas sesiones, ejecutar los

ejercicios guiados por vídeo y visualizar de forma clara su evolución a partir de las evaluaciones registradas.

Las principales funcionalidades que aporta la herramienta son:

- Gestionar usuarios y roles (administrador, profesional y paciente), controlando el acceso mediante autenticación y decoradores específicos por tipo de usuario.
- Permitir que los profesionales sanitarios creen ejercicios en vídeo, organicen sesiones personalizadas y las asignen a sus pacientes.
- Registrar la realización de las sesiones y asociar las grabaciones de los ejercicios al historial de cada paciente, facilitando su posterior evaluación.
- Presentar a cada paciente un entorno simplificado donde pueda consultar la planificación, realizar los ejercicios y revisar su progreso a lo largo del tiempo.

## 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).
- 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.
- La documentación técnica incluida en los anexos, donde se detallan la arquitectura del sistema, los modelos de datos y los controladores implementados.
- 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 y de 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.

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.

# 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.
- Facilitar la creación y programación de sesiones personalizadas de rehabilitación, 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 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.

## 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, especialmente pensada para pacientes con posibles dificultades motoras 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.

## 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.

# 3. Conceptos teóricos

En este capítulo se revisan los conceptos teóricos básicos que sirven de soporte al proyecto TerapiTrack. Primero se abordan nociones relacionadas con la enfermedad de Parkinson y la rehabilitación, y después se describen los principios de accesibilidad, los patrones de diseño y los modelos de datos 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 estrecho [32, 17].

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. 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, 32].

## 3.2. Terapia ocupacional y rehabilitación

La terapia ocupacional es una disciplina sanitaria que ayuda a las personas a participar de forma lo más independiente posible en las actividades significativas de su vida diaria, adaptando tanto las tareas como el entorno cuando existe una enfermedad, lesión o discapacidad [16, 14]. En pacientes con enfermedad de Parkinson, la terapia ocupacional trabaja aspectos como el vestido, la higiene personal, la movilidad en el hogar o la organización de rutinas, con el objetivo de prolongar la autonomía en las actividades básicas y sociales [14].

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 [10].

## 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 [28]. Dentro de este ámbito, la telerehabilitación se centra en el diseño, la supervisión y la evaluación de programas de rehabilitación mediante herramientas remotas, como videoconferencia o plataformas web específicas [28].

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 [10, 11].

## 3.4. Accesibilidad y usabilidad en aplicaciones sanitarias

Las aplicaciones orientadas a la salud deben diseñarse pensando en usuarios con diferentes capacidades motoras, sensoriales o cognitivas, y

también en personas de edad avanzada con poca experiencia en el uso de ordenadores o dispositivos móviles. Ni la edad ni el nivel de conocimiento tecnológico deberían convertirse en una barrera a la hora de acceder a programas de telerehabilitación [28, 12].

Los estándares de accesibilidad web, como las pautas WCAG, recogen cuatro principios generales que sirven de guía para el diseño de interfaces accesibles [33].

- **Perceptibilidad**: La información y los componentes de la interfaz deben presentarse de forma que puedan percibirse por distintos tipos de usuarios, por ejemplo cuidando el contraste de color, el tamaño de la tipografía o el uso de textos alternativos en imágenes [33].
- **Operabilidad**: Los controles deben poder manejarse con diferentes dispositivos de entrada (ratón, teclado, pantallas táctiles o mandos) y con un número reducido de acciones, algo especialmente relevante en personas con limitaciones motoras [28, 12].
- **Comprensibilidad**: Los textos, mensajes de error y flujos de navegación han de ser claros y coherentes, evitando tecnicismos innecesarios y sobrecarga de información [33].
- **Robustez**: El contenido debe mostrarse correctamente en distintos navegadores y dispositivos, y ser compatible con tecnologías de apoyo como lectores de pantalla o ampliadores de pantalla [33].

Estos principios han guiado el diseño de la interfaz de TerapiTrack, priorizando menús sencillos, textos claros y un número reducido de acciones por pantalla.

## 3.5. Patrones de diseño y arquitectura Modelo–Vista–Controlador

En el desarrollo de aplicaciones web es habitual apoyarse en patrones de diseño que separan responsabilidades y facilitan la evolución del sistema. Uno de los más extendidos es el patrón Modelo–Vista–Controlador (MVC), que organiza la aplicación en tres componentes principales [13]:

- **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 pueda evolucionar de manera relativamente independiente, algo especialmente útil en proyectos que pueden ampliarse en el futuro [13].

## 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 resulta adecuado para almacenar información clínica y de seguimiento, donde es importante mantener un historial coherente de usuarios, sesiones y evaluaciones.

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. Aplicar niveles adecuados de normalización ayuda a mantener la coherencia de los datos y a simplificar su mantenimiento a largo plazo [6].

Una **relación de muchos a muchos** aparece cuando varios registros de una tabla pueden asociarse con varios registros de otra. En estos casos se introduce una tabla intermedia que almacena los pares de elementos relacionados y permite gestionar de forma clara y trazable esas asociaciones (por ejemplo, la vinculación entre pacientes y profesionales o la asignación de ejercicios a sesiones) [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 [20].

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 [20].

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 de TerapiTrack. 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 descritas en los capítulos siguientes [8, 9].

## 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 [20]. 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 [20]. 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 microframework web **Flask** [23]. Flask 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 [23, 25].

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 [22, 27]. 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 [21]. 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 [24].

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 [21].

## 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 [30, 29].

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 [25].

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 y mensajes. 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 [20]. Sobre Bootstrap se han aplicado temas de **Bootswatch**, que proporcionan estilos alternativos listos para usar sin modificar el marcado HTML [19].

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 [33].

## 4.5. Testing y aseguramiento de la calidad

### Pytest

Las pruebas automáticas se han realizado con **Pytest**, un marco de testing para Python que permite definir casos de prueba como funciones y agruparlos en módulos dentro de la carpeta `tests` [26]. Pytest ofrece utilidades para preparar datos de prueba mediante *fixtures*, parametrizar pruebas y generar informes con el resultado de la ejecución [26].

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, 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) [30]. 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 [29].

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 [13]. 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 [9].

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**: 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**: 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 [18, 8].
- **Draw.io**: 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 descrito 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 |
|---|---|---|
| Flask | Backend | Lógica de servidor y gestión de rutas y vistas web. |
| SQLAlchemy | Base de datos | ORM para definir modelos y gestionar consultas sobre bases de datos relacionales. |
| SQLite / PostgreSQL | Base de datos | Motores relacionales usados respectivamente en desarrollo local y en despliegue en la nube. |
| DB Browser for SQLite | Base de datos | Cliente gráfico para inspección, consulta y edición de la base de datos SQLite. |
| Jinja | Plantillas | Motor de plantillas para generar dinámicamente las páginas HTML desde Flask. |
| HTML5 / JavaScript | Frontend | Estructura del contenido y lógica en el navegador para dotar de interactividad a la interfaz. |
| Bootstrap / Bootswatch | Frontend | Framework CSS y temas visuales para construir una interfaz adaptable y coherente. |
| Cloudinary | Almacenamiento | Servicio externo para almacenar y servir archivos multimedia generados por la aplicación. |
| Jira | Gestión | Planificación, seguimiento de tareas y organización en sprints. |
| Git | Versionado | Sistema de control de versiones distribuido para gestionar la evolución del código. |
| Git Bash | Versionado | Uso de Git desde la línea de comandos en el entorno local. |
| GitHub | Repositorio | Alojamiento remoto del código, sistema de *issues* y revisión de cambios. |
| Visual Studio Code | Desarrollo | Entorno integrado de edición, depuración y ejecución del código fuente. |
| LaTeX Workshop / Overleaf | Documentación | Herramientas para la redacción y compilación de la memoria y los anexos en LaTeX. |
| Heroku | Despliegue | Plataforma PaaS utilizada para ejecutar la aplicación Flask y la base de datos en la nube. |

Tabla 4.1: Herramientas utilizadas en el proyecto

# 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 [20].

Desde el principio se utilizó Jira para organizar el trabajo en tareas manejables. Al comienzo había simplemente una lista de tareas generales (por ejemplo, çrear 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 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 cerradas [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` [9]. 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 cobertura del 100 %, 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 integrará en `main` mediante un merge final justo antes de la entrega del proyecto, momento en el que el historial completo de desarrollo quedará reflejado en la rama principal y visible en el repositorio para futuras consultas o ampliaciones del sistema [9, 1].

## 5.2. Arquitectura y organización del código

La aplicación sigue una arquitectura basada en el patrón Modelo–Vista–Controlador adaptado a Flask, en el que la lógica de presentación, los datos y el control

de flujo se mantienen separados [13]. 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 se encuentra organizado en la carpeta `src`, que contiene varios subdirectorios:

- `modelos`: Define las entidades del dominio (Usuario, Paciente, Profesional, Ejercicio, Sesion, Evaluacion, VideoRespuesta) y las tablas de asociación para las relaciones de muchos a muchos.
- `controladores`: Agrupa las rutas y la lógica de negocio por rol (`auth_controlador.py`, `admin_controlador.py`, `profesional_controlador.py`, `paciente_controlador.py`).
- `vistas`: Contiene las plantillas Jinja organizadas en subcarpetas por rol (`admin`, `profesional`, `paciente`), todas ellas heredando de `base.html`.
- `static`: Almacena los recursos estáticos (CSS, imágenes, vídeos de ejemplo).

Además, en la raíz de `src` se encuentran ficheros de configuración y utilidades:

- `app.py`: Punto de entrada de la aplicación, donde se crea la instancia de Flask, se registran los blueprints y se configuran las extensiones.
- `extensiones.py`: Inicializa las extensiones (SQLAlchemy, Flask-Login, Flask-WTF, Bcrypt) de forma centralizada para reutilizarlas en todos los módulos.
- `config.py`: Define la configuración del entorno (base de datos, claves secretas, rutas de subida de archivos).
- `decoradores.py`: Contiene decoradores personalizados para comprobar roles y condiciones de acceso antes de ejecutar las vistas.
- `poblar_bd.py`: Script que inicializa la base de datos con datos de prueba (usuarios, pacientes, ejercicios, sesiones) para facilitar el desarrollo y las demostraciones.

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*, 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 [22, 27].

Por ejemplo, `profesional_controlador.py` define un blueprint que agrupa las rutas para crear ejercicios, asignar sesiones y evaluar el desempeño de los pacientes, mientras que `paciente_controlador.py` contiene las rutas para consultar sesiones asignadas, ejecutar ejercicios y ver el historial de evaluaciones. De esta forma, cada controlador puede evolucionar de manera relativamente independiente sin mezclar código de distintos roles en un mismo fichero.

### 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 en `decoradores.py` que comprueban el rol del usuario antes de ejecutar una vista.

Por ejemplo, el decorador `@admin_required` 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, los decoradores `@profesional_required` y `@paciente_required` aseguran que las rutas de cada controlador 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]. 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:

- **Paciente-Profesional**: Un paciente puede estar asignado a varios profesionales y un profesional puede atender a varios pacientes.
- **Ejercicio-Sesion**: Una sesión puede incluir varios ejercicios y un mismo ejercicio puede aparecer en distintas sesiones. La tabla intermedia `EjercicioSesion` también almacena el orden de los ejercicios dentro de la sesión.

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.

Durante el desarrollo se detectó que algunas consultas sobre el historial de sesiones de un paciente eran lentas, y la creación de un índice sobre la columna de fecha resolvió el problema sin necesidad de cambiar el modelo.

## 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.

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 el orden en que deben ejecutarse.

Al guardar la sesión, el sistema crea un registro en la tabla `Sesion` y registros asociados en la tabla intermedia `EjercicioSesion` para cada ejercicio incluido, almacenando también su posición dentro de la secuencia.

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 para el día actual. Al seleccionar una sesión, se muestra una página (`ejecutar_sesion.html`) que presenta los ejercicios en orden, con la posibilidad de ver el vídeo de ejemplo, grabar la ejecución propia del paciente y pasar al siguiente 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 se suben directamente desde el navegador del paciente 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 (por ejemplo, de 1 a 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 `EjercicioSesion`, 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.

## 5.5. 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 [28, 33].

### 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 "Mis sesiones", .Ejercicios disponiblesz "Mi perfil", 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) 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 pantalla táctil. 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) [33].

### Feedback visual claro

Durante la ejecución de una sesión, el sistema muestra de forma clara en qué ejercicio se encuentra el paciente y cuántos quedan pendientes. 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 ha resultado 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.

## 5.6. 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, consultas) y algunos de los flujos principales de los controladores [26].

La cobertura de estas pruebas se ha monitorizado mediante la herramienta `pytest-cov`, y en el momento de escribir esta memoria se ha alcanzado una cobertura del 100 % sobre los módulos más críticos (modelos, decoradores, configuración). 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.

### Pruebas manuales en entornos reales

Además de las pruebas automáticas, se han realizado pruebas manuales en el entorno de 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.

Estas pruebas han permitido detectar problemas que no eran evidentes en el entorno de desarrollo local, como tiempos de respuesta más largos al subir vídeos grandes o errores de permisos en rutas específicas. La corrección de estos problemas se ha registrado en Jira y se ha integrado en la rama `Pruebas` siguiendo el ciclo de desarrollo iterativo descrito en la primera sección de este capítulo.

## 5.7. Lecciones aprendidas y aspectos mejorables

El desarrollo de TerapiTrack ha sido una experiencia de aprendizaje continuo en la que se han ido descubriendo soluciones a medida que surgían los problemas.

Entre los aspectos que han funcionado bien destacan:

- La organización del código en blueprints y el uso de decoradores para control de acceso, que han simplificado el mantenimiento y la evolución del sistema.
- La decisión de normalizar el modelo de datos y utilizar tablas intermedias para las relaciones de muchos a muchos, que ha facilitado añadir nuevas funcionalidades sin tener que reestructurar la base de datos.
- La integración de Cloudinary para el almacenamiento de vídeos, que ha resuelto los problemas de escalabilidad y despliegue que surgían al guardar los archivos en local.

Entre los aspectos mejorables se encuentran:

- La gestión de ramas en Git podría haber sido más disciplinada desde el principio, realizando merges parciales más frecuentes entre `Pruebas` y `main` en lugar de acumular todos los cambios en una sola rama.
- La sincronización entre las tareas de Jira y los commits de GitHub no siempre ha sido perfecta, y algunas tareas completadas no tienen un commit asociado claramente identificable.
- La cobertura de pruebas podría ampliarse a los controladores completos y a los flujos de interacción entre vistas, aunque esto requeriría configurar un entorno de pruebas más complejo con navegadores automatizados (Selenium, Playwright).

A pesar de estas limitaciones, el resultado final es un sistema funcional que cumple los objetivos planteados al inicio del proyecto y que puede servir como base para futuras ampliaciones, como la integración de análisis automático de los vídeos mediante inteligencia artificial o la incorporación de funcionalidades de comunicación en tiempo real entre paciente y profesional.

# 6. Trabajos relacionados

El desarrollo de TerapiTrack se enmarca en un ámbito de investigación activo en el que se han realizado diversos trabajos previos centrados en la telerehabilitación, el análisis de ejercicios mediante vídeo y la aplicación de técnicas de inteligencia artificial para evaluar el desempeño de pacientes. 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.

## 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.

### Evaluación de ejercicios de rehabilitación en vídeo

El Trabajo Fin de Grado de Lucía Núñez Calvo abordó la evaluación automática de ejercicios de rehabilitación a partir de vídeos grabados por los pacientes [2]. 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.

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.

### Detección de ejercicios en vídeos de rehabilitación

El Trabajo 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 [34]. Este proyecto permitió validar que es posible clasificar ejercicios de forma fiable utilizando modelos entrenados sobre conjuntos de datos etiquetados.

TerapiTrack no implementa detección automática de ejercicios, pero el modelo de datos está preparado para almacenar esta información si en el futuro se decide integrar un módulo de clasificación que ayude al profesional a identificar qué ejercicios ha realizado el paciente cuando este graba varios seguidos sin indicar explícitamente cuál corresponde a cada uno.

### Infraestructura para telerehabilitación y análisis online

El Trabajo 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 [15]. 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.

### Detección de poses con Detectron2

El Trabajo 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 [31]. 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.

## 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.

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 [5]. El sistema utiliza redes neuronales convolucionales para analizar vídeos y clasificar la calidad de los movimientos, proporcionando feedback automático al paciente y al profesional.

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.

## 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.
- **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 [33, 28].

- **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.

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.

# 7. Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

# Bibliografía


[1] Atlassian. Jira software: proyecto terapitrack. https://terapitrack.atlassian.net/jira/your-work, 2025. Último acceso: 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. Último acceso: 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. Último acceso: 8-enero-2026.

[4] Cloudinary. Cloudinary console. https://console.cloudinary.com/, 2025. Último acceso: 8-enero-2026.

[5] Esther Cubo and Álvaro y otros García-Bustillo. 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. Último acceso: 8-enero-2026.

[8] Universidad de Burgos. Plantilla latex para trabajos fin de grado. https://github.com/ubutfgm/plantillaLatex/tree/master, 2015. Último acceso: 8-enero-2026.

[9] Alberto Lanchares Díez. Terapitrack: herramienta web para rehabilitación online. https://github.com/lanchares/TerapiTrack, 2025. Repositorio del proyecto. Último acceso: 8-enero-2026.

[10] Autor Ejemplo. Effect of telerehabilitation in parkinson disease: A systematic review and meta-analysis. *Movement Disorders*, 2025.

[11] Autor Ejemplo2. Telerehabilitation for people with parkinson's disease: protocol of a randomized trial. *Journal of Neurology*, 2025.

[12] Example Organization. Disability and virtual healthcare: Accessibility features that matter. https://securemedical.com/telemedicine/disability-and-virtual-healthcare-accessibility-features-that-matter/, 2025. Último acceso: 8-enero-2026.

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

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

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

[16] National Board for Certification in Occupational Therapy. Occupational therapy. https://www.nbcot.org/occupational-therapy, 2015. Último acceso: 8-enero-2026.

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

[18] Overleaf. Overleaf online latex editor. https://www.overleaf.com/project, 2025. Último acceso: 8-enero-2026.

[19] Thomas Park. Bootswatch. https://bootswatch.com/, 2025. Último acceso: 8-enero-2026.

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

[21] SQLAlchemy Project. Sqlalchemy. https://www.sqlalchemy.org/, 2025. Último acceso: 8-enero-2026.

[22] Pallets Projects. Flask blueprints. https://flask.palletsprojects.com/en/stable/blueprints/, 2025. Último acceso: 8-enero-2026.

[23] Pallets Projects. Flask documentation. https://flask.palletsprojects.com/en/stable/, 2025. Último acceso: 8-enero-2026.

[24] Pallets Projects. Flask patterns: Sqlalchemy. https://flask.palletsprojects.com/en/stable/patterns/sqlalchemy/#model-methods, 2025. Último acceso: 8-enero-2026.

[25] Pallets Projects. Jinja documentation. https://jinja.palletsprojects.com/en/stable/, 2025. Último acceso: 8-enero-2026.

[26] pytest Dev Team. pytest documentation: Capturing warnings. https://docs.pytest.org/en/stable/how-to/capture-warnings.html, 2025. Último acceso: 8-enero-2026.

[27] Real Python. Using flask blueprints to organize your application. https://realpython.com/flask-blueprint/, 2024. Último acceso: 8-enero-2026.

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

[29] Salesforce. Aplicación heroku `terapitrack-tfg`. https://terapitrack-tfg-dccabad31cfb.herokuapp.com/, 2025. Último acceso: 8-enero-2026.

[30] Salesforce. Heroku dashboard. https://dashboard.heroku.com/apps#, 2025. Último acceso: 8-enero-2026.

[31] 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. Último acceso: 8-enero-2026.

[32] World Health Organization. Parkinson disease. https://www.who.int/news-room/fact-sheets/detail/parkinson-disease, 2023. Último acceso: 8-enero-2026.

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

[34] 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. Último acceso: 8-enero-2026.