

**DPTO INFORMÁTICA – IES TRAFALGAR  
MÓDULO – PROYECTO INTEGRADO  
C.F.G.S. DESARROLLO DE APLICACIONES WEB**

**pharmAI**

**Autor: José Diego Junquera Lobón  
Fecha: Marzo-Junio de 2024  
Tutor: Teófilo Rojas Mata**

<b>Título del Proyecto:</b> pharmAI	
<b>Autor:</b> José Diego Junquera Lobón	<b>Fecha:</b> Marzo-Junio 2024
<b>Tutor:</b> Teófilo Rojas Mata	
<b>Ciclo Formativo:</b> Desarrollo de Aplicaciones Web	<b>Curso:</b> 2 Daw
<b>Palabras Clave:</b>  Inteligencia Artificial, Farmacia, Astro, Ollama, Endpoints, CIMA, Frontend, Javascript	
<b>Resumen del proyecto:</b>  pharmAI nació con la filosofía de unir la inteligencia artificial con el mundo de la Farmacia, permitiendo al usuario tener una aplicación cuyo pilar es la integración con la IA, en la que además puede ver los prospectos de los medicamentos y datos relevantes con una simple búsqueda, poner recordatorios o ver las farmacias cercanas.  pharmAI se basa en la sencillez de la interacción con el usuario, menús muy intuitivos e interactivos y una interfaz amigable para que navegar por la web sea algo sencillo y una experiencia inmersiva en la plataforma, siempre acompañado de la mano de pharmaBot.  Si estás harto de webs destinadas a la salud estáticas y poco llamativas, sin asistencia de la IA por medio de un chat, pharmAI es tu opción.	

# ÍNDICE

<b>1. INTRODUCCIÓN</b>	<b>3</b>
1.1 Objetivos del proyecto	4
1.2 Beneficios del proyecto	5
1.3 Motivaciones Personales	6
<b>2. ESTUDIO DE LA VIABILIDAD</b>	<b>7</b>
2.1 Análisis de la problemática y oportunidades	7
<b>3. ALTERNATIVAS Y SELECCIÓN DE LA SOLUCIÓN</b>	<b>10</b>
3.1 Análisis DAFO de pharmAI	11
<b>4. ANÁLISIS Y DISEÑO DE LA SOLUCIÓN ADOPTADA</b>	<b>13</b>
4.1 ¿Por qué Astro?	15
4.1.1 Ventajas de Astro	16
4.1.2 Justificación de la Elección de Astro	16
4.2 Ollama, ejecuta un LLMs en local	17
4.3 Otras APIs que dan vida a pharmAI	18
4.3.1 API de CIMA para obtener información sobre medicamentos	18
4.3.2 Geoapify para localizaciones de farmacias	19
4.3.3. API custom con Python-Flask para frases de farmacología aleatorias	19
4.4 Interfaz, minimalista e interactiva	19
4.5 pharmAI, sus puntos fuertes	22
<b>5. IMPLEMENTACIÓN</b>	<b>23</b>
5.1 ¿Cómo se ha hecho el proyecto?	23
5.1.1 Git, mantén tu proyecto a salvo	23
5.2 Construyendo con Astro	25
5.3 Ollama en Astro	29
5.4 Astro DB, una SQLite en Astro	32
5.4.1 Gestión de usuarios: Lucia Auth	33
5.4.2 Sintaxis y Drizzle ORM	34
5.4.3 Diagrama entidad-relación	36
5.5 Otras funcionalidades, Cuestión de APIs	38
5.6 Resumen del funcionamiento de PharmAI	40
5.7 Implementando con identidad	41
<b>6. PRUEBAS</b>	<b>44</b>
6.1 Responsividad, algo básico	45
<b>7. COSTES/PRESUPUESTO</b>	<b>47</b>
<b>8. CONCLUSIONES</b>	<b>50</b>
<b>9. BIBLIOGRAFÍA</b>	<b>52</b>
<b>10. GLOSARIO</b>	<b>53</b>
<b>11. ANEXOS</b>	<b>55</b>

## 1. INTRODUCCIÓN

Cuando se planteó la realización del proyecto a los alumnos, siempre tuve claro que la idea del mismo debía girar en torno a la **Inteligencia Artificial (IA)**.

Desde que empecé el módulo en 2022 he visto en estos dos años como la IA ha crecido de forma imparable y se ha convertido en un tema mainstream en la sociedad de hoy en día. Actualmente la IA se ha convertido en un producto que atrae sólo con mencionarla en la publicidad, que te permite personalizar experiencias, y darle al usuario lo que busca de una forma más sencilla, más natural. La IA “entiende” lo que le pides, lo procesa y te da una respuesta adaptada a ti. El funcionamiento de un modelo de procesamiento de lenguaje natural (**NLP**) se detalla en el **anexo 1**.

Esto supone un nuevo paradigma en la interacción del usuario con la máquina, ya que por ejemplo, si vas a Google y haces una búsqueda, te arrojará resultados cercanos a lo que buscas, pero tendrás que ser tú el que filtre la información hasta llegar a obtener una respuesta que satisfaga la duda inicial que tenías. Con la IA esto ha cambiado, ya que la IA se ha entrenado con esos mismos datos, pero ahora es capaz de entender lo que le estás diciendo y adaptar esa información a tu pregunta, obteniendo al final, una experiencia más personal y simplificando la tarea de búsqueda del usuario, incluso creando una experiencia más accesible (véase la página de [rabbit](#)).

Además, la IA no solo se limita al sector de la informática. Su aplicación se ha extendido a numerosos campos como la medicina, donde se utiliza para diagnosticar enfermedades con mayor precisión; el transporte, con el desarrollo de vehículos autónomos; y la educación, proporcionando tutores virtuales personalizados que se adaptan al ritmo de aprendizaje de cada estudiante. Esta versatilidad hace que la IA sea una herramienta imprescindible en la transformación digital de nuestra sociedad.

Todo esto, sumado a mi otra faceta de estudiante, en el sector sanitario, me ha permitido conectar los dos campos a través de este proyecto.

La idea del proyecto pasa por modernizar el sector de la farmacia, ampliamente distribuido en la sociedad, pero no integrado de una forma tan eficaz con las nuevas tecnologías que surgen en el mundo de la informática, justificando esto la creación de la idea de producto que surge en este proyecto:

Una plataforma cuyo principal reclamo es la asistencia de la IA en preguntas sobre medicamentos, enfermedades etc. Su objetivo no es sustituir a los profesionales, es más bien un complemento para los usuarios y estos mismos profesionales a la hora de obtener información médica y consejo farmacéutico de forma rápida y sencilla.

Además haciendo énfasis en el concepto de plataforma, se le han integrado más funcionalidades con el objetivo de que el usuario no tenga que salir de ella para tener una información de calidad de los medicamentos, recordatorios o las farmacias disponibles en un punto geográfico.

## 1.1 Objetivos del proyecto

El proyecto de la plataforma nace con las siguientes ideas:

- Implementar la IA de forma gratuita dentro de la plataforma, permitiendo al usuario charlar con ella y que lo asesore de una forma sencilla. Para ello se pretende crear un chat llamativo e intuitivo de la mano de **pharmaBot**.
- Crear una web dinámica, que no sea una web más, estática y aburrida, como las típicas webs de consejo farmacéutico y médico, con textos interminables y de difícil acceso para personas que no estén familiarizadas con el uso de las nuevas tecnologías.
- Realizar la web con tecnologías actuales, primando la sencillez (minimalismo) y la interactividad. Se pretende que la navegación por la plataforma sea una experiencia divertida y que el usuario tenga acceso a la información de valor con pocos clicks, haciendo esto un proceso dinámico (La información va a la mano del usuario de forma rápida).
- Crear más funciones dentro de la plataforma, la IA será el reclamo para atraer a los usuarios, pero se le dará más opciones dentro de la plataforma, por lo que será importante además la adaptación de la web a los distintos dispositivos actuales (normalmente ordenadores de sobremesa, portátiles, tablets y smartphones).

## 1.2 Beneficios del proyecto

La creación del proyecto reporta una serie de beneficios que repercuten tanto en el usuario como en la comunidad sanitaria que lo rodea, ya que tiene una serie de implicaciones:

### **Beneficios Sociales:**

- **Asesoramiento Rápido y Preciso:** La integración de un asistente virtual (pharmaBot) permite a los usuarios recibir asesoramiento inmediato sobre medicamentos y enfermedades, mejorando la gestión de su salud.
- **Accesibilidad y Educación:** Al proporcionar una plataforma intuitiva y amigable, se mejora la accesibilidad de información médica y farmacéutica, educando a los usuarios sobre el uso correcto de medicamentos y la localización de farmacias cercanas por si necesitan asesoramiento profesional.

### **Beneficios Económicos:**

- **Ahorro de Tiempo y Recursos:** La plataforma permite a los usuarios obtener información rápidamente sin la necesidad de largas búsquedas, lo que ahorra tiempo y recursos tanto para los usuarios como para los profesionales de la salud.
- **Impulso a la Economía Local:** Al integrar la localización de farmacias cercanas, pharmAI puede impulsar el tráfico y las ventas en farmacias locales, beneficiando a la economía local. Esto está enfocado en personas que no residen en la localidad y quieren buscar información de las farmacias cercanas sin salir del ecosistema de la plataforma.

### **Beneficios Tecnológicos:**

- **Innovación y Modernización:** pharmAI utiliza tecnologías avanzadas de IA para proporcionar una experiencia de usuario moderna y eficiente, estableciendo un plataforma de información farmacéutica amigable con el usuario.

- **Experiencia con nuevas tecnologías:** El proyecto fomenta el desarrollo de habilidades en tecnologías actuales como IA, frontend y backend, tanto para los desarrolladores como para los usuarios que interactúan con la plataforma.

#### Beneficios para la Salud:

- **Mejora de la Calidad de Vida:** pharmAI facilita el acceso a información médica relevante y a prospectos de medicamentos, permitiendo a los usuarios obtener respuestas rápidas y precisas a sus consultas de salud.
- **Recordatorios de Medicación:** La funcionalidad de recordatorios ayuda a los usuarios a seguir sus tratamientos de manera correcta y puntual, mejorando la adherencia a los regímenes de medicación.

#### Beneficios Ambientales:

- **Reducción del Uso de Papel:** Al proporcionar prospectos y otra información de manera digital, se reduce la necesidad de imprimir documentos, contribuyendo a la conservación de recursos naturales.

### 1.3 Motivaciones Personales

La principal motivación para la realización de este proyecto es unir mi faceta pasada de estudiante de la carrera de Farmacia con la nueva incorporación en el mundo del desarrollo de aplicaciones web. Mi experiencia por el paso de la rama sanitaria me ha servido para comprobar que tanto en la administración pública como en las farmacias privadas aún se tiene muy arraigados los métodos tradicionales y ofrecen cierta resistencia a las nuevas tecnologías.

El surgimiento de la IA y la facilidad para conseguir su implementación en proyectos de desarrollo ha impulsado que me decante por este proyecto y rompa la barrera tradicional a las nuevas tecnologías, mostrando un acercamiento entre ambos mundos.

## 2. ESTUDIO DE LA VIABILIDAD

La IA ha transformado diversos sectores, incluyendo el sanitario, mejorando significativamente el diagnóstico médico, la personalización de tratamientos y la atención al paciente. No obstante, el sector farmacéutico aún no ha aprovechado completamente el potencial de la IA para mejorar la experiencia del usuario y la gestión de información médica. Las plataformas actuales suelen ser estáticas, con grandes bloques de texto que pueden resultar abrumadores y de difícil navegación, especialmente para usuarios sin conocimientos técnicos.

El auge de la tecnología móvil y la demanda de acceso instantáneo a la información han puesto de manifiesto la necesidad de soluciones más avanzadas y accesibles. Sin embargo, muchas plataformas farmacéuticas no son llamativas, lo que limita su usabilidad. Además, la información médica y farmacéutica suele estar fragmentada en múltiples fuentes, obligando a los usuarios a realizar búsquedas extensas y aumentando el riesgo de encontrar información incorrecta o desactualizada.

A pesar de la resistencia a la adopción de nuevas tecnologías en el sector farmacéutico, hay un creciente reconocimiento de los beneficios que la IA puede ofrecer. Factores como la falta de formación, preocupaciones sobre la seguridad de los datos y la percepción de complejidad tecnológica contribuyen a esta resistencia. En este contexto, pharmAI se presenta como una solución innovadora, diseñada para mejorar la accesibilidad de la información, proporcionar asesoramiento personalizado y promover la modernización del sector, respondiendo a la demanda de experiencias más interactivas y eficientes.

### 2.1 Análisis de la problemática y oportunidades

Existen múltiples desafíos significativos en el sector farmacéutico que destacan la necesidad de desarrollar una solución como pharmAI para mejorar la experiencia del usuario y la eficiencia en la gestión de la información:

1. **Interacción Limitada:** Las plataformas actuales carecen de interactividad y personalización, lo que impide que los usuarios encuentren fácilmente la información específica que necesitan. Estas plataformas suelen ser estáticas y no ofrecen una experiencia dinámica ni intuitiva, lo que resulta en una navegación tediosa y poco eficiente.
2. **Información Dispersa:** La información farmacéutica está fragmentada en múltiples fuentes, obligando a los usuarios a invertir tiempo en búsquedas exhaustivas para encontrar respuestas concretas. Esta problemática no solo es tediosa para el usuario, sino que también aumenta el riesgo de acceder a información incorrecta o desactualizada, dificultando la toma de decisiones informadas por parte de los usuarios.
3. **Resistencia al Cambio:** Tanto en la administración pública como en las farmacias privadas, persiste una notable resistencia a adoptar nuevas tecnologías que podrían mejorar significativamente la atención y gestión de la información. Esta resistencia puede estar motivada por la falta de formación en nuevas tecnologías, preocupaciones sobre la seguridad y privacidad de los datos, y la percepción de que la implementación de nuevas herramientas es compleja y costosa.
4. **Necesidad de Asesoramiento Rápido:** Los usuarios requieren asesoramiento inmediato y preciso sobre medicamentos y enfermedades, algo que las plataformas actuales no proporcionan de manera eficaz. La falta de un sistema interactivo y personalizado para responder a consultas médicas y farmacéuticas en tiempo real limita la capacidad de los usuarios para gestionar su salud de manera proactiva y eficiente.

Una vez analizadas estas problemáticas es fácil entender por qué pharmAI tiene sentido como plataforma y viable como proyecto:

1. **Crecimiento y Adopción de IA:** La inteligencia artificial está experimentando un crecimiento exponencial en la sociedad. La demanda de soluciones basadas en IA está en auge, lo que crea un entorno favorable para la adopción de pharmAI. La plataforma puede aprovechar esta tendencia, ofreciendo una solución innovadora que responde a las necesidades emergentes del mercado tecnológico.

2. **Preferencia por Experiencias Personalizadas:** Los usuarios actuales valoran las experiencias personalizadas y el acceso rápido a la información. pharmAI, con su asistente virtual pharmaBot, puede ofrecer asesoramiento inmediato y adaptado a las necesidades individuales de los usuarios. Esta capacidad de personalización no solo mejora la experiencia del usuario, sino que también aumenta la satisfacción y fidelización de los usuarios, lo que es crucial para el éxito de la plataforma.
3. **Optimización para Dispositivos Móviles:** El uso de dispositivos móviles para acceder a información sanitaria está en constante aumento. pharmAI está diseñada para ser completamente compatible con dispositivos móviles, lo que asegura que los usuarios puedan acceder a la plataforma en cualquier momento y lugar. Esta accesibilidad mejora la usabilidad y aumenta el alcance potencial de la plataforma.
4. **Integración de Tecnologías Avanzadas:** pharmAI utiliza tecnologías modernas para el desarrollo de la plataforma, asegurando una plataforma dinámica y eficiente. La implementación de modelos de procesamiento de lenguaje natural (NLP) permite interacciones fluidas y naturales con pharmaBot, lo que mejora la experiencia del usuario y la eficacia del asesoramiento proporcionado.
5. **Fuentes de Datos Fiables:** La plataforma se integrará con APIs confiables como [CIMA](#) ([anexo 2](#)) para proporcionar información precisa y actualizada sobre medicamentos. Esto no solo mejora la calidad de la información disponible para los usuarios, sino que también aumenta la confianza en la plataforma.
6. **Viabilidad Económica:** El proyecto puede ser financiado a través de diversas fuentes, incluyendo subvenciones, inversores privados y crowdfunding. Además, se puede implementar un modelo de negocio freemium, donde la plataforma básica es gratuita y se ofrecen funciones premium por una tarifa (modelos de NLP más avanzados por ejemplo). Este modelo no solo atrae a un amplio rango de usuarios, sino que también genera ingresos sostenibles a largo plazo.

En resumen, **pharmAI** tiene sentido como solución innovadora para estos desafíos porque ofrece una plataforma interactiva y personalizada, integra información dispersa en un solo lugar, facilita la adopción de tecnología avanzada en el sector farmacéutico y proporciona asesoramiento rápido y preciso. Esto no solo mejora la experiencia del usuario, sino que también promueve una gestión más eficiente y moderna de la información farmacéutica.

### 3. ALTERNATIVAS Y SELECCIÓN DE LA SOLUCIÓN

En esta sección se presentan y analizan diferentes alternativas disponibles para abordar los desafíos identificados en el sector farmacéutico. La evaluación se realiza en términos de planificación, riesgos, costes y conclusiones, lo que permite justificar la selección de pharmAI como la solución óptima. A continuación, se detalla esta evaluación en formato de tabla:

<i>Alternativa</i>	<i>Descripción</i>	<i>Riesgos y Costes</i>	<i>Conclusión</i>
Plataforma Tradicional de Información	Mejorar las plataformas existentes sin integrar IA.	<ul style="list-style-type: none"> <li>- Interacción limitada</li> <li>- Alta competencia</li> <li>- Desarrollo y mejora: Medio</li> <li>- Mantenimiento: Medio</li> </ul>	Mejora las plataformas existentes pero no resuelve los desafíos de interactividad y personalización que los usuarios actuales demandan.
Aplicación Móvil Especializada	Desarrollar una app móvil para información de medicamentos y farmacias, sin IA.	<ul style="list-style-type: none"> <li>- Interacción limitada</li> <li>- Necesidad de actualizaciones constantes</li> <li>- Desarrollo: Alto</li> <li>- Mantenimiento: Medio</li> </ul>	Proporciona una solución móvil accesible, pero carece de la interactividad y personalización que puede ofrecer una plataforma integrada con IA.
Plataforma Integrada con IA (pharmAI)	Crear pharmAI, una plataforma web y móvil con IA para información sobre medicamentos, asesoramiento y localización de farmacias.	<ul style="list-style-type: none"> <li>- Complejidad técnica</li> <li>- Necesidad de seguridad robusta</li> <li>- Desarrollo: Alto</li> <li>- Mantenimiento: Medio-Alto</li> <li>- Marketing efectivo</li> </ul>	Aborda los desafíos de interactividad, personalización y accesibilidad, ofreciendo una experiencia avanzada, atractiva y eficiente, posicionándose como la solución más innovadora.

Después de evaluar las alternativas, pharmAI destaca como la opción más viable debido a su capacidad para ofrecer una experiencia personalizada y eficiente, aprovechando las tecnologías avanzadas de IA. Esta solución mejora significativamente la interacción y accesibilidad para los usuarios, posicionando la plataforma en un mercado en crecimiento y aprovechando las tendencias actuales en tecnología y salud.

### 3.1 Análisis DAFO de pharmAI

#### Fortalezas (Strengths)

1. **Innovación Tecnológica:** Integración de inteligencia artificial, proporcionando una experiencia personalizada y eficiente.
2. **Accesibilidad:** Plataforma optimizada para dispositivos móviles y de escritorio, accesible en cualquier momento y lugar.
3. **Interfaz Intuitiva:** Diseño amigable y fácil de usar, mejorando la experiencia del usuario.
4. **Información Centralizada:** Integra información de diversas fuentes fiables en un solo lugar, facilitando el acceso a datos precisos y actualizados.
5. **Asesoramiento Personalizado:** Proporciona respuestas rápidas y adaptadas a las necesidades específicas de los usuarios a través de pharmaBot.

#### Debilidades (Weaknesses)

1. **Complejidad Técnica:** Desarrollo e implementación de tecnologías avanzadas de IA requieren altos conocimientos técnicos y recursos.
2. **Resistencia al Cambio:** Posible resistencia de usuarios y profesionales del sector farmacéutico a adoptar nuevas tecnologías.
3. **Dependencia de Datos:** Necesidad de mantener información actualizada y precisa, lo que requiere integración continua con bases de datos y APIs confiables.

#### Oportunidades (Opportunities)

1. **Crecimiento del Mercado de IA en la Sociedad:** Aumento de la demanda de soluciones basadas en IA en la sociedad.

2. **Tendencia hacia la Digitalización:** Creciente tendencia a adoptar tecnologías digitales en sectores donde antes no se usaban.
3. **Necesidad de Información Rápida y Precisa:** Demanda de los usuarios por acceso rápido a información fiable sobre medicamentos y tratamientos.
4. **Expansión a Nuevos Mercados:** Posibilidad de expandir la plataforma a otros mercados geográficos o sectores relacionados con la salud.

## Amenazas (Threats)

1. **Competencia en el Mercado:** Presencia de competidores que también buscan integrar IA en soluciones de salud y farmacia.
2. **Regulaciones y Normativas:** Cambios en las regulaciones del sector de la salud y la privacidad de los datos pueden afectar el funcionamiento de la plataforma.
3. **Seguridad y Privacidad:** Riesgos relacionados con la protección de datos personales y la seguridad de la información.
4. **Dependencia de Terceros:** Dependencia de APIs y bases de datos externas para mantener la información actualizada y precisa.

## 4. ANÁLISIS Y DISEÑO DE LA SOLUCIÓN ADOPTADA

Una vez justificada la existencia de esta propuesta de proyecto y la viabilidad del mismo, es momento de ahondar en el análisis de los requisitos para llevarlo a cabo y justificar las tecnologías utilizadas para su implementación.

Teniendo en cuenta que estamos en el Ciclo Formativo de Grado Superior en Desarrollo de Aplicaciones Web, la primera necesidad del proyecto es evidente, un framework web sobre el que construir la plataforma. Fue la primera pregunta que me hice al empezar el proyecto, ya que la decisión condicionaría el desarrollo de la aplicación significativamente.

Para responder a esta pregunta tuve que hacerme otra, una de las preguntas claves de este proyecto: **¿Cómo implementar una IA en la plataforma?**

Y ese fue el primer paso que di para elegir el framework web que tendría mi proyecto. Estudiando las implementaciones, me di cuenta que todas las implementaciones se hacían con **EndPoints** (una **URL** específica en una **API** donde se puede acceder o interactuar con recursos del servidor mediante solicitudes **HTTP**).

Esto me dio la clave para entender que mi proyecto se iba a basar en EndPoints, ya que requiere de backends que gestionen una gran cantidad de información, como un NLP o la base de datos de medicamentos de CIMA.

Por otro lado, se ha hecho énfasis en la puesta de valor del proyecto, que uno de los principales baluartes del mismo es el atractivo visual, la sencillez y la interactividad, dándole a la web una apariencia moderna, minimalista y muy llamativa.

Con todo esto sobre la mesa, se plantean dos opciones que se han visto durante la formación:

- **Html, Css, Javascript:** Lo mínimo para realizar un proyecto, pero es ideal para algo sencillo, tareas de clase por ejemplo, pero no para un proyecto serio de esta envergadura.

- **PHP con Laravel:** Framework visto en clase con un *Modelo Vista Controlador (MVC)* en el cual el software de la aplicación se separa en tres componentes interconectados: el Modelo, que maneja la lógica de datos; la Vista, que presenta la interfaz de usuario; y el Controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando la entrada del usuario y actualizando tanto la Vista como el Modelo. Esta separación facilita la gestión y escalabilidad del código.

Tras analizar estas opciones y conocer que muchos de mis compañeros han realizado la aplicación en Laravel, yo no decidí esa opción.

Utilizar Laravel sería montar una infraestructura innecesaria para construir mi plataforma, ya que Laravel tiene un amplio marcado carácter de funcionalidades backend y no se ha visto en clase de forma muy enfatizada el otro lado de la moneda, la integración de un framework de frontend, como React por ejemplo.

En este punto vuelvo a recordar que las principales funcionalidades de mi plataforma vienen de Endpoints. Y esto abre la llave a encontrar la solución a gran parte de la pregunta, ¿qué framework usar? Y la respuesta empieza a ser clara, un **framework frontend**.

Otro de los pilares fundamentales del proyecto, es la innovación, por lo que se abre la puerta a un framework novedoso, basado en JavaScript (durante el verano, a petición mía a la empresa, hice cursos de JavaScript, por lo que tengo nociones de este lenguaje de programación).

La elección del framework sigue acotándose, y a los ya conocidos React, Angular, Vue se unen los nuevos Next.js por ejemplo, desarrollado por [Vercel](#).

Llegados aquí, hay que entender la envergadura del proyecto, y los conocimientos que tengo de los framework que se han nombrado, por lo que me pareció demasiado tedioso meterme en alguno de ellos, viniéndome a la cabeza que los principales desarrolladores de frontend en comunidades como YouTube hablaban muy bien de un framework, para webs que cumplen los criterios que he mencionado en los puntos anteriores, y además destaca por la sencillez de creación de webs, haciendo el proceso de desarrollo sencillo una vez entiendes su funcionamiento, y esta fue mi elección.

Estoy hablando de **Astro**, un framework de desarrollo web basado en JavaScript que permite crear sitios web pre-renderizados utilizando componentes de diferentes frameworks, generando HTML estático para optimizar el rendimiento y reducir la carga de JavaScript en el cliente.

## 4.1 ¿Por qué Astro?

[Astro](#) es un moderno framework de desarrollo web basado en JavaScript que permite crear sitios web altamente optimizados y pre-renderizados. A diferencia de otros frameworks, Astro genera HTML estático en el tiempo de compilación, lo que reduce la carga de JavaScript en el cliente y mejora significativamente el rendimiento y la velocidad de carga de las páginas. Una de las características distintivas de Astro es su capacidad para integrar componentes de diversos frameworks populares como React, Vue, Svelte, entre otros, ofreciendo gran flexibilidad y eficiencia en el desarrollo. Este enfoque no solo mejora el rendimiento del sitio, sino que también facilita la creación de interfaces modernas y minimalistas, proporcionando una experiencia de usuario atractiva y rápida.



**Imagen 1:** Logo de Astro.

#### 4.1.1 Ventajas de Astro

1. **Optimización del Rendimiento:** Astro genera HTML estático, lo que reduce significativamente la cantidad de JavaScript necesario en el cliente, mejorando la velocidad de carga y el rendimiento general del sitio.
2. **Compatibilidad con Varios Frameworks:** Permite el uso de componentes de frameworks populares como React, Vue, Svelte, entre otros, ofreciendo flexibilidad en el desarrollo.
3. **Sencillez de Creación:** Astro simplifica el proceso de desarrollo web, haciendo que la creación de sitios sea más accesible y menos tediosa, especialmente una vez que se entiende su funcionamiento.
4. **Atractivo Visual y Interactividad:** Facilita la creación de sitios modernos y minimalistas con alta interactividad, cumpliendo con los requisitos estéticos del proyecto.

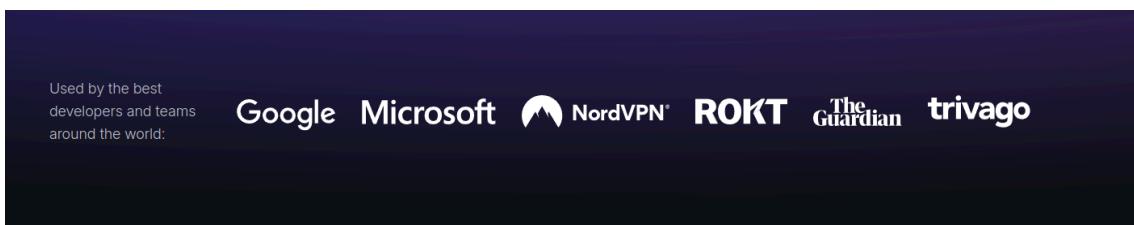
#### 4.1.2 Justificación de la Elección de Astro

La elección de Astro se basa en varios factores clave:

1. **Enfoque en EndPoints:** Dado que la plataforma depende en gran medida de EndPoints para gestionar la información, Astro se alinea bien con esta necesidad, permitiendo una integración eficiente con APIs.
2. **Innovación y Modernidad:** Uno de los pilares fundamentales del proyecto es la innovación. Astro, siendo un framework novedoso y moderno, se ajusta perfectamente a esta filosofía.
3. **Sencillez y Eficiencia:** A diferencia de frameworks más complejos como React, Angular, o Vue; Astro ofrece una curva de aprendizaje más suave y un proceso de desarrollo más simple, lo cual es ideal considerando los conocimientos previos y el tiempo disponible para el proyecto.
4. **Rendimiento y SEO:** La generación de HTML estático mejora el rendimiento del sitio y su posicionamiento en motores de búsqueda (SEO), lo cual es crucial para la accesibilidad y visibilidad del proyecto.

5. **Moderno:** Astro permite crear sitios web con un diseño moderno y minimalista, cumpliendo con uno de los objetivos principales del proyecto: proporcionar una interfaz atractiva y fácil de usar.

Por todo esto, Astro ha conseguido la atención de grandes empresas como se ve en la **Imagen 2**



**Imagen 2:** Empresas que usan Astro.

#### 4.2 Ollama, ejecuta un LLMs en local

Una vez elegido el framework, hay que responder a la segunda gran pregunta de este proyecto, ya tengo el molde sobre el que construir, ahora cómo integrar la IA que dará vida a pharmabot y creo que si estás en este mundillo, se te viene a la cabeza de primeras la opción más famosa, [chatGPT](#).

Fue la primera opción que valoré, ya que todos los proyectos de esta envergadura suelen usar chatGPT como proveedor delEndPoint que gestiona el NLP.

Y mis primeras misiones consistieron en investigar el código para implementarlo, consiguiendo hacer llamadas pero no dándome la respuesta que quería, tenía que meter un método de pago.

Y no está dentro de la filosofía de proyecto utilizar pagos para permitir una funcionalidad, y menos en un proyecto académico, por lo que aunque parecía la opción más factible, la descarté.

Sin chatGPT, ¿ Ahora qué ? volvemos a recordar uno de los lemas de este proyecto, la innovación, y una de las innovaciones que han salido hoy en día en el mundo de la IA, es la capacidad de ejecutar modelos de NLP en local, dónde tú decides cómo implementarlo. Y en este escenario aparece el segundo gran actor de este proyecto, [Ollama](#).

El proyecto Ollama es una plataforma diseñada para facilitar el uso de modelos de lenguaje de gran tamaño (LLMs) de manera local en tu propio ordenador. Ollama permite a los usuarios ejecutar, personalizar y crear modelos de IA como Llama 3, Phi 3, Mistral, y Gemma. La plataforma es compatible con sistemas operativos como macOS, Linux y Windows, y ofrece aceleración GPU para mejorar el rendimiento de los modelos. Ollama proporciona una API sencilla para la gestión de estos modelos.

Ollama será el segundo actor de la plataforma y mediante un **Endpoint** se comunicará con Astro para darle vida a Pharmabot.



**Imagen 3:** Logo de ollama

#### 4.3 Otras APIs que dan vida a pharmAI

En la integración de pharmAI, varias APIs juegan un papel crucial para ofrecer funcionalidades avanzadas y proporcionar una experiencia de usuario completa. A continuación, se describen tres de estas APIs, explicando su uso específico dentro del ecosistema de pharmAI.

#### 4.3.1 API de CIMA para obtener información sobre medicamentos

La API de CIMA (Centro de Información sobre Medicamentos de la AEMPS) proporciona datos exhaustivos sobre medicamentos disponibles en el mercado. Esta API permite a pharmAI acceder a información detallada sobre los medicamentos, incluyendo su composición, indicaciones, contraindicaciones, dosificación, y efectos secundarios. Al integrar esta API, pharmAI puede:

- **Consultar datos de medicamentos:** Proveer información actualizada y precisa sobre medicamentos a los usuarios.
- **Verificar la disponibilidad:** Chequear la disponibilidad de medicamentos en tiempo real.

#### 4.3.2 Geoapify para localizaciones de farmacias

[Geoapify](#) es una API de localización que ofrece servicios de geocodificación, búsqueda de lugares, y rutas. En el contexto de pharmAI, esta API se utiliza para:

- **Encontrar farmacias cercanas:** Permitir a los usuarios localizar farmacias cercanas en función de un nombre de ubicación.
- **Mostrarlas en un mapa de la ciudad:** Gracias a [Leaflet](#), se muestran en un mapa interactivo de la ciudad.

#### 4.3.3. API custom con Python-Flask para frases de farmacología aleatorias

PharmAI también incluye una API personalizada creada con **Python** y Flask que genera y sirve frases aleatorias de farmacología. Esta API es útil para:

- **Educar y entretenecer a los usuarios:** Proporcionar datos interesantes y educativos sobre farmacología en cada recarga de la página.
- **Mantener la interfaz dinámica:** Hacer que la experiencia de usuario sea más atractiva con contenido siempre nuevo.

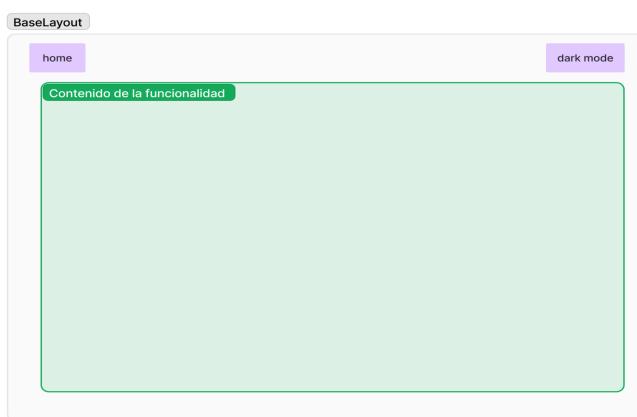
- **Promover el conocimiento farmacéutico:** Ayudar a los usuarios a aprender sobre farmacología de una manera interactiva y accesible.

#### 4.4 Interfaz, minimalista e interactiva

En esta sección, se presentarán los bocetos de la interfaz, diseñados con un enfoque minimalista e interactivo. Estos diseños buscan simplificar la experiencia del usuario, destacando elementos esenciales y eliminando distracciones innecesarias. La interactividad se ha priorizado para asegurar una navegación fluida y una experiencia intuitiva, permitiendo al usuario interactuar de manera eficiente con el sistema.

Uno de los aspectos más innovadores de esta interfaz es la incorporación de efectos de scroll y animaciones, que no solo mejoran la estética visual sino que también contribuyen a una navegación más dinámica y envolvente. Estos efectos están cuidadosamente diseñados para guiar la atención del usuario y proporcionar feedback visual en tiempo real, mejorando así la usabilidad y la experiencia general.

La combinación de un diseño minimalista con estos elementos interactivos crea una interfaz moderna y atractiva que facilita la interacción del usuario con la plataforma. A continuación, se detallan los bocetos (Hechos con [Figma](#)) que ilustran esta visión, destacando cómo cada componente visual y animado se integra para ofrecer una experiencia de usuario óptima.



**Imagen 4:** Boceto del Baselayout de donde van a extender las páginas de funcionalidades, volcando el contenido en el apartado verde.

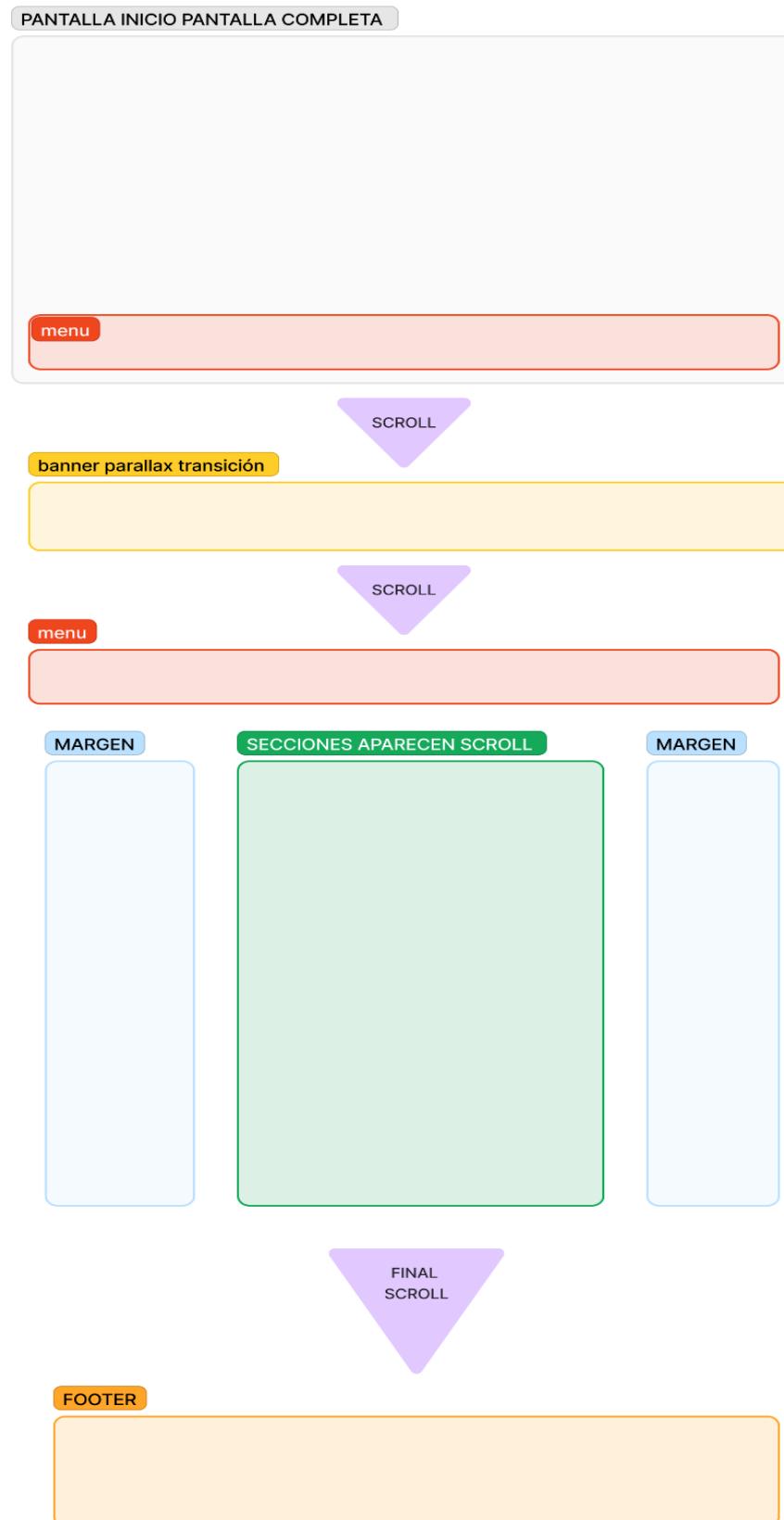


Imagen 5: Boceto de la Landing Page, el eje central de pharmAI

## 4.5 pharmAI, sus puntos fuertes

El fuerte del proyecto pasa porque utiliza tecnologías sólidas para gestionar el backend, como **Ollama**, permitiendo integrar una IA de la siguiente forma:

- **Local:** Ollama permite que la inteligencia artificial se ejecute directamente en el ordenador o servidor del usuario, eliminando la necesidad de depender de servidores externos y aumentando la privacidad y seguridad de los datos.
- **Escalable:** La capacidad de Ollama es escalable, lo que significa que al aumentar las capacidades del hardware, se pueden implementar IA más potentes. Esto permite adaptar la solución a las necesidades crecientes del proyecto, aprovechando al máximo los recursos disponibles.
- **Gratis:** A diferencia de muchas soluciones en el mercado que requieren suscripciones costosas o pagos por uso, Ollama proporciona una solución gratuita para integrar inteligencia artificial, lo que hace que sea accesible para proyectos de todos los tamaños y presupuestos.
- **Novedoso:** Ollama introduce métodos innovadores para la implementación y gestión de inteligencia artificial, permitiendo a los desarrolladores experimentar y adaptar las tecnologías de IA de manera flexible y eficiente.

A esto se suma las capacidades de **Astro** para construir una web moderna e interactiva con un rendimiento óptimo:

- **Rendimiento Óptimo:** Astro está diseñado para maximizar el rendimiento de la web, cargando solo el contenido necesario y optimizando la entrega de los recursos. Esto asegura una experiencia rápida y eficiente para el usuario final.
- **Moderno e Interactivo:** Utilizando las últimas tecnologías web, Astro permite crear interfaces modernas que son altamente interactivas, mejorando la experiencia del usuario con transiciones suaves y una navegación intuitiva.
- **Componentes Reutilizables:** Astro facilita la creación de componentes reutilizables, permitiendo una mayor consistencia y eficiencia en el desarrollo de la interfaz de usuario.
- **Integración con Múltiples Tecnologías:** Astro se integra perfectamente con una variedad de tecnologías modernas, como React, Vue, Svelte, y más, proporcionando una flexibilidad excepcional en la elección de herramientas y frameworks para el desarrollo web.

## 5. IMPLEMENTACIÓN

Una vez analizados en detalle los requisitos del proyecto y diseñada la solución basada en las tecnologías previamente expuestas, en este apartado se procederá a un análisis más exhaustivo de la implementación de dichas tecnologías hasta alcanzar el diseño final de la web. Se explicarán aspectos técnicos y de codificación relevantes que resultaron fundamentales para la integración de las diferentes funcionalidades de la plataforma. Esta sección abarca desde la estructura general del proyecto, pasando por el desarrollo del frontend y backend, hasta la integración de servicios externos y la gestión de la **base de datos**. Cada uno de estos elementos será detallado con ejemplos de código, decisiones arquitectónicas y mejores prácticas seguidas para garantizar una implementación eficiente, escalable y mantenible.

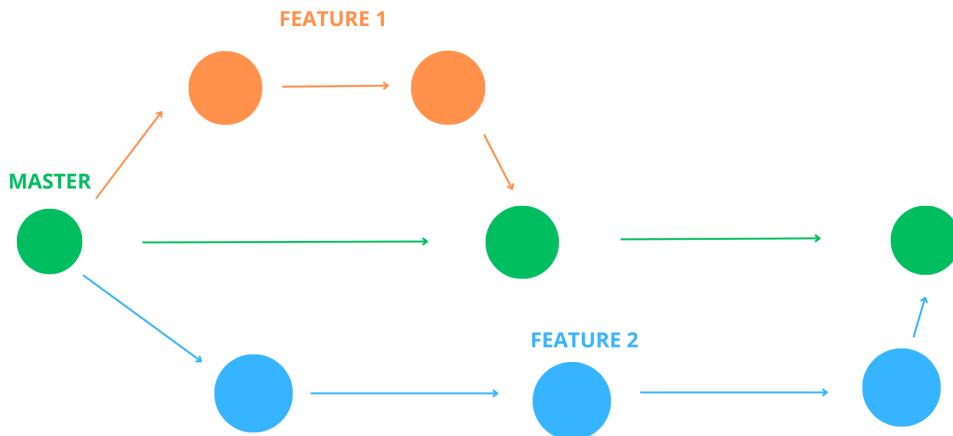
### 5.1 ¿Cómo se ha hecho el proyecto?

SISTEMA OPERATIVO	UBUNTU 22.04
EDITOR DE CÓDIGO	<a href="#"><u>VISUAL STUDIO CODE</u></a> <i>(CONFIGURADO PARA ASTRO)</i>
CONTROL DE VERSIONES	GIT Y GITHUB
HERRAMIENTAS UTILIZADAS	<i>TAILWIND PARA LA MAQUETACIÓN CSS, Y LIBRERÍAS JAVASCRIPT COMO SCROLL REVEAL, PARTICLES, LEAFLET, O TYPED PARA FUNCIONALIDADES DE LA WEB</i>

#### 5.1.1 Git, mantén tu proyecto a salvo

[Git](#) es un sistema de control de versiones distribuido que permite a los desarrolladores trabajar sin conexión, crear ramas para funcionalidades aisladas, y mantener un historial detallado de cambios. Facilita la colaboración y revisión de código mediante plataformas como GitHub, y se integra con herramientas de CI/CD para automatizar pruebas y despliegues. Su diseño garantiza la integridad y seguridad del proyecto.

Para la realización del proyecto se han creado dos repositorios, uno en fase temprana para hacer demos de funcionalidades aisladas, en el que el flujo del trabajo al ser más inestable era el que se visualiza en la **Imagen 6**.



**Imagen 6:** Flujo de git inicial. La rama master no recibe commits y siempre es estable. Se trabaja en las ramas de feature que una vez estén terminadas y estables se merguean en Master, siendo encargada cada rama de realizar una nueva implementación. Flujo seguido en el primer repositorio en la época temprana de desarrollo.

En la época final del desarrollo (segundo repositorio) y la integración de la landing page, se ha seguido un flujo básico de commits a master, ya que los cambios han estado más acotados y eran ajustes y nuevos añadidos básicos (**Imagen 7**). Durante esta fase, se priorizó la estabilidad y la corrección de errores menores.



**Imagen 7:** Flujo final realizando commits a master.

## 5.2 Construyendo con Astro

Imagina que quieres construir una imagen con piezas, lo que tradicionalmente se conoce como un puzzle. Cada pieza tendrá una forma y unas características determinadas, e individualmente estará definida mediante esos valores que he mencionado anteriormente. Entonces llega el momento de crear el puzzle y cuándo unes las piezas, ¡Oh, forman una imagen! Has creado una entidad superior con la unión de entidades menores, y así es como funciona Astro.

Y además Astro te permite reutilizar cuántas veces quieras esas piezas para crear las imágenes que quieras, pues son piezas que al tener un entidad propia, pueden reutilizarse y encajar en muchas imágenes diferentes. Por ejemplo piensa en un Sol, este sol puede encajar en muchas imágenes de paisajes, pero siempre será el mismo Sol.

Una vez introducido esto, hay que entender que en Astro a las piezas se le llaman **componentes** (Pueden haber componentes dentro de componentes, como una muñeca rusa) y a las imágenes se les llama **páginas**. Al final la unión de componentes crea una página y las páginas pueden reutilizar componentes. Y este juego de puzzles es la base de creación de un proyecto en Astro.

A continuación se detallan las características más relevantes de estos componentes:

1. **Isomorfismo:** Los componentes en Astro se pueden renderizar tanto en el servidor como en el cliente, optimizando la entrega de contenido y el rendimiento del sitio web.
2. **Reutilización Multiplataforma:** Los componentes son reutilizables en múltiples páginas, promoviendo la eficiencia en el desarrollo y la consistencia del diseño.
3. **Compatibilidad con JSX y TSX:** Los componentes pueden escribirse en JSX o TSX, facilitando la integración con proyectos existentes y el uso por desarrolladores familiarizados con React y TypeScript.
4. **Hidratación Parcial:** Astro permite la hidratación parcial, renderizando solo los componentes interactivos en el cliente, lo que reduce la carga de JavaScript y mejora el rendimiento.

5. **Integración con Frameworks Populares:** Astro es compatible con componentes de frameworks como React, Vue y Svelte, facilitando la integración y migración de proyectos.
6. **Estilos Encapsulados:** Los componentes pueden incluir estilos CSS exclusivos, evitando conflictos de estilo y asegurando una apariencia coherente.
7. **Optimización del Rendimiento:** El enfoque en la generación de sitios estáticos y la optimización de recursos mejora significativamente el rendimiento del sitio web.
8. **Concepto de Islas:** Astro utiliza el concepto de "islas", donde los componentes interactivos son aislados y renderizados independientemente, permitiendo una carga más eficiente y una mejor experiencia de usuario.

Es mejor ilustrar esto con un ejemplo sencillo sacado del código del proyecto (**Imagen 10**). En Astro, el bloque delimitado por `---` se llama "**Frontmatter**". Este bloque se utiliza para incluir configuraciones, metadatos y declaraciones de variables que afectan a la página o componente. Dentro de este bloque también se pueden importar módulos y componentes de JavaScript, permitiendo que se renderice JavaScript en la página. Después de esto, se visualiza el apartado donde se renderiza el html que va a servir el componente en la web, pero como ves, hay una etiqueta llamada Icon, que no es HTML, pues ¡Es una pieza del puzzle! y viene dada porque la hemos importado, es decir, hemos creado antes esa pieza para luego crear esta, que es una imagen superior de la anterior.

```
import { Icon } from "astro-icon/components";
```

**Imagen 8:** Importaciones en astro.

Este componente luego se invocará en el HTML con una sintaxis similar a las etiquetas HTML, renderizando el mismo en el lugar donde le indiques que tiene que estar.

```
<Icon name="mdi:home" />
```

**Imagen 9:** Llamada al componente en el HTML.

Por último recuerda que al añadir los estilos en los componentes, estos son similares a [Vue](#), se puede añadir tal y como se ve en el componente y son estilos que están encapsulados, evitando las colisiones con estilos de otros componentes o generales.



```

1  ---
2  import { Icon } from "astro-icon/components";
3  ---
4
5  <a href="/">
6      <button class="home-button animate__animated animate__fadeIn">
7          <Icon name="mdi:home" />
8      </button>
9  </a>
10
11 <style>
12     .home-button {
13         background: none;
14         border: none;
15         color: rgb(109, 112, 118);
16         display: block;
17         position: absolute;
18         top: 1.25rem;
19         left: 1.25rem;
20         --size: 2.25rem;
21         width: var(--size);
22         height: var(--size);
23         cursor: pointer;
24         z-index: 1;
25         font-size: var(--size);
26     }
27     .home-button:hover {
28         color: #20BF5B;
29     }
30     .home-button .astro-icon {
31         width: var(--size);
32         height: var(--size);
33     }
34 </style>
35

```

**Imagen 10:** Código del componente completo que renderiza el Logo de la página.

Es importante saber que estos componentes y las páginas tienen una extensión **.astro** si son integraciones propias de astro y siguen el siguiente esquema de directorios:

/project-root

```

|   └── /public      # Archivos estáticos (imágenes, fuentes, etc.)
|   └── /src        # Código fuente del proyecto
|       |   └── /components # Componentes reutilizables
|       |   └── /layouts    # Layouts de página
|       |   └── /pages     # Páginas del sitio
|       |   └── /styles    # Archivos CSS y estilos
|   └── /astro.config.mjs# Configuración de Astro
└── /package.json  # Dependencias y scripts del proyecto

```

Una vez entendido esto, es sencillo entender cómo se construye una página, pues... ¡Es igual! Solo tienes que crear un archivo en el **/pages** con el nombre que quieras que tenga la url, e importar componentes dentro del HTML y estilos quieras que tenga la página.

---

```
import Header from './components/Header.astro';
import Footer from './components/Footer.astro';
```

---

```
<Header />
<main>
  <h1>About Us</h1>
  <p>Welcome to the about page.</p>
</main>
<Footer />
```

o puedes crear layouts que luego extenderán las páginas.

---

```
import MainLayout from './layouts/MainLayout.astro';
```

---

```
<MainLayout>
  <h1>About Us</h1>
  <p>Welcome to the about page.</p>
</MainLayout>
```

Así es como se ha creado este proyecto en Astro, aplicando las bases de lo que he explicado y combinándolas para crear una web llamativa y moderna. Además para simplificar el uso de estilos, se ha usado [Tailwind](#) permitiendo una maquetación más rápida y eficiente.

Por último existe un archivo muy importante a la hora de trabajar con Astro, y es el [astro.config.mjs](#).

Este es el archivo de configuración principal para un proyecto Astro. Este archivo se utiliza para definir diversas opciones y ajustes que determinan cómo Astro construye y despliega tu sitio web.

Este archivo juega un papel importante en este proyecto porque aunque existen páginas estáticas, también tiene páginas dinámicas de funcionalidades que dan lugar a renderizado en el servidor, por lo que existe una directiva llamada **output** que le indicará a Astro como tiene que renderizar la salida del proyecto:

- '**static- '**server- '**hybrid******

### 5.3 Ollama en Astro

Una vez entendido cómo se construye en Astro, es importante entender cómo se crea la comunicación con los Endpoints para que Ollama funcione en el proyecto y sea un proceso interactivo con el usuario final. Pero para entender esto, primero tenemos que entender cómo se descarga y configura Ollama.

Y es que la descarga en Linux es muy sencilla, sólo tienes que seguir lo que te dice en este enlace y ya tendrás Ollama:

<https://ollama.com/download>

Una vez lo tengas, te levantará un servicio que escucha en el **puerto 11434**, y esa será la puerta de entrada para el Endpoint, pero nos falta algo fundamental, el **LLM**.

Sólo con poner **ollama** en la terminal podrás ver todo lo que te deja hacer, y si estás familiarizado con [Docker](#), verás que es muy similar en los comandos que utiliza.

### ¿Dónde encuentro los modelos?

Ollama ya pensó eso por ti, e hizo un marketplace gratuito de modelos, y aquí es dónde encontrarás todos los modelos disponibles:

<https://ollama.com/library>

Y descargar uno es tan sencillo como hacer un comando de este estilo en la terminal:

### ollama run gemma

y ya tendrás ese LLM en local, y podrás ejecutarlo, y preguntarle lo que quieras.

Al llegar a este punto hay que entender que ejecutar estos modelos, al ser un proceso en local, dependerá del hardware de tu máquina, y además esto permitirá la escalabilidad de forma sencilla en base a los recursos de las máquinas que tengas. Es decir, si me descargo un LLM muy pesado de ejecutar en mi ordenador con recursos limitados, irá muy lento y consumirá muchos recursos, pero ese mismo modelo en una máquina de altas especificaciones funcionará bien.

Y esto es una cosa muy a tener en cuenta, porque dependiendo de los billones de parámetros que tenga (**B**), será más pesado o rápido de correr y te requerirá más capacidad de hardware.

Tras hacer un estudio de diferentes modelos y la ejecución en mi máquina, decidí quedarme con la versión light de Gemma (**Imagen 11**) ya que no es pesada de emular y da respuestas que están bien en relación al tiempo que tarda en procesarse, pero hay que tener en cuenta que es un modelo pequeño y no es igual que por ejemplo [llama3](#) de facebook, que tiene **8B** de parámetros, la versión de Gemma que tiene el proyecto es de **2B**.



**Imagen 11:** Versión de LLM elegida para el proyecto.

Pero no te preocupes, no sólo puedes tener un modelo, puedes tener todos los que quieras, por lo que volviendo a lo de la escalabilidad, puedes conseguir que en un mismo proyecto y máquina, coexistan múltiples modelos a la vez, como las diferentes versiones de chatGPT, pero para esta demo técnica, el modelo **gemma:2B** es ideal.

## ¿ Cómo conectar esto con pharmAI ?

La respuesta a esta pregunta ya se ha contestado, por medio de un Endpoint, pero mejor ilustrarlo.



```

1 export async function makeAskRequest(question) {
2   const url = "http://localhost:11434/api/generate";
3   const headers = {
4     "Content-Type": "application/json",
5     "Accept": "application/json"
6   };
7
8   const body = JSON.stringify({
9     model: "gemma:2b",
10    prompt: question,
11    stream: false // Asegurando que se incluya el campo 'stream' como en el ejemplo original
12  });
13
14  try {
15    const response = await fetch(url, {
16      method: "POST",
17      headers: headers,
18      body: body
19    });
20
21    if (!response.ok) {
22      const errorData = await response.json(); // Asumiendo que la API devuelve un JSON con detalles del error
23      const error = new Error(`Error ${errorData.code}: ${errorData.description}`);
24      throw error;
25    }
26
27    const data = await response.json(); // Asumiendo éxito, se parsea la respuesta JSON
28    return data; // Retornar los datos JSON parseados
29  } catch (error) {
30    console.error("Error en la petición:", error);
31    throw error; // Re-lanzar el error para manejo externo
32  }
33}
34

```

**Imagen 12:** Función que hace la llamada al Endpoint de Ollama.

Y como ves en la imagen 12, en el body de la petición a Ollama, hay un apartado donde especifica el modelo que quieras utilizar. Pero si quieras usar otro, simplemente lo descargarías en Ollama y lo especificamos aquí, y listo ¡Ya tendrías otro Modelo!

Además puedes personalizar la pregunta para que te hable con el tono que quieras o sea más específico en campos sanitarios por ejemplo.

Y este es el archivo **Javascript** o **TypeScript** que se encargará de hacer la llamada y traérsela a Astro para que la renderice.

## 5.4 Astro DB, una SQLite en Astro

Durante todo el desarrollo de esta documentación se ha hecho énfasis en que las principales funcionalidades de este proyecto vienen dadas por Endpoints, y esto justifica la elección de Astro como el framework web elegido para desarrollar pharmAI. Y es cierto que la funcionalidad estrella y las demás complementarias vienen dadas por APIs u Ollama. Entonces en este punto, se pensó en cómo se podría integrar una base de datos en este proyecto, ya que es uno de los puntos a valorar del mismo y tras valorar la utilidad de la misma, la respuesta fue clara... **Los usuarios.** Si pharmAI es una plataforma es importante poner en valor el papel de los usuarios dentro de ella y darle funcionalidades en base a su perfil.

Llegando aquí, volví a recordar haber visto en YouTube que en Abril de 2024 aproximadamente, Astro sacó una implementación a su ecosistema, [Astro DB](#).

Astro DB es una implementación de [LibSQL](#) (un fork de SQLite que hereda su base de código pero incorpora cambios y mejoras específicas) que permite crear una base de datos ligera y basada en el ecosistema SQL de las bases de datos relacionales. Este sistema proporciona una solución eficiente y compacta para gestionar datos, aprovechando la simplicidad y familiaridad de SQL, lo que facilita el desarrollo y mantenimiento de aplicaciones. Astro DB es ideal para proyectos que requieren una base de datos con bajo consumo de recursos, portabilidad y capacidad de integración con diversas plataformas y lenguajes de programación. Su diseño flexible y extensible permite personalizar la funcionalidad según las necesidades específicas del proyecto.

Si este proyecto iba a estar programado en Astro, y la integración con Astro DB es sencilla, el uso de Astro DB estaba justificado y además es novedoso, pues apenas tiene dos meses de recorrido, viéndose a día de hoy implementaciones sencillas y testeos de funcionalidades, pero que para la gestión de usuarios es más que suficiente.

Y si la funcionalidad de la base de datos será la gestión de usuarios, debe haber un actor secundario, el código que se encarga de toda esta gestión de los mismos, encontrando que en Astro DB se usa [lucia Auth](#).

#### 5.4.1 Gestión de usuarios: Lucia Auth

La implementación de Lucia Auth permite:

- **Autenticación:** Proporciona métodos seguros para el registro, inicio de sesión y cierre de sesión de usuarios.
- **Autorización:** Permite la gestión de roles y permisos para controlar el acceso a diferentes partes de la aplicación.
- **Gestión de Sesiones:** Maneja sesiones de usuario de forma segura, con opciones para mantener sesiones activas y gestionar expiraciones.
- **Seguridad:** Implementa prácticas de seguridad estándar, como hashing de contraseñas, y soporta autenticación multifactor (MFA).

### PASOS PARA CONFIGURAR LUCIA AUTH

1. **Instalación:** Primero, instalar Lucia Auth en el proyecto Astro usando un gestor de paquetes como npm o pnpm.
2. **Configuración:** Configurar Lucia Auth para que funcione con la base de datos y establecer una clave secreta para firmar tokens de autenticación. También determinar el entorno (desarrollo o producción) para ajustar configuraciones específicas de seguridad.
3. **Registro de Usuarios:** Implementar un sistema de registro donde los usuarios pueden crear cuentas proporcionando un nombre de usuario y una contraseña. Lucia Auth se encarga de almacenar esta información de forma segura.
4. **Inicio de Sesión:** Proporcionar una interfaz de inicio de sesión para que los usuarios puedan autenticarse. Lucia Auth verifica las credenciales y crea sesiones seguras para los usuarios.
5. **Gestión de Sesiones:** Lucia Auth maneja las sesiones de usuario, incluyendo la creación, expiración y cierre de sesiones, garantizando que las sesiones sean seguras.
6. **Roles y Permisos:** Permite definir roles y permisos en los proyectos para controlar el acceso a diferentes partes de la aplicación. Por ejemplo, algunos usuarios pueden tener permisos de administrador, mientras que otros solo tienen acceso como usuarios normales (No se ha aplicado en el proyecto porque sólo queremos a un tipo de usuario).

7. **Medidas de Seguridad:** Implementa medidas de seguridad adicionales, como la autenticación multifactor (MFA), para proteger las cuentas de los usuarios y la aplicación en general.

#### 5.4.2 Sintaxis y Drizzle ORM

Astro DB usa [Drizzle](#), un ORM (Object-Relational Mapper) para interactuar con bases de datos SQL utilizando código JavaScript o TypeScript. Drizzle facilita las operaciones de bases de datos proporcionando una API moderna y sencilla para realizar consultas, insertar, actualizar y eliminar datos sin tener que escribir consultas SQL directamente.

Con Astro DB y Drizzle estás haciendo consultas de SQL con una sintaxis similar a esta, pero con una capa de abstracción mayor y en JavaScript o TypeScript. Por ejemplo, en la **Imagen 13** se ve la definición de una tabla del proyecto, la de recordatorios de medicamentos.



```

1 // Definir tabla de recordatorios de medicación
2 const MedicationReminder = defineTable({
3   columns: {
4     id: column.number({ primaryKey: true, autoIncrement: true }), // Autoincremental
5     userId: column.text({ optional: false, references: () => User.columns.id }), // Referencia a User
6     description: column.text(),
7     medication: column.text(),
8     time: column.date(),
9   }
10 });

```

**Imagen 13:** Definición de una tabla en Astro DB Con Drizzle ORM, la sintaxis de JavaScript con conceptos de SQL.

Esto creará una tabla en la base de datos cuando se levante el servidor, y en este proceso también entra en juego los **imports y exports** de JavaScript, pues para utilizar estos conceptos hay que importarlos.

```
import { defineDb, defineTable, column } from "astro:db";
```

**Imagen 14:** Importaciones en el db/config.ts

y una vez creada la tabla hay que exportarla, tal y como se ve en el código del proyecto en la siguiente imagen:



```

1 // Exportar la configuración de la base de datos
2 export default defineDb({
3   tables: [
4     User,
5     Session,
6     MedicationReminder,
7     Post,
8     Comment,
9     Like,
10    ],
11  });
12

```

**Imagen 15:** Exportaciones de las tablas.

Teniendo en cuenta esto, es fácil entender cómo funcionarán procedimientos como insertar datos , eliminarlos (**Imagen 16**) o seleccionar datos de la base de datos (**Imagen 17**). En este caso no es necesario exportar nada, pero el funcionamiento es igual, y la sintaxis es similar a la explicada en la **Imagen 13**.



```

1 if (Astro.request.method === 'POST') {
2   const formData = await Astro.request.formData();
3   const action = formData.get('action');
4
5   if (action === 'insert') {
6     const description = formData.get('description');
7     const medication = formData.get('medication');
8     const time = formData.get('time');
9     const userId = user.id;
10
11    if (
12      typeof description === 'string' &&
13      typeof medication === 'string' &&
14      typeof time === 'string' &&
15      typeof userId === 'string'
16    ) {
17      await db.insert(MedicationReminder).values({
18        description,
19        medication,
20        time: new Date(time),
21        userId,
22      });
23    }
24  } else if (action === 'delete') {
25    const id = formData.get('id');
26    if (typeof id === 'string') {
27      await db.delete(MedicationReminder).where(eq(MedicationReminder.id, Number(id)));
28    }
29  }
30}

```

**Imagen 16:** Proceso de inserción o eliminación de datos en la base de datos.



The screenshot shows a mobile application interface with a dark background. At the top left are three colored dots (red, yellow, green). Below them is a code block in a monospaced font:

```
1 if (user) {  
2     reminders = await db.select()  
3         .from(MedicationReminder)  
4         .where(eq(MedicationReminder.userId, user.id))  
5         .orderBy(desc(MedicationReminder.id));  
6 }
```

**Imagen 17:** Proceso de selección de datos en la base de datos.

Si quieras saber más al respecto siempre puedes visitar las páginas de Astro DB para ver ejemplos de implementaciones o Drizzle ORM.

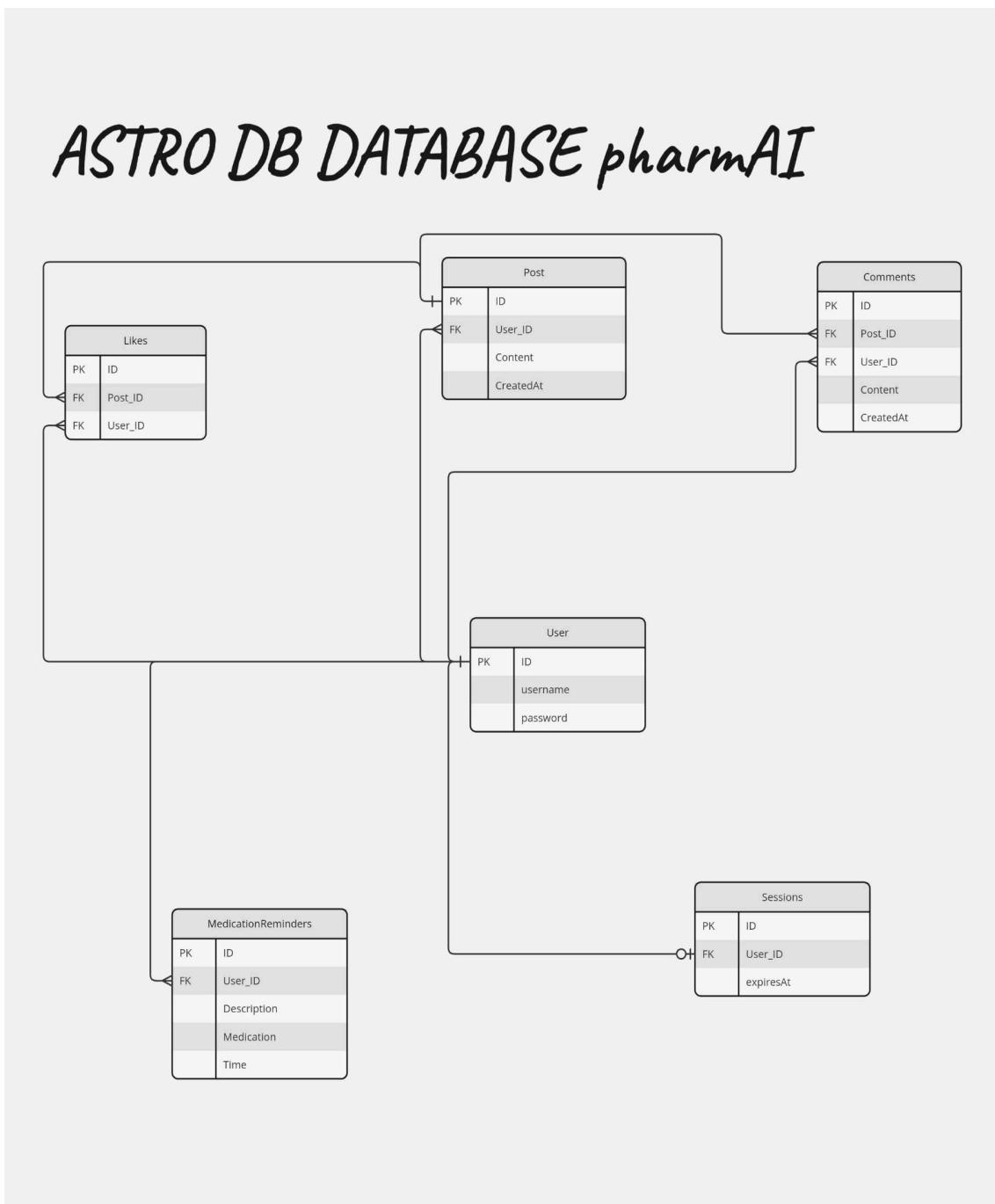
#### 5.4.3 Diagrama entidad-relación

Antes de introducir el diagrama entidad-relación de la base de datos de pharmAI, hay que entender que todo en la base de datos gira en torno a la entidad del **Usuario**.

Si no te logeas, no puedes hacer ninguna funcionalidad relacionada con la base de datos, por lo que en este esquema la tabla **Usuario** y el **ID** (Primary Key) del mismo siempre serán los datos que identifiquen inequívocamente las relaciones, por lo que siempre existirá una Foreign Key del usuario en cada tabla, y todas las relaciones será **1:N**, el usuario puede poner muchos recordatorios pero los recordatorios sólo pertenecen a un usuario, excepto la relación con la tabla **Sesiones**, ya que un usuario puede tener o no una sesión, pero como he remarcado antes, en esta relación el ID del usuario pasa a la tabla de Sesiones como Foreign Key.

Otra tabla relevante es la de los **Post**, pues tienen likes y comentarios, pero siguen siendo relaciones **1:N** en la que el ID del post va a las respectivas tablas, pues el Post es el que identifica inequívocamente los datos.

Al final todo gira en torno al usuario y es por ello que es la tabla que interactúa con todas. Además los Posts también tienen relaciones con otras tablas de la misma índole.



**Imagen 18:** Esquema entidad-relación de la base de datos PharmAI.

Esta implementación permite a los usuarios poner **recordatorios** que son únicos de cada usuario y mediante llamadas recursivas cada minuto, verificar si ya ha pasado y si es así mostrarlo en la pantalla y eliminarlo.

Además permite a los usuarios tener un apartado de **comunidad** donde escribir posts y poner comentarios y likes, siempre manteniendo la integridad de los datos filtrados por cada usuario.

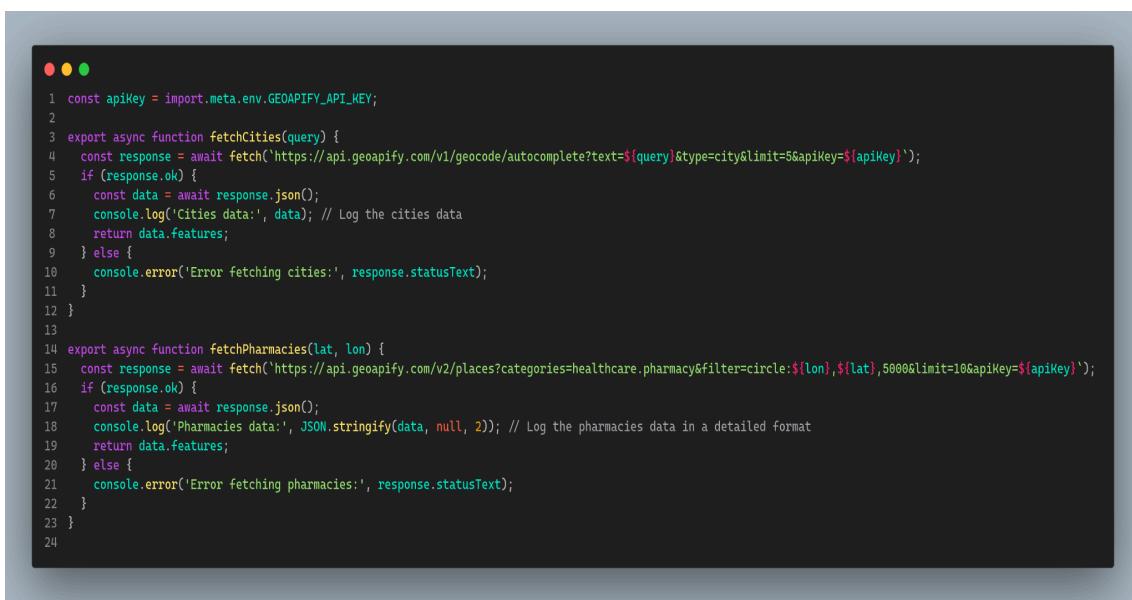
## 5.5 Otras funcionalidades, Cuestión de APIs

Cumpliendo con el concepto integral de plataformas, en este proyecto se han incluido más funcionalidades que se han creído que pueden ser útiles para un usuario que entra en pharmAI y quiere encontrar:

- **Frases sobre Farmacología** para aprender un poco de la misma cada vez que entra en la landing page.
- **Información detallada de los medicamentos** de la mano de CIMA, así como sus prospectos con un simple click.
- **Farmacias cercanas** en un mapa interactivo creado con Leaflet.

Y todas estas implementaciones son similares a lo ya visto con Ollama, son EndPoints que se ejecutan cuando el usuario hace alguna acción, ya sea escribir en el buscador o entrar en la landing page.

En el caso de CIMA y las farmacias cercanas, el funcionamiento es similar, las llamadas se hacen en archivos Typescript que están metidos en **src/pages/api** y es el punto de intercambio de datos entre Astro y las diferentes Apis. Una vez que tiene los datos Astro se encarga de construirlos en la web y así se establece el intercambio.



```

1 const apiKey = import.meta.env.GEOAPIFY_API_KEY;
2
3 export async function fetchCities(query) {
4   const response = await fetch(`https://api.geocapify.com/v1/geocode/autocomplete?text=${query}&type=city&limit=5&apiKey=${apiKey}`);
5   if (response.ok) {
6     const data = await response.json();
7     console.log('Cities data:', data); // Log the cities data
8     return data.features;
9   } else {
10     console.error('Error fetching cities:', response.statusText);
11   }
12 }
13
14 export async function fetchPharmacies(lat, lon) {
15   const response = await fetch(`https://api.geocapify.com/v2/places?categories=healthcare.pharmacy&filter=circle:${lon},${lat},5000&limit=10&apiKey=${apiKey}`);
16   if (response.ok) {
17     const data = await response.json();
18     console.log('Pharmacies data:', JSON.stringify(data, null, 2)); // Log the pharmacies data in a detailed format
19     return data.features;
20   } else {
21     console.error('Error fetching pharmacies:', response.statusText);
22   }
23 }
24

```

**Imagen 19:** Código de implementación de las llamadas a las APIs que construye Astro.

Tanto en la API de CIMA como en la de GeoApify para la obtención de farmacias cercanas, el procedimiento es el mismo. Sin embargo en el caso de la API que ofrece frases random de farmacología, todo se ejecuta en local, pues la API la he creado yo con **Python y Flask** para que corra en el **puerto 11111 (Imagen 20)**.



```
● ● ●

1 from flask import Flask, jsonify
2 from flask_cors import CORS
3 import random
4
5 app = Flask(__name__)
6 CORS(app) # Esta línea permite todas las solicitudes CORS
7
8 # Lista extensa de frases de farmacología
9 phrases = [
10     "aqui van las frases"
11 ]
12
13 @app.route('/random_phrase', methods=['POST'])
14 def get_random_phrase():
15     phrase = random.choice(phrases)
16     return jsonify({"phrase": phrase})
17
18 if __name__ == '__main__':
19     app.run(port=11111)
20
21
```

**Imagen 20:** Código en Python de la API que se sirve en el puerto 11111 en local.

Todas estas funcionalidades son accesibles desde la landing page, siguiendo un **esquema Radial** como se ve en el **anexo 3**.

## 5.6 Resumen del funcionamiento de PharmAI

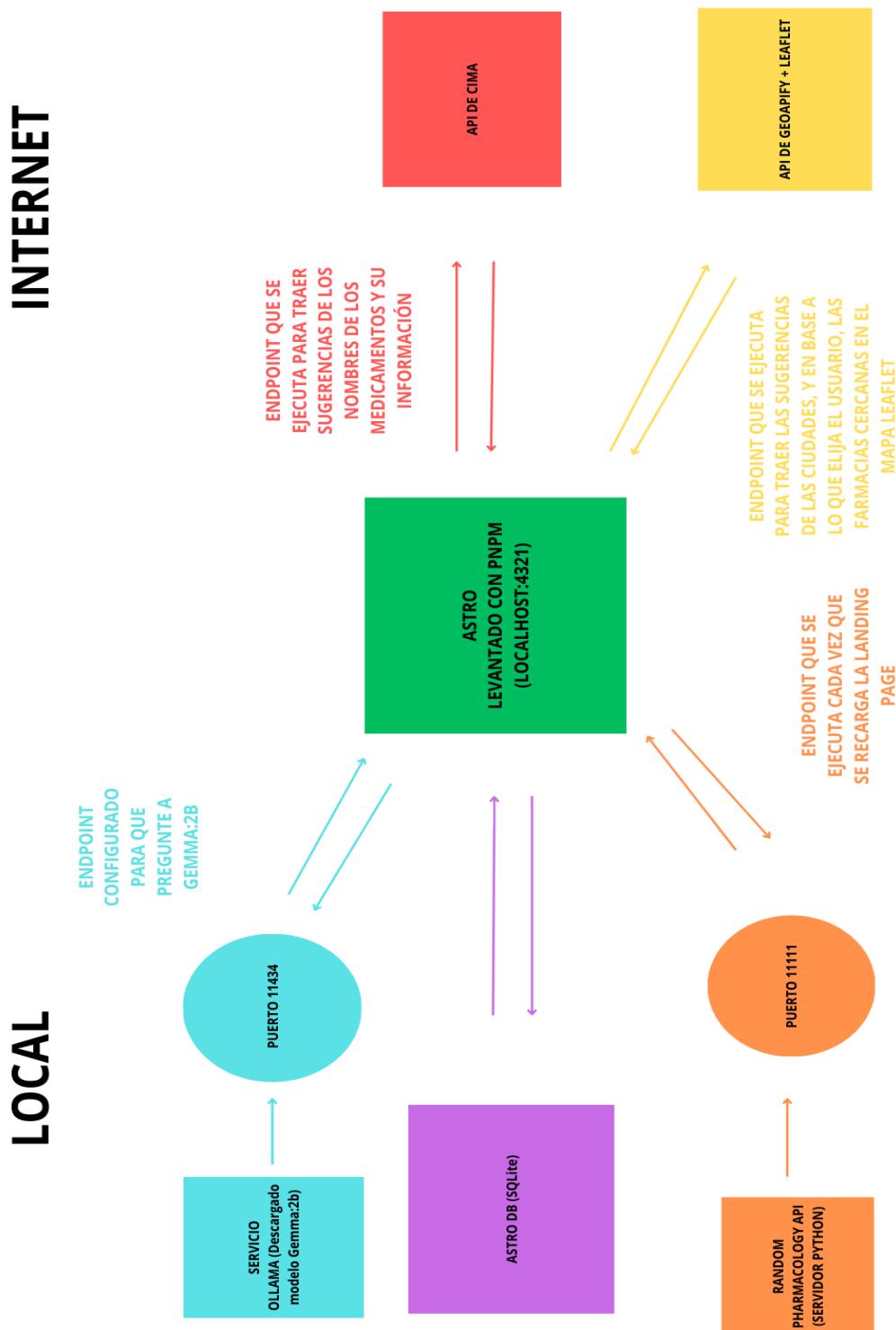


Imagen 21: Esquema resumen del funcionamiento de pharmAI.

## 5.7 Implementando con identidad

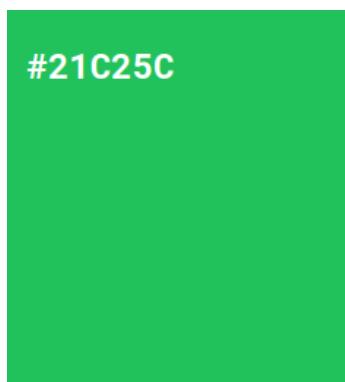
Cubiertos todos los aspectos técnicos del proyecto, queda un apartado que es también muy importante en el diseño de una web: **Tener una identidad propia**.

Por eso en pharmAI siempre se buscó el diseño de una identidad propia desde que entras en la aplicación y esto se consigue mediante la introducción de **pharmaBot** y esquemas de colores que identifican la marca pharmAI.



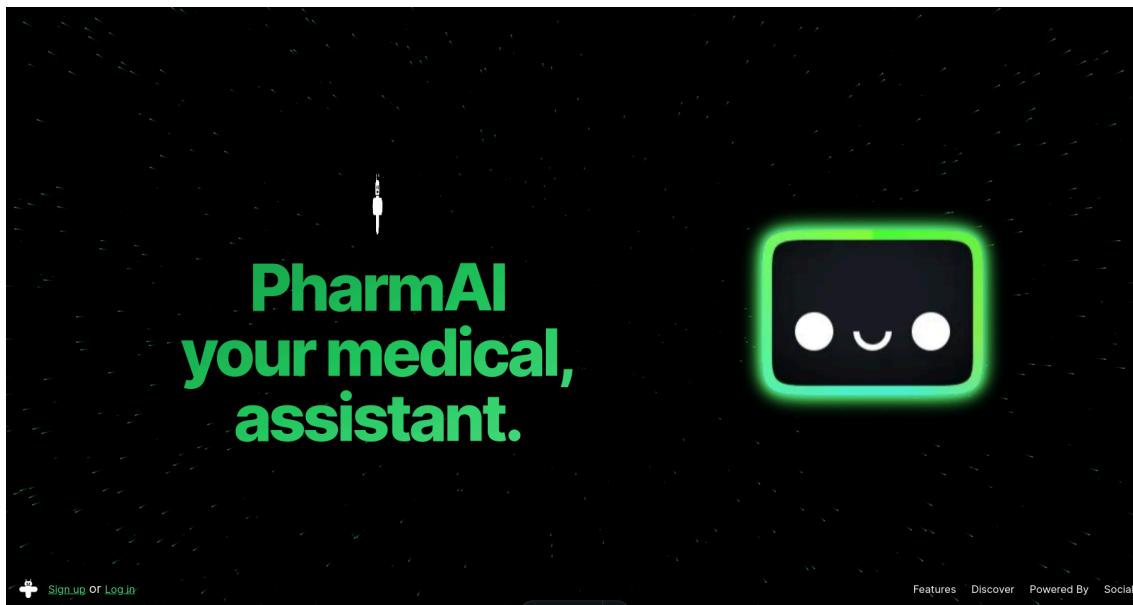
**Imagen 22:** pharmAI, el asistente virtual de pharmAI.

Además se ha seguido un esquema de verdes en la web acordes al siguiente color:



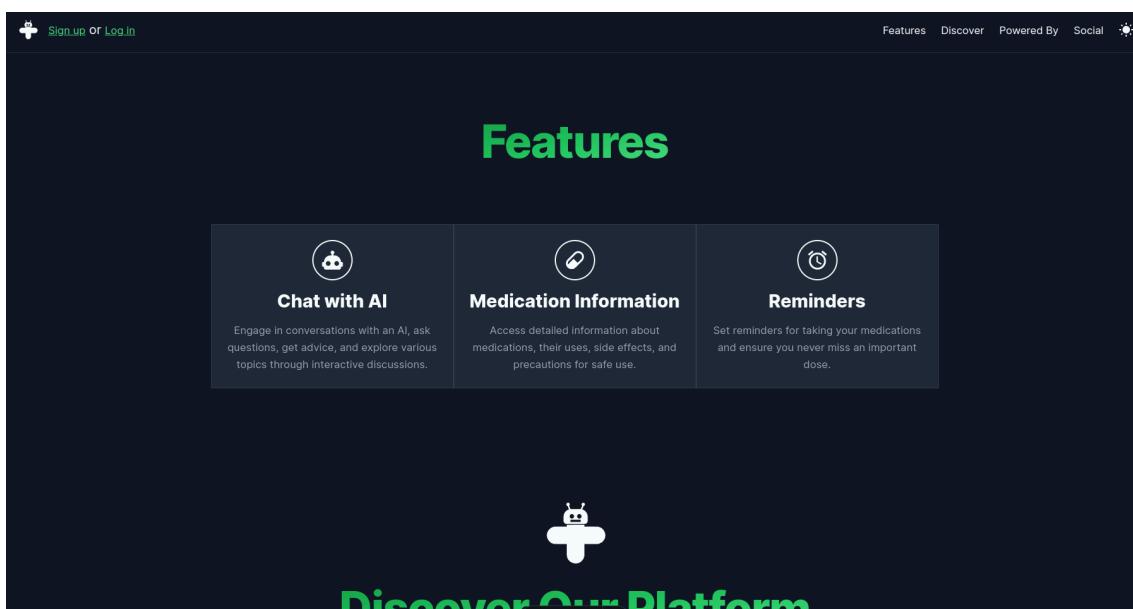
**Imagen 23:** Tonalidad base elegida sobre el que gira todo el contenido de la web.

La web tiene una identidad propia visible desde que entras en ella, implementando una pantalla de inicio llamativa cuando el usuario entra con efectos visibles de esteroides viniendo hacia él, haciendo un símil de cómo la tecnología viene al campo de la farmacia de la mano de este proyecto.



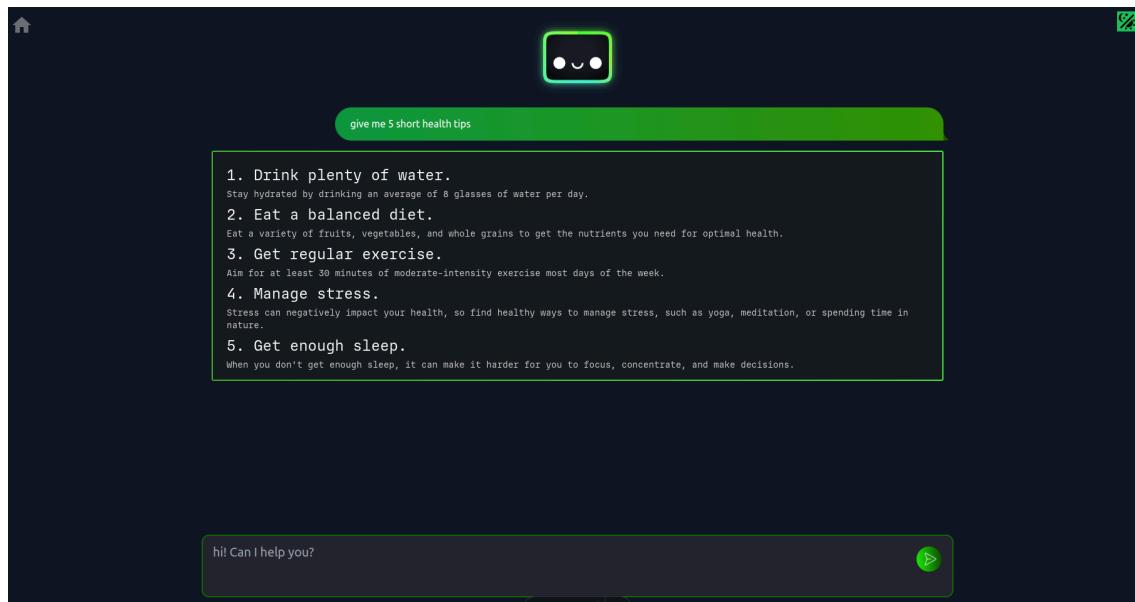
**Imagen 24:** Pantalla de inicio de pharmAI.

Con los diferentes efectos de scroll aparecen las demás secciones de la landing, pues la landing es el eje central de pharmAI.



**Imagen 25:** Ejemplo de sección de la landing con estilo moderno y minimalista.

El chat también destaca por su aspecto moderno e interactivo, siempre con **pharmaBot** arriba presente y una salida clara y concisa (En código me he encargado de formatear la salida estándar de Ollama para hacerla acorde y bonita en el chat).



**Imagen 26:** Ejemplo de una charla sencilla con pharmabot.

La salida de información es clasificada para que llegue al usuario de la forma más clara posible y no tenga que llevarse largos períodos de tiempo buscando los datos como en otras plataformas de Farmacia.

The screenshot shows a search results page for the term "paracetamol". The top section is titled "Search Information" and has a search bar containing "paracetamol". Below the search bar is a green horizontal bar with the text "Search".

The search results are displayed in three cards:

- PARACETAMOL ABAMED 1 G COMPRIMIDOS EFG**  
Laboratorio titular: Abamed Pharma S.L.  
Prescripción: Medicamento Sujeto A Prescripción Médica  
Comercializado: No  
Necesita receta: Sí  
Vía de administración: VÍA ORAL  
Forma farmacéutica: COMPRIMIDO  
[Ficha Técnica PDF](#) [Web Ficha Técnica](#)
- PARACETAMOL ACCORD 10 mg/ml SOLUCION PARA PERFUSION EFG**  
Laboratorio titular: Accord Healthcare S.L.U.  
Prescripción: Uso Hospitalario  
Comercializado: No  
Necesita receta: Sí  
Vía de administración: VÍA INTRAVENOSA  
Forma farmacéutica: SOLUCIÓN PARA PERFUSIÓN  
[Ficha Técnica PDF](#) [Web Ficha Técnica](#)
- PARACETAMOL B.BRAUN 10 mg/ml SOLUCION PARA PERFUSION EFG**  
Laboratorio titular: B. Braun Melsungen Ag

Each card includes a product image, a ruler scale for reference, and a detailed product label.

**Imagen 27:** Salida de una búsqueda sencilla.

Estos ejemplos ilustran un poco la filosofía que se comentó a la hora de crear pharmAI, y son sólo ilustrativos. El proyecto ha sido diseñado para que luzca cuándo se está usando y si quieras ver su verdadero encanto y descubrir todos los secretos de la plataforma, recomiendo probarla en vivo, pues es el fuerte del proyecto.

## 6. PRUEBAS

El desarrollo con Astro ha sido bastante ameno, por lo que el tiempo invertido en testear la aplicación se ha reducido considerablemente al crear funcionalidades por separado lo más sencillas posibles y luego adaptarlas a pharmAI y Astro para conseguir la implementación deseada dentro de la aplicación, por lo que los testeos de la aplicación han consistido basicamente en tres casuísticas:

- **Debug de código** en búsqueda de errores apoyándome en la consola de desarrollador de Google Chrome o plugins de Visual Studio Code como Console Ninja con el objetivo ver qué tienen las variables y si es lo esperado.
- **Pruebas como si fuera un usuario de la plataforma:** Estas pruebas han sido las más extensas a la hora de depurar la aplicación, probando cómo salen los mensajes en el chat para ajustarlo, poner recordatorios con varios usuarios, para ver que todo va bien y testeo del apartado de comunidad e información de medicamentos.
- Ajustes de maquetación para que la web tenga una apariencia sólida y **responsiva**.

Para chequear en todo momento la base de datos, se ha creado una ruta no accesible desde la navegación de la web, que permite ver el estado de la base de datos en tiempo real, siendo esta **/dbcheck**.

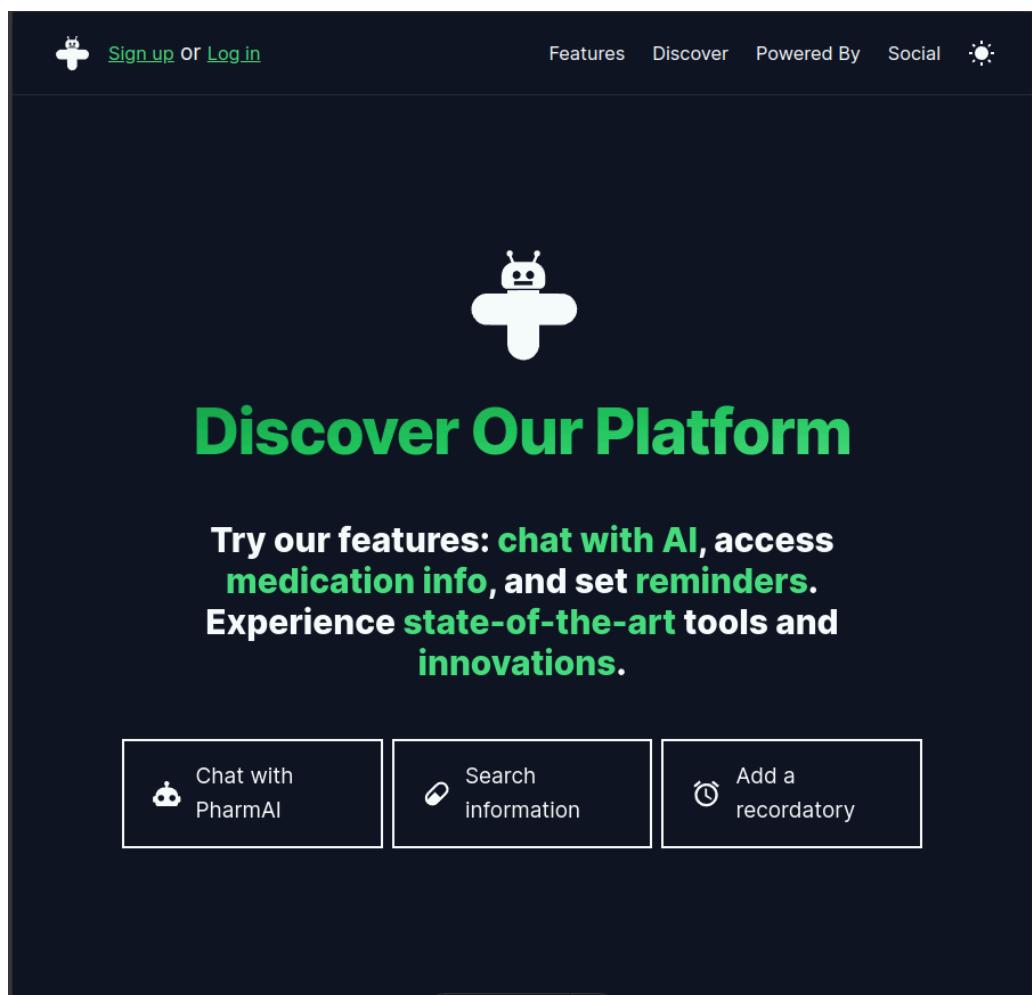
Database Schema					
Sessions	Users	Medication Reminders	Posts	Comments	Likes
<b>ID (PK)</b> : Unique identifier for session	<b>ID (PK)</b> : Unique identifier for user	<b>ID (PK)</b> : Unique identifier for reminder	<b>ID (PK)</b> : Unique identifier for post	<b>ID (PK)</b> : Unique identifier for comment	<b>ID (PK)</b> : Unique identifier for like
<b>User ID (FK)</b> : Identifier of the user	<b>Username</b> : Username of the user	<b>User ID (PK)</b> : Identifier of the user	<b>User ID (FK)</b> : Identifier of the user	<b>Post ID (PK)</b> : Identifier of the post	<b>Post ID (FK)</b> : Identifier of the post
<b>Expires At</b> : Expiration date and time of the session	<b>Password</b> : Password of the user	<b>Description</b> : Description of the reminder	<b>Content</b> : Content of the post	<b>Content</b> : Content of the comment	<b>User ID (FK)</b> : Identifier of the User
		<b>Medication</b> : Name of the medication	<b>Created At</b> : Date and time when the post was created	<b>Created At</b> : Date and time when the comment was created	
		<b>Time</b> : Date and time of the reminder			
<b>Users</b>					
ID		Username		Password	
<b>Sessions</b>					
ID	User ID			Expires At	
<b>Medication Reminders</b>					
ID	User ID	Description	Medication		Time
<b>Posts</b>					
ID	User ID	Content		Created At	

**Imagen 28:** Ruta para chequear la base de datos en tiempo real.

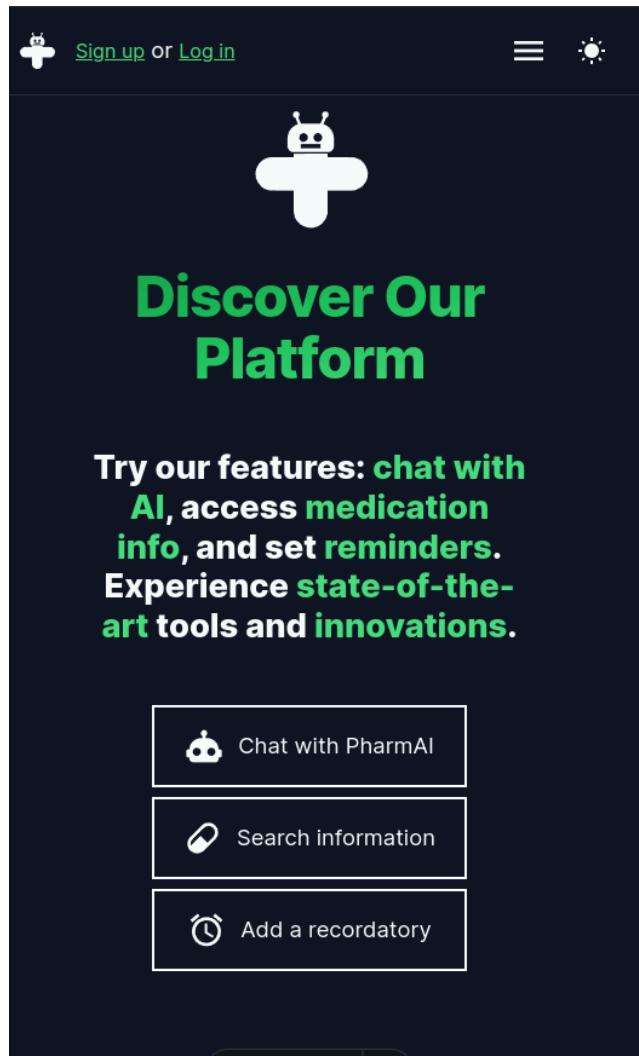
## 6.1 Responsividad, algo básico

La responsividad se ha convertido en un pilar esencial en el diseño y desarrollo de sitios web modernos debido a su impacto significativo en la accesibilidad y usabilidad. Un sitio web responsive es aquel que se adapta de manera fluida a diferentes tamaños de pantalla y dispositivos, proporcionando una experiencia de usuario coherente y satisfactoria sin importar si se accede desde una computadora de escritorio, una tableta o un teléfono móvil.

Sabiendo esto, si una aplicación actual quiere alcanzar al máximo número de usuarios, tiene que ser responsive y tener una buena apariencia en el móvil. Por esto, durante el desarrollo se han estado haciendo pruebas constantes para garantizar la responsividad en toda la plataforma.



**Imagen 29:** Diseño responsive en tablet.



**Imagen 30:** Diseño responsive en móvil

## 7. COSTES/PRESUPUESTO

El presupuesto para el desarrollo y mantenimiento de pharmAI se desglosa en las siguientes categorías: desarrollo, infraestructura, licencias y servicios, marketing y promoción, y otros gastos. En lugar de enfocarse en los precios específicos, me centraré en las necesidades y requerimientos del proyecto si se quiere profesionalizar, con una mirada hacia futuras expansiones.

### Necesidades de Desarrollo

- Diseño y desarrollo de la interfaz de usuario (UI/UX).
- Implementación de funciones avanzadas de IA y chat interactivo.
- Creación de bases de datos y gestión de usuarios.
- Desarrollo de funcionalidades adicionales.

Concepto	Detalles
Diseño de Interfaz	Crear una UI/UX intuitiva y amigable.
Desarrollo Frontend	Implementar la interfaz y la experiencia de usuario.
Desarrollo Backend	Manejo de bases de datos y lógica del servidor.
Integración de IA	Implementar funciones de IA con modelos locales.

### Necesidades de Infraestructura

- Servidores para alojar la plataforma.
- Almacenamiento de datos.
- Servicios de seguridad y backups.

Concepto	Detalles
Servidores	Hospedaje para la plataforma web.
Almacenamiento	Espacio para bases de datos y archivos.
Seguridad y Backups	Protección de datos y copias de seguridad.

## Necesidades de Licencias y Servicios

- Herramientas de desarrollo y colaboración.
- APIs externas para información sobre medicamentos y localización de farmacias.

Concepto	Detalles
Herramientas de Desarrollo	Software como editores de código y plataformas de control de versiones.
APIs Externas	Acceso a APIs de forma profesional que ofrezcan servicios acordes a la plataforma

## Necesidades de Marketing y Promoción

- Estrategias para dar a conocer la plataforma.
- Publicidad en redes sociales y motores de búsqueda.

Concepto	Detalles
Marketing Digital	Campañas en redes sociales y SEO.
Publicidad	Promoción en Google Ads y similares.

## Necesidades de Otros Gastos

- Equipos y software necesarios para el desarrollo.
- Costos legales y administrativos.

Concepto	Detalles
Equipos y Software	Hardware y software adicionales.
Costos Legales y Administrativos	Registro de la plataforma y otros trámites.

## Posibilidades de Expansión

En el futuro, **pharmAI** podría expandirse para incluir:

- **Aplicación Móvil:** Desarrollo de apps para iOS y Android.
- **Conexión con Profesionales:** Integración de consultas en línea con farmacéuticos y médicos.
- **Funciones Avanzadas de IA:** Implementación de modelos de IA más complejos y personalizados.
- **Internacionalización:** Adaptación de la plataforma para diferentes idiomas y mercados.

Concepto	Detalles
Aplicación Móvil	Desarrollo de apps para diversas plataformas.
Consultas en Línea	Conexión con profesionales de la salud.
IA Avanzada	Modelos más complejos y personalizados.
Internacionalización	Adaptación a diferentes idiomas y mercados.

Este desglose proporciona una visión clara de las necesidades y requerimientos del proyecto **pharmAI**, y ofrece una guía para futuras expansiones.

## 8. CONCLUSIONES

Desarrollar **pharmAI** ha sido una experiencia sumamente enriquecedora y desafiante. Este proyecto nació de la idea de combinar mis conocimientos en el campo de la farmacia con las habilidades adquiridas en el desarrollo de aplicaciones web, y ha evolucionado hasta convertirse en una plataforma innovadora que utiliza la inteligencia artificial para mejorar la accesibilidad a la información médica y farmacéutica.

A lo largo del desarrollo de **pharmAI**, he podido explorar y aplicar diversas tecnologías modernas, desde frameworks de frontend como Astro hasta la integración de modelos de IA locales con Ollama. Cada decisión técnica, desde la elección del framework hasta la implementación de APIs, fue estudiada para garantizar que la plataforma no solo fuera funcional, sino también accesible, intuitiva y eficiente.

Uno de los aspectos más gratificantes de este proyecto ha sido ver cómo una idea puede transformarse en una herramienta práctica que tiene el potencial de impactar positivamente en la vida de las personas. **pharmAI** no solo facilita el acceso a la información sobre medicamentos y farmacias cercanas, sino que también ayuda a educar a los usuarios y mejorar su gestión de la salud mediante recordatorios y un asistente virtual personalizado, **pharmaBot**, la funcionalidad estrella de la plataforma.

La creación de **pharmAI** también ha subrayado la importancia de la innovación y la adaptación en el campo de la farmacia. La resistencia al cambio es un desafío real en muchos sectores, incluyendo el farmacéutico, pero proyectos como este demuestran que la integración de nuevas tecnologías puede ser de ayuda tanto para los profesionales de la salud como para los pacientes.

En términos personales, este proyecto ha sido una oportunidad para profundizar en mi capacidad de resolver problemas, aprender nuevas tecnologías y aplicar conocimientos teóricos en un entorno práctico. Ha sido un aprendizaje continuo que me ha preparado mejor para futuros desafíos en el campo del desarrollo de aplicaciones web, especialmente en el campo frontend.

Mirando hacia el futuro, veo numerosas posibilidades de expansión para pharmAI. Desde el desarrollo de una aplicación móvil hasta la internacionalización de la plataforma. pharmAI puede seguir creciendo y adaptándose para satisfacer las necesidades de los usuarios, produciendo un impacto positivo en la comunidad sanitaria.

En resumen, pharmAI es más que un proyecto académico; es una herramienta innovadora con el potencial de mejorar la manera en que las personas acceden y gestionan la información sobre su salud y los medicamentos que toman día a día.

Para finalizar, estos son los puntos clave que hacen diferente este trabajo:

1. **Innovación y Conexión de Sectores:** pharmAI combina inteligencia artificial con el sector farmacéutico para modernizar el acceso a la información médica.
2. **Filosofía del Proyecto:** Facilitar al usuario el acceso a datos relevantes sobre medicamentos, integrando la IA de forma intuitiva con pharmaBot.
3. **Objetivos del Proyecto:** Crear una plataforma dinámica y accesible, utilizando tecnologías modernas y enfocada en la experiencia del usuario.
4. **Beneficios Sociales:** Asesoramiento rápido y preciso, mejor accesibilidad a la información médica y educación sobre el uso de medicamentos.
5. **Motivaciones Personales:** Unir mi experiencia en farmacia con el desarrollo de aplicaciones web para superar la resistencia a las nuevas tecnologías en el sector.
6. **Viabilidad del Proyecto:** Aumento de la adopción de la IA, preferencia por experiencias personalizadas y optimización para todo tipo de dispositivos.
7. **Elección de Tecnologías:** Uso de Astro para el frontend, Ollama para la IA local, Astro DB para la gestión de usuarios, y APIs para diversas funcionalidades.
8. **Interfaz de Usuario:** Diseño minimalista, interactivo y llamativo para facilitar la navegación y mejorar la experiencia del usuario.

## 9. BIBLIOGRAFÍA

- Astro Build. (n.d.). Configuring Astro. Recuperado de <https://docs.astro.build/es/guides/configuring-astro/>
- Astro Build. (n.d.). Database. Recuperado de <https://astro.build/db/>
- ChatGPT. (n.d.). Recuperado de <https://chatgpt.com/>
- CIMA AEMPS. (n.d.). Recuperado de <https://cima.aemps.es/cima/publico/home.html>
- Code Visual Studio. (n.d.). Recuperado de <https://code.visualstudio.com/>
- Docker. (n.d.). Recuperado de <https://www.docker.com/>
- Drizzle ORM. (n.d.). Recuperado de <https://orm.drizzle.team/>
- Figma. (n.d.). Recuperado de <https://www.figma.com/>
- Geoapify. (n.d.). Recuperado de <https://www.geoapify.com/>
- Git SCM. (n.d.). Recuperado de <https://www.git-scm.com/>
- Leaflet JS. (n.d.). Recuperado de <https://leafletjs.com/>
- Lucia Auth. (n.d.). Username and password tutorial. Recuperado de <https://lucia-auth.com/tutorials/username-and-password/astro>
- Miro. (n.d.). Recuperado de <https://miro.com/app>
- Ollama. (n.d.). Recuperado de <https://ollama.com/>
- Ollama. (n.d.). Download. Recuperado de <https://ollama.com/download>
- Ollama. (n.d.). Llama 3. Recuperado de <https://ollama.com/library/llama3>
- Rabbit Tech. (n.d.). Recuperado de <https://www.rabbit.tech/>
- Tailwind CSS. (n.d.). Recuperado de <https://tailwindcss.com/>
- Turso Database. (n.d.). Recuperado de <https://github.com/tursodatabase/libsql>
- Vercel. (n.d.). Recuperado de <https://vercel.com/>
- Vue.js. (n.d.). Recuperado de <https://vuejs.org/>

## 10. GLOSARIO

- 1. LLM (Large Language Model):** Modelos de lenguaje de gran tamaño utilizados en pharmAI para procesar y generar texto natural, permitiendo interacciones avanzadas y personalizadas a través de pharmaBot.
- 2. Inteligencia Artificial (IA):** Tecnología central en pharmAI que permite automatizar tareas complejas, proporcionando asesoramiento y respuestas personalizadas sobre medicamentos y salud.
- 3. NLP (Natural Language Processing):** Tecnología que permite a pharmAI comprender, interpretar y generar lenguaje humano, facilitando la interacción natural entre usuarios y pharmaBot.
- 4. Ollama:** Plataforma utilizada en pharmAI para ejecutar modelos de lenguaje de gran tamaño (LLMs) localmente, optimizando el procesamiento de lenguaje natural en la aplicación.
- 5. Astro:** Framework de desarrollo web basado en JavaScript utilizado en pharmAI para construir una interfaz web moderna, rápida y eficiente.
- 6. Endpoints:** Puntos de acceso en una API que permiten la comunicación entre el frontend y el backend de pharmAI, facilitando la interacción y el intercambio de datos.
- 7. API (Application Programming Interface):** Conjunto de reglas y protocolos en pharmAI que permite la comunicación y el intercambio de datos entre diferentes sistemas y componentes.
- 8. CIMA (Centro de Información sobre Medicamentos de la AEMPS):** API integrada en pharmAI que proporciona información detallada y actualizada sobre medicamentos.
- 9. Geoapify:** API utilizada en pharmAI para servicios de geolocalización, permitiendo a los usuarios encontrar farmacias cercanas y visualizarlas en un mapa interactivo.

10. **Flask:** Framework de microservicios en Python utilizado para crear APIs personalizadas en pharmAI, como la generación de frases aleatorias de farmacología.
11. **SQLite:** Sistema de gestión de bases de datos relacional ligero utilizado en pharmAI para almacenar y gestionar datos de usuarios y medicamentos.
12. **Astro DB:** Implementación de LibSQL utilizada en pharmAI para gestionar bases de datos, facilitando la creación y manipulación de datos de manera eficiente.
13. **Drizzle ORM:** Object-Relational Mapper utilizado en pharmAI para interactuar con bases de datos SQL, facilitando las operaciones de bases de datos mediante una API en JavaScript o TypeScript.
14. **Frontend:** Parte de pharmAI que interactúa directamente con los usuarios, desarrollada con tecnologías modernas para ofrecer una experiencia de usuario optimizada.
15. **Tailwind CSS:** Framework de diseño CSS utilizado en pharmAI para crear una interfaz de usuario coherente, estilizada y fácil de mantener.
16. **Lucia Auth:** Sistema de autenticación y gestión de usuarios implementado en pharmAI para manejar el registro, inicio de sesión y gestión de sesiones de usuarios de manera segura.
17. **Vercel:** Plataforma de despliegue y hosting utilizada en proyectos predominantemente frontends para alojar y gestionar aplicaciones web de manera eficiente.
18. **Leaflet:** Librería JavaScript utilizada en pharmAI para crear mapas interactivos que muestran la ubicación de farmacias cercanas.
19. **Modelo Freemium:** Estrategia de negocio considerada para pharmAI donde el uso básico de la plataforma es gratuito, con funciones avanzadas disponibles por una tarifa (pensada a futuro como estrategia de negocio).
20. **Diseño Responsivo:** Enfoque de diseño web en pharmAI que asegura que la interfaz se adapte y funcione correctamente en una amplia variedad de dispositivos y tamaños de pantalla.

## 11. ANEXOS



**Anexo 1:** Funcionamiento de un NLP o en español procesamiento del lenguaje natural (PNL). imagen extraida de  
<https://es.shaip.com/blog/what-is-nlp-how-it-works-benefits-challenges-examples/>

The screenshot shows the official website of the Spanish Agency for Medicines and Medical Devices (CIMA). At the top, there are logos for the Government of Spain, the Ministry of Health, and CIMA. The main heading is "ENCUENTRA TU MEDICAMENTO AQUÍ". Below it is a search bar with placeholder text "Busca por medicamento, principio activo, código nacional o número de registro". There is also a "Buscador para profesionales sanitarios >>" button. To the right, there are links for "QUÉ ES CIMA", "NOMENCLÁTOR", "GLOSARIO", "ESPAÑOL ENGLISH", and "Iniciar sesión". The page features a background image of a doctor's hands holding a stethoscope and a white tablet.

**Key statistics displayed:**

- 15.447 Medicamentos \*
- 2.635 Principios activos \*
- 32.941 Presentaciones \*\*
- 408 Biosimilares \*\*
- 274 Huérfanos \*\*

\* Datos de medicamentos autorizados  
\*\* Datos para presentaciones de medicamentos

**Cookie consent banner:**

Esta web utiliza cookies propias para facilitar la navegación y cookies de terceros para obtener estadísticas de uso y satisfacción. Puede obtener más información en el apartado "Cookies" de nuestro aviso legal.

**ÚLTIMAS ACTUALIZACIONES**

CARDYL 40 MG COMPRIMIDOS RECUBIERTOS CON PELÍCULA Viatris Up Ver más Fecha actualización: 31/01/2024 Cambios: Otros

CARDYL 80 MG COMPRIMIDOS RECUBIERTOS CON

**PROBLEMAS DE SUMINISTRO**

AZELASTINA ABAMED 0.5 MG/ML COLIRIO EN SOLUCIÓN, 1 frasco de 6 ml Código nacional: 721774 Ver más Fecha Inicio: 07/06/2024

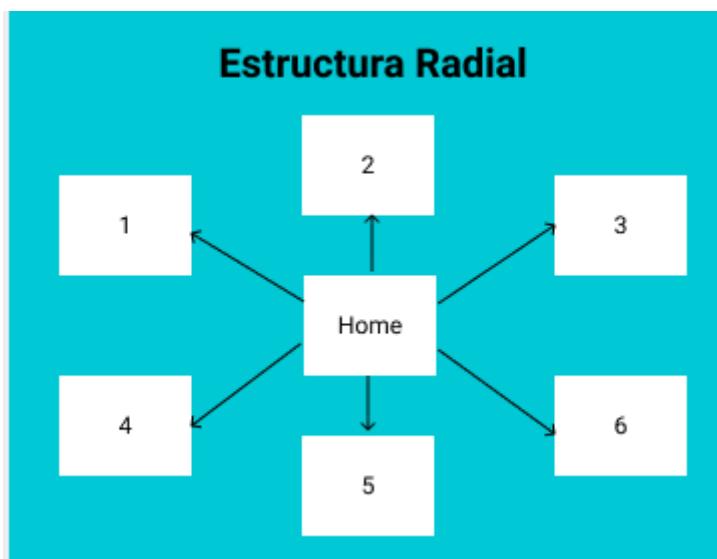
CIDOFOPUR ZENTIVA 75 mg/ml CONCENTRADO PARA SOLUCIÓN PARA INFUSIÓN EFG, 1 vial de 5 ml Código nacional: 712913 Fecha Inicio: 07/09/2024

**NOTAS DE SEGURIDAD**

Recomendaciones sobre el uso de valproato en varones para evitar el problema de desarrollo del neuodevelopimiento en sus hijos tras la exposición materna Nota Informativa MuH (FV), 02/2024 Ver más Publicado en: 15/01/2024

Pseudoefedrina: medidas para minimizar el riesgo de síndrome de encelofagia posterior reversible (PRES) y de

**Anexo 2:** CIMA, de la agencia española de medicamentos y productos sanitarios.  
Imagen extraída de <https://cima.aemps.es/cima/publico/home.html>



**Anexo 3:** Estructura radial en el desarrollo web, la estructura seguida por pharmAI para resaltar su landing page. Imagen extraída de  
<https://javiercatalandlapuente.wordpress.com/2021/07/04/tipos-de-navegacion-web/>