TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

PharmAD

Desarrollo de una plataforma destinada al seguimiento y mejora de la adherencia terapéutica de los pacientes de una farmacia

Autor

VIDICAN, MIHNEA IOAN

Directores

GUIRAO MIRAS, JOSÉ MARÍA

Escuela Técnica Superior de Ingenierías Informática
y de Telecomunicación

—

Granada, a 12 de noviembre de 2024

PharmAD: Desarrollo de una plataforma destinada al seguimiento y mejora de la adherencia terapéutica de los pacientes de una farmacia

MIHNEA IOAN VIDICAN

Palabras clave: PharmAD, adherencia, terapéutica, farmacias, pacientes, seguimiento, cumplimiento.

Resumen

PharmAD aborda el desafío de la adherencia terapéutica en las farmacias comunitarias, entendida como el grado en que los pacientes siguen las recomendaciones de los profesionales de la salud para asegurar la efectividad de los tratamientos médicos. El proyecto se centra en facilitar el seguimiento de tratamientos farmacológicos y no farmacológicos mediante una plataforma web, permitiendo al personal sanitario gestionar de manera eficiente dichos tratamientos, monitorizar la evolución de los pacientes y brindar soporte en la gestión de sus terapias.

Con un enfoque en mejorar la comunicación entre los profesionales de la salud y los pacientes, esta iniciativa pretende contribuir a una atención sanitaria más efectiva y a mejorar la calidad de vida de los pacientes, optimizando el cumplimiento de los tratamientos y reduciendo el riesgo de complicaciones asociadas a un seguimiento inadecuado.

PharmAD: Development of a Platform for Monitoring and Improving Therapeutic Adherence in Pharmacy Patients

MIHNEA IOAN VIDICAN

Keywords: PharmAD, adherence, therapeutic, pharmacies, patients, follow-up, compliance.

Abstract

PharmAD tackles the challenge of therapeutic adherence in community pharmacies, defined as the extent to which patients follow healthcare professionals' recommendations to ensure effective medical treatments. The project focuses on facilitating the monitoring of both pharmacological and non-pharmacological treatments through a web platform, allowing healthcare personnel to efficiently manage these treatments, track patient progress, and provide support in therapy management.

By emphasizing improved communication between healthcare professionals and patients, this initiative seeks to contribute to more effective healthcare delivery and enhance patients' quality of life by optimizing treatment adherence and reducing the risk of complications associated with inadequate monitoring.

Yo, **MIHNEA IOAN VIDICAN**, alumno de la titulación Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con NIE X7527819B, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: MIHNEA IOAN VIDICAN

Granada, a 12 de noviembre de 2024.

D. **JOSÉ MARÍA GUIRAO MIRAS**, Profesor del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Título del proyecto, Subtítulo del proyecto***, ha sido realizado bajo su supervisión por **MIHNEA IOAN VIDICAN**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada, a 12 de noviembre de 2024.

Los directores:

D. JOSÉ MARÍA GUIRAO MIRAS

Agradecimientos

En primer lugar, este proyecto se lo dedico a mi padre, la persona que más ha ansiado verme terminar y que, tristemente, ya no se encuentra entre nosotros.

Me gustaría agradecer a toda mi familia y a mi pareja por todo el apoyo incondicional, la ayuda, el amor y cariño recibido durante todos estos años. Ha sido un largo camino que, por fin, finaliza.

También quiero agradecer a mis amigos más cercanos por haber estado siempre a pie de cañón.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Concepto de adherencia terapéutica

En el ámbito científico, concretamente en el sector sanitario, existen distintos términos utilizados para describir las malas conductas, hábitos o formas no adecuadas de seguir las instrucciones y/o recomendaciones recibidas por parte de los profesionales de la salud. Sin embargo, "adherencia" es el término usado como referencia por la Organización Mundial de la Salud (OMS), y se define como el grado en que la conducta de un paciente, en relación con la toma de medicación, el seguimiento de una dieta o la modificación en los hábitos de vida, se ajusta o corresponde con las recomendaciones acordadas con el profesional sanitario. [1]

A menudo se ha utilizado de manera intercambiable [2] junto al término "cumplimiento", el cual hace alusión a la medida en la que un paciente actúa de acuerdo con la dosis, pauta posológica y plazo prescrito [3]. Con todo, la adherencia comprende una aplicación más amplia que incluye todo el espectro del plan terapéutico, tanto en aspectos farmacológicos como no farmacológicos [4].

Desde sus inicios, las conceptualizaciones del término adherencia abarcaban una variedad de prácticas recomendadas, que incluían desde el cumplimiento de la medicación hasta ajustes en la dieta y otros cambios de estilo de vida, siempre en línea con las prescripciones indicadas. No obstante, estas conceptualizaciones no siempre reflejaban una verdadera sintonía entre las expectativas del profesional sanitario y la participación del paciente, como lo señaló el médico y epidemiólogo clínico Robert B. Haynes en 1979 [4]. En este sentido, la adherencia no farmacológica cobra especial relevancia al enfatizar la importancia de mantener hábitos saludables, como una dieta adecuada, el cese del consumo de tabaco y otras sustancias nocivas, así como la práctica regular de ejercicio físico.

A pesar de las diferencias entre adherencia y cumplimiento, actualmente se acepta cada vez más que la adherencia prevalece sobre el mero cumplimiento de las indicaciones, permitiendo enfrentar de manera más integral el desafío de la adherencia terapéutica [3].

1.2. Predictores de la adherencia y no adherencia

El proceso de adherencia se divide en tres etapas: **iniciación**, cuando el paciente comienza el tratamiento; **ejecución**, que se refiere al cumplimiento continuo del régimen prescrito; y **discontinuación**, que ocurre cuando el tratamiento se interrumpe antes de lo indicado.

Interrupciones como el retraso en empezar el tratamiento o la finalización prematura son ejemplos de fallas en la adherencia que pueden darse en cualquier fase del proceso. No obstante, existe una serie de situaciones que determinan la buena o mala adherencia del paciente, las cuales están ligadas a diferentes factores relacionados con el tratamiento, la sociedad y el sistema sanitario.

En lo que respecta al tratamiento, la incertidumbre generada por el desconocimiento acerca de lo recetado conlleva a una peor adherencia, destacando el impacto que generan los efectos adversos, entre otros. Diversos estudios han demostrado la existencia de una relación entre el número y frecuencia de dosis y el cumplimiento farmacológico, de tal forma que este último se consigue garantizar si el número de dosis es bajo, preferiblemente siendo una única dosis diaria. En otros casos se ve afectado y no se obtiene con pautas de más de dos dosis diarias [5].

En cuanto a los factores relacionados con la sociedad, unos recursos económicos escasos y el precio elevado de los medicamentos, en el caso de un tratamiento farmacológico, están directamente asociados a una peor adherencia. Por lo contrario, la adherencia mejora cuando el paciente convive con miembros familiares [4]. Está también comprobado que el inicio del proceso de adopción de un hábito de vida impuesto por un profesional sanitario se ve facilitado si el paciente se encuentra rodeado de familiares y amistades que apoyen e inciten su cumplimiento.

Uno de los factores más influyentes en la adherencia es la relación entre el profesional sanitario y el paciente, especialmente en el modelo actual de atención centrada en el paciente, donde se promueve la toma de decisiones compartida. En este enfoque el paciente participa activamente en las decisiones relacionadas con su diagnóstico, tratamiento y rehabilitación, permitiéndole expresar sus preferencias, valores y preocupaciones. Sin embargo, su grado de participación en la práctica puede variar según varios factores. Situaciones de emergencia o tratamientos complejos pueden limitar la capacidad del paciente para involucrarse en decisiones críticas. Asimismo, la actitud del profesional sanitario y su disposición para fomentar la participación, así como el nivel de comprensión del paciente y sus preferencias personales, influyen en la medida en que el paciente asume un rol activo. Aun así, la presencia empática y colaborativa del profesional es fundamental durante todo el proceso, mejorando la adherencia y fomentando una relación de confianza. [4]

1.3. Formas de incumplir el tratamiento

Las categorías más comunes de incumplimiento del tratamiento, que reflejan los distintos comportamientos de los pacientes en relación con la adherencia a sus medicamentos y terapias prescritas, se describen a continuación [2]:

- **Incumplimiento parcial.** El paciente no sigue el tratamiento de manera constante, únicamente en algunos momentos.
- **Incumplimiento esporádico.** El paciente omite el tratamiento de manera ocasional, lo cual es más frecuente en personas mayores que pueden olvidar dosis o reduirlas debido a preocupaciones sobre posibles efectos adversos
- **Incumplimiento secuencial.** El paciente interrumpe el tratamiento durante los periodos en los que se siente bien, reanudándolo sólo cuando los síntomas reaparecen. Este patrón es comparable a lo que se conoce por "vacaciones terapéuticas".
- **Cumplimiento de bata blanca.** El paciente se adhiere al tratamiento únicamente en los días cercanos a las visitas médicas. Este comportamiento, junto con el incumplimiento secuencial, es común en enfermedades crónicas como la hipertensión o la dislipemia.
- **Incumplimiento completo.** El paciente abandona el tratamiento de manera indefinida. Este tipo de falta de adherencia es más prevalente entre los jóvenes con enfermedades crónicas, probablemente debido a que los beneficios del tratamiento son a largo plazo, mientras que los costos y efectos adversos son inmediatos.

1.4. Métodos de seguimiento y estimación de la adherencia

Medir la adherencia permite a los profesionales de la salud evaluar hasta qué punto los pacientes siguen las prescripciones de medicamentos según lo indicado. Este proceso de medición puede llevarse a cabo mediante varios métodos, cada uno con sus propias ventajas y limitaciones, desde análisis bioquímicos hasta sistemas electrónicos y cuestionarios. La elección del método adecuado depende de varios factores, incluidos el costo, la facilidad de implementación y la precisión requerida en la medición.

Es importante destacar que ningún método es perfecto por sí solo; por ello, combinar varios métodos puede proporcionar una evaluación más completa y precisa de la adherencia. Además, la medición sistemática de la adherencia permite identificar barreras específicas que enfrentan los pacientes. [2]

En las siguientes tablas se muestran los métodos más usados por los farmacéuticos:

Método	**Ventajas**	**Definición**
Determinación plasmática	▪ Método directo ▪ Detecta toxicidad ▪ Ventaja en población con farmacocinética alterada (embarazo o disfunción hepática)	▪ Caro e invasivo ▪ Recogida e interpretación no estandarizadas
Registros de dispensación	▪ Fácil obtención de datos ▪ Correlación moderada con resultados	▪ Sobrestimación ▪ No mide la frecuencia horaria ▪ Asume que la recogida de medicación equivale a adherencia ▪ No distingue tipos de adherencia
Recuento de medicación	▪ Bajo costo ▪ Correlación moderada con resultados	▪ Requiere colaboración del paciente ▪ Sobrestimación ▪ Asume que el paciente no almacena medicación

Tabla 1: Métodos de estimación de la adherencia (I).

Por ejemplo, los análisis bioquímicos pueden confirmar la presencia de un fármaco en el organismo, pero no indican si se sigue la pauta de dosis exacta. En cambio, los sistemas electrónicos y los cuestionarios permiten monitorear patrones de consumo a lo largo del tiempo, aunque pueden estar sujetos a sesgos de autoinforme o problemas técnicos. La medición de la adherencia es, por tanto, un desafío complejo que requiere un enfoque equilibrado para obtener información útil y confiable [2].

Método	Ventajas	Definición
MEMS/dispositivos electrónicos	▪ Patrones de adherencia en el tiempo ▪ Análisis detallado	▪ Costoso ▪ Infraestimación ▪ No disponible en todos los centros ▪ Vulnerable a fallos tecnológicos
Cuestionario/adherencia autorreferida	▪ Bajo costo ▪ Fácil de implementar	▪ No estandarizado ▪ Sobrestimación ▪ Sensibilidad baja

Tabla 2: Métodos de estimación de la adherencia (II).

La figura inferior es un cuestionario ARMS-e diseñado para medir la adherencia en pacientes con múltiples patologías. Este instrumento evalúa la falta de adherencia de forma multidimensional, lo que permite personalizar las intervenciones según las barreras detectadas en cada individuo. Consta de 12 preguntas y no define un punto de corte específico; en su lugar, una puntuación más baja refleja una mejor adherencia. Para cuantificar el grado de adherencia, a cada opción de respuesta se le asigna un valor de 1 a 4, siguiendo una escala Likert: nunca, algunas veces, casi siempre o siempre. [6]

Responda a las preguntas con una de las siguientes respuestas: Nunca, algunas veces, casi siempre o siempre.

1. ¿Con qué frecuencia olvida tomar sus medicinas?
2. ¿Con qué frecuencia decide no tomar sus medicinas?
3. ¿Con qué frecuencia olvida recoger de la farmacia las medicinas que le han recetado?
4. ¿Con qué frecuencia se queda sin medicinas?
5. ¿Con qué frecuencia se salta una dosis de su medicación antes de ir al médico?
6. ¿Con qué frecuencia deja de tomar sus medicinas cuando se encuentra mejor?
7. ¿Con qué frecuencia deja de tomar sus medicinas cuando se encuentra mal?
8. ¿Con qué frecuencia deja de tomar sus medicinas por descuido?
9. ¿Con qué frecuencia cambia la dosis de su medicación y la adapta a sus necesidades (por ejemplo, cuando se toma más o menos pastillas de las que debería)?
10. ¿Con qué frecuencia olvida tomar sus medicinas cuando debe tomarlas más de una vez al día?
11. ¿Con qué frecuencia retrasa ir a recoger sus medicinas de la farmacia porque cuestan demasiado dinero?
12. ¿Con qué frecuencia planifica recoger de la farmacia sus medicinas antes de que se le acaben?

Figura 1: Cuestionario de adherencia.

1.5. Estado del arte: iniciativas y programas desarrollados

A nivel nacional se encuentran en fase de implantación varios proyectos que posibilitan el servicio de seguimiento farmacoterapéutico [7], como es el caso del programa conSIGUE [8], cuyo objetivo es evaluar y mejorar la adherencia terapéutica en personas mayores, con enfermedades crónicas, polimedicadas e incumplidoras. También existe el programa Adhiérete [9], similar al anterior, y que aprovecha los sistemas personalizados de dosificación para ofrecer un servicio de asistencia domiciliaria y de evaluación personalizada de la calidad de vida del paciente.
A pesar de mostrar resultados favorecedores, se trata de sistemas diseñados y puestos en funcionamiento hace más de una década, orientados mayoritariamente a personas de la tercera edad y pacientes polimedicados y crónicos, sin disponer de plataformas digitales para poder estar al alcance de más rangos de edad.

Un ejemplo de programa que combina las técnicas clásicas con el soporte digital es HazAdherencia, el cual facilita a los colegiados la formación y los medios necesarios para poder atender y prestar asistencia farmacéutica personalizada a pacientes con hipertensión arterial, diabetes o dislipemia [10]. Aunque se tengan las herramientas digitales necesarias, su uso no está dirigido hacia todo tipo de pacientes.

Existen también aplicaciones no oficiales que organizan y alertan sobre la toma de medicación, abiertas a todo el público, pero sin soporte por parte de los profesionales de la salud.

1.6. Motivación

La correcta adherencia a los tratamientos médicos es esencial para garantizar su eficacia. Un cumplimiento deficiente puede llevar a resultados adversos en la salud, incrementar los costos asociados al manejo de la enfermedad y afectar negativamente el ámbito social, tanto a nivel individual como colectivo.

- **Impactos en la salud.** Los medicamentos que han probado su eficacia en ensayos clínicos podrían no ser efectivos en la práctica clínica si no se administran correctamente. Esto incluye tanto las recomendaciones farmacológicas como las no farmacológicas. La reducción en la eficacia depende del nivel de incumplimiento, las características específicas de la enfermedad, las propiedades farmacodinámicas y farmacocinéticas del fármaco, y la condición preexistente del paciente. [4]
- **Implicaciones económicas.** La falta de adherencia se asocia con un incremento en la severidad de las enfermedades, un mayor número de visitas médicas no planificadas, el uso de medicamentos no necesarios y hospitalizaciones, lo cual eleva los gastos de salud significativamente en casos de enfermedades crónicas. Además, los costes indirectos comprenden la pérdida de productividad laboral de los enfermos y sus cuidadores. [4]

Dado que el problema de la falta de adherencia está generalizado y se contempla en más rangos de edad, este proyecto propone un seguimiento y mejora de la adherencia terapéutica para todo tipo de pacientes mediante la creación de un medio digital actual y moderno que facilite a los farmacéuticos la elaboración, preparación y seguimiento de cualquier tipo de tratamiento,

así como la recomendación y advertencia sobre la toma de la medicación, si está incluida, de acuerdo a directrices específicas.

1.7. Objetivos

Los objetivos principales de este proyecto se enumeran como sigue:

1. Ofrecer una atención personalizada a cada paciente, recogiendo información acerca de su estado de salud y recomendaciones sanitarias.
2. Proporcionar información sobre medicación y obtener retroalimentación por parte de los pacientes para posibilitar a los farmacéuticos la labor de realizar un seguimiento del tratamiento vigente.
3. Recordar los tratamientos y hábitos impuestos.
4. Medir la adherencia del paciente mediante una adaptación de los métodos tradicionales.
5. Ofrecer estadísticas acerca de la adherencia terapéutica.
6. Favorecer la relación profesional-paciente mediante un contacto continuo entre ambos.

Capítulo 2

Análisis y especificaciones del proyecto

En la base de todo proyecto informático reside una idea concreta acompañada de objetivos específicos que se desean alcanzar. Para cumplir con lo propuesto, resulta necesario establecer un punto de partida que desarrolle detalladamente dicha idea. Este proceso incluye la identificación de los actores que intervendrán en el proyecto y la comprensión de sus necesidades, las cuales serán aspectos a tener en cuenta para la consecución del producto deseado.

2.1. Identificación de actores

Se definen tres categorías de actores:

- **Personal sanitario.** Dado que este proyecto está orientado de forma exclusiva a farmacias comunitarias[1], se distinguen los siguientes tipos de personal sanitario:
  - **Farmacéutico.** Es la persona cualificada y encargada para elaborar y dispensar medicación, recomendar la implantación de hábitos de vida, dietas, controlar las recetas médicas dispensadas y realizar un seguimiento del tratamiento en los pacientes con el fin de determinar la adherencia y ofrecer recomendaciones para mejorarla. A los farmacéuticos se les atribuye más tareas y cargos dentro de una oficina de farmacia, pero las funciones anteriores serán las principales para este proyecto.
  - **Técnico de farmacia.** Es una persona con formación en el campo de la farmacia que asiste al farmacéutico titular en diferentes tareas: dispensación de productos farmacéuticos junto con sus indicaciones de uso, toma de constantes vitales, bajo la supervisión del facultativo, y parámetros somatométricos (como el peso y la altura), además de realizar tareas administrativas.
- **Clientes.** En esta categoría existen dos perfiles:
  - **Paciente.** Se trata del individuo que busca atención o recibe cuidados de salud debido a enfermedades, lesiones, para mejorar su bienestar, para prevenir otras enfermedades o para obtener diagnósticos sobre su estado de salud. [12]

[1]Las farmacias comunitarias son establecimientos sanitarios de carácter privado pero de interés público donde el farmacéutico titular-propietario, asistido, en su caso, de ayudantes o auxiliares, presta servicios básicos a la población: dispensación de los medicamentos y productos sanitarios, control y custodia de las recetas médicas dispensadas, seguimiento de los tratamientos farmacológicos y no farmacológicos a los pacientes, entre otros. [11]

  - **Tutor o acompañante del paciente**. En el caso en que el paciente no sea una persona independiente, como un menor o un anciano, será el tutor, representante o acompañante del mismo quien realizará todas las tareas de recogida de mediación y de proporción de información en su lugar.

- **Administrador del sistema.** Será el encargado de llevar a cabo tareas de monitorización de la actividad de la plataforma final, ofrecer soporte y manejar la creación y eliminación de oficinas de farmacia dentro del proyecto.

2.2. Historias de usuario

Las historias de usuario en el desarrollo de software son descripciones informales expresadas en lenguaje natural sobre una o varias acciones que un actor desea realizar y su justificación. Sirven como referencia para crear un requisito que deberá cumplir el sistema final.

Para este proyecto se detallan las siguientes historias de usuario, las cuales irán nombradas de la forma HU-XX, siendo XX el número de historia en orden creciente:

- **HU-01: Farmacéutico.** El farmacéutico titular de una farmacia atiende a una variedad de clientes a lo largo de un día. La mayoría de ellos acude a recoger la medicación prescrita, otros para que se les dispense una medicación personalizada y algunos llegan para aclarar sus dudas o incertidumbres sobre la toma de medicamentos y comunicar sus avances en cuanto a todo tipo de tratamientos. El farmacéutico despacha a todas las personas por igual, aunque emplea un mayor tiempo en aquellas consultas más complejas, como es el caso del seguimiento de un tratamiento, su cumplimiento o el estudio del uso individualizado de los medicamentos, con el propósito de detectar reacciones adversas que puedan producirse. Para poder ser más eficiente en su jornada laboral, requiere que muchas de estas tareas se automaticen, pues algunos de los datos que recoge manualmente son de carácter estadístico y no se precisa su presencia obligatoria.

- **HU-02: Técnico de farmacia.** El técnico de farmacia es incorporado a una farmacia por el farmacéutico titular y, aunque no tiene las mismas responsabilidades que este, también puede dispensar medicamentos y atender consultas de los pacientes sobre sus tratamientos en curso. El farmacéutico delega en el técnico diversas funciones, como contribuir al análisis del cumplimiento de tratamientos, dietas o hábitos de vida, y medir constantes vitales y parámetros somatométricos. Además, el técnico puede ser responsable de registrar a los pacientes en el sistema de la farmacia.

- **HU-03. Paciente.** Un paciente llega a la farmacia para retirar su medicación prescrita. Tras comenzar el tratamiento, puede regresar para que se le dispense nuevamente el medicamento necesario o para realizar un seguimiento de su tratamiento o de los hábitos recomendados. Sin embargo, en algunas ocasiones no es necesaria la dispensación de nueva medicación, y el paciente solo necesita informar sobre el cumplimiento de lo prescrito. Estas tareas son algo que el paciente puede, y a menudo prefiere, realizar desde su domicilio, sin necesidad de acudir presencialmente a la farmacia.

- **HU-03. Acompañante del paciente.** Un paciente menor de edad necesita informar sobre su estado de salud unos días después de iniciar su tratamiento. Sin embargo, debido a su corta edad, son sus padres quienes deben realizar esta tarea, ya que el menor no puede acudir solo a la farmacia. Los padres preferirían poder gestionar esto desde su domicilio

o desde cualquier lugar donde se encuentren, dado que podrían estar lejos de la farmacia donde se registró el inicio del tratamiento.

2.3. Requisitos

Los requisitos son especificaciones que el sistema debe cumplir para satisfacer las necesidades de los usuarios finales. El objetivo que plantean es guiar el diseño, el desarrollo y las pruebas y mantenimiento del software.

2.3.1. Requisitos funcionales

Los requisitos funcionales especifican lo que debe realizar el sistema en cuanto a funciones o servicios. Los que se listan a continuación vienen nombrados por RFXX, donde XX es el número del requisito en orden creciente.

- **RF1: Registro de pacientes.** El sistema permitirá al personal sanitario registrar pacientes en la farmacia.
  - **RF1.1**: El farmacéutico y el técnico de farmacia podrán registrar nuevos pacientes en el sistema.
  - **RF1.2**: El sistema permitirá al farmacéutico y al técnico de farmacia actualizar la información de pacientes existentes.
  - **RF1.3**: El sistema permitirá registrar la información de contacto del tutor o acompañante del paciente, en caso de que el paciente sea un menor o una persona dependiente.
- **RF2: Dispensación de medicamentos.** El sistema permitirá la dispensación de medicamentos recetados a los pacientes.
  - **RF2.1**: El farmacéutico podrá registrar la dispensación de medicamentos recetados a los pacientes.
  - **RF2.2**: El técnico de farmacia podrá asistir en la dispensación de medicamentos, bajo la supervisión del farmacéutico.
  - **RF2.3**: El sistema registrará la fecha y hora de cada dispensación, así como la identidad del personal que realizó la dispensación.
- **RF3: Análisis de la adherencia.** El sistema permitirá al personal sanitario realizar el seguimiento de los tratamientos de los pacientes para determinar el nivel de adherencia.
  - **RF3.1**: El farmacéutico podrá registrar observaciones sobre el progreso del tratamiento de un paciente.
  - **RF3.2**: El técnico de farmacia podrá ingresar datos de seguimiento como constantes vitales y parámetros somatométricos del paciente.
  - **RF3.3**: El paciente, o su tutor o acompañante, podrá ingresar datos de sequimiento de un tratamiento en el sistema y notificar las dificultades surgidas.
  - **RF3.4**: El sistema generará alertas y recordatorios para el seguimiento de tratamientos a pacientes con condiciones críticas o crónicas.

  - **RF3.5**: El sistema realizará un análisis del seguimiento del tratamiento e informará sobre las estadísticas obtenidas para determinar el grado de adherencia

- **RF4: Consulta de información médica.** El sistema permitirá al personal sanitario y a los pacientes consultar información sobre los tratamientos y medicamentos.

  - **RF4.1**: El farmacéutico podrá consultar el historial de tratamiento de un paciente.
  - **RF4.2**: El técnico de farmacia podrá acceder a la información relevante para la dispensación y seguimiento del tratamiento, bajo la supervisión del farmacéutico.
  - **RF4.3**: El paciente podrá consultar su historial de medicación y tratamiento en el sistema.
  - **RF4.4**: El acompañante del paciente, en caso de ser un menor o dependiente, podrá acceder a la información médica del paciente mediante un acceso autorizado.

- **RF5: Automatización de tareas administrativas.** El sistema automatizará ciertas tareas administrativas para mejorar la eficiencia del personal sanitario.

  - **RF5.1**: El sistema automatizará la generación de informes estadísticos sobre la dispensación de medicamentos.
  - **RF5.2**: El sistema permitirá la programación automática de seguimientos para tratamientos continuos o crónicos.

- **RF6: Gestión de usuarios del sistema.** El sistema permitirá la gestión de usuarios, incluyendo la creación, modificación y eliminación de cuentas para el personal sanitario y administrativo.

  - **RF6.1**: El administrador del sistema podrá crear, modificar y eliminar cuentas de usuarios para farmacéuticos y técnicos de farmacia.
  - **RF6.2**: El sistema permitirá asignar roles y permisos específicos según el perfil del usuario.

- **RF7: Seguimiento de tratamientos no farmacológicos.** El sistema permitirá al personal sanitario registrar y realizar el seguimiento de tratamientos no farmacológicos.

  - **RF7.1**: El farmacéutico podrá registrar recomendaciones de hábitos de vida saludables, como dietas, ejercicio físico, y otros tratamientos no farmacológicos.
  - **RF7.2**: El técnico de farmacia podrá asistir en el seguimiento del cumplimiento de estos hábitos de vida por parte del paciente.
  - **RF7.3**: El sistema permitirá al paciente o su tutor registrar su progreso en la adopción de estos hábitos de vida en el mismo.
  - **RF7.4**: El sistema generará alertas o recordatorios para el paciente sobre el cumplimiento de los tratamientos no farmacológicos, como recordar la realización de ejercicios o adherencia a una dieta específica.

2.3.2. Requisitos no funcionales

- **RFN1: Seguridad de la información.** El sistema garantizará la seguridad y confidencialidad de la información médica y personal de los pacientes.
  - **RFN1.1**: El sistema utilizará un cifrado seguro para proteger los datos sensibles almacenados y transmitidos.
  - **RFN1.2**: Solo el personal autorizado, como farmacéuticos, técnicos de farmacia, y administradores del sistema, tendrán acceso a la información médica y personal de los pacientes. Los pacientes también tendrán acceso a dicha información, pero no podrán consultar la información de otros pacientes si no son tutores o acompañantes autorizados.
  - **RFN1.3**: El sistema requerirá autenticación para acceder a funciones administrativas o a información confidencial.
- **RFN2: Disponibilidad del sistema.** El sistema deberá estar disponible para su uso en las farmacias comunitarias en todo momento.
  - **RFN2.1**: El sistema garantizará una alta disponibilidad para asegurar que los servicios no se vean interrumpidos durante el horario de atención de las farmacias.
- **RFN3: Rendimiento.** El sistema deberá ser capaz de manejar un alto volumen de transacciones y consultas sin afectar su rendimiento.
  - **RFN3.1**: El sistema deberá procesar la dispensación de medicamentos y la asignación de tratamientos no farmacológicos en un tiempo mínimo por transacción.
  - **RFN3.2**: El sistema deberá soportar la gestión simultánea de un número elevado de usuarios sin degradación en el rendimiento.
- **RFN4: Usabilidad.** El sistema deberá ser fácil de usar para todo el personal sanitario, considerando su nivel de familiaridad con las tecnologías.
  - **RFN4.1**: La interfaz de usuario deberá ser intuitiva, permitiendo a los usuarios realizar con fluidez sus tareas.
  - **RFN4.3**: El sistema deberá ser compatible con una amplia gama de dispositivos, permitiendo a los usuarios acceder a sus funciones de manera flexible y desde cualquier lugar.
- **RFN5: Escalabilidad.** El sistema deberá ser escalable para adaptarse al crecimiento del número de usuarios y farmacias integradas.
  - **RFN5.1**: El sistema deberá poder integrar nuevas farmacias comunitarias sin necesidad de interrupciones en el servicio.
  - **RFN5.2**: El sistema deberá poder integrar nuevo personal sanitarios y nuevos pacientes sin necesidad de interrupciones en el servicio.
- **RFN6: Cumplimiento normativo.** El sistema deberá cumplir con todas las normativas y regulaciones vigentes en el sector de la salud y la farmacia.

  - **RFN6.1**: El sistema deberá cumplir con las normativas de protección de datos personales, como el Reglamento General de Protección de Datos (GDPR) en Europa.
  - **RFN6.2**: El sistema deberá cumplir con las normativas de almacenamiento y dispensación de medicamentos impuestas por las autoridades sanitarias locales.

- **RFN7: Mantenimiento.** El sistema deberá ser fácilmente mantenible para asegurar su operación continua y eficiente.

  - **RFN7.1**: El sistema deberá permitir actualizaciones y parches sin necesidad de detener las operaciones críticas.

2.4. Casos de uso

Los casos de uso son una técnica utilizada para describir cómo los usuarios interactúan con un sistema para lograr un objetivo específico. Un caso de uso representa una secuencia de acciones o pasos que detallan la interacción entre un actor y el sistema en sí.

Caso de uso	**Descripción detallada**	**Requisitos satisfechos**
CU01: Registro de pacientes	**Descripción:** El personal sanitario podrá registrar y actualizar los datos de pacientes en el sistema. Además, se incluirá la opción de añadir la información del tutor o acompañante en caso de que el paciente lo requiera. **Precondiciones:** Usuario autenticado como farmacéutico o técnico de farmacia. **Postcondiciones:** Paciente registrado en el sistema. **Flujo principal:** 1. El usuario selecciona la opción de registrar paciente. 2. El usuario ingresa los datos del paciente. 3. El sistema valida la información. 4. El sistema guarda los datos del paciente.	RF1, RFN1, RFN4

CU02: Dispensación de medicamentos	**Descripción:** El sistema registrará la dispensación de medicamentos, incluyendo detalles como fecha, hora e identidad del personal que realizó la dispensación. **Precondiciones:** Prescripción válida. **Postcondiciones:** Registro de dispensación almacenado. **Flujo principal:** 1. El usuario selecciona la opción de dispensar medicamento. 2. El usuario ingresa la prescripción y la identifica con el paciente. 3. El sistema valida la prescripción. 4. El sistema guarda el registro de dispensación.	RF2, RFN3, RFN6
CU03: Análisis de adherencia	**Descripción:** El sistema permitirá al personal sanitario realizar el seguimiento de los tratamientos de los pacientes para determinar la adherencia. Se podrán registrar observaciones, ingresar datos de seguimiento como constantes vitales y generar alertas de seguimiento. **Precondiciones:** Paciente registrado en el sistema. **Postcondiciones:** Nivel de adherencia calculado y registrado. **Flujo principal:** 1. El usuario selecciona al paciente y el tratamiento a seguir. 2. El usuario ingresa los datos de seguimiento (vitales, somatométricos, etc.). 3. El sistema genera alertas o recordatorios si es necesario. 4. El sistema calcula y muestra el nivel de adherencia.	RF3, RF4, RFN5, RFN7

CU04: Consulta de información médica	**Descripción:** El sistema permitirá a los actores consultar información sobre tratamientos y medicamentos, accesible de manera segura y según los permisos correspondientes. **Precondiciones:** Usuario autenticado con permisos de consulta. **Postcondiciones:** Información médica consultada. **Flujo principal:** 1. El usuario inicia sesión y selecciona la opción de consulta. 2. El sistema verifica los permisos del usuario. 3. El usuario selecciona el tipo de información a consultar (tratamientos, medicamentos). 4. El sistema muestra la información solicitada.	RF4, RFN1, RFN2, RFN4
CU05: Automatización de tareas administrativas	**Descripción:** El sistema automatizará ciertas tareas administrativas, como la generación de informes y la programación de seguimientos, mejorando la eficiencia y reduciendo la carga de trabajo manual. **Precondiciones:** Usuario autenticado con permisos de administrador. **Postcondiciones:** Tareas administrativas automatizadas. **Flujo principal:** 1. El usuario selecciona la opción para generar informes o programar seguimientos. 2. El sistema recopila y procesa la información necesaria. 3. El sistema genera el informe o programa el seguimiento automáticamente. 4. El usuario recibe una notificación del proceso completado.	RF5, RFN3, RFN5, RFN6

CU06: Gestión de usuarios del sistema	**Descripción:** El administrador del sistema podrá gestionar las cuentas de usuarios, incluyendo la creación, modificación y eliminación de cuentas, así como la asignación de roles y permisos. **Precondiciones:** Usuario autenticado como administrador. **Postcondiciones:** Cambios en las cuentas de usuario aplicados. **Flujo principal:** 1. El administrador selecciona la opción de gestión de usuarios. 2. El administrador crea, modifica o elimina una cuenta de usuario. 3. El sistema valida los cambios y los guarda. 4. El usuario afectado recibe una notificación de los cambios.	RF6, RFN1, RFN7
CU07: Seguimiento de tratamientos no farmacológicos	**Descripción:** El sistema permitirá registrar y realizar el seguimiento de tratamientos no farmacológicos, como hábitos de vida saludables. Se generarán alertas y recordatorios para el cumplimiento de estos tratamientos. **Precondiciones:** Paciente registrado y tratamiento asignado. **Postcondiciones:** Seguimiento del tratamiento registrado. **Flujo principal:** 1. El usuario selecciona el paciente y el tratamiento no farmacológico. 2. El usuario ingresa o consulta los datos de seguimiento. 3. El sistema genera alertas o recordatorios si es necesario. 4. El sistema registra el progreso del paciente.	RF7, RFN1, RFN3, RFN4

Tabla 3: Casos de uso del sistema de forma detallada.

Capítulo 3

Planificación

Para que este proyecto se ejecute con éxito, se requiere una planificación meticulosa basada en una metodología específica, dado el amplio espectro de requisitos que deben cumplirse. En las secciones siguientes se detallan tanto la metodología seleccionada como el cronograma de trabajo propuesto.

3.1. Metodología a utilizar

La selección de una metodología de desarrollo adecuada es esencial para el éxito de este proyecto. Dado que buscamos flexibilidad, adaptación a cambios en los requisitos y entrega continua de valor, hemos optado por implementar la metodología ágil **Scrum**.

3.1.1. Introducción a SCRUM

Scrum es un marco de trabajo que promueve el desarrollo iterativo e incremental, centrado en la colaboración, la comunicación constante y la capacidad de adaptarse rápidamente a cambios. Esta metodología es especialmente adecuada para este tipo de proyectos, donde los requisitos pueden evolucionar y se necesita una entrega temprana de funcionalidades.

Se ha adoptado Scrum de manera adaptada para un desarrollo individual. A pesar de ser un proyecto realizado en solitario, se han integrado los principios y prácticas de Scrum, utilizando historias de usuario propias y los requisitos funcionales y no funcionales, y casos de uso descritos anteriormente.

- **Roles**
  - **Desarrollador**: Se han asumido las responsabilidades del *Product Owner*, del *Scrum Master* y del equipo de pesarrollo. Se ha gestionado el *Product Backlog*, planificado los sprints y llevado a cabo el desarrollo técnico del proyecto.
  - **Tutor**: Ha actuado como *stakeholder* y asesor, proporcionando orientación y retroalimentación en las reuniones periódicas, asegurando que el proyecto cumpla con los objetivos académicos y técnicos.

- **Artefactos**
  - **Product Backlog**: Lista priorizada y dinámica de todo lo que se requiere en el producto, incluyendo funcionalidades, mejoras y correcciones, basada en las historias de usuario y requisitos definidos.
  - **Sprint Backlog**: Conjunto de elementos del *Product Backlog* seleccionados para ser desarrollados en el sprint actual, junto con un plan detallado para su implementación.
  - **Incremento**: Suma de todos los elementos completados durante un sprint, representando un avance tangible en el proyecto.
- **Eventos**
  - **Planificación del sprint**: Al inicio de cada sprint, se han definido los objetivos y las tareas a realizar, priorizando según la importancia y la viabilidad.
  - **Revisión del sprint**: Al finalizar cada sprint, se han llevado a cabo reuniones con el tutor para presentar los avances, recibir retroalimentación y ajustar el *Product Backlog* si es necesario.
  - **Retrospectiva del sprint**: Se ha reflexionado sobre el proceso de trabajo, identificando áreas de mejora para incrementar la eficiencia y la calidad en sprints futuros.

3.1.2. Adaptación de Scrum al contexto del proyecto

Dado que el proyecto se basa en desarrollar un *Producto Mínimo Viable* (MVP) y luego expandirlo, Scrum ofrece las herramientas necesarias para cumplir con este objetivo. Las razones principales por las que se ha escogido este tipo de metodología son:

- **Priorización de funcionalidades críticas**: Al enfocarse en las historias de usuario y casos de uso más relevantes, se asegura que el MVP cumpla con los requisitos esenciales.
- **Entrega temprana y continua**: Los sprints permiten entregar incrementos funcionales en cortos períodos de tiempo, proporcionando valor desde las primeras etapas del proyecto.
- **Retroalimentación constante**: Las reuniones regulares con el tutor permiten ajustar el desarrollo según sus recomendaciones y expectativas.
- **Gestión efectiva de cambios**: Scrum facilita la incorporación de nuevos requisitos o modificaciones sin afectar significativamente el cronograma general.

3.1.3. Planificación con Scrum

La implementación de Scrum en el proyecto ha seguido un ciclo iterativo compuesto por sprints de duración de aproximadamente dos semanas. Cada sprint ha incluido las siguientes fases:

1. **Planificación del sprint**:
   - Selección de elementos del *Product Backlog* para el sprint.
   - Definición de objetivos claros y alcanzables.
   - Elaboración de un plan de trabajo detallado.

2. **Ejecución del sprint**:
    - Desarrollo de las funcionalidades seleccionadas.
    - Seguimiento diario del progreso y ajuste del plan según sea necesario.
3. **Revisión del sprint**:
    - Presentación del incremento al tutor.
    - Recopilación de retroalimentación para futuras mejoras.
4. **Retrospectiva del sprint**:
    - Evaluación del proceso y las herramientas utilizadas.
    - Identificación de acciones para mejorar en el siguiente sprint.

3.2. Organización y planificación temporal

Las especificaciones del sistema para este proyecto, detalladas en el capítulo anterior, se han organizado por hitos o *milestones*, los cuales agrupan uno o varios casos de uso para seguir la estructura de Scrum. A su vez, estos hitos han incluido etapas de estudio, investigación y documentación.

- **Hito 0**: Inicio del proyecto.
    - Investigación sobre el concepto central del proyecto.
    - Estudio del estado del arte y definición del dominio del problema.
    - Identificación de actores y requisitos.
    - Especificación de necesidades: historias de usuario, requisitos funcionales y no funcionales, y casos de uso.
    - Planificación del desarrollo.
- **Hito 1**: Elección de herramientas de trabajo.
    - Estudio acerca de la implementación deseada.
    - Comparación y elección de herramientas disponibles.
- **Hito 2**: Desarrollo del backend.
    - Diseño de la base de datos.
    - Diseño de API.
    - Implementación de procedimientos de autenticación y autorización para API.
- **Hito 3**: Desarrollo inicial del frontend.
    - Bocetos de las vistas.
    - Registro y login para usuarios.
    - Primera versión de la página de inicio (landing).
- **Hito 4**: Primeras funcionalidades.

  - Implementación de CU01, CU02, CU07 y CU06

- **Hito 5**: Funcionalidades para el seguimiento de la adherencia.
  - Implementación de CU03 y CU04.
  - Implementación de CU07.
- **Hito 6**: Documentación y despliegue.
  - Documentación del proceso de desarrollo.
  - Despliegue del sistema.

3.3. Control de versiones

La herramienta elegida para controlar las versiones y llevar a cabo un desarrollo organizado del proyecto ha sido GitHub. En el repositorio `TFG` se encuentra el proyecto: `https://github.com/johnwaves/TFG`.

Para la consecución de cada hito, se han creado diversas ramas (*branches*) en el repositorio. Cada rama ha estado dedicada al desarrollo de funcionalidades específicas dentro de la implementación de casos de uso concretos. Al finalizar el trabajo en una rama, se ha generado un *Pull Request* hacia la rama principal (`main`), permitiendo revisar y fusionar los cambios de manera controlada.

Los *Pull Requests* han contenido cambios significativos que representan avances importantes en el proyecto. Antes de fusionarlos, se ha realizado una revisión exhaustiva del código para asegurar la calidad, detectar posibles errores y mantener la coherencia con el resto del sistema.

Además, se han utilizado los *issues* para registrar y rastrear tareas pendientes, reportar errores y proponer nuevas funcionalidades. Cada *issue* ha estado asociado a etiquetas (*labels*) y se ha asignado a hitos (*milestones*) correspondientes, mejorando la organización y priorización del trabajo.

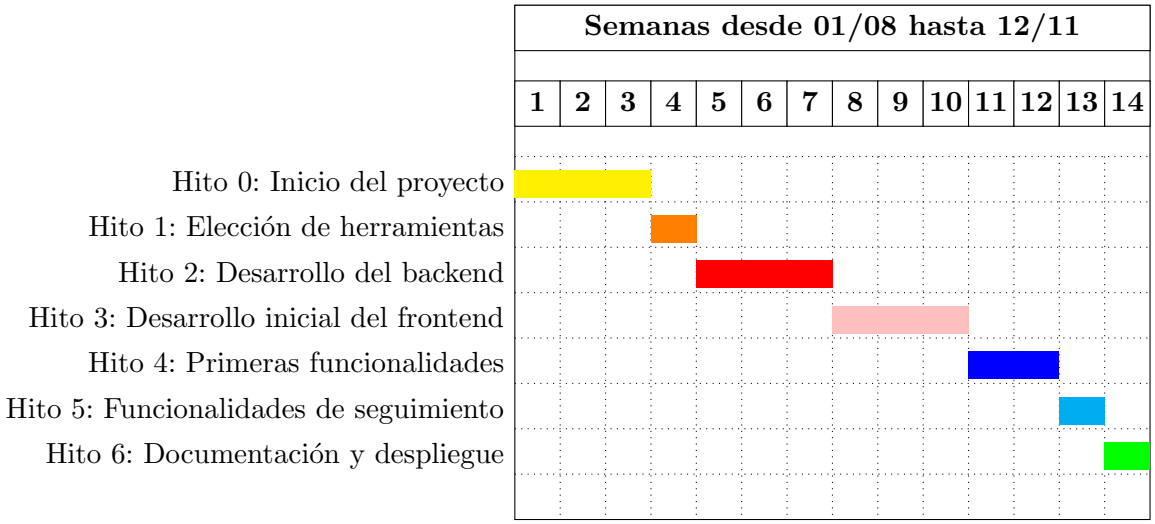

Figura 2: Diagrama de Gantt.

Capítulo 4

Diseño general y herramientas

4.1. Entorno de diseño general

Tras analizar detenidamente las especificaciones y los requisitos del sistema, se ha decidido desarrollar el proyecto en un entorno web en lugar de crear una aplicación nativa para dispositivos móviles. Se presentan las razones que sustentan esta decisión:

1. **Accesibilidad multidispositivo y compatibilidad (RFN4.3)** Un entorno web permite que el sistema sea accesible desde una variedad de dispositivos, como ordenadores de sobremesa, portátiles, tabletas y una amplia gama de *smartphones*. Esto es especialmente relevante porque tanto el personal sanitario como los pacientes pueden utilizar diferentes dispositivos para acceder al sistema. La compatibilidad con múltiples plataformas facilita el cumplimiento del requisito de usabilidad y ofrece flexibilidad de acceso desde cualquier lugar.

2. **Facilidad de actualización y mantenimiento (RFN7)**

   El desarrollo web centraliza el mantenimiento y las actualizaciones del sistema. Cualquier mejora o corrección se implementa directamente en el servidor, y los cambios se reflejan de inmediato para todos los usuarios sin que se requieran acciones adicionales por su parte. Esto cumple con el requisito de mantenimiento, que exige actualizaciones sin interrupciones en las operaciones críticas.

3. **Usabilidad para el personal sanitario (RFN4)**

   El personal sanitario en las farmacias está acostumbrado a utilizar sistemas basados en navegadores web en ordenadores de sobremesa o tablets. Una interfaz web intuitiva facilita la adopción del sistema y reduce la curva de aprendizaje, alineándose con el requisito de usabilidad que considera el nivel de familiaridad con las tecnologías existentes.

4. **Seguridad y cumplimiento normativo (RFN1 y RFN6)**

   Dado que el sistema manejará información médica y personal sensible, la seguridad es un aspecto fundamental. Un entorno web permite implementar medidas de seguridad robustas en el servidor, como cifrado de datos y autenticación segura, controlando eficazmente el acceso y protegiendo los datos conforme a normativas vigentes como el Reglamento General de Protección de Datos (RGPD). El control centralizado facilita el cumplimiento normativo y la aplicación coherente de políticas de seguridad.

5. **Disponibilidad y rendimiento (RFN2 y RFN3)**

   Un sistema web alojado en servidores confiables puede garantizar una alta disponibilidad y un rendimiento óptimo, aspectos esenciales para las operaciones continuas en las farmacias comunitarias. La infraestructura puede escalarse según sea necesario para manejar altos volúmenes de transacciones y consultas sin afectar el rendimiento, satisfaciendo así los requisitos de disponibilidad y rendimiento.

6. **Escalabilidad e integración (RFN5)**

   El entorno web facilita la escalabilidad del sistema para adaptarse al crecimiento en el número de usuarios y farmacias integradas. Nuevas farmacias y personal pueden añadirse sin necesidad de desplegar o actualizar aplicaciones en múltiples dispositivos, lo que agiliza la expansión y adopción del sistema.

7. **Gestión centralizada de usuarios (RF6)**

   La gestión de usuarios es más eficiente en un entorno web, donde las cuentas y permisos pueden administrarse de forma centralizada. Esto permite mantener un control adecuado sobre el acceso al sistema y garantiza que sólo el personal autorizado tenga acceso a la información sensible.

8. **Acceso para pacientes y tutores o acompañantes**

   Dado que el sistema también será utilizado por pacientes y, en algunos casos, por sus tutores o acompañantes, el acceso web elimina la necesidad de descargar e instalar una aplicación móvil. Esto facilita la participación de los pacientes en el seguimiento de sus tratamientos y mejora la comunicación con el personal sanitario.

9. **Costos y tiempo de desarrollo**

   El desarrollo de una aplicación web suele ser más rápido y económico que crear aplicaciones nativas para múltiples plataformas móviles (iOS, Android). Esto permite optimizar los recursos y cumplir con los plazos del proyecto sin sacrificar la calidad o funcionalidad del sistema.

10. **Evitar dependencia de plataformas y tiendas de aplicaciones**

    Las aplicaciones móviles dependen de las tiendas de aplicaciones para su distribución, lo que puede introducir retrasos y requisitos adicionales para su aprobación. Además, existen políticas y restricciones que pueden afectar la implementación de ciertas funcionalidades. Un entorno web elimina estas dependencias, permitiendo un control total sobre el despliegue y las actualizaciones del sistema.

11. **Facilidad de integración con sistemas existentes**

    Las farmacias pueden contar con sistemas existentes con los que el nuevo sistema debe interactuar. Las aplicaciones web son más flexibles para integrarse con otros sistemas y bases de datos, facilitando la interoperabilidad y cumpliendo con los requisitos de escalabilidad e integración.

12. **Consistencia en la experiencia de usuario**

    Un diseño web responsivo garantiza una experiencia de usuario consistente en todos los dispositivos. Esto es importante para mantener la eficiencia en las operaciones diarias del personal sanitario y en la interacción de los pacientes con el sistema.

4.2. Selección de herramientas y tecnologías para el desarrollo

En esta sección se presenta la elección de las herramientas y tecnologías que se utilizarán para el desarrollo del sistema, basándose en los requisitos funcionales y no funcionales establecidos. La selección busca cumplir con los objetivos de eficiencia, escalabilidad y seguridad, según las necesidades específicas del proyecto.

4.2.1. Base de datos: PostgreSQL

Para almacenar y gestionar los datos se ha seleccionado **PostgreSQL**, un sistema de gestión de bases de datos relacionales de código abierto que garantiza el cumplimiento de los estándares SQL, asegurando la integridad y consistencia de los datos. Esta característica es especialmente relevante para manejar información médica (RFN1, RFN2, RFN3, RFN6). Además, su extensibilidad permite definir tipos de datos personalizados y funciones que facilitan la adaptación a las necesidades específicas del sistema, garantizando así su escalabilidad (RFN5). El soporte comunitario de PostgreSQL también asegura que la base de datos reciba actualizaciones y mantenimiento constantes, alineándose con los requisitos de mantenimiento (RFN7).

4.2.2. Contenedor de base de datos: Docker Compose

Para simplificar la configuración, despliegue y mantenimiento de la base de datos **PostgreSQL** se ha utilizado un contenedor Docker Compose, mediante el cual se permite una fácil replicación del entorno de desarrollo y se asegura que el sistema pueda desplegarse de forma consistente en distintos entornos. Esto contribuye al cumplimiento de los requisitos de disponibilidad (RFN2) y escalabilidad (RFN5), permitiendo que el sistema se adapte a nuevas farmacias y usuarios sin interrupciones en el servicio.

4.2.3. ORM: PrismaJS

Para interactuar con la base de datos de manera eficiente y segura se ha optado por **PrismaJS** como el ORM (Object-Relational Mapping) del sistema. Esta herramienta proporciona tipado estático y autocompletado, lo cual minimiza errores y mejora la productividad en el desarrollo (RFN7, RFN4). También incluye funcionalidades para gestionar migraciones de esquema de manera segura y eficiente, lo que se ha usado para el mantenimiento y actualización de la estructura de la base de datos. Asimismo, sus consultas optimizadas contribuyen al rendimiento del sistema (RFN3).

4.2.4. Entorno de ejecución: Node.js

Node.js se ha elegido como entorno de ejecución del servidor debido a su modelo basado en eventos, el cual permite manejar un gran número de conexiones simultáneas sin degradar el rendimiento. De esta forma se cumple con los requisitos de rendimiento y escalabilidad (RFN3, RFN5) del sistema. Este entorno permite unificar el stack tecnológico en JavaScript tanto para el lado del cliente como del servidor, mejorando así la productividad y alineándose con los requisitos de mantenimiento (RFN7).

4.2.5. Framework Backend: Fastify

Para el desarrollo del backend se ha optado por **Fastify**, un framework para Node.js que destaca por resultar una solución ideal para sistemas que requieren tiempos de respuesta mínimos y eficiencia en el procesamiento, satisfaciendo los requisitos de rendimiento del proyecto (RFN3). Una característica que ha resultado primordial ha sido la estructura modular que presenta Fastify gracias a su sistema de plugins, facilitando la expansión del sistema conforme crezca en funcionalidad y permitiendo un fácil mantenimiento (RFN5, RFN7). Además, Fastify incorpora validación de esquemas JSON, lo cual refuerza la seguridad y confiabilidad de las API, alineándose con los requisitos de seguridad (RFN1).

4.2.6. Generador de sitios estáticos: Astro

Para el desarrollo de la interfaz de usuario y generación de contenido estático se ha seleccionado **Astro** como generador de sitios estáticos. Astro se ha diseñado para crear sitios web rápidos y optimizados, lo cual mejora la experiencia del usuario y cumple con los requisitos de rendimiento (RFN3) al minimizar la carga de JavaScript en el cliente, lo que se conoce como *Server Side Rendering*. Astro permite utilizar componentes de diferentes frameworks, como React, ofreciendo flexibilidad en el desarrollo del frontend. Esta característica permite construir interfaces de usuario modernas y responsivas, alineándose con los requisitos de usabilidad (RFN4).

4.2.7. Biblioteca frontend: React

Para añadir interactividad en la interfaz de usuario se ha elegido **React**, una biblioteca de JavaScript ampliamente adoptada, la cual ofrece un enfoque basado en componentes a la par que facilita la creación de interfaces reutilizables y mantenibles (RFN7).

4.2.8. Framework de diseño: Tailwind CSS

Para el diseño y estilización de la interfaz se ha optado por **Tailwind CSS**. Este framework permite crear diseños gracias a su sistema de utilidades predefinidas, característica que simplifican la personalización de estilos.

Capítulo 5

Desarrollo e implementación

Una vez elegidas todas las herramientas y tecnologías necesarias para el desarrollo del sistema, se procede a la fase de implementación. En esta sección se detallan los componentes del backend, así como el uso de contenedores Docker para la base de datos y su administración.

5.1. Desarrollo del backend

5.1.1. Docker

Para facilitar la gestión y despliegue de la base de datos se ha configurado **PostgreSQL** en un contenedor Docker, acompañado de **pgAdmin** como herramienta de administración de base de datos. Esta estructura en contenedores asegura que el entorno sea replicable y consistente en distintas fases del desarrollo y despliegue.

La configuración de estos servicios en Docker se define en un archivo 'docker-compose.yml', que permite instanciar tanto la base de datos como la interfaz de administración en contenedores separados pero conectados entre sí. A continuación, se presenta el contenido del archivo 'docker-compose.yml' utilizado para la configuración:

Listing 5.1: Archivo docker-compose.yml para PostgreSQL y pgAdmin

```
1  version: '3.8'
2
3  services:
4  db:
5  image: postgres
6  restart: always
7  shm_size: 128mb
8  environment:
9  POSTGRES_USER: root
10 POSTGRES_PASSWORD: root
11 POSTGRES_DB: postgres
12 ports:
13 - "5432:5432"
14 volumes:
15 - postgres_data:/var/lib/postgresql/data
16
17 pgadmin:
18 image: dpage/pgadmin4
19 restart: always
20 environment:
21 PGADMIN_DEFAULT_EMAIL=admin@admin.com
22 PGADMIN_DEFAULT_PASSWORD=root
```

```
23          ports:
24          - "5050:80"
25          volumes:
26          - pgadmin_data:/var/lib/pgadmin
27
28          volumes:
29          postgres_data:
30          driver: local
31          pgadmin_data:
32          driver: local
```

El servicio db se encarga de ejecutar la instancia de PostgreSQL configurada para estar accesible en el puerto 5432 de la máquina local. Además, utiliza un volumen persistente `postgres` data para asegurar que los datos de la base de datos se mantengan incluso si el contenedor se detiene o reinicia.

Por otro lado, el servicio pgadmin proporciona una interfaz gráfica para la administración de la base de datos, accesible en el puerto 5050 de la máquina local. Este servicio también utiliza un volumen persistente `pgadmin` data para almacenar sus configuraciones de forma persistente.

Esta configuración en Docker permite separar los entornos de desarrollo y producción, facilitando la administración y el despliegue seguro de la base de datos y sus herramientas de administración.

5.1.2. Prisma

Para llevar a cabo la conexión entre Prisma y el contenedor de Docker donde está desplegada la base de datos PostgreSQL, se ha creado el archivo schema.prisma. Este archivo define el modelo de datos utilizado en el sistema y permite a Prisma interactuar con la base de datos de manera estructurada y eficiente.

Después de configurar el esquema en schema.prisma, se procedió a instalar el paquete `@prisma/client`. Este cliente permite realizar consultas a la base de datos de forma sencilla y directa desde el código de Node.js, utilizando Prisma como ORM. La instalación y configuración del cliente Prisma permitieron verificar que la conexión con PostgreSQL y con pgAdmin estaba funcionando correctamente, lo que permitió visualizar y administrar los datos almacenados tanto en pgAdmin como en Prisma.

La verificación de esta conexión incluyó la comprobación de que los datos ingresados o modificados en la base de datos a través de Prisma aparecían correctamente en pgAdmin y viceversa, confirmando así la correcta sincronización y funcionalidad de ambos entornos. Como resultado de esta configuración y de la estructura de schema.prisma, se obtuvo la estructura de tablas que se muestra en la imagen, donde se representa el modelo de datos en sus diferentes entidades y relaciones, necesarias para cumplir con los requisitos del sistema.

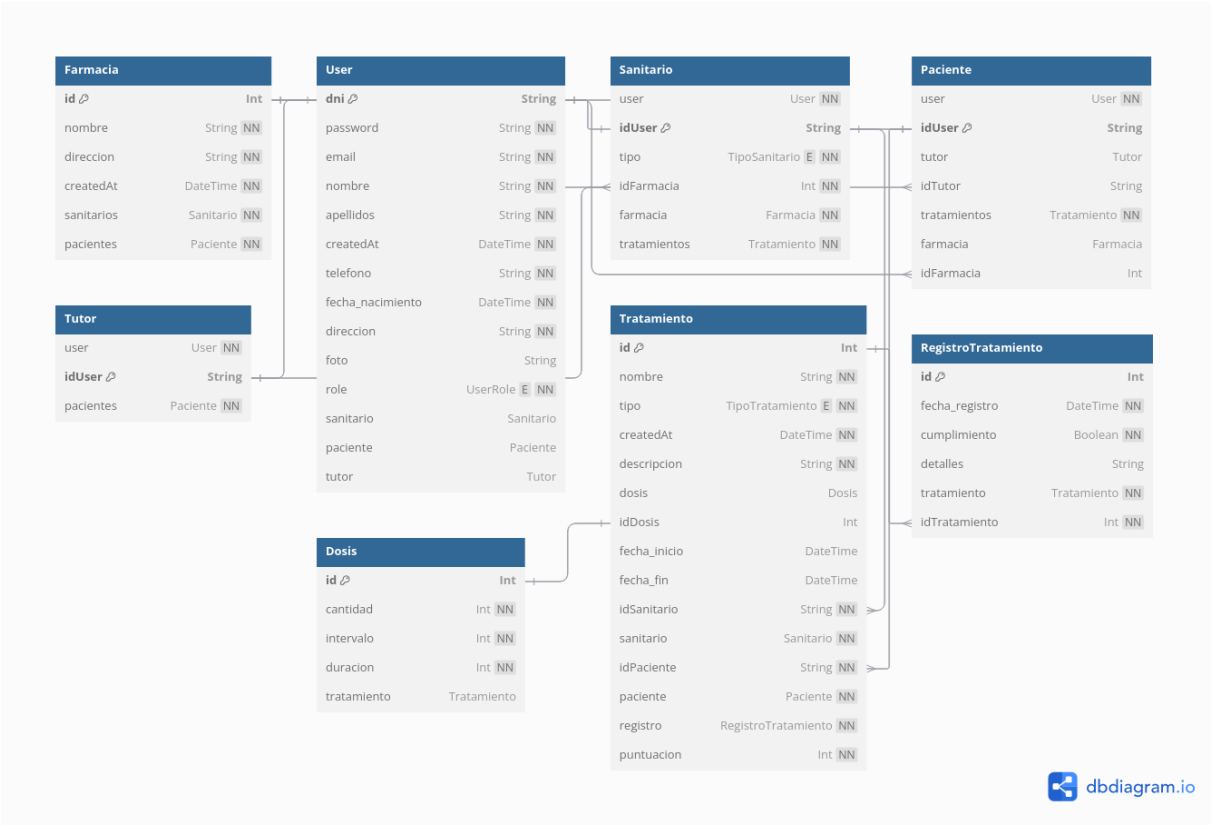

Figura 3: Tablas de la base de datos.

El modelo de datos establece varias relaciones clave: una farmacia puede tener múltiples sanitarios y pacientes asociados, lo que organiza a ambos grupos bajo un establecimiento específico. Cada sanitario y paciente está registrado como usuario en el sistema, permitiendo gestionar distintos roles y accesos. Los tutores pueden estar vinculados a pacientes que requieren asistencia, facilitando el seguimiento. Además, un sanitario es responsable de los tratamientos de los pacientes, cada uno de los cuales tiene su pauta de dosis y registros de adherencia, lo que permite monitorizar la evolución del tratamiento.

5.1.3. Fastify, desarrollo de API

Para configurar Fastify en este proyecto se han establecido cinco componentes principales: controladores, plugins, rutas, tests y utils. A continuación, se explica brevemente la función de cada uno y se muestra un ejemplo de código asociado.

- **Controladores**. Los controladores gestionan la lógica de negocio de las operaciones solicitadas por el cliente. En este caso, el controlador maneja la creación de usuarios, validando roles y permisos de acceso según el tipo de usuario que solicita la acción.

Listing 5.2: Ejemplo de controlador para crear un usuario

```
1  const createUser = async (req, reply) => {
2      try {
3          const { dni, email, password, nombre, apellidos, fechaNac,
               telefono, direccion, role, tipoSanitario, idFarmacia,
               foto, dniPaciente } = req.body
4          const userCreator = req.user
```

```
5
6                       const existingUser = await prisma.user.findUnique({
7                               where: { dni }
8                       })
9
10                      const fechaNacimiento = new Date(fechaNac)
11                      if (isNaN(fechaNacimiento.getTime())) {
12                              return reply.status(400).send({ error: 'Fecha de
                                    nacimiento no es válida.' })
13                      }
14
15                      if (existingUser) return reply.status(400).send({ error: '
                            Ya existe un usuario con este DNI.' })
16
17                      let allowedRoles = []
18
19                      if (userCreator.role === ROLES.ADMIN) {
20                              allowedRoles = createUserPermissions[ROLES.ADMIN]
21                      } else if (userCreator.role === ROLES.SANITARIO) {
22                              if (!userCreator.sanitario) {
23                                      userCreator.sanitario = await prisma.
                                            sanitario.findUnique({
24                                              where: { idUser: userCreator.dni }
25                                      })
26
27                                      if (!userCreator.sanitario)
28                                      return reply.status(400).send({ error: 'El
                                            usuario sanitario no tiene información
                                            de tipo de sanitario.' })
29
30                              }
31                              allowedRoles = createUserPermissions[userCreator.
                                    sanitario.tipo] || []
32
33                      } else {
34                              return reply.status(403).send({ error: '
                                    UNAUTHORIZED. Only an ADMIN or SANITARIO can
                                    create users.' })
35                      }
36
37                      if (!allowedRoles.includes(role)) {
38                              return reply.status(403).send({ error: '
                                    UNAUTHORIZED. You are not allowed to create a
                                    user with this role.' })
39                      }
40
41                      if (role === ROLES.SANITARIO && !tipoSanitario) {
42                              return reply.status(400).send({ error: 'El tipo de
                                    sanitario es requerido.' })
43                      }
44
45                      if (role === ROLES.SANITARIO && !Object.values(
                            TIPO_SANITARIO).includes(tipoSanitario)) {
46                              return reply.status(400).send({ error: 'El tipo de
                                    sanitario no es válido.' })
47                      }
48
49                      if (role === ROLES.SANITARIO && !idFarmacia) {
50                              return reply.status(400).send({ error: 'El id de la
                                     farmacia es requerido para sanitarios.' })
51                      }
52
53                      if (idFarmacia) {
54                              const farmacia = await prisma.farmacia.findUnique({
55                                      where: { id: idFarmacia }
56                              })
57                              if (!farmacia) return reply.status(400).send({
                                    error: 'La farmacia no existe.' })
58                      }
59
```

```
60                          if (role === ROLES.TUTOR) {
61                                  if (!dniPaciente) {
62                                          return reply.status(400).send({ error: 'El
                                                DNI del paciente es requerido.' })
63                                  }
64
65                                  const existingPatient = await prisma.paciente.
                                        findUnique({
66                                          where: { idUser: dniPaciente }
67                                  })
68                                  if (!existingPatient) return reply.status(400).send
                                        ({ error: 'El paciente no existe.' })
69                          }
70
71                          const hashedPassword = await hashPassword(password)
72
73                          const newUser = await prisma.user.create({
74                                  data: {
75                                          dni,
76                                          email,
77                                          password: hashedPassword,
78                                          nombre,
79                                          apellidos,
80                                          fecha_nacimiento: fechaNacimiento,
81                                          telefono,
82                                          direccion,
83                                          foto,
84                                          role,
85                                          sanitario: role === ROLES.SANITARIO ? {
86                                                  create: {
87                                                          tipo: tipoSanitario,
88                                                          idFarmacia
89                                                  }
90                                          } : undefined,
91                                          tutor: role === ROLES.TUTOR ? {
92                                                  create: {
93                                                          pacientes: { connect: {
                                                                idUser: dniPaciente } }
94                                                  }
95                                          } : undefined,
96                                          paciente: role === ROLES.PACIENTE ? {
97                                                  create: {
98                                                          idFarmacia
99                                                  }
100                                         } : undefined
101                                 }
102                         })
103
104                         if (role === ROLES.TUTOR) {
105                                 await prisma.paciente.update({
106                                         where: { idUser: dniPaciente },
107                                         data: { idTutor: newUser.dni }
108                                 })
109                         }
110
111                         return reply.status(201).send({ message: 'User created
                                successfully.', user: newUser })
112
113                 } catch (error) {
114                         console.error(error)
115                         reply.status(500).send({ error: 'Error creating user.' })
116                 }
117         }
```

- **Plugins o *Middleware***. Los plugins extienden la funcionalidad de Fastify. En este ejemplo, se ha configurado un plugin para manejar JWT (JSON Web Tokens) y así asegurar rutas.

Listing 5.3: Plugin para JWT

```
import fp from 'fastify-plugin'
import jwt from '@fastify/jwt'

export default fp(async function (fastify, options) {
        fastify.register(jwt, {
                secret: process.env.JWT_SIGNING_SECRET
        })

        fastify.decorate("jwtAuth", async function (request, reply) {
                try {
                        await request.jwtVerify()
                } catch (err) {
                        reply.status(401).send({ err })
                }
        })

})
```

- **Rutas**. Las rutas definen los puntos de acceso a los controladores. Se utiliza preValidation con jwtAuth para proteger las rutas, asegurando que solo usuarios autenticados puedan acceder.

Listing 5.4: Definición de rutas para usuarios

```
import userController from '../controllers/userController.js'

async function userRouters(fastify, options) {
        fastify.post('/users/create', { preValidation: [fastify.jwtAuth] }, userController.createUser)
        fastify.get('/users/:dni', { preValidation: [fastify.jwtAuth] }, userController.getUserByDNI)
        fastify.get('/users', { preValidation: [fastify.jwtAuth] }, userController.getAllUsers)
        fastify.put('/users/:dni', { preValidation: [fastify.jwtAuth] }, userController.updateUser)
        fastify.delete('/users/:dni', { preValidation: [fastify.jwtAuth] }, userController.deleteUser)
        fastify.get('/users/sanitarios/:dni', { preValidation: [fastify.jwtAuth] }, userController.getSanitarioData)
        fastify.get('/users/pacientes/:dni', { preValidation: [fastify.jwtAuth] }, userController.getPacienteData)
        fastify.get('/users/tutores/:dni', { preValidation: [fastify.jwtAuth] }, userController.getTutorData)
        fastify.get('/users/pacientes/sinfarmacia', { preValidation: [fastify.jwtAuth] }, userController.getPacientesSinFarmacia)
        fastify.put('/users/:dni/password', { preValidation: [fastify.jwtAuth] }, userController.updatePassword)
}

export default userRouters
```

- **Tests**. Los tests validan la funcionalidad del sistema, asegurando que las operaciones cumplen con las expectativas. En este ejemplo, se verifica que un farmacéutico pueda crear un tratamiento farmacológico para un paciente.

Listing 5.5: Ejemplo de test para creación de tratamiento

```
it('Debería permitir al farmacéutico crear un tratamiento farmacológico para un paciente', async () => {
    const farmaceuticoLogin = await fastify.inject({
        method: 'POST',
        url: '/api/login',
        payload: { dni: '33445566G', password: 'farmaceutico567' }
    })
    expect(farmaceuticoLogin.statusCode).to.equal(200)
    const farmaceuticoToken = JSON.parse(farmaceuticoLogin.payload).token

    const tratamientoFarmacologicoResponse = await fastify.inject({
        method: 'POST',
        url: '/api/tratamientos/create',
        headers: { Authorization: `Bearer ${farmaceuticoToken}` },
        payload: {
            nombre: 'Tratamiento Antibiotico',
            tipo: 'FARMACOLOGICO',
            descripcion: 'Tratamiento para infección',
            idPaciente: pacienteId1,
            dosis: {
                cantidad: 500,
                intervalo: 8,
                duracion: 10
            }
        }
    })
    expect(tratamientoFarmacologicoResponse.statusCode).to.equal(201)
    const tratamiento = JSON.parse(tratamientoFarmacologicoResponse.payload)
    tratamientoId1 = tratamiento.id
})
```

- **Utils**. Los utils agrupan funciones de utilidad que se utilizan en distintos puntos del código. En este caso, se incluyen funciones para encriptar y comparar contraseñas.

Listing 5.6: Funciones de utilidad para contraseñas

```
import bcrypt from 'bcrypt';

const SALT_ROUNDS = 12

export async function hashPassword(password) {
    try {
        return await bcrypt.hash(password, SALT_ROUNDS)
    } catch (error) {
        console.error('Error hashing password:', error)
        throw new Error('Failed to hash password')
    }
}

export const comparePassword = async (password, hashedPassword) => {
    return bcrypt.compare(password, hashedPassword)
}
```

- **Configuración del servidor**. En el archivo `server.js` se configura el servidor Fastify, que actúa como el núcleo de la aplicación, integrando diferentes funcionalidades a través

de plugins, rutas y configuraciones específicas.

En `server.js` primero se importa y configura Fastify, iniciando el servidor con un logger detallado para rastrear eventos y errores. Luego, se registra el plugin cors, permitiendo el acceso desde distintas fuentes con los métodos HTTP especificados. También se habilita el plugin JWT para manejar la autenticación con tokens, asegurando que solo los usuarios autorizados puedan acceder a ciertas rutas.

A continuación, se registra un parser para solicitudes en formato JSON. Después se configuran las rutas principales del servidor (farmaciaRoutes, userRoutes, tratamientoRoutes y authRoutes), todas accesibles bajo el prefijo /api. Finalmente, se define una ruta por defecto para probar el funcionamiento del servidor y se implementa una función para iniciar el servidor en el puerto indicado, gestionando cualquier error que ocurra al iniciar.

Listing 5.7: Archivo server.js - Configuración del servidor Fastify

```
import Fastify from 'fastify'
import { PORT } from './config/init.js'
import cors from '@fastify/cors';
import farmaciaRoutes from './routes/farmaciaRoutes.js'
import userRoutes from './routes/userRoutes.js'
import tratamientoRoutes from './routes/tratamientoRoutes.js'
import authRoutes from './routes/authRoutes.js'
import jwtPlugin from './plugins/jwtPlugin.js'

const fastify = Fastify({
        logger: {
                level: 'trace',
                transport: {
                        target: 'pino-pretty',
                }
        }
})


fastify.register(cors, {
        origin: true,
        methods: ['POST', 'GET', 'PUT', 'DELETE', 'OPTIONS'],
})

fastify.register(jwtPlugin)

// Parser para contenido JSON
fastify.addContentTypeParser('application/json', { parseAs: 'string' },
    fastify.getDefaultJsonParser('strict'))

fastify.register(farmaciaRoutes, { prefix: '/api' })
fastify.register(userRoutes, { prefix: '/api' })
fastify.register(tratamientoRoutes, { prefix: '/api' })
fastify.register(authRoutes, { prefix: '/api' })

// Ruta por defecto
fastify.get('/', async function handler (request, reply) {
        return { hello: 'world' }
})

const startServer = async () => {
        try {
                await fastify.listen({ port: PORT })
                console.log(`Server is running on port ${ PORT }`)

        } catch (err) {
                fastify.log.error(err)
                process.exit(1)
        }
```

```
49          }
50
51          export default fastify
52          startServer()
```

5.1.4. Dependencias de Node.js

Las dependencias son módulos externos que un proyecto necesita para funcionar correctamente. Estas se especifican en el archivo `package.json` bajo las secciones `dependencies` y `devDependencies`.

- **@fastify/cors**: Permite configurar CORS (Cross-Origin Resource Sharing) en Fastify, lo que habilita solicitudes desde distintos dominios y métodos HTTP específicos.
- **@fastify/jwt**: Gestiona la autenticación mediante JSON Web Tokens (JWT), proporcionando seguridad al sistema al verificar usuarios.
- **@prisma/client**: Cliente de Prisma para interactuar con la base de datos para realizar consultas y operaciones sobre el esquema definido en `schema.prisma`.
- **bcrypt**: Utilizado para encriptar contraseñas.
- **fastify-plugin**: Facilita la creación y uso de plugins en Fastify.
- **pino-pretty**: Formatea los logs generados por Fastify para hacerlos más legibles durante el desarrollo.
- **chai**: Biblioteca de aserciones para escribir pruebas, proporcionando múltiples estilos de aserción para verificar resultados.
- **mocha**: Marco de pruebas que permite ejecutar y estructurar pruebas, facilitando la verificación de la funcionalidad del sistema.
- **prisma-dbml-generator**: Genera diagramas de entidad-relación (ERD) en formato DBML, visualizando el modelo de datos definido en Prisma.

5.1.5. Flujo de peticiones a API

El flujo de una petición a la API en Fastify comienza cuando un cliente envía una solicitud a una ruta específica, como POST `/api/users/create`. Primero, la ruta definida en el archivo de rutas (por ejemplo, userRoutes.js) intercepta la solicitud. Si la ruta está protegida, se aplica una pre-validación, utilizando el plugin de autenticación JWT (como jwtPlugin.js), que verifica que el cliente esté autenticado. Una vez autenticado, el controlador correspondiente (en este caso, createUser en userController.js) se encarga de procesar la lógica principal de la solicitud, como validar datos y realizar operaciones en la base de datos usando Prisma. Finalmente, la respuesta se envía al cliente, indicando el éxito o el error de la operación. Este flujo asegura un manejo estructurado y seguro de las solicitudes en la API.

5.1.6. Organización de directorios

Finalmente, la estructura de directorios para el backend se muestra como sigue:

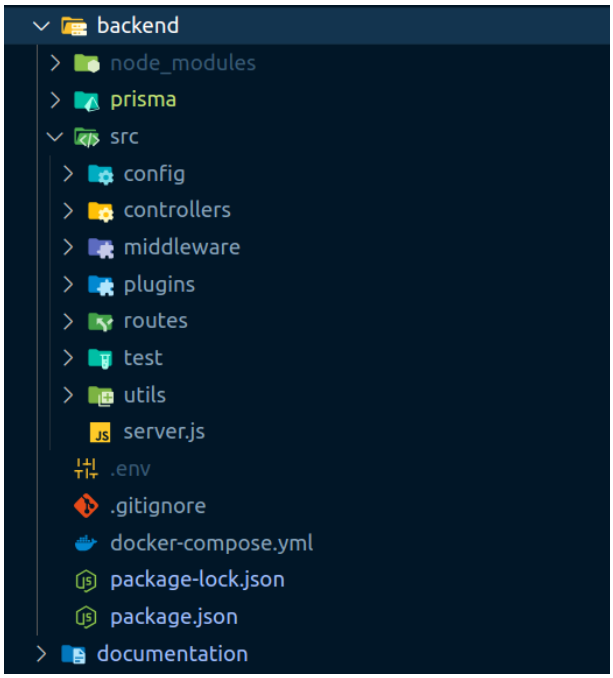

Figura 4: Directorios del backend.

5.2. Desarrollo del frontend

5.2.1. Desarrollo del frontend con Astro

En el desarrollo del frontend se ha utilizado Astro y se ha configurado para que tenga un layout base que permita establecer una estructura y diseño consistentes en todas las páginas de la aplicación. Este layout se configura en el archivo `Layout.astro` y contiene componentes comunes como la barra de navegación, pie de página y verificación de autenticación.

Configuración del layout base

El layout base incluye los componentes esenciales y define cuándo deben mostrarse, basándose en la ruta actual. A continuación se presenta el código del archivo `Layout.astro`:

Listing 5.8: Layout base en Astro

```
---
import Navbarreact from "../components/ui/Navbar";
import Footer from "../components/ui/footer.astro";
import AuthCheck from "../components/sections/checks/AuthCheck";

interface Props {
        title: string;
}

```

```
10          const { title } = Astro.props;
11          const currentPath = Astro.url.pathname;
12          const showNavbar = currentPath !== "/" && currentPath !== "/login";
13          const showLimit = showNavbar;
14          const authCheck = showNavbar;
15          ---
16
17          <!doctype html>
18          <html lang="es" class="lenis lenis-smooth scroll-pt-5">
19          <head>
20          <meta charset="UTF-8" />
21          <meta name="description" content="Astro description" />
22          <meta name="viewport" content="width=device-width" />
23          <link rel="icon" type="image/svg+xml" href="/favicon.svg" />
24          <meta name="author" content="VIDICAN, MIHNEA IOAN" />
25          <meta name="generator" content={Astro.generator} />
26          <title>{title}</title>
27          </head>
28          <style>
29          html {
30                  overflow-y: scroll;
31          }
32          </style>
33          <body class="bg-base-100 w-full">
34          <div class="grid min-h-screen grid-rows-[auto_1fr_auto]">
35          {showNavbar && <Navbarreact client:load />}
36          {authCheck && <AuthCheck client:load/>}
37
38          <main>
39          <div class={showLimit ? "max-w-screen-xl mx-auto" : ""}>
40          <slot />
41          </div>
42          </main>
43
44          <Footer />
45          </div>
46          </body>
47          <script>
48          import "../../assets/scripts/lenisSmoothScroll.js";
49          </script>
50          </html>
```

En este layout, se configuran los componentes comunes `Navbarreact`, `Footer`, y `AuthCheck`, que se muestran solo en determinadas rutas. La etiqueta `<slot />` permite incluir el contenido de cada vista específica dentro de esta estructura.

Uso del layout en una vista específica

Para cada página, se utiliza el layout importándolo y envolviendo el contenido dentro de él. A continuación, se muestra cómo se utiliza el layout en la página de inicio de sesión, `login.astro`:

Listing 5.9: Uso del Layout en login.astro

```
1           ---
2           import Layout from "../layouts/Layout.astro";
3           import LoginForm from "../components/sections/LoginForm.jsx";
4           ---
5
6           <Layout title="Inicio de sesión">
7           <LoginForm client:load />
8           </Layout>
```

- Aquí, el componente `LoginForm` se incluye dentro del layout, configurando así la página

de inicio de sesión con el mismo estilo y estructura que el layout base.

- Esto permite mantener una estructura común en toda la aplicación.

5.2.2. Uso de React

Para añadir interactividad en el frontend, como se ha mencionado previamente, se ha utilizado **React**. A continuación se muestra un ejemplo del archivo `TratamientosForm.jsx`, un componente que permite a un sanitario, bien sea un farmacéutico o técnico de farmacia, crear un tratamiento específico para un paciente. Este formulario incluye validación de datos y mensajes de éxito o error según el resultado de la petición.

Listing 5.10: TratamientoForm.jsx - Formulario de creación de tratamiento para un paciente

```
1  import { useState, useEffect } from "react";
2  import SanitarioCheck from "../checks/SanitarioCheck";
3
4  const TratamientoForm = () => {
5      const [nombre, setNombre] = useState("");
6      const [descripcion, setDescripcion] = useState("");
7      const [tipo, setTipo] = useState("");
8      const [dosis, setDosis] = useState({ cantidad: "", intervalo: "",
           duracion: "" });
9      const [fechaFin, setFechaFin] = useState("");
10     const [idFarmacia, setIdFarmacia] = useState(null);
11     const [pacientes, setPacientes] = useState([]);
12     const [idPaciente, setIdPaciente] = useState("");
13     const [isLoading, setIsLoading] = useState(false);
14     const [errorMessage, setErrorMessage] = useState("");
15     const [successMessage, setSuccessMessage] = useState("");
16
17     useEffect(() => {
18         const fetchSanitarioData = async () => {
19             const user = JSON.parse(sessionStorage.getItem("user"));
20             if (user && user.dni) {
21                 try {
22                     const token = sessionStorage.getItem("
                           jwtToken");
23                     const response = await fetch(`http://
                           localhost:3000/api/users/sanitarios/$
                           {user.dni}`, {
24                         headers: { Authorization: `Bearer
                               ${token}` },
25                     });
26                     const data = await response.json();
27                     if (response.ok) {
28                         setIdFarmacia(data.idFarmacia);
29                         fetchPacientes(data.idFarmacia,
                               token);
30                     } else {
31                         setErrorMessage(data.error || "No
                               se pudo obtener el ID de la
                               farmacia.");
32                     }
33                 } catch (error) {
34                     console.error("Error al obtener los datos
                           del sanitario:", error);
35                     setErrorMessage("Error de conexión al
                           obtener los datos del sanitario.");
36                 }
37             } else {
38                 setErrorMessage("No se pudo obtener el DNI del
                       usuario logeado.");
39             }
40         };
```

```
41
42                      fetchSanitarioData();
43              }, []);
44
45              const fetchPacientes = async (farmaciaId, token) => {
46                      try {
47                              const response = await fetch(`http://localhost:3000/api/
                                    farmacias/${farmaciaId}/pacientes`, {
48                                      method: "GET",
49                                      headers: { Authorization: `Bearer ${token}` },
50                              });
51                              const data = await response.json();
52                              if (response.ok) {
53                                      setPacientes(data);
54                              } else {
55                                      setErrorMessage(data.error || "Error al obtener
                                            los pacientes.");
56                              }
57                      } catch (error) {
58                              console.error("Error fetching pacientes:", error);
59                              setErrorMessage("Hubo un problema con la conexión. Inté
                                    ntelo de nuevo más tarde.");
60                      }
61              };
62
63              const validateForm = () => {
64                      if (!nombre || !descripcion || !tipo || !idPaciente) {
65                              setErrorMessage("Todos los campos son obligatorios.");
66                              return false;
67                      }
68                      if (tipo === "FARMACOLOGICO" && (!dosis.cantidad || !dosis.
                            intervalo || !dosis.duracion)) {
69                              setErrorMessage("Todos los campos de dosis son
                                    obligatorios para un tratamiento farmacológico.");
70                              return false;
71                      }
72                      if (tipo === "NO_FARMACOLOGICO" && !fechaFin) {
73                              setErrorMessage("La fecha de fin es obligatoria para un
                                    tratamiento no farmacológico.");
74                              return false;
75                      }
76                      setErrorMessage("");
77                      return true;
78              };
79
80              const handleSubmit = async (e) => {
81                      e.preventDefault();
82                      if (!validateForm()) return;
83
84                      setIsLoading(true);
85                      setErrorMessage("");
86                      setSuccessMessage("");
87
88                      const data = {
89                              nombre,
90                              descripcion,
91                              tipo,
92                              idPaciente,
93                              dosis: tipo === "FARMACOLOGICO" ? dosis : null,
94                              fecha_fin: tipo === "NO_FARMACOLOGICO" ? fechaFin : null,
95                      };
96
97                      try {
98                              const response = await fetch("http://localhost:3000/api/
                                    tratamientos/create", {
99                                      method: "POST",
100                                     headers: {
101                                             "Content-Type": "application/json",
102                                             Authorization: `Bearer ${sessionStorage.
                                                    getItem("jwtToken")}`,
```

```
103                                         },
104                                         body: JSON.stringify(data),
105                                 });
106
107                                 const responseData = await response.json();
108                                 if (response.ok) {
109                                         setSuccessMessage("Tratamiento creado con éxito")
                                                ;
110                                         setTimeout(() => setSuccessMessage(""), 3000);
111                                         setNombre("");
112                                         setDescripcion("");
113                                         setTipo("");
114                                         setDosis({ cantidad: "", intervalo: "", duracion:
                                                 "" });
115                                         setFechaFin("");
116                                         setIdPaciente("");
117                                 } else {
118                                         setErrorMessage(responseData.error || "Error al
                                                crear el tratamiento");
119                                 }
120                         } catch (error) {
121                                 console.error("Hubo un error de conexión:", error);
122                                 setErrorMessage("Hubo un problema con la conexión. Inté
                                        ntelo de nuevo más tarde.");
123                         } finally {
124                                 setIsLoading(false);
125                         }
126                 };
127
128                 return (
129                 <SanitarioCheck>
130                 <div className="flex justify-center text-center m-10">
131                 <div className="breadcrumbs text-xl">
132                 <ul>
133                 <li><a href="/dashboard">Panel de control</a></li>
134                 <li><a href="/tratamientos">Tratamientos</a></li>
135                 <li>Crear tratamiento a paciente</li>
136                 </ul>
137                 </div>
138                 </div>
139
140                 <div className="flex items-center justify-center m-10">
141                 <div className="card bg-base-100 w-full max-w-xl shadow-2xl p-8">
142                 <form onSubmit={handleSubmit}>
143                 </form>
144                 </div>
145                 </div>
146                 </SanitarioCheck>
147                 );
148         };
149
150         export default TratamientoForm;
```

5.2.3. Uso de Tailwind CSS y DaisyUI

Para la gestión de estilos en el frontend se ha optado por **Tailwind CSS**, un framework CSS que permite aplicar estilos mediante clases predefinidas. Además, se ha integrado la librería **DaisyUI** sobre Tailwind CSS, la cual proporciona componentes predefinidos y personalizables, como botones, tarjetas, formularios, entre otros.

A continuación se presenta un ejemplo de cómo se usa DaisyUI para el componente de tarjeta en el formulario de tratamiento:

Listing 5.11: Ejemplo de componente de tarjeta con DaisyUI

```
<div className="card bg-base-100 w-full max-w-xl shadow-2xl p-8">
<form onSubmit={handleSubmit}>
<div className="form-control mb-4">
<input
type="text"
className="input input-bordered input-primary"
placeholder="Nombre del tratamiento"
value={nombre}
onChange={(e) => setNombre(e.target.value)}
required
/>
</div>

<div className="form-control mb-4">
<textarea
className="textarea textarea-primary"
placeholder="Descripción"
value={descripcion}
onChange={(e) => setDescripcion(e.target.value)}
required
></textarea>
</div>

<div className="form-control mt-4">
<button className="btn btn-primary text-white" disabled={isLoading}>
{isLoading ? (
        <span className="loading loading-dots loading-lg text-white"></span>
        ) : (
        "Crear tratamiento"
        )}
</button>
</div>
</form>
</div>
```

5.2.4. Dependencias adicionales en el Frontend

Para el desarrollo del frontend, además de **Astro** y **DaisyUI**, se han integrado otras dependencias que complementan la funcionalidad y estilización de la aplicación. A continuación se detallan:

- **lenis**: Esta biblioteca permite suavizar el desplazamiento dentro de la aplicación, proporcionando una experiencia de usuario fluida al navegar entre secciones.
- **react-dom**: textttreact-dom Facilita el renderizado de los componentes interactivos de React en el DOM.
- **react-router-dom**: Esta biblioteca gestiona el enrutamiento de la aplicación para permitir la navegación entre diferentes vistas.

5.2.5. Organización de directorios

Finalmente, la estructura de directorios para el frontend se muestra como sigue:

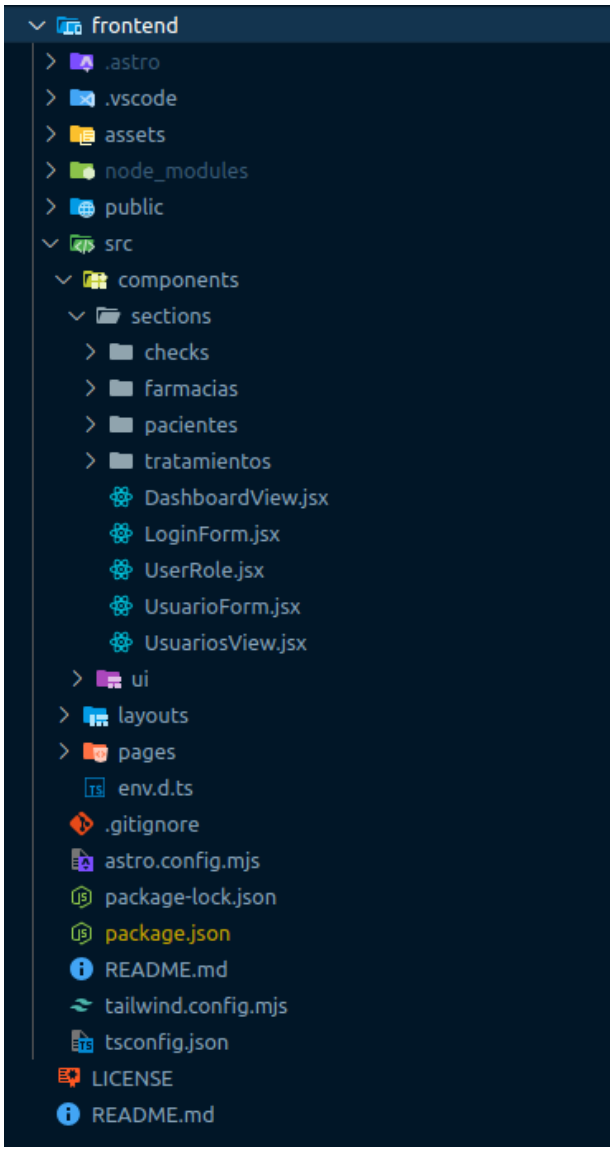

Figura 5: Directorios del frontend.

5.3. Diseño de la interfaz de usuario

. La interfaz de usuario se ha visto inspirada en el aspecto visual de un proyecto de código abierto, ScrewFast [13], el cual combina el uso de Astro y Tailwind CSS para proporcionar una página web rápida y atractiva.

Para PharmAD predomina los colores verde, en diferentes tonalidades, y blanco. Se muestran imágenes acerca de las diferentes secciones de la plataforma.

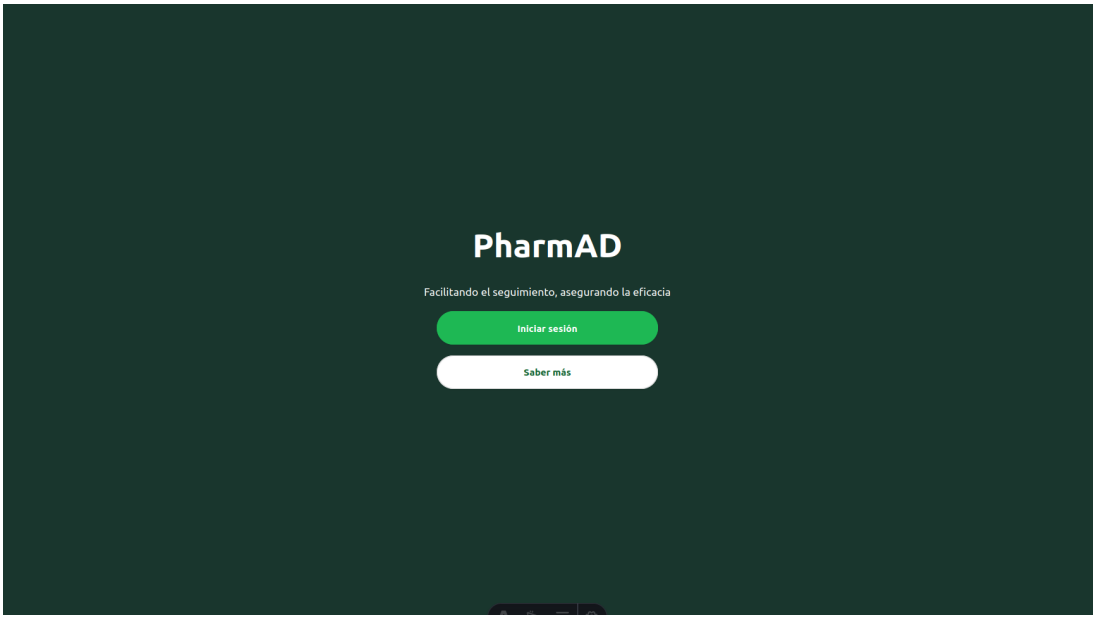

Figura 6: Landing page.

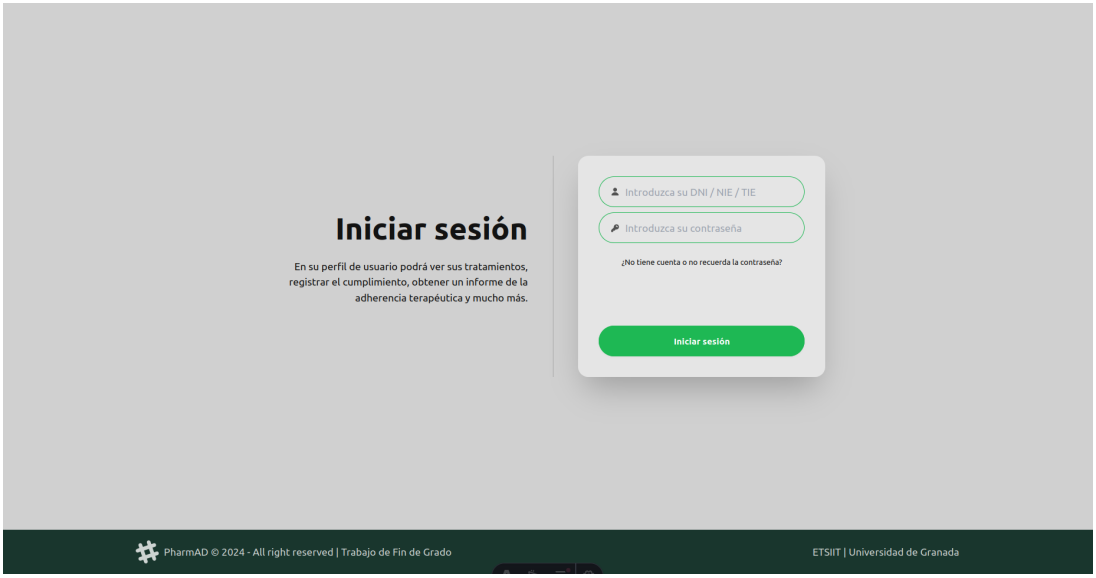

Figura 7: Pantalla de inicio de sesión.

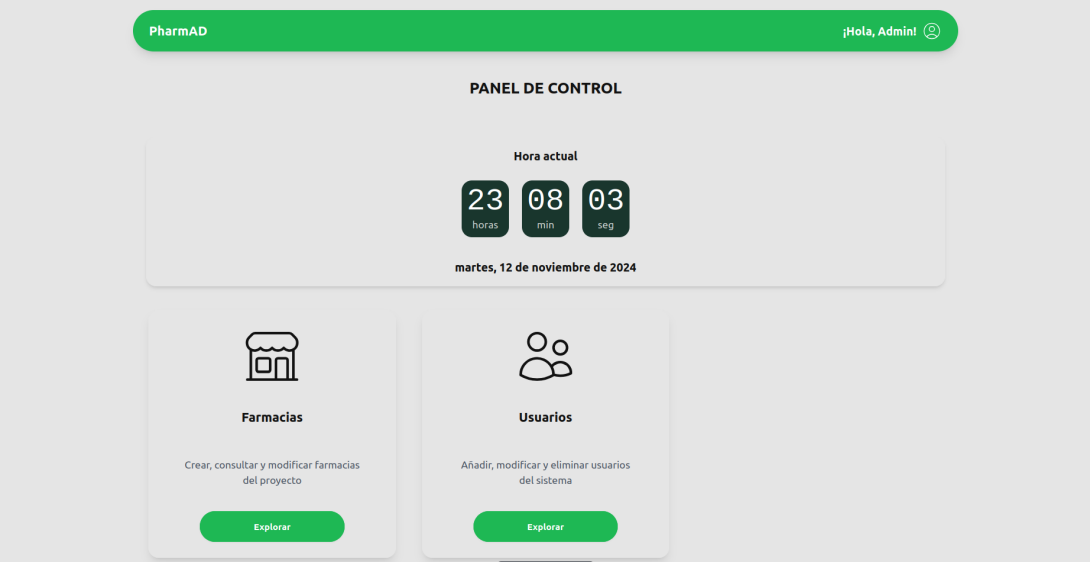

Figura 8: Panel de control del administrador.

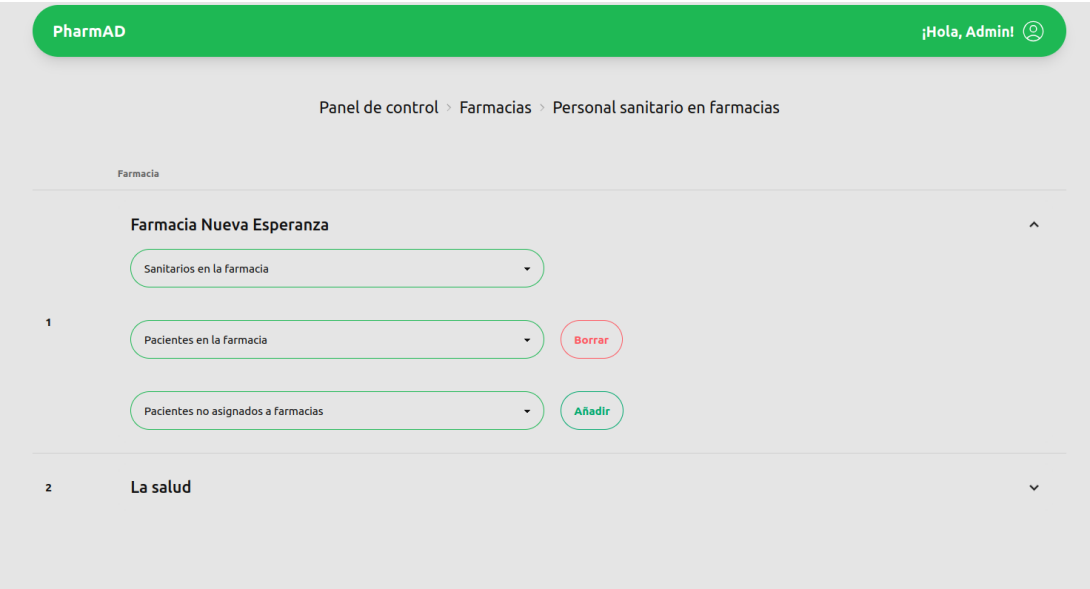

Figura 9: Lista de personal de farmacias para el administrador.

Figura 10: Listado de farmacias para el administrador

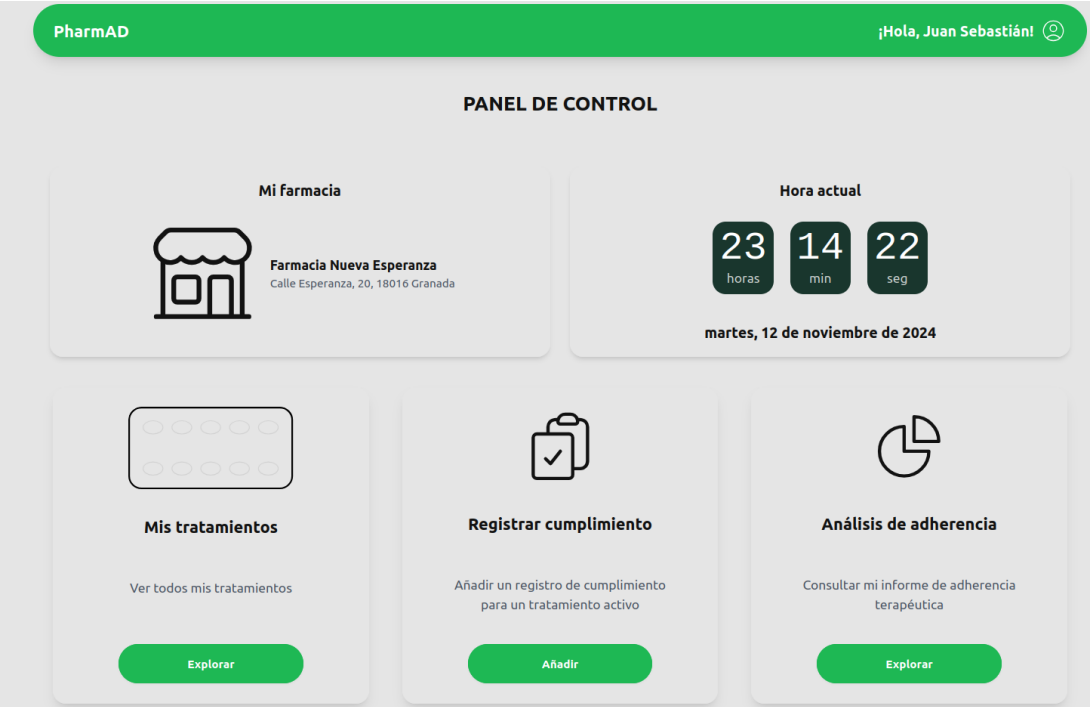

Figura 11: Panel de control del usuario.

Figura 12: Vista de registro de cumplimiento para el usuario.

Figura 13: Vista del informe de adherencia terapéutica para el usuario.

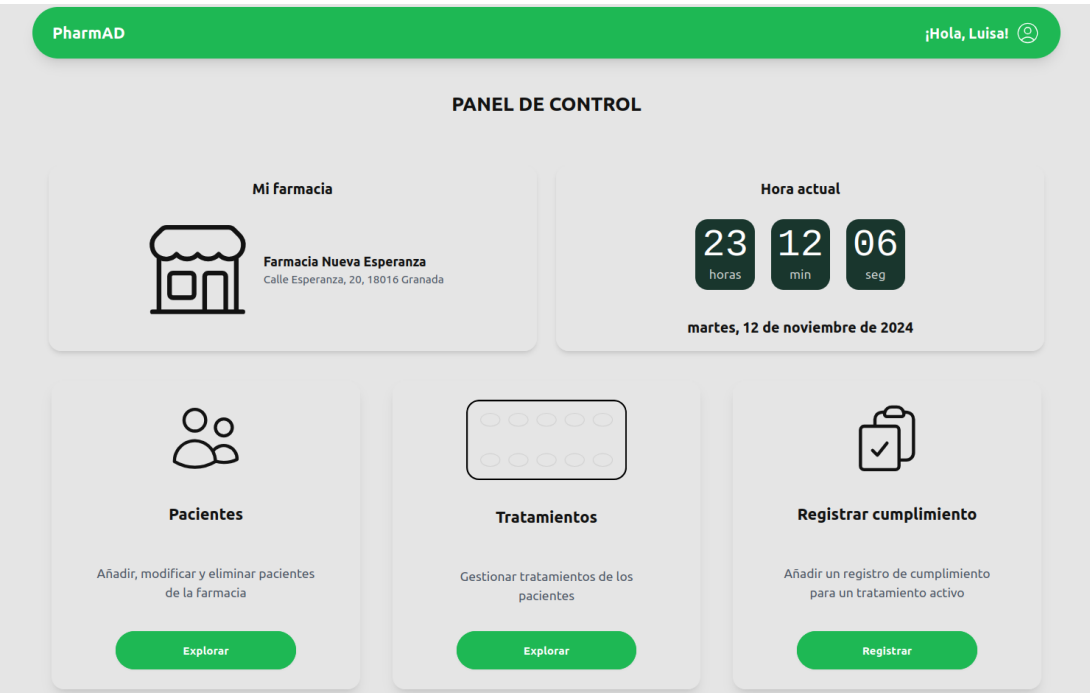

Figura 14: Panel de control para el farmacéutico.

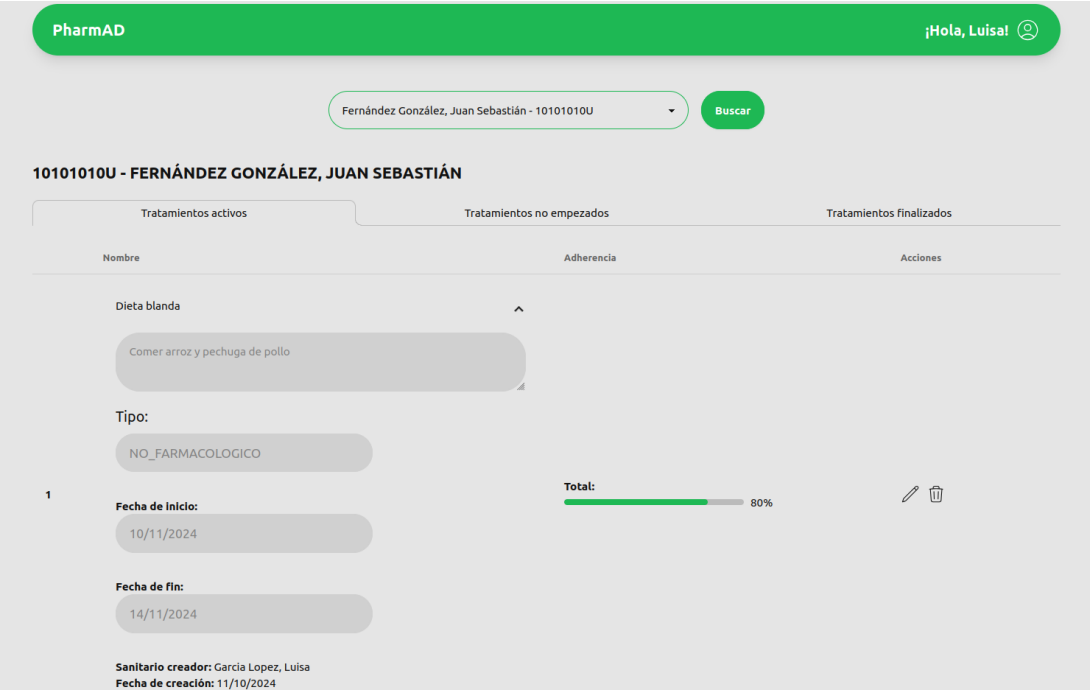

Figura 15: Vista de gestión de tratamientos para un farmacéutico.

Capítulo 6

Conclusiones

El desarrollo de este proyecto ha culminado en la creación de una plataforma que integra un backend escalable y eficiente con un frontend interactivo y funcional, satisfaciendo plenamente todos los requisitos establecidos inicialmente. El sistema es capaz de gestionar de forma segura y efectiva información crítica de pacientes y personal sanitario en el contexto de una farmacia comunitaria.

6.1. Despliegue

Durante el desarrollo del proyecto se tomó la decisión de mantener esta plataforma como exclusiva para el uso en las farmacias inscritas. Se trata de un soporte orientado a facilitar el trabajo del personal sanitario de las farmacias comunitarias, y la peculiaridad que lo caracteriza es que no todo el público, sin previo registro y alta como paciente de una farmacia, podría tener acceso. Esta característica proporciona una grado más de seguridad para proteger la información personal y médica de cada persona.

6.2. Dificultades encontradas

El proyecto ha enfrentado desafíos significativos, especialmente en lo referente a la configuración e integración de las diversas tecnologías utilizadas. La implementación de Docker Compose para gestionar PostgreSQL y pgAdmin, junto con la integración de Prisma y Fastify, requirió un esfuerzo para garantizar que todos los servicios se comunicaran correctamente y de manera segura. Asimismo, la implementación de controladores y rutas en Fastify, así como la validación de usuarios y permisos, presentó complejidades tanto técnicas como lógicas. Esto fue particularmente relevante durante el desarrollo de mecanismos de autenticación y autorización utilizando JSON Web Tokens (JWT) para controlar el acceso a los datos de los usuarios.

En el ámbito del frontend, la configuración de Astro y su integración con React añadieron niveles adicionales de complejidad, especialmente en la estructuración de componentes y la optimización del rendimiento de la interfaz de usuario.

6.3. Conclusión final

A nivel personal, la satisfacción con el resultado obtenido es alta. Gracias a todo el aprendizaje obtenido se podrán ampliar y mejorar todas las funcionalidades que ofrece la plataforma y realizar un despliegue real en el futuro.

Bibliografía

[1] World Health Organization (WHO). *Adherence to long-term therapies: evidence for action*. WHO Library Cataloguing-in-Publication Data. 2003.

[2] Olatz Ibarra Barrueta and Ramón Morillo Verdugo. *Lo que debes saber sobre la adherencia al tratamiento*. Sociedad Española de Farmacia Hospitalaria, 2017. Grupo de Adherencia Terapéutica ADHEFAR de la SEFH, Edición patrocinada por Boehringer Ingelheim.

[3] Farmaindustria. Plan de adherencia al tratamiento. `https://www.farmaindustria.es/adherencia/`, 2023. Accedido en agosto de 2024.

[4] Grupo OAT Observatorio de la Salud and Fundación Weber. *Libro Blanco de la Adherencia en España*. Grupo OAT Observatorio de la Salud, S.L.; Fundación Weber, Las Rozas de Madrid; Madrid, 2021. Depósito Legal: M-10617-2022.

[5] A. J. Claxton, J. Cramer, and C. Pierce. A systematic review of the associations between dose regimens and medication compliance. *Clinical Therapeutics*, 23(8):1296–1310, 2001.

[6] Neus Pagès-Puigdemont and M. Isabel Valverde-Merino. Métodos para medir la adherencia terapéutica. *Ars Pharm*, 59(3):163–172, 9 2018.

[7] P. Conthe, E. Márquez Contreras, A. Aliaga Pérez, B. Barragán García, M.N. Fernández de Cano Martín, M. González Jurado, M. Ollero Baturone, and J.L. Pinto. Adherencia terapéutica en la enfermedad crónica: estado de la situación y perspectiva de futuro. *Revista Clínica Española*, 214(6):336–344, 2014.

[8] Farmacéuticos.com. Programa consigue. `https://www.farmaceuticos.com/farmaceuticos/farmacia/farmacia-asistencial/proyectos-de-investigacion/programa-consigue/`, abril 2016. Último acceso: [21 abril 2024].

[9] El Farmacéutico. El programa "adhiérete" confirma la influencia positiva del farmacéutico en la adherencia a los tratamientos. `https://www.elfarmaceutico.es/actualidad/noticias-novedades/el-programa-adhierete-confirma-la-influencia-positiva-del-farmaceutico-en-la-adherencia106040_102.html`, junio 2015. Último acceso: [21 abril 2024].

[10] Consejo General de Colegios Oficiales de Farmacéuticos. Hazadherencia: Servicio de adherencia terapéutica a pacientes con enfermedades crónicas. `https://www.farmaceuticos.com/farmaceuticos/agenda/hazadherencia-servicio-de-adherencia-terapeutica-a-pacientes-con-enfermedades-cronicas-`2024. Accedido en agosto de 2024.

[11] Ministerio de Sanidad y Consumo. Ley 16/1997, de 25 de abril, de regulación de servicios de las oficinas de farmacia, 1997. Accedido en agosto de 2024.

[12] Clínica Universidad de Navarra. Paciente. diccionario médico, 2024. Accedido: 1-sep-2024.

[13] Screwfast Foundations Ltd. Screwfast foundations: Helical piles and pile foundations. `https://screwfast.uk/`.