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
Documentación Técnica

Presentado por Alberto Lanchares Diez
en Universidad de Burgos — 8 de diciembre de 2025
Tutor: José Luis Garrido Labrador y José Miguel Ramírez Sanz

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

Este anexo describe el plan de proyecto seguido para el desarrollo de TerapiTrack, incluyendo la planificación temporal del trabajo y el estudio de viabilidad correspondiente. El propósito es mostrar de forma ordenada cómo se ha organizado el proceso de desarrollo para alcanzar los objetivos planteados y justificar que el proyecto resulta asumible desde el punto de vista económico y adecuado en términos legales, teniendo en cuenta que se tratan datos de carácter sanitario.

A.2. Planificación temporal

El desarrollo de *TerapiTrack* se gestionó mediante una metodología ágil, basada principalmente en Scrum, empleando Jira para la organización y seguimiento de tareas. Aunque la planificación inicial se estructuró en sprints, el avance práctico fue flexible y se adaptó constantemente al progreso real y a los retos técnicos del desarrollo individual. Por ello, la documentación en Jira refleja tanto la hoja de ruta como los reajustes efectuados en el proyecto.

El desarrollo de TerapiTrack se ha organizado siguiendo una metodología ágil inspirada en Scrum, utilizando Jira para planificar y hacer el seguimiento del trabajo. La planificación inicial se estructuró en sprints, pero el avance real se fue ajustando a medida que aparecían imprevistos técnicos y se concretaban mejor los requisitos. Por este motivo, el tablero de Jira refleja

tanto la planificación prevista como los cambios introducidos durante el proyecto.

Se ha dado prioridad a completar primero los módulos y tareas considerados esenciales, procurando mantener una entrega de funcionalidad lo más continua posible. En las páginas siguientes se describen los sprints realizados y se incluyen los gráficos de evolución extraídos de Jira, donde puede verse el detalle de las tareas planificadas, su fecha de finalización y la progresión del trabajo en cada iteración.

Aunque se ha utilizado la estructura de sprints para organizar el proyecto, la ejecución no ha seguido de forma estricta el calendario inicial. En varios casos la mayor parte del trabajo se concentró al final del sprint y algunos se cerraron más tarde de la fecha prevista. Esto se debe a que el desarrollo se ha realizado de manera individual, lo que ha obligado a reorganizar el tiempo disponible y a reagrupar tareas. Aun así, la planificación por sprints y el uso de Jira han resultado útiles para mantener una visión global del avance y de las prioridades en cada momento, por lo que los gráficos deben interpretarse como una guía del trabajo realizado más que como una aplicación estricta de Scrum.

Sprint 1: Configuración inicial (11–17 marzo 2025)

Este primer sprint se dedicó a poner en marcha el entorno de trabajo y a conocer las tecnologías básicas del proyecto. Se configuró el entorno con Flask, se creó el repositorio en GitHub y se definió una estructura inicial de carpetas y dependencias. Paralelamente se revisó la documentación de Flask y Jinja y se realizaron las primeras pruebas de conexión con SQLite para comprobar la persistencia de datos.[10, 9]

Quedaron pendientes tareas como el diseño completo de la base de datos y la documentación detallada de requisitos, que se trasladaron al sprint siguiente. El gráfico burndown muestra que la mayor parte del trabajo se concentró al final del periodo, algo lógico en esta fase inicial de familiarización con las herramientas.

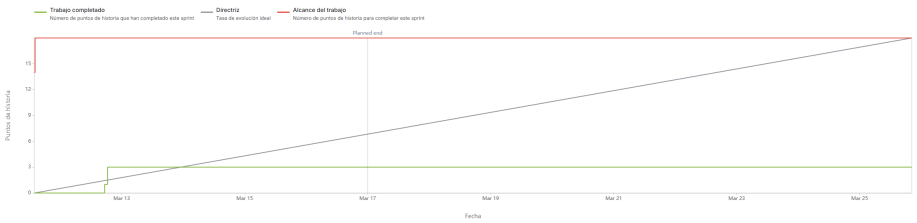

Figura A.1: Gráfico burndown del Sprint 1 obtenido de Jira.

Fecha	Evento	Actividad
Tue, Mar 11 2025, 1:53pm	Sprint iniciado	TFGTT-52 Estudiar documentación sobre Flask TFGTT-4 Configurar entorno Flask TFGTT-54 Estudiar documentación sobre SQL Alchemy TFGTT-53 Estudiar documentación sobre Jinja TFGTT-3 Configurar repositorio GitHub TFGTT-55 Probar conexión entre Flask y SQLite con DB-Browser TFGTT-57 Documentar requisitos funcionales y no funcionales
Tue, Mar 11 2025, 2:13pm	Añadida al sprint	TFGTT-5 Diseñar base de datos
Wed, Mar 12 2025, 5:21pm	Actividad completada	TFGTT-3 Configurar repositorio GitHub
Wed, Mar 12 2025, 6:29pm	Actividad completada	TFGTT-4 Configurar entorno Flask
Tue, Mar 25 2025, 8:23pm	Sprint completado	TFGTT-52 Estudiar documentación sobre Flask TFGTT-4 Configurar entorno Flask TFGTT-54 Estudiar documentación sobre SQL Alchemy TFGTT-53 Estudiar documentación sobre Jinja TFGTT-3 Configurar repositorio GitHub TFGTT-55 Probar conexión entre Flask y SQLite con DB-Browser TFGTT-57 Documentar requisitos funcionales y no funcionales TFGTT-5 Diseñar base de datos

Figura A.2: Resumen de tareas del Sprint 1 en Jira.

Sprint 2: Estudio (18–24 marzo 2025)

En el segundo sprint el esfuerzo se centró en profundizar en las tecnologías clave y en definir con más detalle el modelo de datos. Se completó el estudio práctico de Flask, SQLAlchemy y Jinja mediante ejemplos y pequeñas pruebas, se diseñó la base de datos y se elaboró el primer modelo entidad–relación.[10, 3, 9] También se validó la integración entre Flask y SQLite con pruebas de lectura y escritura, apoyándose en herramientas como DB Browser for SQLite para inspeccionar la base de datos.[4]

Algunas tareas de implementación de modelos en SQLAlchemy no llegaron a completarse y se arrastraron al sprint 3, donde se terminó de consolidar la capa de datos. El burndown refleja que las finalizaciones se concentran al final del sprint, más como registro de lo realizado que como una ejecución estricta del plan diario.

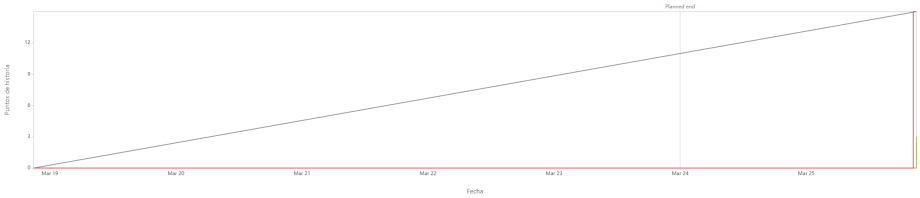

Figura A.3: Gráfico burndown del Sprint 2 obtenido de Jira.

Fecha	Evento	Actividad
Tue, Mar 18 2025, 8:55pm	Sprint iniciado	
Tue, Mar 25 2025, 8:23pm	Añadida al sprint	TFGTT-52 Estudiar documentación sobre Flask
Tue, Mar 25 2025, 8:23pm	Añadida al sprint	TFGTT-54 Estudiar documentación sobre SQL Alchemy
Tue, Mar 25 2025, 8:23pm	Añadida al sprint	TFGTT-5 Diseñar base de datos
Tue, Mar 25 2025, 8:23pm	Añadida al sprint	TFGTT-53 Estudiar documentación sobre Jinja
Tue, Mar 25 2025, 8:23pm	Añadida al sprint	TFGTT-55 Probar conexión entre Flask y SQLite con DB-Browser
Tue, Mar 25 2025, 8:23pm	Añadida al sprint	TFGTT-57 Documentar requisitos funcionales y no funcionales
Tue, Mar 25 2025, 8:24pm	Añadida al sprint	TFGTT-9 Diseñar modelo entidad-relación (ER) para usuarios.
Tue, Mar 25 2025, 8:31pm	Añadida al sprint	TFGTT-10 Implementar modelos SQLAlchemy para Usuario, Paciente, Profesional según diagrama relacional
Tue, Mar 25 2025, 8:58pm	Actividad completada	TFGTT-52 Estudiar documentación sobre Flask
Tue, Mar 25 2025, 8:58pm	Sprint completado	TFGTT-52 Estudiar documentación sobre Flask TFGTT-54 Estudiar documentación sobre SQL Alchemy TFGTT-5 Diseñar base de datos TFGTT-53 Estudiar documentación sobre Jinja TFGTT-55 Probar conexión entre Flask y SQLite con DB-Browser TFGTT-57 Documentar requisitos funcionales y no funcionales TFGTT-9 Diseñar modelo entidad-relación (ER) para usuarios. TFGTT-10 Implementar modelos SQLAlchemy para Usuario, Paciente, Profesional según diagrama relacional

Figura A.4: Resumen de tareas del Sprint 2 en Jira.

Sprint 3: Diseño (25–31 marzo 2025)

El tercer sprint se orientó al diseño detallado del sistema y a la implementación de los primeros modelos completos. Se ampliaron y ajustaron las tablas de la base de datos, se implementaron los modelos principales (usuario, paciente, profesional) en SQLAlchemy y se comprobó la estabilidad de la integración con SQLite.[3] Además, se avanzó en el diseño de las plantillas de Jinja y se revisó la documentación de requisitos para alinearla con las decisiones tomadas.[9]

La creación de un sistema de permisos basado en Flask-Login quedó únicamente esbozada en este sprint y se abordó con más profundidad en los siguientes.[2] El gráfico burndown muestra cómo gran parte de los puntos de historia se cierran hacia el final, cuando se integran y prueban los distintos elementos de diseño.

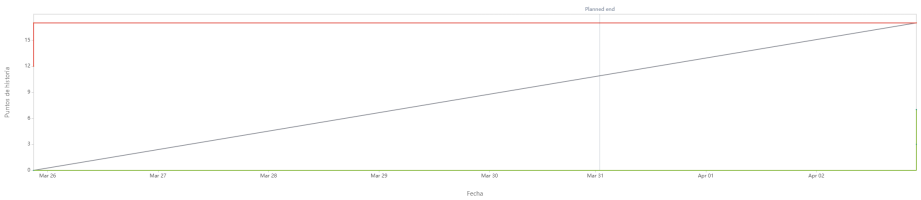

Figura A.5: Gráfico burndown del Sprint 3 obtenido de Jira.

Fecha	Evento	Actividad
Tue, Mar 25 2025, 8:59pm	Sprint iniciado	TFGTT-54 Estudiar documentación sobre SQL Alchemy TFGTT-10 Implementar modelos SQLAlchemy para Usuario, Paciente, Profesional según diagrama relacional TFGTT-5 Diseñar base de datos TFGTT-53 Estudiar documentación sobre Jinja TFGTT-55 Probar conexión entre Flask y SQLite con DB-Browser TFGTT-9 Diseñar modelo entidad-relación (ER) para usuarios. TFGTT-57 Documentar requisitos funcionales y no funcionales
Tue, Mar 25 2025, 8:59pm	Añadida al sprint	TFGTT-6 Crear sistema de permisos por roles usando Flask-Login
Wed, Apr 02 2025, 9:43pm	Actividad completada	TFGTT-53 Estudiar documentación sobre Jinja
Wed, Apr 02 2025, 9:43pm	Actividad completada	TFGTT-5 Diseñar base de datos
Wed, Apr 02 2025, 9:43pm	Sprint completado	TFGTT-54 Estudiar documentación sobre SQL Alchemy TFGTT-10 Implementar modelos SQLAlchemy para Usuario, Paciente, Profesional según diagrama relacional TFGTT-5 Diseñar base de datos TFGTT-53 Estudiar documentación sobre Jinja TFGTT-55 Probar conexión entre Flask y SQLite con DB-Browser TFGTT-9 Diseñar modelo entidad-relación (ER) para usuarios. TFGTT-57 Documentar requisitos funcionales y no funcionales TFGTT-6 Crear sistema de permisos por roles usando Flask-Login

Figura A.6: Resumen de tareas del Sprint 3 en Jira.

Sprint 4: Base de datos (2–7 abril 2025)

En el sprint 4 se consolidó el trabajo previo sobre la base de datos y se empezó a reforzar la capa de seguridad y pruebas. Se completó la definición del modelo relacional, se ajustaron las relaciones entre entidades y se mejoró la integración con SQLAlchemy.[3] También se avanzó en la definición de permisos por rol y se continuó actualizando la documentación funcional y técnica.

Las pruebas unitarias sobre los modelos y la redacción sistemática de la memoria en LaTeX comenzaron en este sprint pero se fueron extendiendo a los posteriores.

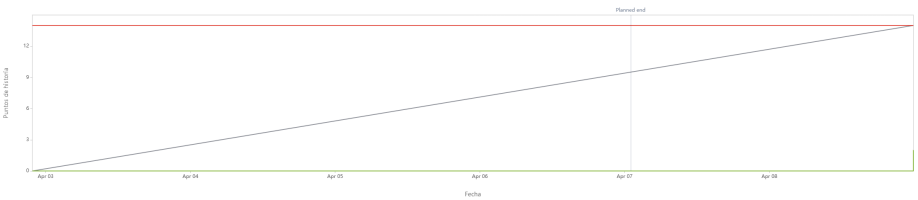

Figura A.7: Gráfico burndown del Sprint 4 obtenido de Jira.

Fecha	Evento	Actividad
Wed, Apr 02 2025, 9:47pm	Sprint iniciado	TFGTT-6 Crear sistema de permisos por roles usando Flask-Login TFGTT-54 Estudiar documentación sobre SQL Alchemy TFGTT-10 Implementar modelos SQLAlchemy para Usuario, Paciente, Profesional según diagrama relacional TFGTT-55 Probar conexión entre Flask y SQLite con DB-Browser TFGTT-9 Diseñar modelo entidad-relación (ER) para usuarios. TFGTT-57 Documentar requisitos funcionales y no funcionales TFGTT-58 Implementar tests unitarios para modelos SQLAlchemy TFGTT-56 Crear memoria del TFG usando LaTeX en Overleaf
Tue, Apr 08 2025, 11:52pm	Actividad completada	TFGTT-9 Diseñar modelo entidad-relación (ER) para usuarios.
Tue, Apr 08 2025, 11:52pm	Actividad completada	TFGTT-54 Estudiar documentación sobre SQL Alchemy
Tue, Apr 08 2025, 11:53pm	Sprint completado	TFGTT-6 Crear sistema de permisos por roles usando Flask-Login TFGTT-54 Estudiar documentación sobre SQL Alchemy TFGTT-10 Implementar modelos SQLAlchemy para Usuario, Paciente, Profesional según diagrama relacional TFGTT-55 Probar conexión entre Flask y SQLite con DB-Browser TFGTT-9 Diseñar modelo entidad-relación (ER) para usuarios. TFGTT-57 Documentar requisitos funcionales y no funcionales TFGTT-58 Implementar tests unitarios para modelos SQLAlchemy TFGTT-56 Crear memoria del TFG usando LaTeX en Overleaf

Figura A.8: Resumen de tareas del Sprint 4 en Jira.

Sprint 5: Diseño de la interfaz (8–13 abril 2025)

El quinto sprint se centró en la parte visible de la aplicación. Se diseñó la interfaz de administración de usuarios, se trabajó en los formularios y en la validación básica de datos y se fueron integrando estos elementos con la capa de modelos existente.[10, 2] También se revisaron los requisitos funcionales para asegurarse de que la interfaz cubría los flujos principales de gestión.

Algunas tareas de pruebas unitarias y de depuración más exhaustiva quedaron pendientes y se trasladaron a sprints posteriores.

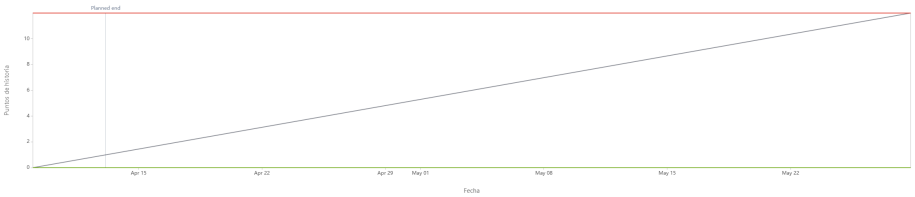

Figura A.9: Gráfico burndown del Sprint 5 obtenido de Jira.

Fecha	Evento	Actividad
Tue, Apr 08 2025, 11:55pm	Sprint iniciado	TFGTT-6 Crear sistema de permisos por roles usando Flask-Login TFGTT-55 Probar conexión entre Flask y SQLite con DB-Browser TFGTT-58 Implementar tests unitarios para modelos SQLAlchemy TFGTT-57 Documentar requisitos funcionales y no funcionales TFGTT-10 Implementar modelos SQLAlchemy para Usuario, Paciente, Profesional según diagrama relacional TFGTT-56 Crear memoria del TFG usando LaTeX en Overleaf TFGTT-12 Diseñar interfaz de administración de usuarios con formularios validados
Wed, May 28 2025, 7:36pm	Sprint completado	TFGTT-6 Crear sistema de permisos por roles usando Flask-Login TFGTT-55 Probar conexión entre Flask y SQLite con DB-Browser TFGTT-58 Implementar tests unitarios para modelos SQLAlchemy TFGTT-57 Documentar requisitos funcionales y no funcionales TFGTT-10 Implementar modelos SQLAlchemy para Usuario, Paciente, Profesional según diagrama relacional TFGTT-56 Crear memoria del TFG usando LaTeX en Overleaf TFGTT-12 Diseñar interfaz de administración de usuarios con formularios validados

Figura A.10: Resumen de tareas del Sprint 5 en Jira.

Sprint 6: Base funcional (3–9 junio 2025)

Tras una pausa en el calendario académico, el sprint 6 se dedicó a dotar al sistema de la base funcional completa. Se implementaron los modelos relacionados con ejercicios, sesiones, evaluaciones y la relación paciente–profesional, y se definieron los flujos principales de trabajo (asignación de ejercicios, registro de sesiones, evaluación, etc.). Además, se incorporó el cifrado de contraseñas con *bcrypt* y se reforzó la autenticación de usuarios.[2]

Quedaron para más adelante algunos aspectos de la interfaz de administración y parte de las pruebas automáticas. El burndown de este sprint muestra un incremento importante de trabajo completado en pocos días, reflejando que muchas tareas se habían preparado en los sprints anteriores y se integraron en este bloque.

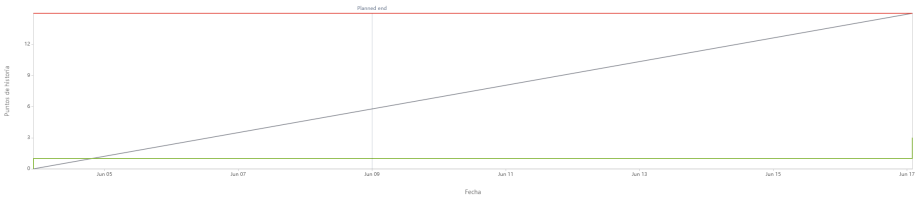

Figura A.11: Gráfico burndown del Sprint 6 obtenido de Jira.

Fecha	Evento	Actividad
Tue, Jun 03 2025, 10:29pm	Sprint iniciado	TFGTT-10 Implementar modelos SQLAlchemy para Usuario, Paciente, Profesional según diagrama relacional TFGTT-55 Probar conexión entre Flask y SQLite con DB-Browser TFGTT-57 Documentar requisitos funcionales y no funcionales TFGTT-102 Crear diagrama relacional TFGTT-101 Crear diagrama entidad-relacion TFGTT-34 Implementar modelo Ejercicio con campos (nombre, descripción, tipo, video, duración) TFGTT-23 Implementar modelo Evaluacion con puntuacion, comentarios y fecha_evaluacion TFGTT-32 Implementar modelo Ejercicio_Sesion para duración específica por ejercicio TFGTT-89 Crear modelo Paciente_Profesional para tabla de vinculación TFGTT-14 Crear modelo Sesion con relaciones a Paciente y estado (PENDIENTE/COMPLETADA/CANCELADA) TFGTT-103 Crear modelo Video_Respuesta con ruta_almacenamiento y fecha_expiracion TFGTT-90 Implementar cifrado de contraseñas con bcrypt TFGTT-6 Crear sistema de permisos por roles usando Flask-Login TFGTT-88 Implementar gestión de sesiones Flask con tiempo de expiración TFGTT-92 Configurar protección CSRF en formularios Flask TFGTT-87 Crear rutas para gestión de roles (/usuarios/<id>/rol, /usuarios/<id>/estado) TFGTT-18 Implementar rutas CRUD para usuarios (/usuarios, /usuarios/<id>, POST, PUT, DELETE) TFGTT-35 Crear rutas CRUD para evaluaciones (/evaluaciones, /evaluaciones/<id>) TFGTT-28 Implementar rutas CRUD para sesiones (/sesiones, /sesiones/<id>) TFGTT-47 Implementar rutas CRUD para ejercicios (/ejercicios, /ejercicios/<id>) TFGTT-7 Crear diagramas UML TFGTT-58 Implementar tests unitarios para modelos SQLAlchemy TFGTT-12 Diseñar interfaz de administración de usuarios con formularios validados TFGTT-48 Crear template base con navegación coherente

Figura A.12: Resumen de tareas del Sprint 6 (vista 1) en Jira.

Tue, Jun 03 2025, 10:29pm	Actividad completada	TFGTT-57 Documentar requisitos funcionales y no funcionales
Tue, Jun 03 2025, 10:30pm	Actividad completada	TFGTT-101 Crear diagrama entidad-relacion
Tue, Jun 03 2025, 10:30pm	Actividad completada	TFGTT-102 Crear diagrama relacional
Tue, Jun 03 2025, 10:34pm	Añadida al sprint	TFGTT-73 Creación de rutinas personalizadas de ejercicios
Tue, Jun 03 2025, 10:34pm	Añadida al sprint	TFGTT-70 Subida de vídeos de ejercicios por profesionales sanitarios
Tue, Jun 03 2025, 10:34pm	Añadida al sprint	TFGTT-65 Alta de usuarios con roles específicos (admin, paciente, profesional)
Tue, Jun 03 2025, 10:34pm	Añadida al sprint	TFGTT-68 Autenticación segura de usuarios por email y contraseña
Tue, Jun 17 2025, 1:26am	Actividad completada	TFGTT-10 Implementar modelos SQLAlchemy para Usuario, Paciente, Profesional según diagrama relacional
Tue, Jun 17 2025, 1:33am	Actividad completada	TFGTT-34 Implementar modelo Ejercicio con campos (nombre, descripción, tipo, video, duración)
Tue, Jun 17 2025, 1:34am	Actividad completada	TFGTT-32 Implementar modelo Ejercicio_Sesion para duración específica por ejercicio
Tue, Jun 17 2025, 1:34am	Actividad completada	TFGTT-14 Crear modelo Sesion con relaciones a Paciente y estado (PENDIENTE/COMPLETADA/CANCELADA)
Tue, Jun 17 2025, 1:36am	Actividad completada	TFGTT-103 Crear modelo Video_Respuesta con ruta_almacenamiento y fecha_expiracion
Tue, Jun 17 2025, 1:36am	Actividad completada	TFGTT-23 Implementar modelo Evaluacion con puntuacion, comentarios y fecha_evaluacion
Tue, Jun 17 2025, 1:37am	Actividad completada	TFGTT-89 Crear modelo Paciente_Profesional para tabla de vinculación
Tue, Jun 17 2025, 1:55am	Actividad completada	TFGTT-55 Probar conexión entre Flask y SQLite con DB-Browser
Tue, Jun 17 2025, 1:58am	Sprint completado	TFGTT-10 Implementar modelos SQLAlchemy para Usuario, Paciente, Profesional según diagrama relacional TFGTT-55 Probar conexión entre Flask y SQLite con DB-Browser TFGTT-57 Documentar requisitos funcionales y no funcionales TFGTT-102 Crear diagrama relacional TFGTT-101 Crear diagrama entidad-relacion TFGTT-34 Implementar modelo Ejercicio con campos (nombre, descripción, tipo, video, duración) TFGTT-23 Implementar modelo Evaluacion con puntuacion, comentarios y fecha_evaluacion TFGTT-32 Implementar modelo Ejercicio_Sesion para duración específica por ejercicio TFGTT-89 Crear modelo Paciente_Profesional para tabla de vinculación TFGTT-14 Crear modelo Sesion con relaciones a Paciente y estado (PENDIENTE/COMPLETADA/CANCELADA) TFGTT-103 Crear modelo Video_Respuesta con ruta_almacenamiento y fecha_expiracion TFGTT-90 Implementar cifrado de contraseñas con bcrypt TFGTT-6 Crear sistema de permisos por roles usando Flask-Login TFGTT-88 Implementar gestión de sesiones Flask con tiempo de expiración TFGTT-92 Configurar protección CSRF en formularios Flask TFGTT-87 Crear rutas para gestión de roles (/usuarios/<id>/rol, /usuarios/<id>/estado) TFGTT-18 Implementar rutas CRUD para usuarios (/usuarios, /usuarios/<id>, POST, PUT, DELETE) TFGTT-35 Crear rutas CRUD para evaluaciones (/evaluaciones, /evaluaciones/<id>) TFGTT-28 Implementar rutas CRUD para sesiones (/sesiones, /sesiones/<id>) TFGTT-47 Implementar rutas CRUD para ejercicios (/ejercicios, /ejercicios/<id>) TFGTT-7 Crear diagramas UML TFGTT-58 Implementar tests unitarios para modelos SQLAlchemy TFGTT-12 Diseñar interfaz de administración de usuarios con formularios validados TFGTT-48 Crear template base con navegación coherente TFGTT-73 Creación de rutinas personalizadas de ejercicios TFGTT-70 Subida de vídeos de ejercicios por profesionales sanitarios TFGTT-65 Alta de usuarios con roles específicos (admin, paciente, profesional) TFGTT-68 Autenticación segura de usuarios por email y contraseña

Figura A.13: Resumen de tareas del Sprint 6 (vista 2) en Jira.

Sprint 7: Base funcional 2 (17–22 junio 2025)

En el sprint 7 se completaron y pulieron muchas de las funcionalidades introducidas en el sprint anterior. Se reforzó la seguridad (protección CSRF, gestión avanzada de sesiones), se implementaron rutas CRUD para usuarios, sesiones, ejercicios y evaluaciones, y se añadieron características clave como la subida de vídeos por parte de los profesionales y la generación de diagramas UML de apoyo.[2, 8] También se avanzó en la documentación de casos de uso y en la preparación de material para la memoria.

Algunas pruebas unitarias siguieron abiertas y se terminaron de consolidar en la fase final del proyecto.

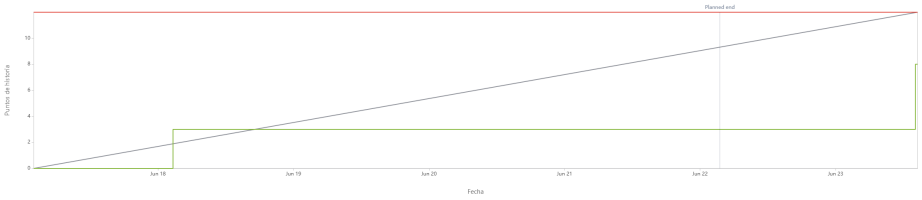

Figura A.14: Gráfico burndown del Sprint 7 obtenido de Jira.

Fecha	Evento	Actividad
Tue, Jun 17 2025, 1:59am	Sprint iniciado	TFGTT-90 Implementar cifrado de contraseñas con bcrypt TFGTT-6 Crear sistema de permisos por roles usando Flask-Login TFGTT-73 Creación de rutinas personalizadas de ejercicios TFGTT-7 Crear diagramas UML TFGTT-87 Crear rutas para gestión de roles (/usuarios/<id>/rol, /usuarios/<id>/estado) TFGTT-65 Alta de usuarios con roles específicos (admin, paciente, profesional) TFGTT-92 Configurar protección CSRF en formularios Flask TFGTT-70 Subida de vídeos de ejercicios por profesionales sanitarios TFGTT-48 Crear template base con navegación coherente TFGTT-18 Implementar rutas CRUD para usuarios (/usuarios, /usuarios/<id>, POST, PUT, DELETE) TFGTT-28 Implementar rutas CRUD para sesiones (/sesiones, /sesiones/<id>) TFGTT-12 Diseñar interfaz de administración de usuarios con formularios validados TFGTT-88 Implementar gestión de sesiones Flask con tiempo de expiración TFGTT-58 Implementar tests unitarios para modelos SQLAlchemy TFGTT-47 Implementar rutas CRUD para ejercicios (/ejercicios, /ejercicios/<id>) TFGTT-35 Crear rutas CRUD para evaluaciones (/evaluaciones, /evaluaciones/<id>) TFGTT-68 Autenticación segura de usuarios por email y contraseña TFGTT-104 Creación del diccionario de datos
Tue, Jun 17 2025, 2:01am	Actividad completada	TFGTT-104 Creación del diccionario de datos
Tue, Jun 17 2025, 8:15pm	Añadida al sprint	TFGTT-105 Documentar casos de uso
Wed, Jun 18 2025, 2:40am	Actividad completada	TFGTT-58 Implementar tests unitarios para modelos SQLAlchemy
Wed, Jun 18 2025, 2:40am	Actividad completada	TFGTT-105 Documentar casos de uso

Figura A.15: Resumen de tareas del Sprint 7 (vista 1) en Jira.

Mon, Jun 23 2025, 2:08pm	Actividad completada	TFGTT-6 Crear sistema de permisos por roles usando Flask-Login
Mon, Jun 23 2025, 2:08pm	Actividad completada	TFGTT-90 Implementar cifrado de contraseñas con bcrypt
Mon, Jun 23 2025, 2:31pm	Sprint completado	TFGTT-90 Implementar cifrado de contraseñas con bcrypt TFGTT-6 Crear sistema de permisos por roles usando Flask-Login TFGTT-73 Creación de rutinas personalizadas de ejercicios TFGTT-7 Crear diagramas UML TFGTT-87 Crear rutas para gestión de roles (/usuarios/<id>/rol, /usuarios/<id>/estado) TFGTT-65 Alta de usuarios con roles específicos (admin, paciente, profesional) TFGTT-92 Configurar protección CSRF en formularios Flask TFGTT-70 Subida de vídeos de ejercicios por profesionales sanitarios TFGTT-48 Crear template base con navegación coherente TFGTT-18 Implementar rutas CRUD para usuarios (/usuarios, /usuarios/<id>, POST, PUT, DELETE) TFGTT-28 Implementar rutas CRUD para sesiones (/sesiones, /sesiones/<id>) TFGTT-12 Diseñar interfaz de administración de usuarios con formularios validados TFGTT-88 Implementar gestión de sesiones Flask con tiempo de expiración TFGTT-58 Implementar tests unitarios para modelos SQLAlchemy TFGTT-47 Implementar rutas CRUD para ejercicios (/ejercicios, /ejercicios/<id>) TFGTT-35 Crear rutas CRUD para evaluaciones (/evaluaciones, /evaluaciones/<id>) TFGTT-68 Autenticación segura de usuarios por email y contraseña TFGTT-104 Creación del diccionario de datos TFGTT-105 Documentar casos de uso

Figura A.16: Resumen de tareas del Sprint 7 (vista 2) en Jira.

Sprint 8: Controladores y vistas (23–29 junio 2025)

En este sprint se continuó cerrando la integración entre controladores y vistas y se añadieron funcionalidades de apoyo a la explotación diaria de la herramienta. Se completaron los paneles de gestión para profesionales y pacientes, se refinaron los decoradores de autorización, se incorporó lógica de notificaciones y se implementó el borrado automático de vídeos pasados ciertos plazos para controlar el espacio de almacenamiento.[8, 11] Paralelamente se realizaron ajustes en los formularios y en la validación de datos en cliente y servidor.

Durante este periodo también se avanzó en tareas de documentación técnica y en la elaboración del diccionario de datos, aunque parte de este trabajo se siguió afinando después fuera del marco de los sprints. Los gráficos de Jira muestran una evolución casi plana hasta las fechas cercanas al cierre,

en las que se registran la mayoría de los puntos completados, reflejando la concentración de integración y pruebas en los últimos días.

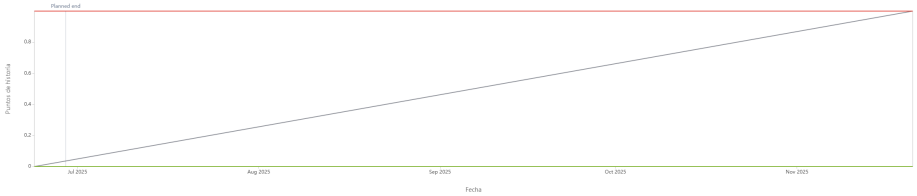

Figura A.17: Gráfico burndown del Sprint 8 obtenido de Jira.

Fecha	Evento	Actividad
Mon, Jun 23 2025, 3:18pm	Sprint iniciado	TFGTT-73 Creación de rutinas personalizadas de ejercicios TFGTT-92 Configurar protección CSRF en formularios Flask TFGTT-18 Implementar rutas CRUD para usuarios (/usuarios, /usuarios/<id>, POST, PUT, DELETE) TFGTT-12 Diseñar interfaz de administración de usuarios con formularios validados TFGTT-35 Crear rutas CRUD para evaluaciones (/evaluaciones, /evaluaciones/<id>) TFGTT-65 Alta de usuarios con roles específicos (admin, paciente, profesional) TFGTT-87 Crear rutas para gestión de roles (/usuarios/<id>/rol, /usuarios/<id>/estado) TFGTT-48 Crear template base con navegación coherente TFGTT-28 Implementar rutas CRUD para sesiones (/sesiones, /sesiones/<id>) TFGTT-88 Implementar gestión de sesiones Flask con tiempo de expiración TFGTT-47 Implementar rutas CRUD para ejercicios (/ejercicios, /ejercicios/<id>) TFGTT-68 Autenticación segura de usuarios por email y contraseña TFGTT-17 Crear rutas para vinculación pacientes-profesionales (/asignaciones) TFGTT-20 Implementar interfaz de creación/asignación de sesiones TFGTT-93 Crear decoradores de autorización por rol TFGTT-15 Desarrollar lógica de eliminación automática de videos TFGTT-56 Crear memoria del TFG usando LaTeX en Overleaf TFGTT-67 Vinculación de pacientes con personal sanitario TFGTT-75 Acceso del paciente a sus rutinas programadas TFGTT-69 Gestión de sesiones de pacientes en dispositivos TFGTT-66 Activación/desactivación de cuentas de usuario y notificación TFGTT-29 Crear rutas para asignación de sesiones (/pacientes/<id>/sesiones) TFGTT-39 Crear panel de gestión de ejercicios para profesionales con subida de archivos TFGTT-91 Crear middleware de validación de entrada de datos TFGTT-86 Diseñar formularios accesibles con validación client-side

Figura A.18: Resumen de tareas del Sprint 8 (vista 1) en Jira.

Thu, Nov 20 2025, 7:00pm	Sprint completado	TFGTT-73 Creación de rutinas personalizadas de ejercicios TFGTT-92 Configurar protección CSRF en formularios Flask TFGTT-18 Implementar rutas CRUD para usuarios (/usuarios, /usuarios/<id>, POST, PUT, DELETE) TFGTT-12 Diseñar interfaz de administración de usuarios con formularios validados TFGTT-35 Crear rutas CRUD para evaluaciones (/evaluaciones, /evaluaciones/<id>) TFGTT-65 Alta de usuarios con roles específicos (admin, paciente, profesional) TFGTT-87 Crear rutas para gestión de roles (/usuarios/<id>/rol, /usuarios/<id>/estado) TFGTT-48 Crear template base con navegación coherente TFGTT-28 Implementar rutas CRUD para sesiones (/sesiones, /sesiones/<id>) TFGTT-88 Implementar gestión de sesiones Flask con tiempo de expiración TFGTT-47 Implementar rutas CRUD para ejercicios (/ejercicios, /ejercicios/<id>) TFGTT-68 Autenticación segura de usuarios por email y contraseña TFGTT-17 Crear rutas para vinculación pacientes-profesionales (/asignaciones) TFGTT-20 Implementar interfaz de creación/asignación de sesiones TFGTT-93 Crear decoradores de autorización por rol TFGTT-15 Desarrollar lógica de eliminación automática de videos TFGTT-56 Crear memoria del TFG usando LaTeX en Overleaf TFGTT-67 Vinculación de pacientes con personal sanitario TFGTT-75 Acceso del paciente a sus rutinas programadas TFGTT-69 Gestión de sesiones de pacientes en dispositivos TFGTT-66 Activación/desactivación de cuentas de usuario y notificación TFGTT-29 Crear rutas para asignación de sesiones (/pacientes/<id>/sesiones) TFGTT-39 Crear panel de gestión de ejercicios para profesionales con subida de archivos TFGTT-91 Crear middleware de validación de entrada de datos TFGTT-86 Diseñar formularios accesibles con validación client-side

Figura A.19: Resumen de tareas del Sprint 8 (vista 2) en Jira.

A.3. Estudio de viabilidad

Antes de abordar los aspectos economicos y legales, en este apartado se valora si el proyecto *TerapiTrack* seria viable en un escenario real, suponiendo que su desarrollo y despliegue se encargan a una empresa de software a partir de los requisitos y del prototipo implementado en este Trabajo Fin de Grado.

Viabilidad economica

La viabilidad economica de *TerapiTrack* se ha planteado considerando un escenario en el que una empresa de desarrollo recibe el encargo de construir una solucion similar a la descrita en este TFG. En ese contexto, el sistema se concibe como una herramienta web para el seguimiento terapeutico remoto de pacientes que viven en entornos rurales o alejados de los hospitales de referencia, permitiendo que puedan realizar parte de su rehabilitacion desde casa con el apoyo de un mando tipo SNES y la supervision a distancia de profesionales sanitarios.

Tomando como referencia la planificacion temporal ya ejecutada y el alcance funcional previsto, puede estimarse que el desarrollo de una primera version completa de *TerapiTrack* requeriria alrededor de seis meses de trabajo de un desarrollador junior, con un esfuerzo equivalente a unas 400 horas efectivas dedicadas al proyecto. Esta estimacion incluye el analisis de requisitos, el diseno de modelos y controladores, la implementacion de vistas y formularios, la integracion con la base de datos, la logica asociada al mando, la configuracion de pruebas automatizadas y las tareas basicas de despliegue. Si se considera un coste medio en torno a 24 euros por hora para este tipo de perfil, dentro de los rangos habituales para desarrolladores web junior en Espana, el coste de personal para esta primera version se situaria aproximadamente en 9 600 euros.[1]

En cuanto a recursos materiales para el desarrollo, la empresa necesitara al menos un equipo de trabajo estandar (ordenador portatil o de sobremesa) valorado en torno a 1 000 euros. Suponiendo una amortizacion de tres anos y que el proyecto ocupa aproximadamente medio ano de dedicacion, el coste imputable de hardware a este desarrollo se situaria en torno a 170 euros. Ademas, para que los pacientes puedan utilizar la aplicacion en su domicilio se requiere que dispongan de un ordenador o portatil con navegador web y camara. Si se parte de portatiles de gama media-baja en torno a 400 euros por unidad y se considera un piloto inicial con dieciseis pacientes a los

que se les proporciona dicho equipo, el coste de estos terminales se situaria alrededor de 6 400 euros.

A estos equipos habria que sumar los mandos de juego necesarios para realizar los ejercicios. En el mercado existen packs de dos mandos USB estilo SNES con precios ajustados; por ejemplo, se puede encontrar un pack de dos mandos por 12,95 euros, lo que permite equipar a varios pacientes con un coste muy reducido por dispositivo.[6] En el escenario de dieciseis pacientes, bastaria con adquirir ocho packs de este tipo (dieciseis mandos en total), con un coste aproximado de 103,60 euros, una cifra asumible dentro del presupuesto global del proyecto.

Un aspecto que ayuda a contener los costes de desarrollo es el uso intensivo de tecnologias y herramientas de codigo abierto. El backend esta desarrollado en Python utilizando Flask para la gestion de rutas y controladores, mientras que la persistencia se resuelve mediante SQLAlchemy sobre una base de datos ligera como SQLite, lo que evita costes de licenciamiento en las primeras fases del proyecto.[10, 3] La interfaz web se apoya en plantillas Jinja, HTML y CSS, empleando Bootstrap y temas de Bootswatch para lograr una presentacion adaptada a las necesidades de accesibilidad de los pacientes.[9, 7] Durante el desarrollo se utilizan herramientas de soporte como Visual Studio Code, GitHub y plataformas colaborativas de edicion en sus modalidades gratuitas, de modo que no se generan costes directos asociados a licencias de software de desarrollo.

En lo relativo a costes recurrentes, los elementos mas relevantes serian la infraestructura de despliegue y el almacenamiento de los videos terapeuticos generados durante las sesiones. En una fase de pruebas o piloto, el sistema podria alojarse en planes de hosting compartido o en un pequeño servidor en la nube, suficientes para ejecutar la aplicacion Flask y la base de datos SQLite en un mismo entorno, con precios para proyectos pequenos que suelen situarse entre 2 y 15 euros al mes.[5] A esto habria que anadir el espacio necesario para almacenar los videos y un sistema basico de copias de seguridad periodicas de la base de datos y del contenido multimedia. Si se asume un escenario conservador de unos 50 euros mensuales para cubrir hosting, almacenamiento y copias de seguridad durante los ultimos meses de desarrollo y una fase piloto limitada, el coste de infraestructura para esta primera version se situaria en el entorno de 150–200 euros, en funcion de la duracion efectiva del despliegue inicial.

La tabla A.1 resume de forma orientativa los principales conceptos de coste considerados para una primera version profesional de *TerapiTrack* en un horizonte de unos seis meses y un piloto con dieciseis pacientes:

Concepto	Coste estimado
Coste de personal (400 h a 24 €/h)	9 600 €
Coste imputable de hardware de desarrollo	170 €
Dispositivos para pacientes (16 portatiles de 400 €)	6 400 €
Mandos SNES USB para pacientes (16 mandos)	103,60 €
Infraestructura y servicios (despliegue inicial)	150–200 €
Coste total aproximado primera version	16 423,60–16 473,60 €

Tabla A.1: Resumen orientativo de costes de la primera version de *TerapiTrack* en un escenario de seis meses de proyecto y piloto con dieciseis pacientes.

Combinando el coste de personal, la amortizacion de hardware, la provision de ordenadores y mandos para un grupo inicial de dieciseis pacientes y un presupuesto razonable de infraestructura para el despliegue inicial y una fase piloto, puede concluirse que el coste total de una primera version profesional de *TerapiTrack* se situaria en torno a 16 500 euros, sin incluir servicios posteriores de soporte continuado, mantenimiento evolutivo o integracion con otros sistemas clinicos.

Viabilidad legal

El desarrollo de *TerapiTrack* se ha llevado a cabo utilizando de forma casi exclusiva software libre y de código abierto, lo que ha permitido reducir significativamente los costes de licenciamiento y facilita tanto la explotación académica como un eventual despliegue profesional sin restricciones legales importantes.

El backend está construido sobre Python y el microframework Flask para la gestión de rutas y controladores, mientras que la persistencia de datos se resuelve con SQLAlchemy sobre SQLite.[10, 3] Estas tecnologías fundamentales se distribuyen bajo licencias permisivas de tipo BSD o MIT, que autorizan el uso, modificación y redistribución del software siempre que se mantengan los avisos de derechos de autor y las notas de licencia de los autores originales.

Flask se apoya en varias bibliotecas del ecosistema Pallets que se utilizan de forma indirecta a través del framework. Werkzeug proporciona la infraestructura WSGI de bajo nivel para gestionar peticiones y respuestas HTTP, cookies y depuración. Jinja2 actúa como motor de plantillas para la generación de vistas HTML. Click gestiona la interfaz de línea de comandos,

ItsDangerous se encarga de la firma segura de tokens, y MarkupSafe proporciona funciones de escape para prevenir inyecciones de código.[10, 9] Todas estas dependencias se distribuyen bajo licencias permisivas, por lo que su presencia en el proyecto no introduce restricciones adicionales.

Para el control de acceso de usuarios se emplean Flask-Login para la gestión de sesiones, Flask-WTF y WTForms para la validación de formularios, y bcrypt junto con Flask-Bcrypt para el cifrado robusto de contraseñas.[2] Estas herramientas se publican bajo licencias MIT o BSD, permitiendo su integración en proyectos propios sin obligación de publicar el código fuente, siempre que se conserven las atribuciones exigidas.

La interacción con el mando tipo SNES se resuelve mediante Pygame, una biblioteca de código abierto basada en SDL distribuida bajo licencia LGPL. Esta licencia permite el uso en aplicaciones propietarias siempre que la biblioteca se enlace de forma dinámica. En el caso de *TerapiTrack*, Pygame se utiliza sin modificar su código interno, únicamente para gestionar la comunicación con el mando, por lo que su integración es totalmente compatible con un despliegue profesional.

Además de las dependencias listadas, el proyecto utiliza herramientas de desarrollo como Visual Studio Code, Git, GitHub y pytest, todas ellas accesibles bajo modalidades gratuitas o de código abierto sin coste de licencia. El análisis conjunto de todas las dependencias y herramientas confirma que *TerapiTrack* es completamente viable desde el punto de vista legal, siempre que se respeten los avisos de licencia de terceros y se cumpla la normativa de protección de datos aplicable al tratamiento de información sanitaria.

La tabla A.2 recoge el análisis de licencias de todas las dependencias principales del proyecto.

Dependencia	**Versión**	**Licencia y uso**
Flask	3.0.0	BSD-3-Clause. Microframework web.
Werkzeug	3.1.4	BSD-3-Clause. Librería WSGI (uso indirecto vía Flask).
Jinja2	3.1.2	BSD. Motor de plantillas.
SQLAlchemy	2.0.23	MIT. ORM para BD relacional.
Flask-SQLAlchemy	3.1.1	BSD. Integración SQLAlchemy–Flask.
SQLite	(embebido)	Dominio público. BD embebida.
Flask-Login	0.6.3	MIT. Gestión de sesiones de usuario.
Flask-WTF	1.2.1	BSD-3-Clause. Formularios con CSRF.
WTForms	3.1.1	BSD-3-Clause. Validación de formularios.
bcrypt	5.0.0	Apache-2.0. Cifrado de contraseñas.
Flask-Bcrypt	1.0.1	BSD. Integración bcrypt–Flask.
Pygame	2.5.2	LGPL. Comunicación con el mando SNES.
Bootstrap	5.3.x	MIT. Framework CSS para la interfaz.
Bootswatch	(temas)	MIT. Colección de temas para Bootstrap.
Click	8.3.1	BSD. CLI interna de Flask.
ItsDangerous	2.2.0	BSD. Firma segura de tokens.
MarkupSafe	3.0.3	BSD. Escape de HTML en plantillas.
Blinker	1.9.0	MIT. Sistema de señales en Flask.
Greenlet	3.2.4	MIT. Soporte de contextos ligeros.
Colorama	0.4.6	BSD. Colores en la salida de terminal.
Typing-Extensions	4.15.0	PSF. Tipos adicionales para Python.

Tabla A.2: Análisis de dependencias y licencias de *TerapiTrack*.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este anexo se recogen los requisitos funcionales y no funcionales definidos para el desarrollo de *TerapiTrack*, así como la especificación detallada de los casos de uso principales del sistema.

Estos requisitos se elaboraron durante la fase de análisis y diseño del proyecto, a partir de los objetivos generales y del contexto de uso descritos en la memoria. A partir de ellos se ha ido tomando decisiones sobre la arquitectura y la implementación de la aplicación web, de manera que cada módulo desarrollado responde a necesidades previamente identificadas y no a decisiones improvisadas durante la programación.

B.2. Objetivos generales

El sistema persigue los siguientes objetivos principales:

- Facilitar el acceso a servicios de rehabilitación especializada para pacientes en entornos rurales.
- Ofrecer una plataforma accesible y fácil de usar para personas con dificultades motoras o cognitivas.
- Permitir el seguimiento remoto y personalizado por parte de profesionales sanitarios.
- Garantizar la seguridad y confidencialidad de los datos médicos.

- Mejorar el proceso de evaluación y seguimiento terapéutico mediante el uso de tecnologías digitales.

B.3. Catálogo de requisitos

Requisitos funcionales

RF1: Gestión de usuarios

- **RF1.1**: El administrador creará usuarios asignándoles roles específicos (administrador, paciente, médico, terapeuta, psicólogo, enfermero) mediante formulario validado.
- **RF1.2**: El administrador activará o desactivará cuentas mediante cambio de estado, enviando notificación al afectado.
- **RF1.3**: El administrador vinculará pacientes con profesionales sanitarios mediante interfaz de asignación directa.
- **RF1.4**: Los usuarios accederán mediante autenticación email/contraseña con sesiones temporales según rol.

RF2: Gestión de ejercicios

- **RF2.1**: Los profesionales cargarán vídeos de ejercicios optimizados para visualización estándar.
- **RF2.2**: Los pacientes visualizarán ejercicios filtrados según necesidades terapéuticas y condición médica.

RF3: Gestión de sesiones

- **RF3.1**: Los profesionales crearán sesiones seleccionando ejercicios y definiendo orden, duración y objetivos.
- **RF3.2**: Los profesionales asignarán sesiones a pacientes con fechas y horarios programados.
- **RF3.3**: Los pacientes accederán a sesiones programadas desde fecha actual hasta el mes siguiente.
- **RF3.4**: El sistema registrará automáticamente marcas temporales de cada ejercicio durante la sesión.

RF4: Seguimiento y evaluación

- **RF4.1**: El sistema grabará automáticamente la ejecución de ejercicios mediante cámara del dispositivo.
- **RF4.2**: El sistema almacenará vídeos con identificador único vinculado a sesión y ejercicio.
- **RF4.3**: Los profesionales reproducirán vídeos grabados desde panel de evaluación.
- **RF4.4**: Los profesionales evaluarán desempeño mediante puntuación (1–5) y comentarios.
- **RF4.5**: Los pacientes consultarán evaluaciones históricas con puntuaciones y comentarios.
- **RF4.6**: El sistema generará gráficos de progresión temporal para visualizar evolución.

RF5: Configuración del sistema

- **RF5.1**: El administrador establecerá políticas de retención de vídeos según necesidades institucionales.

RF6: Interfaz de usuario

- **RF6.1**: La interfaz funcionará con controles simplificados adaptados a dispositivos de entrada limitados.
- **RF6.2**: Los pacientes visualizarán simultáneamente vídeo ejemplo y grabación en pantalla dividida.

Requisitos no funcionales

RNF1: Usabilidad

- **RNF1.1**: Interfaz diseñada para personas con dificultades motoras, con elementos visuales accesibles.
- **RNF1.2**: Flujos optimizados para minimizar pasos en tareas principales.

- **RNF1.3**: Navegación coherente permitiendo identificar posición en la aplicación.

- **RNF1.4**: Sesión de acceso gestionada por personal autorizado, sin autenticación manual recurrente.

RNF2: Rendimiento

- **RNF2.1**: Gestión eficiente de archivos multimedia adaptada a conectividad habitual.

- **RNF2.2**: Respuesta ágil de páginas y funcionalidades bajo condiciones normales.

- **RNF2.3**: Comportamiento estable ante concurrencia de múltiples usuarios.

RNF3: Seguridad

- **RNF3.1**: Almacenamiento seguro de credenciales mediante cifrado adecuado.

- **RNF3.2**: Acceso restringido según roles garantizando confidencialidad de datos.

- **RNF3.3**: Comunicaciones protegidas mediante protocolos de cifrado estándar.

- **RNF3.4**: Mecanismos de auditoría para registrar accesos a información sensible.

RNF4: Disponibilidad

- **RNF4.1**: Accesibilidad en periodos necesarios para actividad asistencial.

- **RNF4.2**: Procedimientos de recuperación minimizando tiempo de indisponibilidad.

RNF5: Mantenibilidad

- **RNF5.1**: Arquitectura basada en separación de responsabilidades.
- **RNF5.2**: Registros de actividad para análisis y diagnóstico de incidencias.

B.4. Especificación de requisitos

Esta sección recoge los principales casos de uso del sistema y define cómo interactúan los actores con la aplicación para satisfacer los requisitos funcionales descritos en el catálogo anterior. Cada caso de uso se presenta de forma estructurada, indicando sus precondiciones, flujo de acciones, postcondiciones y relación con los requisitos asociados.

Actores del sistema

En el sistema *TerapiTrack* intervienen los siguientes actores principales:

- **Administrador**: Gestiona los usuarios de la plataforma, las configuraciones globales y las vinculaciones entre pacientes y profesionales.
- **Profesional sanitario**: Médico, terapeuta, psicólogo o enfermero responsable de planificar, programar y evaluar las sesiones terapéuticas de sus pacientes.
- **Paciente**: Usuario final que realiza los ejercicios prescritos, completa las sesiones programadas y consulta su evolución a lo largo del tratamiento.

Diagrama de casos de uso

En la Figura B.1 se representa de forma global cómo interactúan los actores con los principales casos de uso del sistema, agrupados por módulos funcionales.

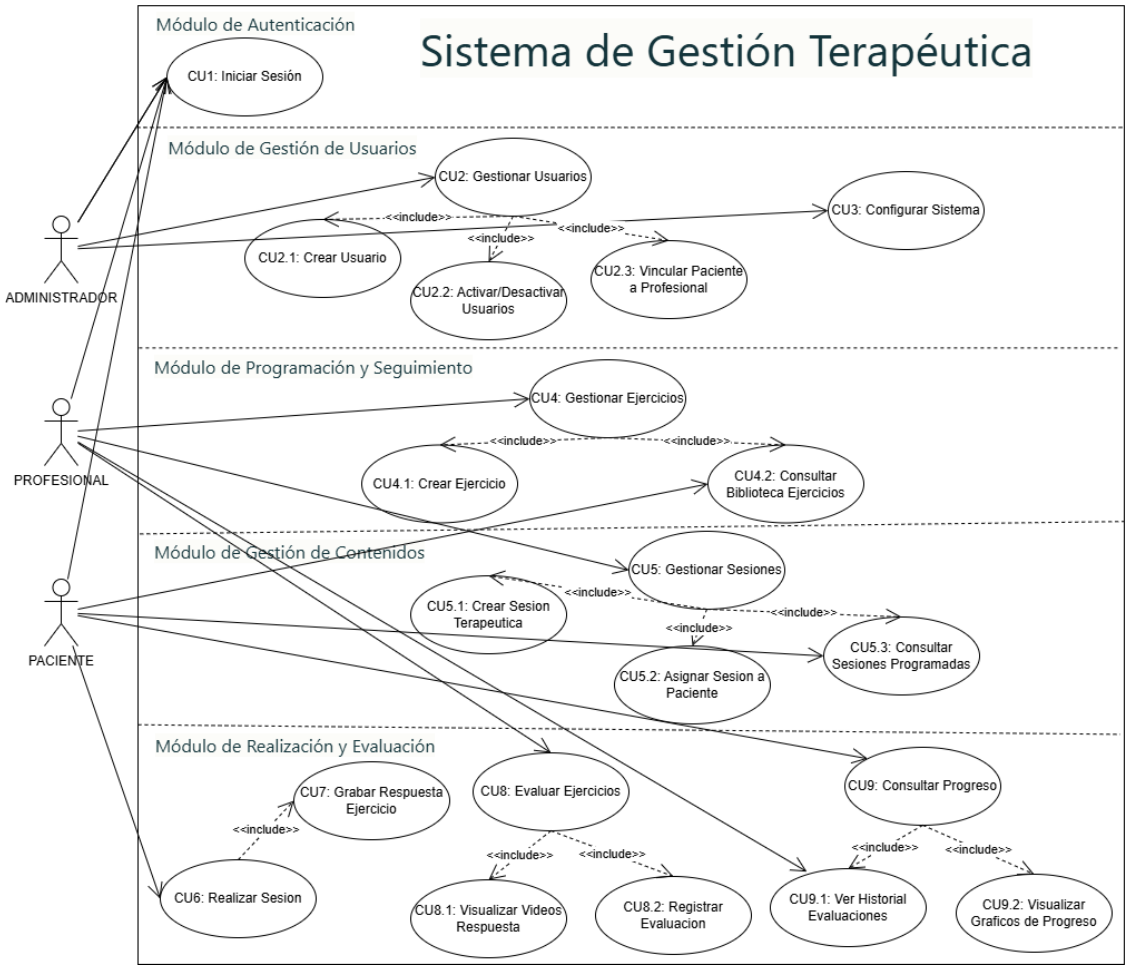

Figura B.1: Diagrama de casos de uso del sistema *TerapiTrack*.

CU-1	**Iniciar sesión**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF1.4
Descripción	Autenticación en el sistema según rol del usuario
Precondición	Usuario registrado y cuenta activa
Acciones	1. El usuario introduce correo y contraseña 2. El sistema valida las credenciales y estado de cuenta 3. El sistema redirige al panel correspondiente según rol
Postcondición	Usuario autenticado con permisos según su rol
Excepciones	Credenciales incorrectas, cuenta inactiva
Importancia	Alta

Tabla B.1: CU-1 Iniciar sesión.

CU-2.1	**Crear usuario**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF1.1
Descripción	Registrar nuevo usuario en el sistema
Precondición	Administrador autenticado (CU-1)
Acciones	1. Accede al formulario de alta 2. Introduce datos personales (nombre, apellidos, email) 3. Asigna contraseña 4. Selecciona rol (Administrador/Profesional/Paciente) 5. Si es Profesional: selecciona tipo y especialidad 6. Si es Paciente: registra fecha nacimiento y condición médica 7. El sistema valida los datos y crea el usuario
Postcondición	Usuario creado en estado activo
Excepciones	Datos inválidos: sistema muestra errores de validación
Importancia	Alta

Tabla B.2: CU-2.1 Crear usuario.

CU-2.2	**Activar/desactivar usuario**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF1.2
Descripción	Modificar el estado de acceso de un usuario
Precondición	Administrador autenticado (CU-1), usuario existente
Acciones	1. Busca y selecciona usuario 2. Modifica estado (activo/inactivo) 3. El sistema actualiza estado y envía notificación al usuario
Postcondición	Estado de usuario actualizado
Excepciones	Ninguna
Importancia	Media

Tabla B.3: CU-2.2 Activar/desactivar usuario.

CU-2.3	**Vincular paciente con profesional**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF1.3
Descripción	Establecer relación terapéutica entre profesional y paciente
Precondición	Administrador autenticado (CU-1), ambos usuarios activos
Acciones	1. Selecciona profesional 2. Selecciona paciente de la lista disponible 3. Confirma vinculación y asigna fecha de inicio 4. El sistema registra la relación en la base de datos
Postcondición	Relación establecida; el profesional puede gestionar al paciente
Excepciones	Ninguna
Importancia	Alta

Tabla B.4: CU-2.3 Vincular paciente con profesional.

CU-3	**Configurar sistema**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF5.1
Descripción	Definir parámetros de configuración global
Precondición	Administrador autenticado (CU-1)
Acciones	1. Accede al panel de configuración 2. Configura parámetros del sistema (política de retención de vídeos) 3. El sistema valida y guarda la configuración
Postcondición	Configuración actualizada
Excepciones	Ninguna
Importancia	Media

Tabla B.5: CU-3 Configurar sistema.

CU-4.1	**Crear ejercicio**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF2.1
Descripción	Añadir nuevo ejercicio al sistema
Precondición	Profesional autenticado (CU-1)
Acciones	1. Accede al formulario de creación 2. Sube vídeo demostrativo 3. Introduce nombre, descripción, tipo y duración 4. El sistema valida el formato y almacena el ejercicio
Postcondición	Ejercicio disponible en la biblioteca del profesional
Excepciones	Ninguna
Importancia	Alta

Tabla B.6: CU-4.1 Crear ejercicio.

CU-4.2	**Consultar biblioteca de ejercicios**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF2.2
Descripción	Visualizar ejercicios disponibles según perfil
Precondición	Usuario autenticado (CU-1)
Acciones	1. Profesional: filtra ejercicios por tipo, duración, condición médica 2. Profesional: ordena por popularidad o eficacia 3. Profesional: visualiza detalles y vídeos de cada ejercicio 4. Paciente: accede mediante navegación secuencial 5. Paciente: visualiza únicamente ejercicios de sus sesiones pasadas
Postcondición	Información del ejercicio accesible según permisos
Excepciones	Ninguna
Importancia	Media

Tabla B.7: CU-4.2 Consultar biblioteca de ejercicios.

CU-5.1	**Crear sesión terapéutica**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF3.1
Descripción	Definir conjunto de ejercicios secuenciales
Precondición	Profesional autenticado (CU-1), ejercicios disponibles
Acciones	1. Accede al creador de sesiones 2. Selecciona ejercicios de la biblioteca (CU-4.2) 3. Establece orden y objetivo terapéutico 4. El sistema valida y guarda la sesión
Postcondición	Sesión creada y lista para asignar
Excepciones	Ninguna
Importancia	Alta

Tabla B.8: CU-5.1 Crear sesión terapéutica.

CU-5.2	**Asignar sesión a paciente**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF3.2
Descripción	Programar sesión para un paciente específico
Precondición	Profesional autenticado (CU-1), paciente vinculado, sesión creada
Acciones	1. Selecciona paciente de su lista 2. Selecciona sesión creada previamente 3. Define fecha y hora programada 4. El sistema agenda la sesión y notifica al paciente
Postcondición	Sesión asignada con estado "Pendiente"
Excepciones	Ninguna
Importancia	Alta

Tabla B.9: CU-5.2 Asignar sesión a paciente.

CU-5.3	**Consultar sesiones programadas**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF3.3
Descripción	Visualizar calendario de sesiones
Precondición	Usuario autenticado (CU-1)
Acciones	1. Paciente: visualiza sus sesiones pendientes y completadas 2. Profesional: visualiza todas las sesiones asignadas a sus pacientes 3. Ambos pueden filtrar por estado (pendiente, completada, cancelada)
Postcondición	Información de sesiones disponible
Excepciones	Ninguna
Importancia	Media

Tabla B.10: CU-5.3 Consultar sesiones programadas.

CU-6	**Realizar sesión**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF3.4, RF6.1, RF6.2
Descripción	Completar sesión terapéutica con grabación
Precondición	Paciente autenticado (CU-1), sesión programada
Acciones	1. Selecciona sesión pendiente 2. El sistema muestra interfaz dividida (vídeo + cámara) 3. Inicia secuencia de ejercicios con controles 4. El sistema registra marcas temporales y graba automáticamente (CU-7) 5. Actualiza el estado de la sesión a Completada
Postcondición	Sesión completada, vídeos almacenados
Excepciones	Ninguna
Importancia	Alta

Tabla B.11: CU-6 Realizar sesión.

CU-7	**Grabar respuesta de ejercicio**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF4.1, RF4.2
Descripción	Almacenar vídeo de ejecución del ejercicio
Precondición	Sesión en progreso, cámara activa
Acciones	1. El sistema inicia grabación al comenzar la sesión 2. Registra fecha de grabación y calcula fecha de expiración 3. Al finalizar el ejercicio, detiene la grabación 4. Almacena el vídeo 5. Genera URL de almacenamiento
Postcondición	Video_Respuesta almacenado y accesible para evaluación
Excepciones	Ninguna
Importancia	Alta

Tabla B.12: CU-7 Grabar respuesta de ejercicio.

CU-8.1	**Visualizar vídeos de respuesta**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF4.3
Descripción	Revisar ejecución del paciente
Precondición	Profesional autenticado (CU-1), sesión completada
Acciones	1. Selecciona paciente y sesión 2. El sistema muestra lista de ejercicios grabados 3. Selecciona ejercicio específico 4. Reproduce vídeo con controles estándar
Postcondición	Vídeo visualizado, listo para evaluar
Excepciones	Ninguna
Importancia	Alta

Tabla B.13: CU-8.1 Visualizar vídeos de respuesta.

CU-8.2	**Registrar evaluación**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF4.4
Descripción	Calificar y comentar ejecución del paciente
Precondición	Vídeo visualizado (CU-8.1)
Acciones	1. Asigna puntuación (1-5) 2. Añade comentarios específicos 3. Registra fecha de evaluación 4. El sistema almacena evaluación y la deja disponible para el paciente
Postcondición	Evaluación accesible para el paciente y el profesional
Excepciones	Ninguna
Importancia	Alta

Tabla B.14: CU-8.2 Registrar evaluación.

CU-9.1	**Ver historial de evaluaciones**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF4.5
Descripción	Consultar evaluaciones anteriores
Precondición	Usuario autenticado (CU-1), evaluaciones existentes
Acciones	1. Selecciona periodo de tiempo 2. El sistema muestra lista de evaluaciones ordenadas cronológicamente 3. Visualiza puntuación y comentarios para cada ejercicio
Postcondición	Información histórica accesible
Excepciones	Ninguna
Importancia	Media

Tabla B.15: CU-9.1 Ver historial de evaluaciones.

CU-9.2	**Visualizar gráficos de progreso**
Versión	1.0
Autor	Alberto Lanchares Diez
R. Asociados	RF4.6
Descripción	Analizar evolución temporal del desempeño
Precondición	Usuario autenticado (CU-1), múltiples evaluaciones
Acciones	1. Selecciona tipo de gráfico y periodo 2. El sistema genera visualización con puntuaciones a lo largo del tiempo 3. Muestra datos estadísticos relevantes
Postcondición	Análisis visual disponible
Excepciones	Ninguna
Importancia	Media

Tabla B.16: CU-9.2 Visualizar gráficos de progreso.

Apéndice C

Especificación de diseño

C.1. Introducción

Este anexo recoge el diseño técnico del sistema *TerapiTrack*, incluyendo la estructura de datos, la arquitectura general y los principales procedimientos implementados. El objetivo es documentar las decisiones de diseño que permiten garantizar la robustez, escalabilidad y mantenibilidad del sistema, así como mostrar la coherencia entre el modelo de datos, los casos de uso y la implementación realizada.

C.2. Diseño de datos

El diseño de datos se ha realizado siguiendo un enfoque relacional, asegurando la integridad y normalización de la información mediante claves primarias, foráneas y restricciones de unicidad y dominio. A continuación se presentan los principales diagramas y el diccionario de datos que describen cómo se modelan usuarios, pacientes, profesionales, ejercicios, sesiones y evaluaciones dentro de la aplicación.

Diagrama entidad-relación

El diagrama entidad-relación representa las entidades conceptuales del sistema (Usuario, Paciente, Profesional, Sesión, Ejercicio, Vídeo de respuesta, Evaluación, etc.) y las relaciones existentes entre ellas. En él se distinguen claramente las relaciones uno a uno (por ejemplo, Usuario–Paciente o Usuario–Profesional), las relaciones muchos a muchos resueltas mediante tablas intermedias (Paciente–Profesional, Sesión–Ejercicio) y las restricciones

principales que garantizan la trazabilidad de cada sesión y de sus ejercicios asociados. Esta representación facilita la comprensión de cómo se vinculan los datos clínicos con los usuarios y las sesiones terapéuticas.

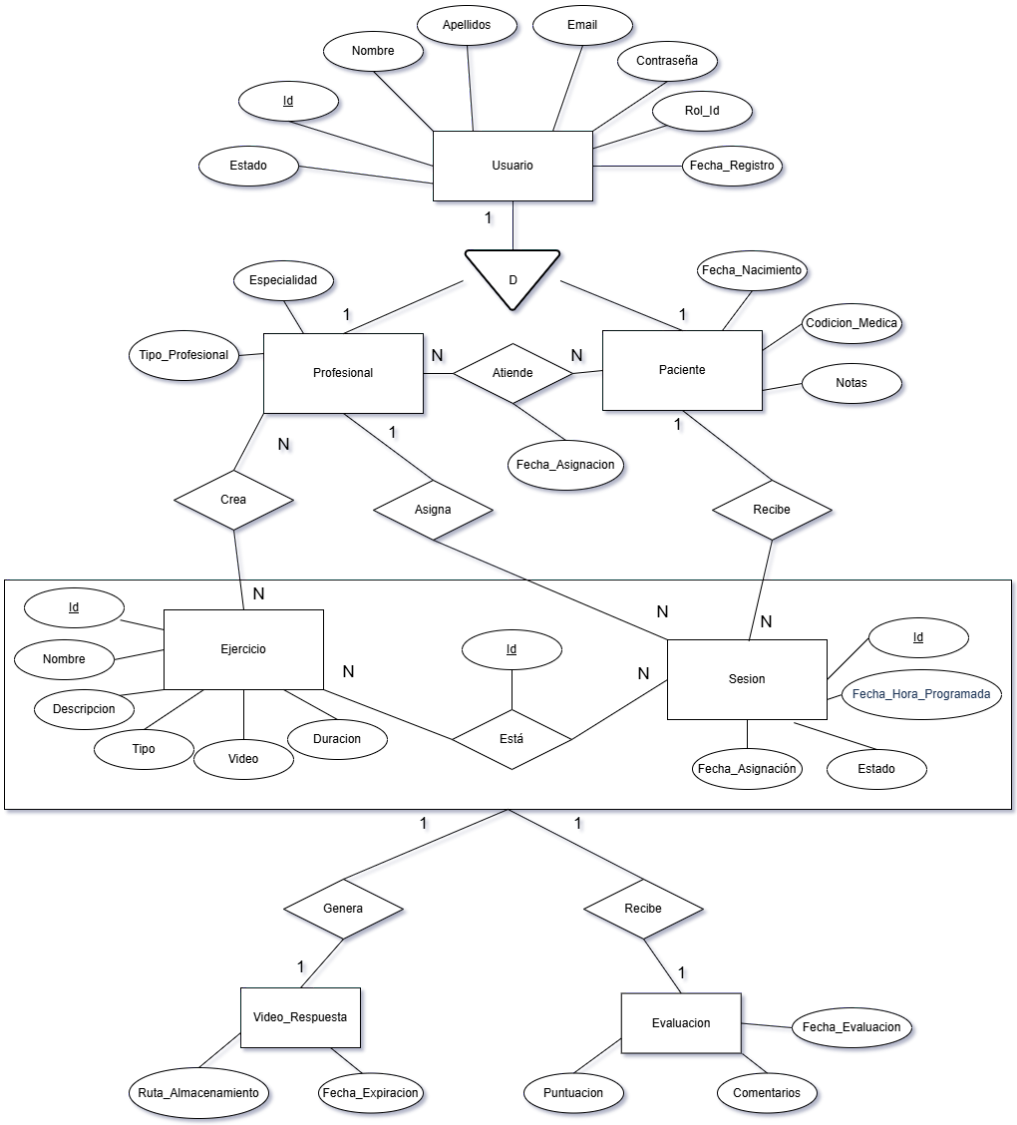

Figura C.1: Diagrama entidad-relación del sistema.

Diagrama relacional

El diagrama relacional muestra la traducción del modelo conceptual a tablas concretas de una base de datos relacional, indicando claves

primarias, claves foráneas e índices relevantes. En él puede verse cómo se han introducido tablas puente para las relaciones muchos a muchos (como `Paciente_Profesional` o `Ejercicio_Sesion`) y cómo se asegura la integridad referencial entre sesiones, ejercicios, vídeos y evaluaciones. Asimismo, el diagrama refleja el uso de campos de estado, fechas y restricciones de dominio que permiten controlar el ciclo de vida de las sesiones y la expiración de los vídeos.

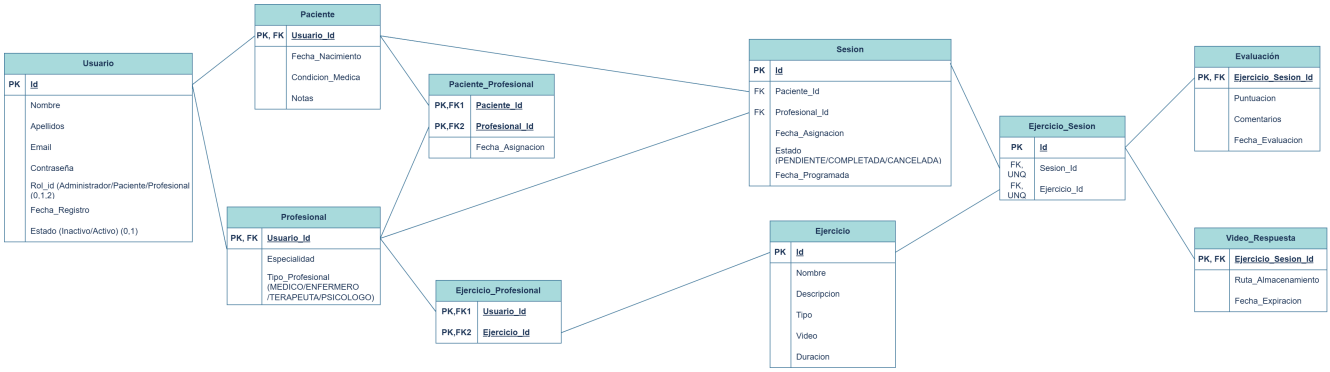

Figura C.2: Diagrama relacional de la base de datos.

Diccionario de datos

El diccionario de datos detalla la estructura de cada tabla, sus campos, tipos de datos, claves y restricciones. A continuación se resumen las principales entidades del sistema:

Campo	**Tipo**	**PK/FK**	**Restricciones**	**Descripción**
Id	integer	PK	autoincrement, not null	Identificador único del usuario
Nombre	text		not null	Nombre del usuario
Apellidos	text		not null	Apellidos del usuario
Email	text		unique, not null	Correo electrónico de acceso
Contrasena	text		not null	Contraseña cifrada
Rol_Id	integer		not null, check (0,1,2)	Rol: 0=Admin, 1=Paciente, 2=Profesional
Fecha_Registro	text		not null	Fecha de alta (YYYY-MM-DD)
Estado	integer		not null, check (0,1)	0=Inactivo, 1=Activo

Tabla C.1: Diccionario de datos de la tabla `Usuario`

Campo	**Tipo**	**PK/FK**	**Restricciones**	**Descripción**
Usuario_Id	integer	PK, FK	fk → Usuario(Id), not null	Identificador (FK `Usuario`)
Fecha_Nacimiento	text		not null	Fecha de nacimiento (YYYY-MM-DD)
Condicion_Medica	text			Condición médica principal
Notas	text			Observaciones adicionales

Tabla C.2: Diccionario de datos de la tabla `Paciente`

Campo	**Tipo**	**PK/FK**	**Restricciones**	**Descripción**
Usuario_Id	integer	PK, FK	fk → Usuario(Id), not null	Identificador (FK `Usuario`)
Especialidad	text		not null	Especialidad del profesional
Tipo_Profesional	text		check ('MEDICO', 'TERAPEUTA', 'ENFERMERO', 'PSICOLOGO'), not null	Tipo de profesional sanitario

Tabla C.3: Diccionario de datos de la tabla `Profesional`

Campo	**Tipo**	**PK/FK**	**Restricciones**	**Descripción**
Paciente_Id	integer	PK, FK	fk → Paciente(Usuario_Id), not null	Identificador (FK `Paciente`)
Profesional_Id	integer	PK, FK	fk → Profesional(Usuario_Id), not null	Identificador (FK `Profesional`)
Fecha_Asignacion	text		not null	Fecha de asignación (YYYY-MM-DD)

Tabla C.4: Diccionario de datos de la tabla `Paciente_Profesional`

Campo	**Tipo**	**PK/FK**	**Restricciones**	**Descripción**
Id	integer	PK	autoincrement, not null	Identificador del ejercicio
Nombre	text		not null	Nombre del ejercicio
Descripcion	text		not null	Descripción del ejercicio
Tipo	text		not null	Tipo o categoría del ejercicio
Video	text		not null	Ruta al vídeo demostrativo
Duracion	integer		not null	Duración estimada (segundos)

Tabla C.5: Diccionario de datos de la tabla `Ejercicio`

Campo	**Tipo**	**PK/FK**	**Restricciones**	**Descripción**
Usuario_Id	integer	PK, FK	fk → Profesional(Usuario_Id), not null	Identificador (FK `Profesional`)
Ejercicio_Id	integer	PK, FK	fk → Ejercicio(Id), not null	Identificador (FK `Ejercicio`)

Tabla C.6: Diccionario de datos de la tabla `Ejercicio_Profesional`

Campo	**Tipo**	**PK/FK**	**Restricciones**	**Descripción**
Id	integer	PK	autoincrement, not null	Identificador de la sesión
Paciente_Id	integer	FK	fk → Paciente(Usuario_Id), not null	Paciente al que se asigna la sesión
Profesional_Id	integer	FK	fk → Profesional(Usuario_Id), not null	Profesional responsable
Fecha_Creacion	text		not null	Fecha de creación (YYYY-MM-DD)
Estado	text		check ('PENDIENTE', 'COMPLETADA', 'CANCELADA'), not null	Estado de la sesión
Fecha_Programada	text		not null	Fecha programada (YYYY-MM-DD)

Tabla C.7: Diccionario de datos de la tabla `Sesion`

Campo	**Tipo**	**PK/FK**	**Restricciones**	**Descripción**
Id	integer	PK	autoincrement, not null	Identificador del ejercicio en sesión
Sesion_Id	integer	FK	fk → Sesion(Id), not null	FK a `Sesion`
Ejercicio_Id	integer	FK	fk → Ejercicio(Id), not null	FK a `Ejercicio`

Tabla C.8: Diccionario de datos de la tabla `Ejercicio_Sesion`

Campo	**Tipo**	**PK/FK**	**Restricciones**	**Descripción**
Ejercicio_Sesion_Id	integer	PK, FK	fk → Ejercicio_Sesion(Id), not null	Identificador (FK `Ejercicio_Sesion`)
Ruta_Almacenamiento	text		not null	Ruta del archivo de vídeo
Fecha_Expiracion	text			Fecha de expiración (YYYY-MM-DD)

Tabla C.9: Diccionario de datos de la tabla `Video_Respuesta`

Campo	Tipo	PK/FK	Restricciones	Descripción
Ejercicio_Sesion_Id	integer	PK, FK	fk → Ejercicio_Sesion(Id), not null	Identificador (FK `Ejercicio_Sesion`)
Puntuacion	numeric		check (>=1 and <=5), not null	Puntuación (1–5) asignada por profesional
Comentarios	text			Observaciones de la evaluación
fechas_Evaluacion	text		not null	Fecha de evaluación (YYYY-MM-DD)

Tabla C.10: Diccionario de datos de la tabla `Evaluacion`

Leyenda de simbología:

- PK: Clave primaria (Primary Key).
- FK: Clave foránea (Foreign Key).
- Unique: Valor único en la tabla.
- Check: Restricción de valores permitidos.
- Autoincrement: Incremento automático de la clave primaria.
- Not null: No puede ser un valor nulo.

C.3. Diseño arquitectónico

El sistema sigue una arquitectura modular basada en el patrón Modelo–Vista–Controlador (MVC), que facilita la separación de responsabilidades y la escalabilidad del desarrollo. En el caso de *TerapiTrack*, esta arquitectura se implementa sobre Flask mediante módulos y *blueprints* que agrupan funcionalidades por dominios (usuarios, sesiones, ejercicios, evaluaciones).

A nivel lógico, la arquitectura se organiza en tres capas principales:

- **Capa de presentación**: Incluye las vistas de Flask y las plantillas Jinja que generan las páginas HTML. En ella se definen los formularios, los mensajes de validación y los componentes visuales construidos con Bootstrap y Bootswatch, diferenciando las interfaces de administrador, profesional y paciente.
- **Capa de negocio**: Implementada en los *blueprints* de `src/controladores`, contiene la lógica de aplicación: gestión de usuarios y roles, asignación

de pacientes a profesionales, planificación de sesiones, grabación y evaluación de ejercicios, así como las comprobaciones de permisos antes de cada operación.

- **Capa de datos**: Formada por los modelos SQLAlchemy de `src/modelos` y por la base de datos SQLite. Se encarga de mapear las entidades del diccionario de datos a tablas, aplicar las restricciones de integridad definidas en el diseño de datos y proporcionar métodos para consultas y actualizaciones transaccionales.

La estructura de carpetas del proyecto refleja esta organización:

- `src/modelos`: Definición de entidades y relaciones (modelos SQLAlchemy) correspondientes a las tablas del diccionario de datos.
- `src/controladores`: Lógica de negocio, rutas y *blueprints* de Flask, incluyendo la gestión de autenticación, permisos y validación de formularios.
- `src/vistas`: Plantillas HTML, ficheros estáticos (CSS, JS) y recursos gráficos empleados para construir la interfaz web.
- `src/tests`: Pruebas unitarias y de integración sobre modelos y controladores, que permiten comprobar la corrección de la lógica implementada.

De forma transversal a estas capas, el sistema incorpora varios servicios comunes. La autenticación y gestión de sesión de usuarios se resuelve mediante Flask-Login, combinada con una tabla de `Usuario` que almacena contraseñas cifradas y un campo de rol que condiciona el acceso a cada sección de la aplicación. La validación de formularios se apoya en WTForms y en comprobaciones adicionales en los controladores para garantizar la coherencia de los datos antes de almacenarlos en la base de datos. Además, la aplicación está preparada para su despliegue en un entorno *Platform as a Service* como Heroku, donde el servidor Flask se expone a través de un servidor WSGI y se configuran las credenciales y rutas de base de datos mediante variables de entorno.

C.4. Diseño procedimental

En esta sección se describen los principales flujos y procedimientos implementados, que se apoyan en la arquitectura anterior y en el modelo de datos descrito:

- **Gestión de usuarios**: Incluye el alta, baja lógica, modificación y autenticación de usuarios, con control de roles y permisos. El flujo típico comienza con el administrador creando una cuenta, continúa con el acceso del usuario mediante correo y contraseña, y termina con la asignación de un panel específico según su rol.
- **Asignación de pacientes a profesionales**: El administrador establece relaciones muchos a muchos entre pacientes y profesionales mediante la tabla `Paciente_Profesional`, permitiendo que un profesional gestione a varios pacientes y que un paciente pueda estar vinculado a diferentes perfiles sanitarios.
- **Gestión de ejercicios y sesiones**: Los profesionales crean ejercicios con su vídeo demostrativo y configuran sesiones terapéuticas combinando varios ejercicios. Posteriormente asignan estas sesiones a pacientes concretos, definiendo fechas programadas y controlando el estado de cada sesión (pendiente, completada o cancelada).
- **Grabación y evaluación**: Durante la realización de una sesión, el sistema registra la ejecución de cada ejercicio mediante la cámara del dispositivo y almacena el vídeo asociado. Más tarde, los profesionales revisan esos vídeos, asignan una puntuación y añaden comentarios, generando un histórico de evaluaciones que se utiliza para visualizar la evolución del paciente a lo largo del tiempo.

En una versión ampliada de este anexo podrían incorporarse diagramas de actividad para estos procedimientos, mostrando para cada caso el actor implicado (paciente, profesional o administrador), las llamadas a los controladores y las interacciones principales con los modelos y la base de datos.

Apéndice D

Documentación técnica de programación

D.1. Introducción

D.2. Estructura de directorios

D.3. Manual del programador

D.4. Compilación, instalación y ejecución del proyecto

D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Apéndice F

Anexo de sostenibilización curricular

F.1. Introducción

Este anexo incluirá una reflexión personal del alumnado sobre los aspectos de la sostenibilidad que se abordan en el trabajo. Se pueden incluir tantas subsecciones como sean necesarias con la intención de explicar las competencias de sostenibilidad adquiridas durante el alumnado y aplicadas al Trabajo de Fin de Grado.

Más información en el documento de la CRUE https://www.crue.org/wp-content/uploads/2020/02/Directrices_Sosteniblidad_Crue2012.pdf.

Este anexo tendrá una extensión comprendida entre 600 y 800 palabras.

Bibliografía

[1] Agencia Homia. Cuanto cobra un desarrollador web en espana, 2025. Consulta: 1 diciembre 2025.

[2] Flask-Login Authors. Flask-login: User session management for flask. https://pypi.org/project/Flask-Login/, 2025. Accedido el 1 de diciembre de 2025.

[3] SQLAlchemy Authors. Sqlalchemy documentation. https://www.sqlalchemy.org/, 2025. Accedido el 1 de diciembre de 2025.

[4] DB Browser for SQLite. Db browser for sqlite. https://sqlitebrowser.org/, 2025. Accedido el 1 de diciembre de 2025.

[5] Hostinger. Cuanto cuesta un hosting en 2025, 2025. Consulta: 1 diciembre 2025.

[6] Omniretro. Mandos usb estilo snes, 2025. Consulta: 1 diciembre 2025.

[7] Thomas Park. Bootswatch: Free themes for bootstrap. https://bootswatch.com/, 2025. Accedido el 1 de diciembre de 2025.

[8] Pallets Projects. Blueprints — flask documentation. https://flask.palletsprojects.com/en/stable/blueprints/, 2025. Accedido el 1 de diciembre de 2025.

[9] Pallets Projects. Documentación de jinja (versión en español). https://jinja.palletsprojects.com/es/estable/, 2025. Accedido el 1 de diciembre de 2025.

[10] Pallets Projects. Flask documentation (stable). https://flask.palletsprojects.com/en/stable/, 2025. Accedido el 1 de diciembre de 2025.

[11] Real Python. Organizing flask applications with blueprints. https://realpython.com/flask-blueprint/, 2025. Accedido el 1 de diciembre de 2025.