UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo e implementación de una plataforma de asesoría financiera basada en inteligencia artificial

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Christian Alejandro Leyva Mercado

ASESOR DE INFORME

Dr. Ismael Everardo Bárcenas Patiño

Ciudad Universitaria, Cd. Mx., 2025

A mis padres, les agradezco profundamente por brindarme no solo la oportunidad de alcanzar este logro, sino también por enseñarme el valor del trabajo duro y la perseverancia. Su incansable esfuerzo y constante sacrificio me han permitido llegar hasta este momento tan significativo. Cada paso que he dado ha sido posible gracias a su guía.

A mi hermana, me enseñaste una nueva forma de amor y responsabilidad. Me hiciste crecer como persona y demostraste que, incluso en los momentos más ocupados, siempre hay tiempo para una sonrisa y un abrazo.

A mi hermano, gracias por ayudarme a ser más empático y mostrarme que el apoyo mutuo trasciende cualquier desacuerdo.

A mis amigos, que fueron mi refugio en los momentos difíciles y mi impulso cuando sentí que no podía más. Gracias por su compañía, sus palabras de aliento y por recordarme siempre que no estaba solo en este viaje.

Este logro también es de ustedes.

Índice general

Introducción vi

1. Historia laboral 1
1.1. BRITA Inteligencia Artificial . . . 1
1.1.1. Historia . . . 1
1.1.2. Filosofía empresarial . . . 2
1.1.3. Organigrama . . . 2
1.2. Historia Laboral en BRITA Inteligencia Artificial . . . 3
1.2.1. Desarrollador Fullstack . . . 4
1.2.2. Líder de desarrollo . . . 5

2. Desarrollo e implementación 7
2.1. Análisis del proyecto . . . 7
2.1.1. Objetivo . . . 7
2.1.2. Visión . . . 7
2.1.3. Antecedentes . . . 8
2.1.4. Análisis de Mercado . . . 8
2.1.5. Definición de Requerimientos . . . 16
2.2. Diseño de la aplicación . . . 18
2.2.1. Diseño de la arquitectura de la aplicación . . . 19
2.2.2. Diseño de la experiencia e interfaz de usuario (UX/UI) de la aplicación . . . 24
2.2.3. Diseño final de la aplicación . . . 26
2.3. Desarrollo de la aplicación . . . 32
2.3.1. Metodología utilizada . . . 32
2.3.2. Desarrollo del Frontend de la aplicación . . . 34
2.3.3. Conexión con Google Firebase . . . 41
2.3.4. Conexión con backend de Inteligencia Artificial . . . 42
2.4. Etapa de pruebas . . . 43
2.5. Despliegue de la aplicación . . . 44

3. Resultados 46

4. Conclusiones 48

Conclusión **50**

Glosario **54**

Bibliografía **55**

Índice de figuras

1.1. Organigrama de BRITA Inteligencia Artificial 2

2.1. Captura del sitio web de Destácame 9
2.2. Sitio web de Zenfi 11
2.3. Sitio web de iKiwi 13
2.4. Capturas de pantalla del funcionamiento de Qiip 14
2.5. Página web de Lounn 15
2.6. Diagrama de la arquitectura de la plataforma 23
2.7. Interfaz del CMS Sanity con proyecto ejemplo. Fuente: Sanity, 2025 . 38
2.8. Diagrama de estrategia de manejo de versiones con Git 40
2.9. Resultados finales de las pruebas de Lighthouse en la aplicación. . . . 44

Introducción

En el dinámico mundo de las finanzas personales y empresariales, la necesidad de asesoramiento financiero preciso y accesible ha crecido exponencialmente. Este trabajo presenta el desarrollo e implementación de una innovadora plataforma de asesoría financiera basada en Inteligencia Artificial (IA, por sus siglas), diseñada para revolucionar la forma en que individuos y organizaciones abordan sus decisiones financieras.

El asesoramiento financiero juega un papel crucial en la orientación de individuos a través de las complejidades inherentes al sistema financiero moderno. Esta importancia se magnifica en el contexto mexicano, como lo demuestra la alarmante estadística proporcionada por GBM: "nuestro país está entre los países con peor calificación en alfabetización financiera dentro del G20, y menos del 3 % de la población invierte" (GBM, 2024). Estos datos subrayan la urgente necesidad de asesores financieros calificados en México, quienes pueden desempeñar un papel fundamental en la educación financiera y en el fomento de una cultura de inversión más robusta entre la población.

Hablando sobre la inteligencia artificial, este es un concepto complejo de definir debido a la diversidad de áreas que abarca. Sin embargo, una definición concisa sería "la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano" (Rouhiainen, 2018).

Entre los servicios ofrecidos por la plataforma de asesoría financiera desarrollada en este trabajo, destaca la búsqueda personalizada de Productos Financieros con ayuda de Inteligencia Artificial. Este servicio está diseñado para atender las necesidades tanto de personas individuales como de empresas, proporcionando una selección cuidadosa de las mejores opciones de productos financieros adaptados a cada situación particular.

Para individuos, la plataforma analiza factores como ingresos, gastos, metas financieras a corto y largo plazo, y tolerancia al riesgo, dependiendo del producto que le interese al cliente. Con base en estos datos, ofrece recomendaciones sobre productos como cuentas de ahorro, inversiones, seguros, y opciones de crédito que mejor se ajusten al perfil financiero del usuario.

En el caso de las empresas, el sistema considera aspectos como el tamaño de la organización, sector de actividad, flujo de caja, proyecciones de crecimiento y necesidades de financiamiento. A partir de este análisis, proporciona sugerencias sobre productos

financieros corporativos, como líneas de crédito, instrumentos de inversión, seguros comerciales y soluciones de gestión de tesorería.

La plataforma de asesoría financiera desarrollada no solo identifica los productos, sino que también realiza comparaciones detalladas entre diferentes opciones, considerando tasas de interés, comisiones, beneficios adicionales y términos contractuales. Esto permite a los usuarios tomar decisiones financieras fundamentadas, gracias a un análisis comparativo y objetivo de las alternativas disponibles en el mercado.

Además, una vez que el usuario ha seleccionado el producto financiero que mejor se adapta a sus necesidades, la plataforma ofrece un servicio adicional de acompañamiento personalizado. Un experto financiero es asignado para asesorar y guiar al cliente a través de todo el proceso, desde la selección inicial hasta la finalización del trámite. Este profesional se encarga de contactar a la entidad financiera y realizar todas las gestiones necesarias en nombre del cliente, asegurando una experiencia sin complicaciones y maximizando los beneficios del producto elegido.

Este informe presenta el desarrollo de la plataforma de asesoría financiera, la cual está diseñada para democratizar el acceso a asesoramiento financiero de calidad, combinando el potencial de la inteligencia artificial con una interfaz intuitiva, accesible y eficiente para el usuario. Se explora el proceso completo de desarrollo del proyecto, desde su conceptualización hasta su implementación, con especial énfasis en el desarrollo de la aplicación web. Asimismo, se abordan los desafíos encontrados, las soluciones aplicadas y el impacto potencial de esta tecnología en el sector financiero. Finalmente, se analizan las implicaciones futuras y las oportunidades de expansión y mejora continua del sistema.

Capítulo 1

Historia laboral

La empresa donde trabajé durante seis meses y realicé el proyecto que describo en este informe es BRITA Inteligencia Artificial (BRITA). BRITA es una consultora de tecnologías de la información líder en el sector de la Inteligencia Artificial en México. Durante mi tiempo en BRITA, participé en diversos proyectos de software y desempeñé diferentes roles laborales.

1.1. BRITA Inteligencia Artificial

1.1.1. Historia

BRITA Inteligencia Artificial, fundada en 2019, es una empresa mexicana de consultoría de software especializada en el campo de la IA. Desde sus inicios, la compañía se ha enfocado en desarrollar soluciones innovadoras, con un énfasis particular en la Inteligencia Artificial Generativa.

La empresa ofrece una amplia gama de servicios que van desde el desarrollo de software inteligente hasta la automatización de procesos empresariales. Estas soluciones están diseñadas para optimizar tiempos y mejorar la toma de decisiones en diversos sectores industriales.

Entre las principales áreas de especialización de BRITA se encuentran el Aprendizaje Automático, el Aprendizaje Profundo, el Procesamiento de Lenguaje Natural (PNL), la Visión Computacional y el desarrollo de *Chatbots*. Estas tecnologías permiten a las empresas mejorar significativamente la experiencia del cliente, automatizar tareas complejas y extraer información valiosa de grandes volúmenes de datos, lo que resulta en una optimización de operaciones y la generación de nuevas oportunidades de crecimiento.

En resumen, desde su creación en 2019, BRITA Inteligencia Artificial se ha consolidado como una empresa líder en el sector de la IA en México, combinando innovación

tecnológica con un fuerte sentido de responsabilidad ética. «Desarrollo de Soluciones con Inteligencia Artificial México», 2023

1.1.2. Filosofía empresarial

Los principales valores de la empresa BRITA Inteligencia artificial son la **ética y la responsabilidad**.

Esto se debe a que, además de su compromiso con la innovación, la empresa ha establecido colaboraciones con entidades nacionales e internacionales, posicionándose como un defensor activo de la regulación adecuada de tecnologías emergentes. Un ejemplo de esto es su trabajo en torno a los *deepfakes*, donde BRITA aboga por medidas que prevengan el uso malintencionado de esta tecnología.

Un caso destacado fue la advertencia que realizó la empresa en 2023, cuando se aproximaban las elecciones presidenciales de México. BRITA alertó que ***"abundarían Deepfakes políticos"***, abogando por la creación de regulaciones tanto a nivel nacional como internacional. Esta advertencia fue crucial, ya que la sociedad mexicana era vulnerable a la proliferación de **contenidos políticos falsos** creados con herramientas sencillas de inteligencia artificial. Estos deepfakes podrían suplantar la identidad de los candidatos políticos, representando una amenaza significativa (Ricardo Quiroga, 2023).

Asimismo, la empresa colabora estrechamente con sus clientes para asegurar que la implementación de sus soluciones tecnológicas se realice de forma equitativa y responsable, promoviendo así un desarrollo sostenible en la sociedad.

1.1.3. Organigrama

El organigrama de la organización, hacia mi equipo de trabajo era el siguiente:

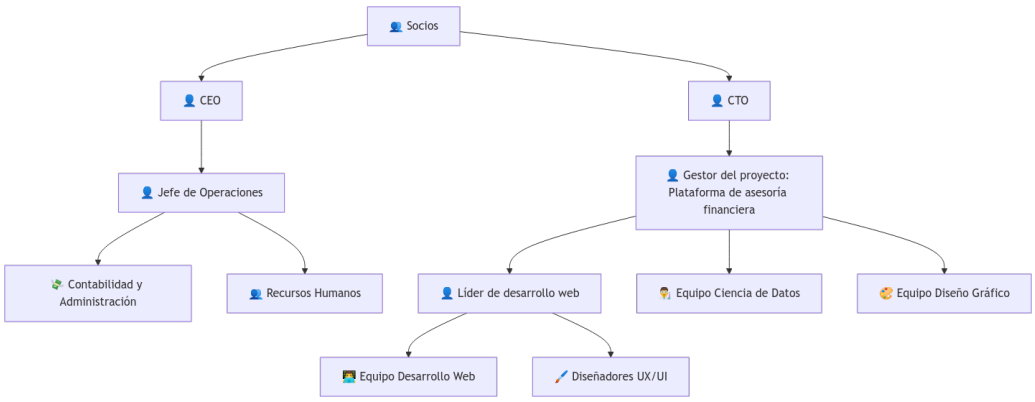

Figura 1.1: Organigrama de BRITA Inteligencia Artificial

La estructura organizacional de BRITA se compone de los siguientes roles clave:

- **Socios:** Son los inversionistas de la empresa que supervisan el rendimiento general y participan en la toma de decisiones estratégicas.
- **CEO (*Chief Executive Officer*):** Es el responsable de liderar la organización, tomar decisiones estratégicas y establecer objetivos corporativos para asegurar el crecimiento de la empresa (UNIR La Universidad en Internet, s.f.).
- **CTO (*Chief Technology Officer*):** Lidera la dirección tecnológica de la empresa. Es responsable de supervisar el desarrollo e implementación de soluciones tecnológicas, evaluar nuevas tecnologías y asegurar la excelencia técnica en todos los proyectos (de Mendieta, 2022).
- **Jefe de Operaciones:** Supervisa las operaciones diarias de la empresa, gestiona los recursos y coordina entre los diferentes departamentos. Entre los equipos que supervisa se encuentra el de contabilidad y administración, además del equipo de recursos humanos.
- **Gerente de Proyecto:** Coordina y gestiona proyectos de software, liderando equipos multidisciplinarios como desarrollo web, Ciencia de datos o como fue el caso de nuestro proyecto, Diseño Gráfico. Es responsable de la planificación, ejecución y entrega exitosa de proyectos, manteniendo la comunicación con clientes y asegurando que se cumplan los objetivos establecidos.

Cabe aclarar que expongo en este organigrama los roles de los cuales yo tenía conocimiento, por lo que puede haber otros roles en la empresa que no se mencionan en este informe.

1.2. Historia Laboral en BRITA Inteligencia Artificial

Formé parte de la división de proyectos de BRITA, donde inicialmente fui contratado como desarrollador Fullstack. En este rol, llevé a cabo un proyecto de desarrollo web para un cliente de la consultora. Posteriormente, ascendí a desarrollador líder para un proyecto de *software*. En esta posición, mis responsabilidades se ampliaron: además de desarrollar el proyecto con mi equipo de trabajo, me encargaba de mantener reuniones con el cliente y tomar decisiones cruciales para el proyecto, como la elección del Stack Tecnológico, la metodología de trabajo y la implementación de cambios solicitados por el cliente.

Más adelante, tras recibir una retroalimentación positiva por mi desempeño como líder en el proyecto anterior, asumí la dirección del desarrollo de la plataforma que describo en este informe.

1.2.1. Desarrollador Fullstack

Un Desarrollador *Fullstack* es un experto tecnológico capaz de gestionar tanto el desarrollo *frontend* (interfaz de usuario) como el *backend* (lógica y base de datos) de aplicaciones o sitios web. Esta versatilidad en habilidades les otorga una visión holística del desarrollo web y de software, permitiéndoles crear soluciones integrales y coherentes (Hireline, 2024).

En mi rol como Desarrollador *Fullstack* en BRITA, fui responsable de la creación completa de un proyecto web para un cliente de la consultora. Mis tareas abarcaron desde el diseño de la interfaz de usuario hasta la implementación de la lógica de negocio y la gestión de bases de datos. Esta experiencia me permitió aplicar mis conocimientos en diversas tecnologías y *Frameworks*, tanto en el lado del cliente como en el servidor. El *stack* tecnológico principal que utilicé durante este tiempo fue *Next.js* para el *frontend* y *Flask* para el *backend* con *Python*, lo que me permitió desarrollar aplicaciones web de alto rendimiento y escalables.

Mantuve este puesto durante un mes, donde logré demostrar mi capacidad para manejar múltiples aspectos del desarrollo de *software*, desde la conceptualización inicial hasta la implementación final. Además, esta posición me brindó la oportunidad de colaborar estrechamente con otros equipos y *Stakeholders*, mejorando mis habilidades de comunicación y trabajo en equipo. La experiencia con *Next.js* y *Flask* también me permitió optimizar el proceso de desarrollo y mejorar la eficiencia del proyecto, aprovechando las fortalezas de ambas tecnologías para crear soluciones robustas y flexibles.

Las principales actividades que realicé en este rol fueron:

- Desarrollo de interfaces gráficas de usuario responsivas y atractivas utilizando *HTML*, *CSS* y *JavaScript*.
- Implementación de lógica del lado del servidor utilizando el backend serverless de Next.js y el backend robusto con Flask.
- Diseño y gestión de bases de datos, tanto *SQL* como *NoSQL*.
- Integración de *APIs* y servicios de terceros.
- Implementación de autenticación y autorización de usuarios.
- Optimización del rendimiento de aplicaciones web.
- Realización de pruebas unitarias y de integración.
- Despliegue y mantenimiento de aplicaciones en entornos de producción.
- Colaboración con diseñadores UX/UI para implementar diseños y funcionalidades.
- Participación en revisiones de código y sesiones de *Pair Programming*.

1.2.2. Líder de desarrollo

Un Líder de Desarrollo es un profesional de *software* que combina habilidades técnicas avanzadas con capacidades de liderazgo y gestión de proyectos. Este rol implica además de escribir código, guiar a un equipo de desarrolladores, tomar decisiones técnicas cruciales y actuar como puente entre el equipo técnico y los *stakeholders* del proyecto (Hack a Boss, 2024).

Como mencioné anteriormente, asumí el rol un mes después de estar como desarrollador *fullstack* en dos proyectos de BRITA. Esta era la primera vez que realizaba las tareas de líder en un rol profesional, por lo que representó un reto para mí. Sin embargo, tomé el cargo con gran entusiasmo y deseo de mejorar y aprender nuevas habilidades "blandas", ya que mi rol siempre había sido técnico.

Cabe aclarar que en este proyecto, mi liderazgo fue en el desarrollo de la aplicación web, es decir, principalmente el *frontend* de la aplicación. Ya que al ser un proyecto amplio, existían 3 equipos colaborando continuamente, los cuales son los siguientes:

- **Equipo de desarrollo de aplicación web:** Como mencioné, este fue el equipo que lideré. Nos encargamos de crear todo el código necesario para el correcto funcionamiento de la aplicación web, conectarla con los servicios internos y externos, y realizar el despliegue y publicación del sitio web.
- **Equipo de desarrollo de identidad:** Este equipo contaba con expertos multidisciplinarios que crearon la identidad de la aplicación. Entre sus actividades se incluían la creación del nombre comercial, el diseño del logotipo, la selección de los colores de la marca, la elaboración de presentaciones y la realización de sesiones de lluvia de ideas, entre otras.
- **Equipo de desarrollo de Inteligencia Artificial:** Este equipo se encargó de desarrollar los modelos de Inteligencia Artificial y la lógica necesaria para su funcionamiento. Además, fue responsable de crear una *API* para conectar estos flujos con la aplicación web.

En mi rol como Líder de Desarrollo en BRITA, asumí responsabilidades adicionales que iban más allá del desarrollo de *software*. Mis principales funciones incluían:

- Liderar el equipo de desarrollo, asignando tareas y supervisando el progreso del proyecto.
- Tomar decisiones técnicas clave, como la elección del *stack* tecnológico y la arquitectura del *software*.
- Mantener reuniones regulares con el cliente para discutir requisitos, avances y cambios en el proyecto.
- Participar activamente en el desarrollo del código del proyecto.
- Implementar y supervisar la metodología ágil *Scrum*, incluyendo:

- Facilitar las ceremonias de Scrum: Planeaciones de *Sprint*, Reuniones Diarias (*Daily meetings*), Revisiones de *Sprint* y Retrospectivas de *Sprint*.
- Asegurar la transparencia del progreso mediante el uso de un Backlog en Notion.
- Mantener y priorizar el *Product Backlog* en colaboración con el *Product Owner*.
- Fomentar la auto-organización del equipo y la mejora continua.
- Realizar revisiones de código y asegurar la calidad del *software* producido por el equipo.
- Mentorizar a otros desarrolladores, fomentando su crecimiento profesional y técnico.
- Gestionar los recursos del proyecto, incluyendo la planificación del tiempo y la asignación de tareas.
- Colaborar con otros departamentos, como diseño y control de calidad, para asegurar una integración fluida de todos los aspectos del proyecto.
- Eliminar impedimentos que pudieran afectar el progreso del equipo durante el *Sprint*.

Este rol me permitió desarrollar habilidades de liderazgo, mejorar mi capacidad de comunicación tanto con el equipo técnico como con los clientes, y profundizar mi comprensión de la gestión de proyectos de *software*.

Mantuve este rol por 5 meses aproximadamente durante el desarrollo del Producto Mínimo Viable (usualmente llamado por su acrónimo en inglés, **MVP**) del proyecto, el *MVP* es la versión más básica de un producto que incluye solo las características esenciales, y se utiliza como una herramienta para evaluar la viabilidad, aplicación y aceptación de un producto en el mercado antes de su desarrollo completo (da Silva, 2021). Además, durante este periodo, tuve la oportunidad de liderar otro proyecto en la consultora y participar en la planeación de nuevos proyectos que llegaban a la consultora de diversos clientes.

Capítulo 2

Desarrollo e implementación de una plataforma de asesoría financiera basada en inteligencia artificial

2.1. Análisis del proyecto

2.1.1. Objetivo

Desarrollar una plataforma financiera innovadora que ofrezca una selección precisa de los mejores productos financieros disponibles en el mercado. Esta plataforma utilizará inteligencia artificial para generar recomendaciones personalizadas basadas en las necesidades del usuario, y proporcionará asesoramiento experto que lo acompañe en el proceso de alcanzar sus metas financieras. El enfoque integral de la plataforma garantiza que el usuario reciba tanto opciones adecuadas a su perfil como orientación especializada que facilite la toma de decisiones financieras fundamentadas y eficaces.

2.1.2. Visión

Ser la plataforma financiera líder que revolucione la interacción de las personas con sus finanzas, destacándonos por nuestra interfaz intuitiva y fácil de usar. Nos comprometemos a empoderar a los usuarios para tomar decisiones informadas y alcanzar sus objetivos financieros con confianza, democratizando el acceso a productos financieros de alta calidad y asesoramiento experto. Utilizando tecnología de vanguardia, proporcionaremos soluciones personalizadas que se adapten a las necesidades únicas de cada usuario, creando una experiencia financiera tan sencilla y accesible que incluso aquellos sin conocimientos previos puedan navegar con éxito en el mundo de las

finanzas personales.

2.1.3. Antecedentes

Nuestro cliente tuvo la idea de desarrollar su propia *Startup* combinando sus conocimientos de finanzas con la informática, para ello se acerco a la consultora BRITA Inteligencia Artificial, donde después de múltiples conversaciones se llego al acuerdo que BRITA desarrollaría el *MVP* del proyecto.

La idea general del cliente era desarrollar una plataforma financiera que utilizara inteligencia artificial para actuar como intermediaria entre entidades financieras y personas físicas o morales. Esta plataforma tendría como objetivo facilitar el acceso a productos financieros adecuados, adaptándose a las necesidades específicas de cada usuario, mediante recomendaciones automatizadas y personalizadas.

Esta plataforma busca diferenciarse de la competencia al actuar como un asesor financiero integral, y no solo como un buscador de productos. Su enfoque principal es proporcionar una interfaz de usuario sencilla y accesible, destacando el uso de inteligencia artificial para generar las mejores opciones de productos financieros adaptadas a cada cliente. La plataforma ofrecerá una amplia gama de servicios, desde cuentas y tarjetas hasta créditos e inversiones para empresas, y préstamos y seguros personales para individuos.

2.1.4. Análisis de Mercado

Uno de los primeros pasos cruciales en nuestro proyecto fue realizar un análisis exhaustivo y revelador del mercado. Este estudio no solo nos permitió identificar a varios actores clave en el sector de servicios financieros digitales, sino que también resultó fundamental para el éxito del proyecto. Un análisis de mercado profundo proporciona información vital sobre las tendencias actuales, las necesidades de los clientes y las oportunidades de negocio, lo que permite tomar decisiones estratégicas informadas y minimizar riesgos.

Al llevar a cabo este análisis, obtuvimos una visión clara y actualizada del panorama competitivo en el que nos desenvolvemos. Esta comprensión del mercado es esencial para desarrollar una propuesta de valor única, identificar nichos de mercado desatendidos y anticipar posibles obstáculos. Además, nos ayuda a ajustar nuestro modelo de negocio, optimizar nuestras estrategias de *marketing* y posicionar eficazmente nuestro producto en el mercado.

Competencia Directa

Se identifico a “Destácame” como el competidor más directo, ofreciendo servicios similares a los que nuestro proyecto pretende brindar.

Destácame es una plataforma financiera digital que opera en México, ofreciendo servicios innovadores para mejorar la salud financiera de sus usuarios. Esta plataforma esta enfocada en un negocio de Empresa a Consumidor (también llamado por sus siglas en inglés **B2C**) ofreciendo herramientas que ayuden a las personas físicas a encontrar las mejores opciones de productos financieros y a la educación financiera de los usuarios (Destácame, 2024a).

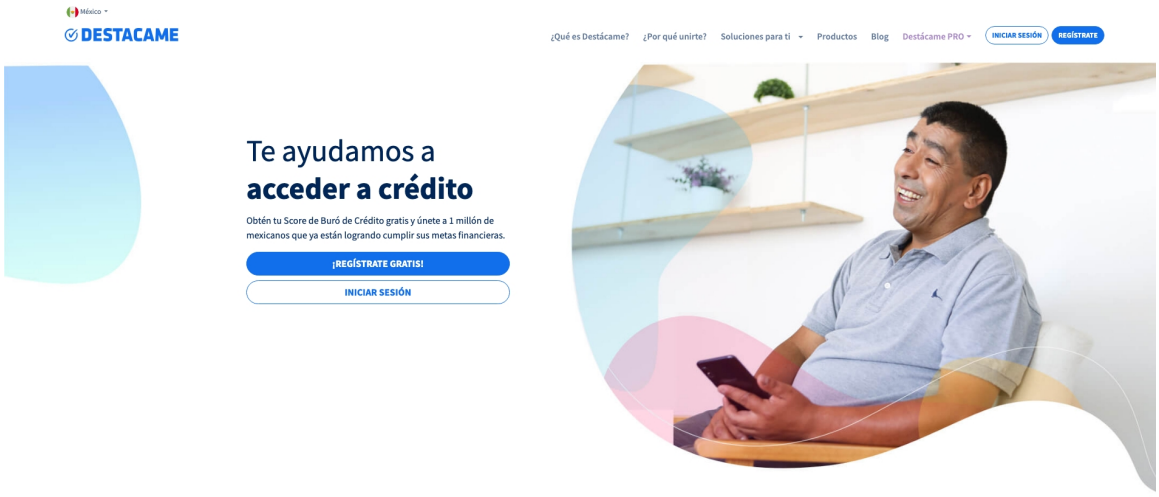

Figura 2.1: Captura del sitio web de Destácame

Sus principales características incluyen:

- **Reporte de buró de crédito gratuito**: Ofrece a los usuarios acceso a su reporte de buró crédito sin costo.
- **Recomendación de productos financieros:** A partir del reporte de buró de crédito, la plataforma recomienda al cliente productos financieros. Los productos que la plataforma ofrece son los siguientes:
  - Tarjetas de crédito
  - Créditos
  - Seguros
  - Ahorro e inversión
  - Tarjetas de débito
- **Educación financiera:** La plataforma ofrece "consejos y herramientas" que permiten al usuario entender y manejar mejor sus finanzas.
- **Liquidación de deudas:** La plataforma ofrece opciones con diferentes entidades bancarias para liquidar las deudas del usuario.

Según lo mencionado en su sitio web (Destácame, 2024b), la plataforma ofrece 2 planes de uso, uno gratuito con el cual el usuario puede acceder a los servicios antes mencionados, donde la plataforma monetiza mediante las comisiones que genera con las entidades bancarias, y una suscripción "Pro" de paga, la cual permite a sus clientes además de acceder a los servicios gratuitos de acceder a los siguientes servicios y/o beneficios:

- Descuentos en deudas
- Revisión de deudas en Buró de crédito
- Recibe tu Score y alertas de atrasos
- Obtén hasta 40 % de descuento
- Asesoría personalizada sobre tus finanzas

Puntos fuertes:

- Reporte gratuito de buró de crédito para el cliente
- Consejos y herramientas de educación financiera
- Variedad de productos financieros ofrecidos
- Interfaz amigable
- Acompañamiento de ejecutivos
- No cobra comisión al usuario por contratación de productos financieros

Puntos débiles:

- No ofrece servicios para personas morales (empresas)
- Solo disponible de forma web
- Requiere de una suscripción para acceder a todos los servicios
- Acompañamiento por ejecutivos y no por asesores financieros
- No cuenta con algoritmos avanzados de búsqueda

Competencia Relevante

Además, se han encontrado varios competidores relevantes como Zenfi, iKiwi México, Lounn, qiip. Cada uno de estos ofrece servicios financieros digitales, aunque con enfoques ligeramente diferentes. Esta diversidad en el mercado sugiere que, si bien existen múltiples actores, aún no hay una solución que abarque de manera integral todas las necesidades del usuario promedio. Esta situación representa una oportunidad para innovar mediante el uso de inteligencia artificial en la recomendación de productos

financieros, así como para ofrecer asesoría personalizada brindada por especialistas. También existe espacio para especializarse en nichos poco atendidos, como el acompañamiento en los procesos de contratación o la integración de múltiples productos financieros dentro de una sola plataforma.

A continuación, se presenta un análisis detallado de los competidores relevantes mencionados:

Zenfi *Zenfi* es una plataforma financiera digital mexicana que se enfoca en ayudar a los usuarios a mejorar su salud financiera y acceder a productos financieros personalizados. La plataforma ofrece una variedad de servicios y herramientas diseñadas para empoderar a los usuarios en la toma de decisiones financieras informadas. Su modelo de negocio se basa principalmente en comisiones por recomendación de productos financieros, lo que permite ofrecer muchos de sus servicios de forma gratuita a los usuarios (Zenfi, 2024).

Figura 2.2: Sitio web de Zenfi

Entre los servicios que ofrece están:

- **Reporte de buró de crédito gratuito:** Zenfi permite a los usuarios acceder a su reporte de crédito sin costo, ayudándoles a entender su situación crediticia actual.
- **Asesoramiento financiero con Inteligencia Artificial:** Zenfi incluye un bot de inteligencia artificial con el que puedes conversar para realizarle preguntas distintos temas, incluyendo tu salud financiera.
- **Conoce tu situación fiscal:** *Zenfi* te permite obtener constancias de situación fiscal, declaraciones, facturas y otros documentos que normalmente encontrarías

en la página del Servicio de Administración Tributaria (SAT), lo cual simplifica el acceso de estos documentos a sus usuarios.

- **Recomendación de productos:** La plataforma permite comparar diferentes productos financieros, incluyendo tarjetas de crédito, préstamos personales y seguros, la plataforma monetiza con esta recomendación ya que obtiene una comisión de las entidades bancarias por cada contratación.

Puntos fuertes:

- Interfaz de usuario intuitiva y fácil de usar.
- Disponible en dispositivos móviles iOS y Android.
- Enfoque en la educación financiera.
- Recomendaciones personalizadas basadas en el perfil financiero del usuario.
- Acceso gratuito al reporte de crédito.
- Amplia gama de productos financieros disponibles.
- Asesoramiento gratuito con IA.

Áreas de mejora:

- No tiene aplicación web.
- No cuenta con opciones de inversión y ahorro.
- No cuenta con la posibilidad de obtener un asesoramiento con una persona real.
- No ofrece servicio para empresas

iKiwi México *iKiwi* México se presenta como "Tu vitamina financiera de cada día" (iKiwi, 2024), ofreciendo una plataforma de servicios financieros digitales, la plataforma esta enfocada a personas físicas en busca de mejorar su salud financiera utilizando sus herramientas.

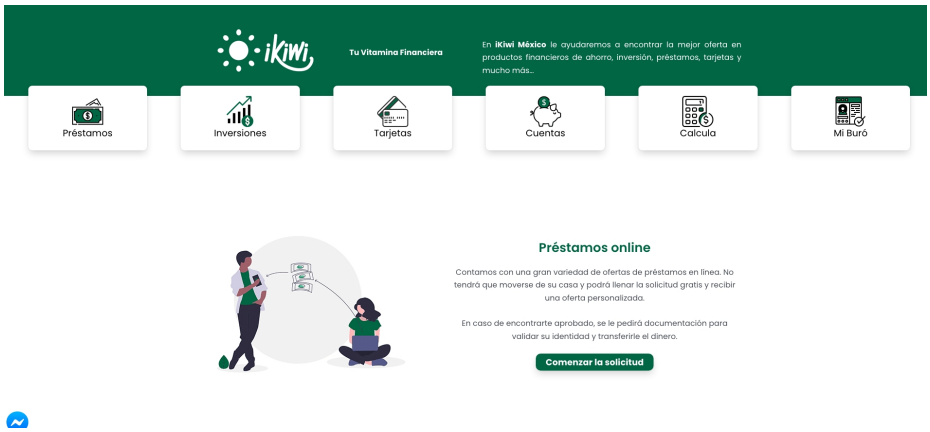

Figura 2.3: Sitio web de iKiwi

De igual forma ofrece algunos productos financieros como créditos, tarjetas de crédito y débito, seguros y servicios de ahorro e inversión. Estos productos no son financiados por *iKiwi*, si no que funciona como intermediario entre las entidades bancarias.

Puntos fuertes:

- Herramientas como calculadoras que ayudan al usuario a realizar sus finanzas y recurrencia en la pagina web.
- Interfaz fácil de usar.
- Variedad de productos financieros

Áreas de mejora:

- El sitio web tiene publicidad externa (Google Ads)
- No ofrece asesoramiento financiero de ningún tipo.
- No ofrecen conocimiento financiero mediante Blogs o paginas informativas.
- Pocas alianzas con entidades bancarias.
- El sitio web es demasiado simple.

Qiip *Qiip* es una plataforma digital de coaching financiero lanzada en 2020 con el objetivo de ayudar a los usuarios a gestionar sus finanzas personales, ofreciendo servicios como la consulta del historial crediticio y sesiones de asesoramiento financiero (Qiip, 2023).

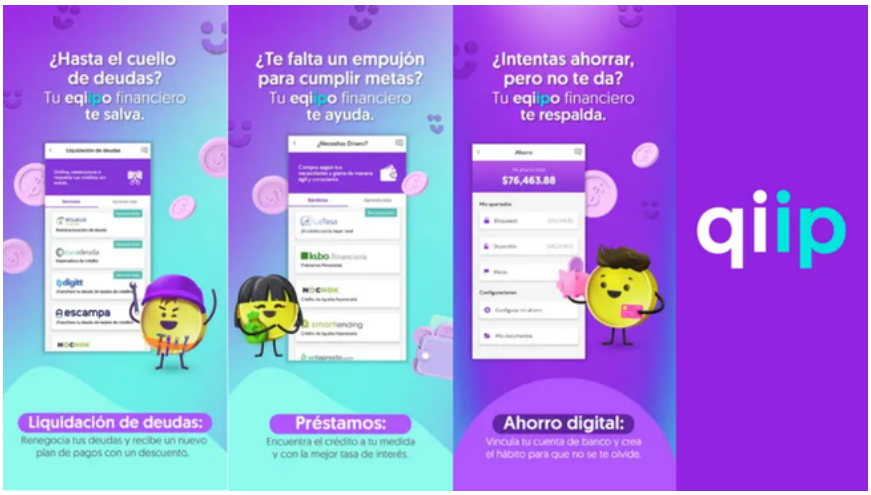

Figura 2.4: Capturas de pantalla del funcionamiento de Qiip

La plataforma esta disponible en México y Colombia, y tiene más de 4.5 millones de usuarios. También *Qiip* ofrece una suscripción de pago llamada "Qiip plus" la cual ofrece mejorar tu historial crediticio mediante consultas y asesoramiento.

Puntos fuertes:

- Aplicación disponible para dispositivos móviles iOS y Android.
- Interfaz fácil de usar.
- Blogs y artículos de educación financiera disponible

Áreas de mejora:

- El asesoramiento o coaching tiene un costo adicional.
- Pocas alianzas con entidades bancarias.
- Poca variedad de productos financieros.

Lounn Lounn es una plataforma financiera digital que ofrece servicios de comparación y contratación de productos financieros, centrándose principalmente en préstamos personales y tarjetas de crédito. Su modelo de negocio está enfocado en *B2B*, lo que significa que ofrece sus servicios a otras empresas, actuando como intermediario entre estas y las entidades bancarias (Lounn, 2022).

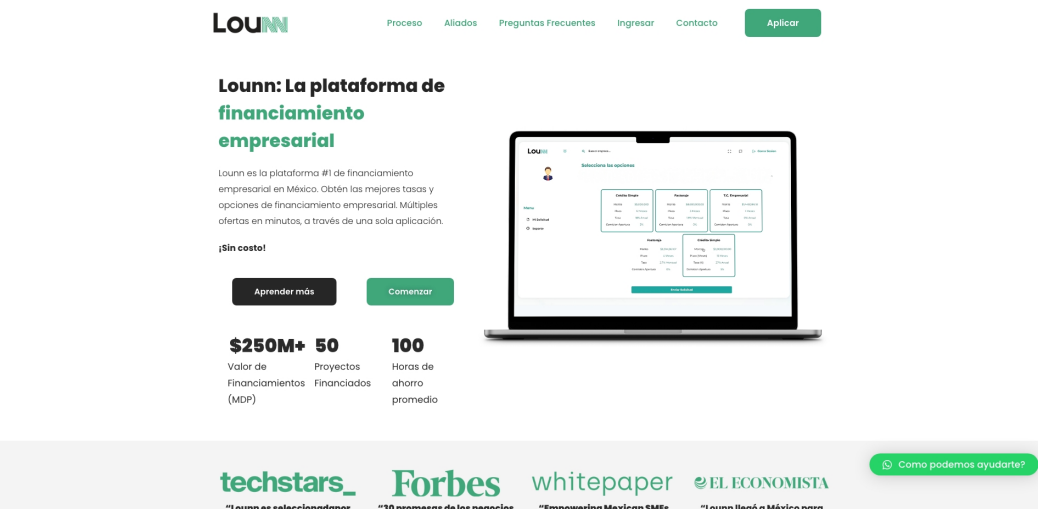

Figura 2.5: Página web de Lounn

Puntos fuertes:

- Variedad de productos para empresas, siendo una de las pocas plataformas que ofrecen servicios a este sector.
- Variedad de alianzas financieras, más de 30 establecidas.

Áreas de mejora:

- No ofrecen ningún tipo de asesoramiento.
- Sitio web y aplicación muy simples, lo cual genera una mala experiencia de usuario.
- Registro muy confuso y que requiere de un registro o contacto manual.

Competencia en segmentos específicos

Se identificaron también competidores que se especializan en segmentos específicos del mercado financiero:

- Alprestamo es un marketplace de productos financieros que conecta a personas que buscan algún préstamo personal. La página simplemente cuenta con un formulario que el usuario debe responder para acceder al listado de productos disponibles, Alprestamo es un intermediario o "Marketplace" entre clientes y empresas o entidades financieras (Al Prestamo, 2024).
- Túdecide era un comparador de créditos y algunos otros productos financieros, sin embargo su sitio web ha pasado a estar inhabilitado, por lo que se puede suponer que ya no esta en operación.

A diferencia de los competidores mencionados, la propuesta de esta plataforma no se limita a ser un simple comparador de productos financieros ni a actuar únicamente como un intermediario. Su aporte distintivo radica en la combinación de un sistema inteligente de comparación de productos financieros, basado en inteligencia artificial, con un servicio de asesoría personalizada proporcionada por expertos humanos. Esta integración permite brindar recomendaciones más ajustadas a las necesidades del usuario y un acompañamiento profesional durante el proceso de selección y contratación, lo que representa un valor agregado en comparación con soluciones existentes en el mercado.

2.1.5. Definición de Requerimientos

Una vez que el cliente y la consultora llegaron a un acuerdo sobre el proyecto, tomando como base las ideas expuestas en los antecedentes y los resultados de la exhaustiva investigación de mercado, se procede a realizar un paso **crucial** en el proceso de desarrollo: la definición de los requerimientos funcionales y no funcionales.

Esta etapa es fundamental debido a que establece los cimientos sobre los cuales se construirá toda la estructura del proyecto, determinando no solo las capacidades y características que el sistema debe poseer, sino también los estándares de calidad, rendimiento y seguridad que deberá cumplir.

Es crucial que los requerimientos sean claros, concisos y verificables. Estos servirán como nuestra guía detallada para ejecutar el proyecto y confirmar que hemos cumplido con todos los aspectos acordados. Además, deben alinearse con los objetivos del negocio y las necesidades de los usuarios finales.

Para definir los requerimientos de la aplicación, se organizaron varias reuniones con el equipo de trabajo completo (incluyendo desarrolladores, diseñadores, y hasta algunos directivos como el *CEO* o el líder del núcleo de proyectos estuvieron presentes en algunas reuniones) con el fin de identificar los puntos más importantes de las ideas del cliente (planteadas en los antecedentes). A este proceso se le dedicaron varios días de la primera semana del proyecto, ya que el cliente nos contrató no solo para desarrollar, sino también para "volver la idea una realidad", por lo que **teníamos que definir y ajustar ideas para lograr un desarrollo viable** (aunque también el cliente contaba con su propio equipo con el que definían algunos conceptos del negocio), considerando el tiempo limitado del que disponíamos. Aunque surgieron muchas buenas ideas y funciones durante este proceso, se decidió que algunas se implementarían en un módulo futuro, enfocándonos en desarrollar las más importantes (y vitales) para el *MVP*.

Durante la segunda semana del proyecto, se llevaron a cabo reuniones con el cliente para presentarle nuestra propuesta de requerimientos y obtener retroalimentación. Este proceso conllevó algunos cambios en los requerimientos, principalmente en aquellos relacionados con el modelo de negocio, ya que el cliente estaba desarrollando con

su equipo una propuesta para implementar un modelo basado en suscripciones. Sin embargo, los requisitos técnicos se modificaron muy poco.

Para esta etapa (y otras que más adelante se mencionarán), se utilizó la herramienta *Notion*, una plataforma de productividad todo en uno que funciona como un espacio de trabajo digital colaborativo. Combina características de procesador de texto, base de datos, wiki, gestor de proyectos y sistema de organización en una única plataforma (Notion, 2024). Esta herramienta nos permitió organizar y documentar todo el proceso de definición de requerimientos, reuniones internas y acuerdos con el cliente, entre otros documentos. Esto facilitó la gestión de la información necesaria para definir los requerimientos.

Tras este proceso, los requisitos definidos para el *MVP* fueron los siguientes:

Requerimientos Funcionales

Los requerimientos funcionales de un proyecto de software, "describen lo que el sistema debe hacer" (Sommerville, 2005). Es decir, estos describen las características y capacidades del software, detallando cómo el sistema debe responder a entradas particulares y cómo debe comportarse en situaciones específicas.

Los requisitos funcionales que se acordaron para el proyecto, fueron:

- Implementar un sistema recomendador de productos financieros basado en un formulario, que utilice Inteligencia Artificial.
- Desarrollar un sistema de reserva de citas con asesores financieros
- Crear un sistema de registro e inicio de sesión para diferentes tipos de usuarios:
  - Personas morales (empresas)
  - Personas físicas (particulares)
  - Asesores
  - Administradores
- Implementar un dashboard de seguimiento de procesos financieros
- Desarrollar perfiles de usuario con las siguientes funcionalidades:
  - Modificación de datos personales
  - Visualización del historial de consultas
  - Acceso a resultados del recomendador de Inteligencia Artificial
- Crear un sistema de gestión de contenidos (CMS) para administrar el contenido del sitio
- Implementar un blog integrado en la plataforma

- Desarrollar un sistema de monetización mediante suscripciones

Requerimientos no Funcionales

Los requerimientos no funcionales definen las propiedades de calidad y restricciones del sistema, como la fiabilidad, el rendimiento, y la capacidad de almacenamiento, que no están directamente relacionadas con funciones específicas del sistema. Estas propiedades emergentes son esenciales para el funcionamiento y la usabilidad del sistema, y su incumplimiento puede llevar a que el sistema sea inutilizable (Sommerville, 2005).

Los requisitos no funcionales que se acordaron en el proyecto son:

- Diseñar una interfaz de usuario intuitiva y fácil de usar
- Implementar un sistema de preferencias de contacto para los usuarios (email, WhatsApp, teléfono)
- Crear un sitio web informativo sobre el producto y sus servicios
- Garantizar la seguridad y privacidad de los datos de los usuarios, cumpliendo con las regulaciones financieras y de protección de datos aplicables
- Asegurar un tiempo de respuesta rápido para las consultas y recomendaciones de la IA, no superando los 3 segundos
- Implementar un diseño *responsive* que se adapte a diferentes dispositivos (desktop, tablet, móvil)
- Garantizar una disponibilidad del sistema del 99.9 % para asegurar el acceso continuo a los servicios financieros
- Asegurar la escalabilidad del sistema para manejar un crecimiento en el número de usuarios y transacciones

2.2. Diseño de la aplicación

En esta sección, abordaré el proceso de diseño del proyecto, este proceso es muy importante en el ciclo de vida del desarrollo de software, ya que es la guía para crear un producto satisfaga los requerimientos que redactamos en la etapa de análisis.

En esta etapa, se sientan las bases que permitirán una implementación fluida y exitosa, así como la creación de una experiencia de usuario capaz de superar las expectativas, logrando una interacción óptima y satisfactoria con la plataforma (Cataldi et al., 1999).

2.2.1. Diseño de la arquitectura de la aplicación

La arquitectura de software es vital para crear nuestro "esqueleto" de lo que será la aplicación. Esta define los componentes principales, sus relaciones y cómo interactúan entre sí. Proporcionando una base sólida para el desarrollo, mantenimiento y evolución del software. Además, influye en atributos de calidad como la escalabilidad, la seguridad y el rendimiento.

De acuerdo a Fernández, 2006, este concepto nace en 1992 a raíz del trabajo de Perry y Wolf, aunque desde finales de la década de los sesenta se empezaba a realizar este proceso llamándolo una "estructuración correcta de los sistemas".

Sin duda, este es un proceso clave para cumplir los requerimientos de nuestra plataforma financiera. Por ello, me involucré profundamente en definir una buena arquitectura para desarrollar nuestro proyecto de forma eficiente, confiable y rápida, dado que contábamos con un tiempo limitado para la creación del *MVP*.

La arquitectura de la aplicación se comenzó a definir formalmente al terminar de establecer los requerimientos, aunque durante ese tiempo ya me encontraba investigando algunas tecnologías que podríamos utilizar para el proyecto.

La arquitectura definida para nuestra aplicación se divide en dos partes principales: el *frontend* y el *backend*.

El *frontend* abarca todas las tecnologías que se ejecutan en el lado del cliente. En nuestra plataforma web, esto se refiere a las tecnologías que funcionan en navegadores como *Google Chrome*, *Microsoft Edge*, entre otros. Es importante destacar que una plataforma web es independiente del dispositivo utilizado para acceder al contenido, siempre y cuando se haga a través de un navegador web. Así, no hay distinción si un usuario accede desde su teléfono inteligente o su computadora; lo crucial es que el contenido se adapte o sea **responsivo**, un término que explicaremos a profundidad más adelante.

El principal objetivo del Frontend es crear una interfaz de usuario gráfica (GUI en inglés) que sea intuitiva y agradable para el usuario, para esto los desarrolladores son apoyados por diseñadores de experiencias e interfaces de usuario, los cuales son comúnmente llamados diseñadores *UX/UI* por su termino en inglés.

El desarrollo web (*Frontend*) se centra principalmente en 3 lenguajes:

- **HTML (HyperText Markup Language):** Este lenguaje es el componente más básico de la Web, definiendo su significado y la estructura de su contenido. Lleva "Hipertexto" en su nombre, ya que "hace referencia a los enlaces que conectan páginas web entre sí, ya sea dentro de un único sitio web o entre sitios web", y utiliza marcas para etiquetar el contenido (Mozilla MDN, 2024b).
- **CSS (Cascading Style Sheets):** Es un lenguaje de estilos que define la presentación visual de lo documentos *HTML* o *XML*. *CSS* permite separar el contenido de la estructura (*HTML*) de su diseño y apariencia, controlando aspectos

como colores, fuentes, espaciado, y disposición de elementos en la página. Esta separación mejora la accesibilidad del contenido web, proporciona mayor flexibilidad y control en la especificación de características de presentación, y reduce la complejidad y repetición en el contenido estructural (Mozilla MDN, 2024a).

- **JavaScript:** Es un lenguaje de programación versátil y dinámico que puede ser interpretado en tiempo real o compilado justo antes de su ejecución (*"just-in-time"*). Su naturaleza multiparadigma, basada en prototipos, ofrece una flexibilidad excepcional, permitiendo diversos estilos de programación como la orientación a objetos, la programación imperativa y la declarativa. *JavaScript* ha revolucionado el desarrollo web moderno, convirtiéndose en un componente esencial debido a su presencia ubicua en los navegadores. Todos los navegadores web actuales incorporan un motor de ejecución de *JavaScript* por defecto, lo que lo convierte en la herramienta predilecta para crear interactividad, manipular el DOM (*Document Object Model*), gestionar eventos del usuario, y realizar solicitudes asíncronas al servidor (Mozilla MDN, 2024c).

Sobre estas 3 tecnologías, cabe aclarar que solamente *JavaScript* es un **lenguaje de programación**, ya que suele ser común la confusión diciendo, pensando que *HTML* y/o *CSS* son lenguajes de programación, sin embargo el primero es como su nombre lo indica un lenguaje de hipertexto, y el segundo es un lenguaje para definir la presentación visual de los documentos con hojas de estilos.

Otra mención importante sobre el desarrollo web actual, es que aunque las 3 tecnologías mencionadas (*HTML, CSS y JS*) son y siguen siendo la base del desarrollo web, estas ya no utilizan de forma directa (también llamada *vanilla*), si no que en la gran mayoría de los casos se utilizan ***Frameworks* o bibliotecas**, esto ayuda a crear aplicaciones web de forma más sencilla y eficiente. De acuerdo con Muente, 2020, "Un *Framework*, es una estructura previa que se puede aprovechar para desarrollar un proyecto". Es decir, funciona como una **plantilla**, que simplifica la elaboración de una tarea, ya que solo es necesario complementarlo de acuerdo a lo que se quiere realizar.

Por otro lado una biblioteca en programación, según educaopen, 2024, "es un conjunto de funciones y rutinas predefinidas que facilitan el desarrollo de software al proporcionar una serie de herramientas listas para usar". La mayor diferencia con los *frameworks*, es que las bibliotecas no se pueden modificar o adaptar a las necesidades de la aplicación, si no que estas solo tienen una función establecida, los *frameworks* en cambio, funcionan como un **entorno de trabajo base para las aplicaciones**.

Ahora, hablando sobre el *Backend*, esta es la capa de lógica de manejo y acceso a los datos de un *software* que permanece invisible para el usuario final. A diferencia del *Frontend*, el *Backend* engloba todas las tecnologías que se ejecutan **en el servidor**, no en el cliente. Esta capa suele involucrar lenguajes de programación, bases de datos, modelos de inteligencia artificial y gestión de servidores, entre otros componentes esenciales (Pérez Ibarra et al., 2021).

De igual forma, a comparación del *Frontend*, el *Backend* no tiene 3 tecnologías o

lenguajes dominantes, sin embargo según Instituto Técnico Superior, 2024, algunos de los lenguajes más utilizados son:

- Python
- JavaScript (Utilizando el Tiempo de Ejecución NodeJS)
- Ruby
- PHP
- Java
- C#
- Go

Una forma sencilla de entender la diferencia entre el *Backend* y *Frontend* es la siguiente:

El *Frontend* es todo lo que el usuario ve y con lo que interactúa, mientras que el Backend es todo lo que "está detrás" o lo que no se ve de forma gráfica en las aplicaciones, siendo lo que proporciona la lógica a estas.

En esta etapa de definición, el líder de proyectos de la empresa se involucró significativamente, principalmente para garantizar la compatibilidad entre la aplicación *frontend* —cuyo desarrollo yo estaba liderando— y el backend, que incluiría los modelos de Inteligencia Artificial para la búsqueda de productos financieros.

Finalmente, tras una exhaustiva deliberación con el equipo, se definió la siguiente arquitectura para la aplicación web (*frontend*):

- **Next.js**: Next.js es un *framework* de *React* de código abierto que facilita la creación de aplicaciones web de alto rendimiento. Este *framework* de *React* fue elegido como la base del frontend, incluyendo su *API serverless*. Next.js ofrece renderizado del lado del servidor, lo que mejora el rendimiento y el *SEO*. Además, su *API serverless* permite crear *endpoints* de *backend* sin necesidad de configurar un servidor separado, lo que puede simplificar la arquitectura y reducir costos (Vercel, 2024a).
- **TypeScript**: Es un lenguaje de programación que expande las posibilidades de *JavaScript*, añadiendo características como tipos estáticos opcionales, clases e interfaces para mejorar la calidad y mantenibilidad del código. La elección de *TypeScript* sobre *JavaScript* sugiere un enfoque en la seguridad de tipos y la mejora de la mantenibilidad del código (Microsoft, 2024).
- **Google Firebase**: *Firebase* es una plataforma de *Google* para el desarrollo de aplicaciones que proporciona una variedad de herramientas y servicios en la nube a un costo sobre demanda. Se utilizó esta plataforma para utilizar diversos servicios de la misma, recortando en gran medida los tiempos de desarrollo (Google, 2024).

- **Vercel:** Es una plataforma de alojamiento y despliegue en la nube optimizada para aplicaciones web modernas, especialmente para proyectos basados en *Next.js*, que ofrece despliegues automáticos, escalabilidad y alto rendimiento (Vercel, 2024b).
- **GitHub**: *Github* es una plataforma en la nube que aloja repositorios de código mediante el sistema de control de versiones **Git**. Esta plataforma, se utiliza para el control de versiones e integración continua y el despliegue continuo (CI/CD, por sus siglas en inglés) en conjunto con Vercel (GitHub, 2024).
- **SanityCMS**: *Sanity* es un sistema de gestión de contenidos (CMS) *headless* que permite a los administradores crear y gestionar contenido de forma flexible y eficiente sin necesidad de un programador. (Sanity, 2024a).

Por oto lado, el backend, se optó por desarrollarlo en **Python**, ya que al ser el lenguaje por excelencia para el desarrollo de modelos de inteligencia artificial, la compatibilidad con este lenguaje es excelente. Además, se utilizaría el *Framework* de *Python* **FastAPI** para crear una **API REST** y conectar mediante *endpoints* el *frontend* y el *backend*. Esta arquitectura, ofrece múltiples ventajas:

- Facilita enormemente la escalabilidad del sistema, permitiendo que la plataforma crezca y se adapte a las demandas cambiantes del mercado
- Proporciona una flexibilidad excepcional, posibilitando que tanto el *frontend* como el *backend* evolucionen de manera independiente sin comprometer la integridad del sistema en su conjunto. Esta separación clara de responsabilidades no solo mejora la mantenibilidad del código, sino que también permite la implementación de actualizaciones y mejoras de forma más ágil y segura en cada componente del sistema.

Para el despliegue del *backend* se utilizó la infraestructura en la nube de ***Amazon Web Services*** (AWS), lo que permitió asegurar la disponibilidad del servicio, facilitar la integración con otros componentes y optimizar la escalabilidad del entorno de producción.

Además para el desarrollo de los modelos de inteligencia artificial fue requerido utilizar otras bibliotecas, algunas de las utilizadas por el equipo de *data science* fueron:

- **Selenium:** Es una herramienta de automatización de navegadores web. Se utiliza principalmente para pruebas automatizadas de aplicaciones web, pero también es útil para *Web Scraping* y automatización de tareas en navegadores (Software Freedom Conservancy, 2024).
- **Pandas:** Es una biblioteca de análisis de datos que proporciona estructuras de datos flexibles y herramientas para trabajar con datos estructurados. Es muy popular para manipulación y análisis de datos (NumFOCUS, Inc., 2024).
- **NumPy:** Es una biblioteca fundamental para la computación científica en Python. Proporciona soporte para arreglos y matrices grandes, junto con una

colección de funciones matemáticas para operar en estos arreglos (NumPy team, 2024).

- **BeautifulSoup:** Es una biblioteca para extraer datos de archivos HTML y XML. Es muy útil para web scraping, permitiendo navegar y buscar en documentos HTML de manera sencilla («Beautiful Soup : ¿cómo aprender a hacer web scraping en Python?», 2022).

De esta forma, nos aseguramos de lograr un desarrollo rápido y eficiente del MVP (Producto Mínimo Viable), ya que cada tecnología seleccionada contribuye a la velocidad de desarrollo, la escalabilidad y la mantenibilidad del proyecto, aspectos cruciales en el contexto de la plataforma que desarrollamos.

A continuación, se presenta un diagrama de la arquitectura de la plataforma, el cual ilustra cómo se integran todas las tecnologías y componentes mencionados:

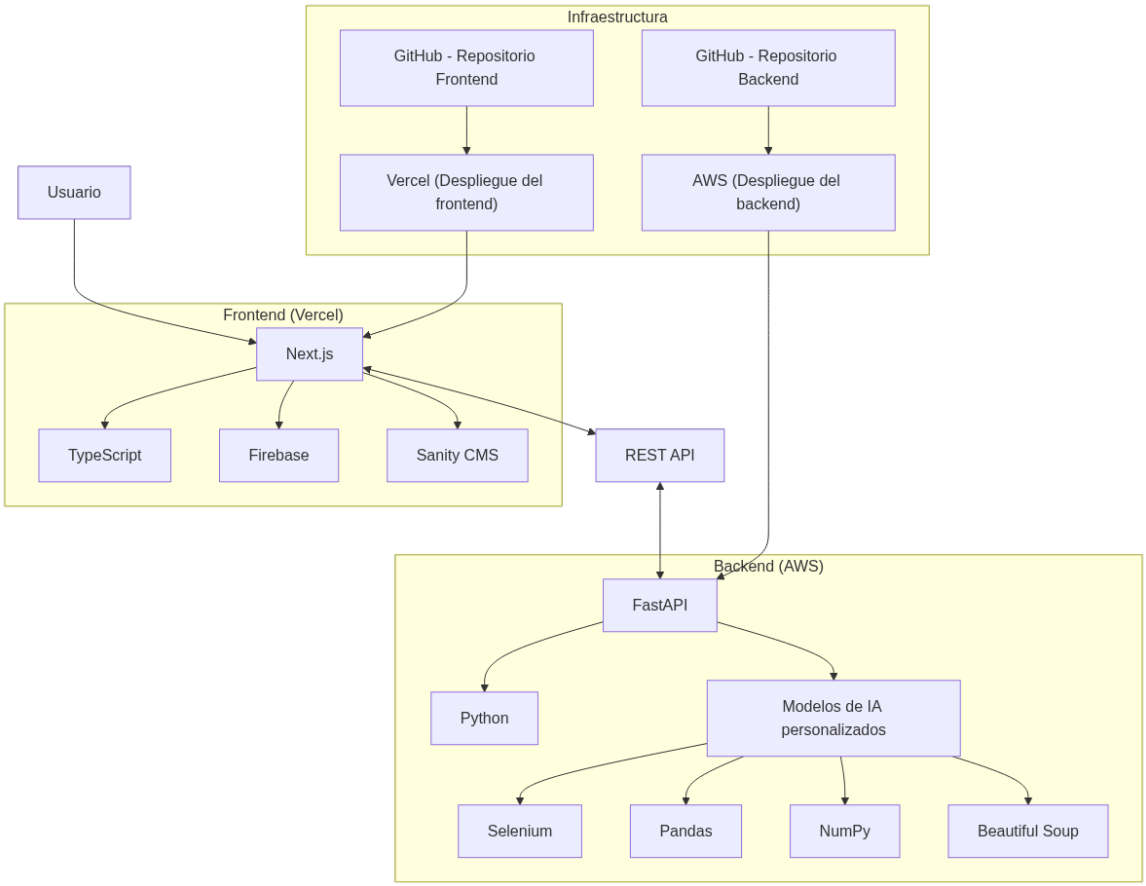

Figura 2.6: Diagrama de la arquitectura de la plataforma

2.2.2. Diseño de la experiencia e interfaz de usuario (UX/UI) de la aplicación

Antes de profundizar, es importante definir qué significa el diseño *UX/UI*. Como se mencionó anteriormente, las siglas *UI* hacen referencia a **Interfaz de Usuario** (del inglés, User Interface). Esta es la interfaz visual con la que el usuario final interactúa en una herramienta de software. Puede ser una aplicación web, de teléfono móvil o incluso en desarrollos más específicos como televisores inteligentes o consolas de videojuegos.

Por otro lado, *UX* hace referencia a la **Experiencia de Usuario** (del inglés, User Experience), que se enfoca en cómo el usuario interactúa con la aplicación antes, durante y después de su uso. Este aspecto es crucial en el desarrollo de aplicaciones modernas, ya que un diseño *UX* eficaz mejora la satisfacción del usuario y aumenta la eficiencia de la aplicación. Al comprender las necesidades y comportamientos de los usuarios, el *UX* permite crear interfaces intuitivas y flujos de trabajo optimizados, lo que se traduce en una mayor retención de usuarios y, en última instancia, en el éxito del producto (Ramírez-Acosta, 2017).

Generalmente, estas dos disciplinas se consideran en conjunto porque son interdependientes. Si un diseñador no tiene en cuenta la experiencia del usuario al crear una interfaz gráfica, puede resultar en una aplicación difícil de usar, lo que conlleva a una baja retención de usuarios. Por otro lado, sin un diseño de interfaz adecuado, no es posible implementar efectivamente los hallazgos obtenidos del análisis y la investigación de la experiencia de usuario.

Esta etapa a mi me gusta definirla como el parteaguas para iniciar el desarrollo de código de una aplicación, ya que los diseñadores que se encargan de esta tarea deben tener un profundo entendimiento tanto de las necesidades del usuario como de las capacidades técnicas del sistema. Su trabajo es crucial para crear una interfaz que no sea solo atractiva visualmente, sino también funcional y fácil de usar. Esto implica un proceso iterativo de diseño, pruebas y refinamiento, donde cada decisión se toma pensando en cómo mejorará la experiencia del usuario final.

En esta etapa de diseño, yo era el principal encargado de dirigir al equipo de diseñadores UX/UI, basándome en los comentarios y necesidades del cliente. Además, estuvieron involucrados nuestro equipo de diseño gráfico y el gerente de proyecto.

Durante esta etapa se utilizó principalmente la herramienta **Figma** para el diseño de las pantallas. *Figma* es una herramienta de diseño de interfaz basada en la nube que permite a los equipos colaborar en tiempo real, además ofrece la posibilidad de crear prototipos interactivos, crear componentes, y hasta crear animaciones (Figma, 2024).

Esta es la principal herramienta utilizada por los diseñadores *UX/UI* en la actualidad, esto se debe a que permite a los equipos trabajar de una forma ágil, colaborativa y asíncrona. Además que no requiere ninguna instalación, su sencillez ayuda a que personas fuera del rubro del diseño gráfico puedan interactuar y crear comentarios en

el trabajo creado por los diseñadores.

En nuestro proyecto, esta herramienta fue clave para iterar sobre los diseños de la interfaz de usuario creada por los diseñadores. La posibilidad de ver los cambios en tiempo real permitió transmitir retroalimentación de forma eficaz y presentar los avances a los *stakeholders* mediante prototipos.

Otra herramienta fundamental en nuestro proceso fue **Notion**. Esta plataforma se utilizó para mantener un registro detallado de las decisiones tomadas y documentar los comentarios del cliente. Además, la empleamos para crear y compartir los textos que los diseñadores posteriormente incorporarían en las pantallas.

La dinámica que llevé a cabo durante esta etapa de diseño, la cual duró aproximadamente mes y medio, fue la siguiente:

- Reunión diaria para revisar avances y comentarios del cliente.
- Reunión cada dos días con el cliente para obtener retroalimentación.
- Reuniones cada dos días con el equipo de desarrollo para recibir comentarios sobre el diseño e iniciar la creación del esqueleto del proyecto.
- Reuniones semanales con el equipo de diseño gráfico para revisar actualizaciones de la identidad de la plataforma y revisar los textos para el sitio web.

Cabe aclarar que la dinámica antes mencionada indica el proceso general que se seguía. Sin embargo, durante todo este periodo hubo una intensa comunicación entre equipos para resolver dudas que surgían a lo largo del diseño de la interfaz.

Este proceso requirió mucho trabajo de todos los involucrados. Por un lado, el equipo de diseño gráfico definía gran parte de cómo debería verse el sitio web, ya que ellos estaban diseñando la identidad de la marca. Por otro, el cliente daba el visto bueno al diseño y solicitaba cambios en aspectos que no le terminaban de agradar. El equipo de desarrollo aportaba comentarios más específicos sobre la ubicación de elementos, animaciones y la interacción entre el usuario y el sitio web. De igual manera, el *project manager* realizaba observaciones sobre aspectos que debían ser iterados en el diseño. En todo este proceso, yo fungía como mediador o dirigente para que el trabajo fluyera de manera correcta, resolviendo dudas e inconvenientes que surgían para evitar tener bloqueantes en esta etapa.

Un desafío que enfrentamos en esta etapa fueron las **iteraciones en la identidad**, ya que el logotipo y los colores de la marca cambiaron en varias ocasiones. Estos cambios fueron a veces bastante radicales. Por ejemplo, se pasó de tener un logotipo con formas muy curvas y colores claros a uno con formas rectas y colores oscuros.

Estos cambios en la identidad visual presentaron desafíos significativos para nuestro equipo de diseño UX/UI, ya que requerían ajustes constantes en la interfaz para mantener la coherencia con la nueva imagen de marca. A pesar de estos retos, estas iteraciones nos permitieron refinar y mejorar continuamente nuestro diseño, asegurando que la interfaz final no solo fuera atractiva, sino que también reflejara fielmente la

visión y valores de la marca. Esta experiencia reforzó la importancia de la flexibilidad y la comunicación efectiva en el proceso de diseño.

2.2.3. Diseño final de la aplicación

En esta sección explicaré en detalle cómo se decidió que funcionaría la aplicación y mostraré algunas capturas y prototipos (versiones previas) del diseño realizado para nuestro cliente.

En primera instancia, identificamos 4 tipos de usuarios en nuestra aplicación:

- **Usuario visitante o sin iniciar sesión:** Cualquier persona que ingrese a nuestro sitio web sin estar registrada o sin haber iniciado sesión. Tendrán acceso a información sobre el funcionamiento de nuestra *app*, educación financiera gratuita sobre productos y servicios financieros, y podrán leer todos los *blogs* publicados en la página web.
- **Usuario con sesión iniciada:** Personas registradas que han iniciado sesión en nuestra aplicación. Además de acceder a todas las características de los usuarios visitantes, pueden buscar productos financieros con nuestro sistema de Inteligencia Artificial, dar seguimiento a sus solicitudes y modificar su perfil.
- **Asesores:** Personal encargado de dar seguimiento y apoyo a las solicitudes de productos financieros de los usuarios. Tienen acceso a todas las funciones de usuarios visitantes y pueden ver las solicitudes que se les asignan.
- **Administradores:** Tienen acceso a todas las funciones disponibles para usuarios visitantes y pueden modificar el contenido de las páginas (agregar, editar o eliminar contenido).

Es importante aclarar que estas son las funciones iniciales de los roles para la versión inicial (MVP). En futuras implementaciones, es probable que se modifiquen las capacidades de cada rol.

Después de todo el riguroso trabajo en equipo, se decidió que la experiencia de la aplicación tuviera una navegación similar a una red social utilizando principalmente la barra de navegación (también llamada *navbar* en inglés) la cual muestra diferentes opciones dependiendo de que tipo de usuario sea.

Cuando el usuario sin sesión iniciada (visitante) tiene en la barra de navegación links hacia paginas informativas sobre el servicio como *Nosotros* o *Precios*, las cuales se colocan de forma vistosa para que los usuarios visitantes conozcan sobre nuestro producto. Además de mostrar 2 botones los cuales permiten al usuario iniciar sesión o registrarse si es su primera vez conociendo el producto.

Por otro lado, cuando un usuario ha iniciado sesión en la aplicación, la *navbar* cambia: se reducen los enlaces informativos al centro y se incorporan más elementos funcionales en la parte derecha. Esto se debe a que, al ya conocer el producto, el usuario no

necesita acceder constantemente a secciones como Nosotros o Precios. Con este ajuste, se optimiza el espacio en la *navbar* y se evita una sobrecarga visual. No obstante, estos enlaces informativos siguen disponibles en el pie de página, por lo que el usuario puede acceder a ellos en cualquier momento si lo desea.

Además, en la *navbar* para los usuarios con sesión iniciada, se muestra su foto de perfil, nombre y correo registrado. Al hacer clic en este espacio, se abre un submenú que contiene botones para acceder a las siguientes secciones o acciones:

- **Mi seguimiento:** En esta sección, el usuario puede revisar el estado de sus solicitudes de productos financieros realizadas mediante la búsqueda con Inteligencia Artificial de la plataforma.
- **Mi perfil:** En este apartado de la aplicación, el usuario puede revisar y/o modificar la información personal que registró en la aplicación. Por ejemplo, nombre(s) y apellido(s), identificación oficial, foto de perfil, entre otros datos.
- **Cerrar sesión:** Esta acción permite al usuario cerrar sesión en el navegador que está utilizando actualmente.

Con la barra de navegación definida, el equipo opto por dividir la aplicación en dos partes principales. La primera es la sección informativa, que contiene detalles sobre la aplicación, sus productos y servicios, preguntas frecuentes, y un *blog* donde se publican artículos sobre diversos temas financieros. La otra parte de la aplicación es la sección transaccional, la cual involucra la búsqueda de productos con IA, el seguimiento de la contratación de productos, historial de consultas y información del perfil del usuario.

A continuación explicare el diseño de estas 2 secciones de la aplicación:

Páginas informativas

En la sección informativa, el principal objetivo es mostrar al usuario nuestro producto, y además enseñar al usuario sobre los productos financieros, tanto para darles conocimientos sobre estos y publicar sus productos en esas páginas

Landing page La Landing Page o página principal es la primera página que se muestra a cualquier tipo de usuario que accede al sitio. En esta página se encuentra información detallada sobre el producto, sus características y beneficios, así como los precios de los diferentes servicios disponibles.

La página principal cuenta con un diseño llamativo y múltiples CTA (*Call to Action*). Estos botones de acción guían al usuario hacia acciones específicas dentro de la página, diseñados estratégicamente para captar nuevos usuarios. Además, la página principal presenta los servicios disponibles tanto para empresas como para personas físicas. Se detalla la oferta de servicios para cada tipo de cliente junto con sus respectivos precios. También se enfatizan las ventajas de utilizar la aplicación en comparación con

métodos tradicionales, destacando principalmente la facilidad de uso para el usuario final.

Listado de productos La plataforma cuenta con una página dedicada a listar los productos financieros disponibles para búsqueda mediante el sistema de inteligencia artificial. Esta página está organizada en dos secciones principales: una para empresas y otra para personas físicas, ya que los productos varían según el tipo de usuario.

Detalle de páginas La página de detalle es aquella que muestra información específica sobre un artículo o producto. Se implementó un diseño estandarizado para todas las páginas de detalle con el fin de agilizar el desarrollo, permitiendo que solo varíe el contenido y las imágenes según sea necesario.

Este diseño se utiliza en múltiples secciones de la aplicación, principalmente en los detalles de cada producto ofrecido y en otras páginas como "Sobre Nosotros", donde se presenta información sobre la aplicación, sus objetivos y visión. El diseño consta de un título, un subtítulo que proporciona información adicional, una imagen de portada y una sección de contenido detallado que combina texto e imágenes complementarias.

Adicionalmente, las páginas que contienen información sobre productos financieros incluyen un banner publicitario de la plataforma. Este banner contiene un botón que invita a los usuarios a crear una cuenta o iniciar una búsqueda financiera utilizando inteligencia artificial. Esta estrategia se implementó para que las páginas funcionen como punto de entrada para usuarios externos que buscan información sobre productos financieros específicos, como tarjetas de crédito, permitiéndoles acceder fácilmente a la plataforma si desean profundizar en los servicios ofrecidos.

Blog La sección de *blog* es un espacio dentro de la plataforma destinado a la publicación de artículos de interés para los usuarios. Los temas son diversos, incluyendo inteligencia artificial, finanzas y otros tópicos relacionados con la plataforma. El listado de blogs está organizado por categorías, mostrando para cada entrada la foto de perfil del autor, fecha de publicación y detalles relevantes.

El diseño implementado es minimalista y atractivo, presentando cada entrada del *blog* con una imagen de portada en miniatura. Para facilitar la navegación, se incluye una barra de búsqueda que permite a los usuarios encontrar contenido específico conforme la biblioteca de artículos crece. Adicionalmente, la clasificación por categorías optimiza la experiencia de búsqueda, permitiendo a los usuarios encontrar fácilmente los temas de su interés.

Detalle de blogs Las páginas de detalle de blog tienen un diseño similar a las páginas convencionales, pero con algunas diferencias notables. El encabezado presenta la imagen de una manera distintiva y muestra información como la categoría del blog y el autor. El contenido se estructura mediante títulos, subtítulos, texto e imágenes

relacionadas. Cada página de blog incluye un banner publicitario de la plataforma, estratégicamente colocado para convertir lectores en potenciales usuarios. Si un visitante llega a la plataforma a través de un artículo interesante, el banner facilita su registro inmediato en el sistema.

Ayuda (FAQs) La página de ayuda (FAQs - Frequently Asked Questions) muestra preguntas frecuentes organizadas por categorías que los usuarios podrían tener sobre la plataforma. Esta sección es fundamental en cualquier aplicación, ya que los usuarios pueden tener dudas sobre cómo utilizar la plataforma o necesitar contactar al equipo de soporte debido a problemas específicos.

Esta página está diseñada para proporcionar respuestas inmediatas a las dudas más comunes, reduciendo la necesidad de contactar directamente al soporte técnico. Sin embargo, si el usuario necesita asistencia adicional, también se proporciona la información de contacto necesaria.

La página presenta un diseño intuitivo donde las preguntas frecuentes están organizadas por categorías. Cada pregunta se puede desplegar mediante un clic o toque en dispositivos móviles, mostrando su respectiva respuesta. Además, la página incluye una sección destacada con la información de contacto de la plataforma y su dirección física.

Pie de página (Footer) El pie de página, o también llamado *footer* en inglés, es un elemento consistente que aparece en la parte inferior de todas las páginas de la aplicación. Este componente contiene enlaces importantes organizados en diferentes categorías para facilitar la navegación del usuario. Las categorías principales incluyen:

- **Información de la aplicación:** Enlaces a páginas como “Sobre Nosotros”, “Contacto” y los servicios ofrecidos.
- **Productos y Servicios:** Links hacia las diferentes categorías de productos financieros disponibles en la plataforma.
- **Recursos:** Accesos al blog, preguntas frecuentes y centro de ayuda.
- **Redes Sociales:** Enlaces a los perfiles oficiales de la empresa en diferentes plataformas sociales.
- **Información Legal:** Enlaces a la recursos como los términos y condiciones, u otros documentos legales importantes.
- **Enlaces a aplicaciones móviles:** Botones para descargar la aplicación móvil en dispositivos *iOS* y *Android*.

El pie de página mantiene un diseño consistente en toda la aplicación, proporcionando una navegación secundaria importante que complementa la barra de navegación principal. También incluye información legal necesaria y el logotipo de la empresa.

Además el pie de página incluye un input para registrarte en el *newsletter* de la aplicación. Cuando un usuario se registra en el *newsletter* le permite recibir información sobre nuevos productos financieros, actualizaciones de la plataforma y otros anuncios importantes directamente en su correo electrónico.

Páginas transaccionales

Las páginas transaccionales son aquellas secciones de la aplicación donde los usuarios realizan acciones específicas que implican interacción directa con el sistema, como iniciar sesión, registrarse, realizar búsquedas o gestionar su perfil. Estas páginas son fundamentales para el funcionamiento principal de la plataforma, ya que permiten a los usuarios acceder y utilizar las características principales del servicio.

A diferencia de las páginas informativas que solo muestran contenido, las páginas transaccionales requieren que el usuario interactúe activamente con la aplicación, ingresando datos, realizando selecciones o confirmando acciones. Estas páginas están diseñadas para ser intuitivas y eficientes, facilitando que los usuarios completen sus objetivos de manera rápida y segura.

Las principales páginas transaccionales de nuestra aplicación incluyen:

- El sistema de inicio de sesión y registro
- La búsqueda de productos financieros mediante inteligencia artificial
- El seguimiento de solicitudes de productos
- La gestión del perfil de usuario

A continuación, se detalla cada una de estas secciones:

Inicio de sesión El inicio de sesión es el proceso mediante el cual un usuario registrado accede a la plataforma ingresando sus credenciales para interactuar con la sección transaccional.

La página cuenta con un diseño minimalista que muestra el nombre y logotipos de la plataforma. Los elementos principales son los *input* para ingresar el correo y contraseña, el botón de inicio de sesión y el botón de registro para nuevos usuarios.

Registro La página de registro permite a los nuevos usuarios crear su perfil en la plataforma. Esta se divide en dos secciones: una para personas físicas y otra para empresas.

El formulario de registro solicita información básica del usuario como nombre, apellidos, correo electrónico, teléfono, edad, género e imagen de perfil. También requiere establecer una contraseña y aceptar la política de privacidad y términos y condiciones de la aplicación.

El proceso de registro para empresas es similar, aunque solicita información específica adicional como el RFC y otros datos relevantes para poder brindar el servicio empresarial de manera adecuada.

Búsqueda de productos (IA) Para el servicio de búsquedas por IA, el diseño se implementó como un formulario intuitivo. Cuando el usuario selecciona esta opción, primero se le pregunta qué tipo de producto está buscando. Dependiendo de su elección, se presenta un formulario sencillo con preguntas cortas enfocadas en identificar sus necesidades específicas.

Una vez completado el formulario, la información se envía al sistema de inteligencia artificial, el cual analiza las respuestas y presenta las cinco mejores opciones para el usuario. Para cada opción, se muestra una descripción detallada y datos relevantes que ayudan en la toma de decisiones.

Si el usuario encuentra una opción que le interesa, puede proceder con el proceso de contratación directamente desde la plataforma. En caso contrario, tiene la flexibilidad de realizar una nueva búsqueda o salir del sistema. Este diseño se inspiró en la simplicidad de los motores de búsqueda como *Google*, permitiendo a los usuarios explorar opciones sin compromiso.

Seguimiento de solicitudes El seguimiento de solicitudes es una sección donde los usuarios pueden monitorear el estado de los productos financieros que han decidido contratar a través de la plataforma. En esta vista, se muestra una lista de los productos en proceso de contratación junto con su estado actual.

Al hacer clic en cualquier solicitud, se abre una ventana lateral que muestra información detallada, incluyendo el estado actual del trámite, el asesor asignado y la opción para subir documentos cuando el asesor los requiera para continuar con el proceso de contratación.

Esta vista también cuenta con una variante para asesores que, aunque mantiene una estructura similar, presenta algunas diferencias importantes. En la parte de detalle, los asesores tienen acceso a botones para modificar el estado de la solicitud, solicitar documentos y revisar los documentos enviados por el usuario. Además, en lugar de mostrar la información del asesor asignado, se visualiza la información y foto de perfil del usuario que está realizando la contratación del producto.

Mi perfil En esta sección de la aplicación, los usuarios pueden modificar su información personal como nombre, correo electrónico, contraseña y teléfono. También tienen la opción de actualizar su foto de perfil y cargar documentos oficiales de identificación (INE) necesarios para los procesos de contratación. Adicionalmente, desde esta vista los usuarios tienen la posibilidad de eliminar su cuenta si así lo desean.

2.3. Desarrollo de la aplicación

En esta sección hablare sobre la etapa de desarrollo del proyecto, describiendo tanto los aspectos técnicos, como los metodológicos del proyecto, incluyendo la metodología de desarrollo utilizada, las diferentes etapas del proceso, las tecnologías implementadas y los procedimientos de control de calidad establecidos.

2.3.1. Metodología utilizada

Para el desarrollo de la plataforma, se implementó la metodología ágil **Scrum**, la cual permitió una gestión iterativa y flexible del proyecto.

Scrum es un marco de trabajo ágil que permite abordar proyectos complejos de manera iterativa e incremental. Se caracteriza por dividir el trabajo en ciclos cortos llamados ***sprints***, generalmente de 2 a 4 semanas, durante los cuales el equipo se enfoca en completar un conjunto específico de objetivos (Martins, 2024).

Los roles definidos en la metodología Scrum son:

- **Product Owner:** Es el responsable de maximizar el valor del producto y el trabajo del equipo de desarrollo. Gestiona el *Product Backlog*, define las prioridades y asegura que las necesidades del cliente sean atendidas.
- **Scrum Master:** Actúa como facilitador del equipo, asegurando que se sigan los principios y prácticas de *Scrum*. Elimina impedimentos, facilita eventos y ayuda al equipo a mejorar sus prácticas.
- **Equipo Scrum:** Es el grupo de profesionales que realizan el trabajo de entregar un incremento potencialmente utilizable del producto al final de cada *Sprint*. Son auto-organizados y multifuncionales.

Los eventos de Scrum son elementos fundamentales que proporcionan regularidad y minimizan la necesidad de reuniones no definidas. Estos eventos son:

- **Sprint Planning:** Reunión que marca el inicio de cada *Sprint*, donde el equipo planifica el trabajo a realizar.
- **Daily Meeting:** Reunión diaria de máximo 15 minutos donde el equipo de desarrollo sincroniza las actividades que realizaron o están realizando.
- **Sprint Review:** Se realiza al final del Sprint para inspeccionar el incremento y adaptar el *Product Backlog* si fuera necesario. Esta reunión suele durar aproximadamente una hora, aunque puede extenderse hasta cuatro horas si el *sprint* es de mayor duración.
- **Retrospectiva de Sprint:** Oportunidad para que el equipo Scrum se inspeccione a sí mismo y cree un plan de mejoras para el siguiente *Sprint*. Ocurre después del *Sprint Review* y antes del siguiente *Sprint Planning*.

Los artefactos de Scrum representan trabajo o valor, y están diseñados para maximizar la transparencia de la información clave. Los principales artefactos son:

- **Product Backlog:** Es una lista ordenada de todo lo que se conoce que es necesario en el producto. Es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El *Product Owner* es responsable de su contenido, disponibilidad y ordenación.
- **Sprint Backlog:** Es el conjunto de elementos del *Product Backlog* seleccionados para el Sprint, más un plan para entregar el Incremento de producto y conseguir el Objetivo del *Sprint*. Es una predicción hecha por el Equipo de Desarrollo.
- **Incremento:** Es la suma de todos los elementos del *Product Backlog* completados durante un *Sprint* y el valor de los incrementos de todos los *Sprints* anteriores. Al final de un *Sprint*, el nuevo Incremento debe estar "Terminado", lo cual significa que está en condiciones de ser utilizado.

Según Schwaber y Sutherland, 2013, se puede resumir la metodología en los siguientes pasos:

1. Un propietario del producto (*Product Owner*) separa las tareas de un problema y las organiza dentro del **Product Backlog**.
2. Durante un *Sprint*, el equipo Scrum se encarga de desarrollar una porción del trabajo priorizado, convirtiéndolo en un resultado medible.
3. El equipo scrum y los *stakeholders* analizan los resultados de cada *Sprint*, realizando los ajustes necesarios para el siguiente mantenga un trabajo constante.

Estos pasos fundamentales se repiten de forma cíclica y continua a lo largo del ciclo de vida del proyecto, lo que permite la evolución constante del producto y una adaptación eficaz a los cambios y necesidades que surgen durante el desarrollo.

Para el desarrollo del *MVP* se eligió la metodología ágil Scrum, debido a su enfoque iterativo e incremental, el cual se adaptaba perfectamente al contexto y necesidades del proyecto. Una de las razones clave para esta elección fue que contábamos con poco tiempo para entregar una versión funcional del producto, y Scrum permite entregar incrementos utilizables de forma rápida, priorizando el valor para el cliente. Además, el equipo ya estaba familiarizado con esta metodología ágil, lo que facilitó su implementación y redujo la curva de adopción. También se buscaba trabajar bajo una estructura organizada que permitiera mantener el orden, claridad en los roles y un seguimiento efectivo de las tareas, especialmente ante cambios frecuentes en los requerimientos. Gracias a la comunicación continua con el cliente y la posibilidad de adaptar el *Product Backlog* en cada Sprint, fue posible ajustar el alcance del *MVP* conforme el cliente validaba avances, optimizando así el desarrollo y enfocándonos en las funcionalidades más relevantes.

Implementación de Scrum dentro del proyecto

El equipo implementó *Scrum* de manera flexible, adaptándolo a las necesidades específicas del proyecto. Como líder de desarrollo web, mis responsabilidades se alineaban principalmente con el rol de *Scrum Master*, facilitando que el equipo siguiera las prácticas de *Scrum*, eliminando impedimentos y manteniendo el *Backlog* actualizado con las actividades del equipo. El *Project Manager* asumió el rol de *Product Owner*, asegurándose de que las necesidades del cliente fueran atendidas y administrando el trabajo de los equipos de *Frontend* y *Backend*.

Además de mis responsabilidades como *Scrum Master*, también participé como miembro del *Scrum Team*, tomando algunas tareas de desarrollo, aunque en menor cantidad que mis compañeros. Esta dualidad de roles permitió una mayor flexibilidad en la gestión del proyecto.

En cuanto a las ceremonias *Scrum*, se realizaban *Daily Meetings* de 15 minutos para dar status sobre el avance de cada miembro del equipo. Las retrospectivas, en lugar de realizarse al final de cada *sprint*, se llevaban a cabo mensualmente debido a la velocidad del desarrollo. En estas sesiones, el equipo utilizaba un Tablero colaborativo para documentar los aspectos positivos, negativos, áreas de mejora y aprendizajes del período.

La *Sprint Planning* se realizaba cada dos semanas, donde el equipo planificaba las tareas del siguiente sprint y creaba los issues correspondientes en *Notion*, plataforma utilizada para el seguimiento de tareas. Durante las retrospectivas mensuales, también se incluía una *Sprint Review* simplificada para evaluar el progreso del equipo y realizar ajustes necesarios en las tareas.

Los roles de *Scrum Master* y *Product Owner* mantenían cierta interrelación, ya que como líder de desarrollo, participaba en reuniones con el cliente para definir prioridades y asegurar su cumplimiento por parte del equipo. Esta flexibilidad en los roles permitió una mejor adaptación a las necesidades del proyecto durante los cuatro meses de desarrollo.

2.3.2. Desarrollo del Frontend de la aplicación

Como se menciono en la sección Subsección 2.2.1, el *frontend* se realizo con el *Framework* de *Next.js* utilizando el lenguaje *TypeScript*.

La elección de *TypeScript* en lugar de *JavaScript* nos proporcionó importantes ventajas durante el desarrollo, principalmente debido a que *TypeScript* detecta errores de tipado durante la programación. Por ejemplo, identifica cuando un desarrollador intenta asignar un valor booleano a una variable de tipo string, o lo que es muy común en el desarrollo web, cuando intenta acceder a una propiedad inexistente en un objeto (ya sea por errores de escritura o problemas al invocar el objeto). Además, el desarrollo se agiliza significativamente gracias al autocompletado *Intellisense* que el

editor de código proporciona con este lenguaje, permitiendo escribir código de manera más eficiente.

Por otro lado, una ventaja clave de haber elegido *Next.js* como *framework* de desarrollo fue la reducción significativa del tiempo de implementación, ya que este incluye de forma predeterminada diversas bibliotecas y funcionalidades esenciales para una aplicación web, las cuales normalmente deben agregarse o configurarse manualmente. Un claro ejemplo de esto es el Enrutamiento, es decir, el mecanismo que permite definir y gestionar las diferentes rutas o páginas de la aplicación web. En una aplicación de *React* convencional, este proceso suele requerir bibliotecas externas (como *React Router*) y una configuración manual. En cambio, *Next.js* ofrece un sistema de enrutamiento automatizado y simplificado, basado en la estructura de archivos del directorio /pages. Esto permite crear nuevas rutas simplemente añadiendo archivos en dicha carpeta, lo que agiliza considerablemente el desarrollo.

Otra de las grandes ventajas de haber utilizado Next.js es el hecho que nos permite crear la posibilidad de crear API Routes dentro del mismo proyecto, es decir, que podemos tener un "backend" (limitado a funciones Serverless) integrado en nuestro mismo proyecto, lo que nos permite crear funciones más complejas en nuestra aplicación sin requerir aumentar la complejidad y costos teniendo un backend a parte.

Un aspecto fundamental de *Next.js* es la optimización que ofrece para mejorar la optimización para motores de búsqueda o también llamada SEO (*Search Engine Optimization*) de las aplicaciones web. Como menciona Mousinho, 2020, el SEO es el conjunto de técnicas y estrategias que buscan optimizar un sitio web para que aparezca en las primeras posiciones de los resultados de búsqueda de los motores como *Google*, lo cual es crucial para aumentar la visibilidad online y atraer más tráfico orgánico al sitio.

Finalmente, una ventaja adicional de Next.js frente a React puro es la capacidad de generar páginas estáticas (también llamado *Static Site Generation* o SSG). En este proceso, Next.js construye las páginas previamente, por lo que estas no se generan de forma dinámica cuando el usuario las visita, sino que ya están precargadas y optimizadas en el servidor. Esto mejora significativamente la velocidad de navegación que experimenta el usuario en nuestra web.

En el caso de nuestro proyecto se utilizo la **versión 13** de Next.js, esto es importante mencionar ya que esta versión incorporó 2 grandes cambios que impactaron la forma de utilizar el *framework*, así como otras ventajas. La primera de ellas es que se cambio el directorio de enrutamiento de llamarse "pages" a "app", además del cambio de nombre la implementación tenia algunos cambios en la forma de utilizar este nuevo enrutamiento, entre los que destaca la incorporación de "layouts", que permiten definir una estructura común para todas las páginas "hijas" que se encuentren dentro de su jerarquía.

La segunda novedad importante fue la introducción del "*Server Side Rendering*" (SSR) por defecto en todos los componentes. Anteriormente, los componentes inicialmente se renderizaban en el cliente (*Client Side Rendering*), y había que especificar

explícitamente si queríamos utilizar SSR, pero ahora esta lógica se invierte. Esta característica ayuda a mejorar significativamente el rendimiento de la aplicación, ya que el servidor puede generar el *HTML* completo antes de enviarlo al cliente, reduciendo el tiempo de carga inicial y mejorando el *SEO*. Además, permite realizar operaciones con los datos directamente en el servidor sin necesidad de exponerlas al cliente, lo cual también nos ayuda a ocultar las direcciones de *API* para agentes mal intencionados (Baláz Orbán, 2022).

Estos cambios hasta cierto punto eran "opcionales", debido a que se el *framework* seguía permitiendo utilizar sin inconvenientes el enrutamiento clásico por "pages", sin embargo, se decidió incorporar el nuevo "app router" en nuestra aplicación debido a los siguientes motivos:

1. **Evitar deuda técnica:** Aunque era muy reciente la incorporación del *app router* en *Next.js*, era evidente que *Next.js* en el futuro iba a apuntar todas sus nuevas características y soporte a este nuevo enrutamiento, por lo que si elegíamos el enrutamiento por "pages" nos hubiera llevado en un futuro a la necesidad de migrar al *app router*.

2. **Utilización de la funcionalidad "layouts":** La novedad de definir una estructura en común con jerarquía nos fue muy útil para definir elementos que siempre se mostraran en todas las páginas, como la barra de navegación (*navbar*) y el pie de página (*footer*). Además, el uso de *layouts* permite evitar el renderizado repetido de estos componentes en cada carga, lo que contribuye al ahorro de memoria del cliente.

3. **Utilización de SSR al manejar datos:** Las páginas con renderizado en el servidor fueron muy útiles para obtener la información de manera sencilla antes de renderizarlas, evitando el uso de cláusulas como *useEffect* de *React*. Además, nos ayudó a ocultar los *Endpoints* del *backend* de IA, protegiéndolo contra posibles acciones maliciosas.

Para facilitar la internacionalización de la aplicación, se implementó un sistema de localización de textos mediante un *Middleware* en *Next.js*. Este *middleware* detecta el idioma preferido del usuario a través de un parámetro en la *URL* por ejemplo "/es" o "/en" que envolvía todas las páginas en el *app router*. Una vez que el *middleware* detectara que el lenguaje seleccionado por el parámetro fuera soportado, obtenía las cadenas de texto traducidas de su archivo JSON correspondiente.

Por otro lado, se utilizó una combinación de páginas estáticas y dinámicas. Anteriormente en la Subsección 2.2.3, se mencionaba que el equipo "separo" a la aplicación en 2, donde tenemos las páginas informativas y las páginas transaccionales. De esta forma se decidió que las páginas informativas se generaran de forma estática utilizando la *SSG* de *Next.js* para optimizar el rendimiento, mientras que las páginas transaccionales se renderizan de forma dinámica en conjunto de utilizar renderizado de servidor en algunas secciones de la página para tener una experiencia óptima.

La organización de los componentes siguió una estructura similar a la utilizada gene-

ralmente en la industria, a partir de un directorio “components”, se creó una carpeta por cada página de la aplicación, la cuales contienen sus componentes específicos. Sin embargo, para los componentes reutilizables (componentes que se empleaban en múltiples páginas) se creo un directorio “General” haciendo referencia a que estos componentes se utilizan en varias secciones de la página, esto para promover la reutilización de código y mantener un estándar consistente en toda la aplicación.

Para el estilizado de la aplicación, se implementó *CSS Modules*, lo que nos que permitió escribir hojas de estilos *CSS* con un alcance local para cada componente, evitando el problema de conflictos de nombres entre estilos (estilos con el mismo nombre) y facilitó el mantenimiento al tener los estilos directamente asociados con sus componentes correspondientes.

La gestión de errores se implementó a través de páginas personalizadas. Se desarrollaron páginas específicas para errores tipo 404 (página no encontrada) y 500 (errores de servidor), ofreciendo una experiencia más amigable y manteniendo la coherencia visual de la aplicación. Estas páginas informan claramente a los usuarios sobre el error ocurrido y los invitan a continuar navegando por la aplicación.

Implementación del CMS

Un Sistema de Gestión de Contenidos (CMS, por sus siglas en inglés *Content Management System*) es una plataforma que permite administrar el contenido de una aplicación mediante una interfaz gráfica intuitiva. Esto elimina la necesidad de contar con desarrolladores para actualizar manualmente el contenido de una aplicación (web, móvil, etc.), reduciendo así los costos y la complejidad de su mantenimiento.

Se decidió implementar un CMS en nuestra aplicación para la **parte informativa**, permitiendo que el cliente cuente con un equipo sin conocimientos técnicos en programación que pueda actualizar constantemente el contenido de la página —como *blogs* y productos financieros— sin requerir la intervención de un programador.

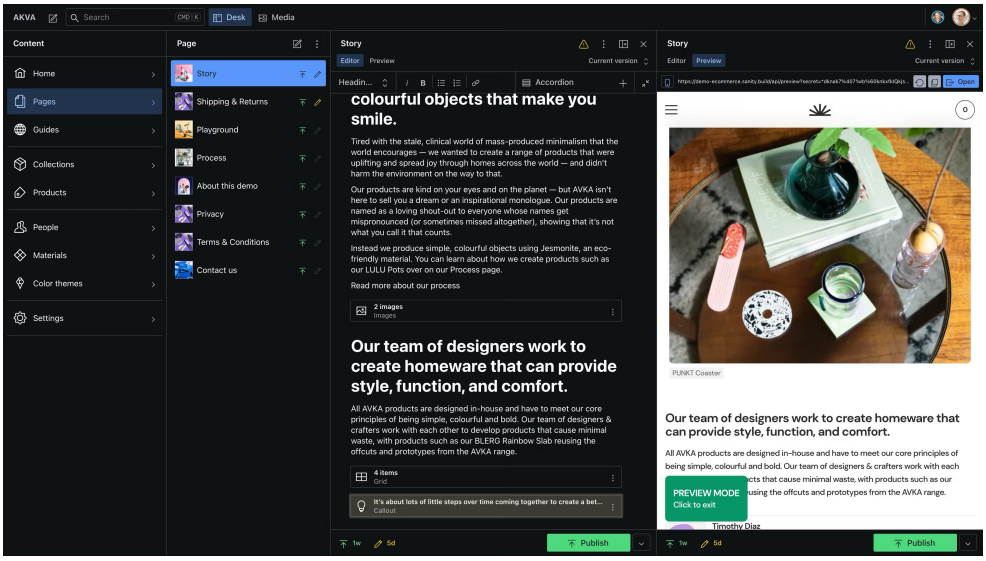

Figura 2.7: Interfaz del CMS Sanity con proyecto ejemplo. Fuente: Sanity, 2025

Como se mencionó en la arquitectura de la aplicación, se utilizo *Sanity* como CMS para aplicación, esto debido principalmente a los siguientes factores:

- **Integración con Next.js:** *Sanity* cuenta con una excelente integración dentro del *framework Next.js*, lo que facilita su implementación y reduce significativamente los tiempos de desarrollo. Además, este CMS era compatible con el nuevo "app" routing de *Next.js* 13.
- **Interfaz moderna e intuitiva:** La interfaz de este CMS es fácil de utilizar para miembros no especializados en software, lo cual es idóneo para su propósito.
- **Precios accesibles:** Los precios del CMS son accesibles y flexibles, adaptándose al tamaño de la aplicación.
- **Muy personalizable:** Se pueden crear tipos de contenido con campos variados y personalizados, además de poder añadir plugins de la comunidad para ampliar la funcionalidad del CMS.
- **Funciones de colaboración:** *Sanity* permite que múltiples personas editen contenido simultáneamente, ofreciendo colaboración en tiempo real similar a plataformas como *Google Docs*.

Sanity se fundamenta en la creación de esquemas de contenido, que son estructuras detalladas donde se especifican múltiples elementos clave, entre algunos de estos se encuentran, el nombre identificativo del tipo de contenido, una descripción comprehensiva que explica su propósito y funcionalidad, y principalmente la serie de campos que definen la estructura y tipos de datos que tendrá cada instancia de ese tipo de contenido. Estos esquemas son realizados en archivos de *TypeScript* y actúan como plantillas que determinan cómo se organizará y presentará la información en el sistema.

Entre las modificaciones realizadas en la estructura base de *Sanity* para cumplir con los requerimientos de la aplicación, se implementó la creación de esquemas *Singleton* o de instancia única. Por defecto, todos los esquemas generan una lista en la interfaz, pero existen casos como las páginas de inicio y "sobre nosotros" que requieren una sola instancia del esquema. Para lograr esto, se modificó la configuración creando una estructura personalizada dentro del archivo "sanity.config.ts" que se encuentra en la raíz del proyecto, definiendo manualmente los elementos *Singleton* que se muestran en el lateral de la interfaz (*sidebar*). Posteriormente, se filtraron los esquemas de contenido que no debían mostrarse en forma de lista de la siguiente manera:

```
export const sanityCustomStructure: StructureResolver = (S) => {
    return S.list()
        .title("Contenido")
        .items([
            // Singleton items
            S.listItem()
                // Singleton Page
                .title("Pagina Singleton")
                .icon(HeartIcon)
                .child(
                    S.document().schemaType("singletonSchemaName")
                    .documentId("singletonPage")
                ),
            S.divider(),
            ...S.documentTypeListItems().filter(
                (listItem) => {
                    return !["singletonSchemaName"]
                        .includes(listItem.getId());
            })
        ]
    );
};
```

Código 2.1: Ejemplo de estructura personalizada de esquemas Singleton en Sanity

Para conectar la información de *Sanity* con la vista del usuario, se utiliza GROQ (*Graph-Relational Object Queries*), el cual es un lenguaje de consulta para obtener información desde el cliente. GROQ es un lenguaje de código abierto desarrollado por *Sanity* cuya principal característica es obtener y filtrar el contenido de archivos JSON. Esto permite solicitar únicamente los datos requeridos, simplificando la obtención de información del CMS de manera eficiente (Sanity, 2024b).

Finalmente, en el administrador de *Sanity* se configuró un *WebHook* que reconstruye la página cada vez que se realiza un cambio en la información de cualquier sección. Esto es necesario porque, como se mencionó anteriormente, la parte informativa de la página funciona de forma estática. El *Trade-off* de esta decisión es que los cambios no se reflejan instantáneamente, sino hasta después de unos minutos, cuando se crea y reemplaza la versión de la página que contiene las modificaciones.

Dinámica de colaboración y manejo de versiones

Como se mencionó anteriormente, durante el desarrollo del proyecto se utilizó *GitHub* como plataforma de almacenamiento y control de versiones. Asimismo, se empleó Notion para el manejo, delegación y seguimiento de las tareas del *Sprint*. Estas dos plataformas contaban con una integración que estructuraba la forma en que el equipo manejaba las versiones del proyecto.

Al utilizar *GitHub* —y por ende Git— como sistema de control de versiones, se estableció una estructura jerárquica de ramas con el objetivo de utilizar esta herramienta de forma eficiente y mantener un control preciso del código. La rama principal, "main", fue destinada exclusivamente al código en producción; es decir, al código compilado que estaría disponible para el usuario final. Para que un cambio pudiera integrarse en esta rama, debía haber sido previamente probado y revisado por al menos otro miembro del equipo, con el fin de garantizar su calidad y estabilidad.

Por debajo de "main", se mantuvo la rama "development" como la rama principal de desarrollo, donde se integraban todas las características nuevas una vez que estaban listas para pruebas. A partir de "development", cada nueva característica o corrección se desarrollaba en una rama específica, cuyo nombre era el ID del ticket correspondiente en *Notion* (por ejemplo, "NOTION-123"). La figura 2.8 demuestra gráficamente la estrategia de manejo de versiones que utilizamos en el proyecto.

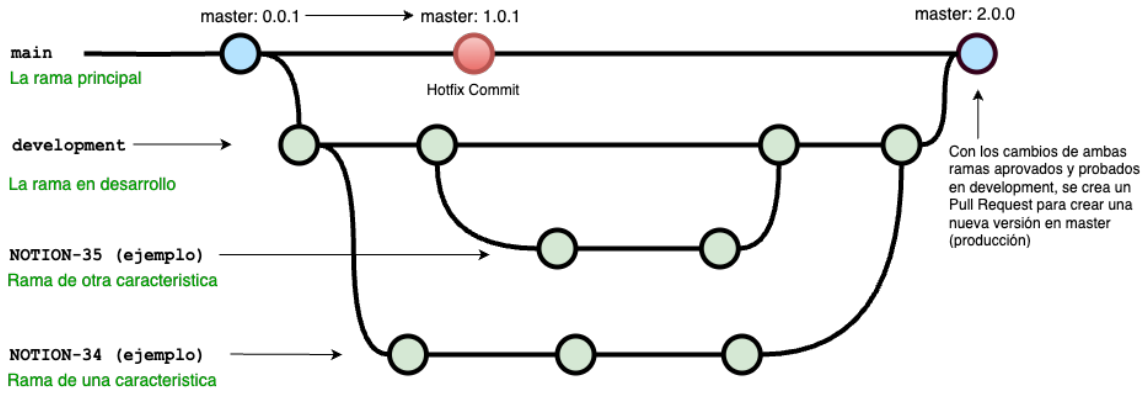

Figura 2.8: Diagrama de estrategia de manejo de versiones con Git

El proceso de integración de código seguía un flujo estructurado mediante *Pull Request (PR)*. Cada PR debía cumplir con un formato específico que incluía:

- **Título del PR:** Compuesto por el ID del ticket de *Notion* seguido del título descriptivo de la tarea (ejemplo: "NOTION-123: Implementación del sistema de autenticación")
- **Descripción del PR:** Una explicación detallada de los cambios realizados, incluyendo:

- Lista de modificaciones principales
- Capturas de pantalla demostrando los cambios visuales cuando aplicaba
- Consideraciones especiales o dependencias

Este sistema permitía mantener una trazabilidad clara entre el código y las tareas en *Notion*, facilitando el seguimiento del progreso y la revisión del código. Además, la integración entre *GitHub* y *Notion* actualizaba automáticamente el estado de las tareas cuando se creaban o fusionaban los PRs.

Los *Pull Requests* requerían al menos una revisión aprobatoria de otro miembro del equipo antes de poder fusionarse con la rama "development". Este proceso de revisión aseguraba la calidad del código y permitía la detección temprana de posibles problemas.

2.3.3. Conexión con Google Firebase

Como se mencionó anteriormente, la aplicación utilizó *Google Firebase* como plataforma para conectar diversos aspectos del sistema. *Firebase* ofrece distintas soluciones para diversos casos de uso, y a continuación se enlistan las que se implementaron en este proyecto:

- **Authentication**: Este sistema maneja la gestión de usuarios y el proceso de inicio de sesión en la aplicación de forma sencilla.
- **Cloud Firestore**: Este servicio es una base de datos *NoSQL*, es decir, es una base de datos no relacional en la nube que se utilizamos en la aplicación para almacenar los datos de los usuarios y sus solicitudes.
- **Cloud Storage**: Este servicio es una nube de almacenamiento que utilizamos principalmente para almacenar contenido multimedia, por ejemplo, las fotos de perfil de los usuarios.
- **Cloud Functions**: Este servicio permite ejecutar código *backend* automáticamente en respuesta a eventos o peticiones HTTP, sin necesidad de administrar servidores (Google, s.f.).

Firebase incluye una biblioteca de *JavaScript* la cual facilita la implementación de todos sus servicios en una aplicación web, por lo que la implementación de *Firebase* esta principalmente basada en esta con algunas particularidades:

Para utilizar *Authentication* dentro de la aplicación, implementamos una *API* de autenticación utilizando la funcionalidad de *API Serverless* de *Next.js*, la cual contaba con tres *endpoints*. El primer *endpoint* realizaba el login: cuando la librería de *Firebase Auth* verificaba que las credenciales eran correctas, la *API* utilizaba el *header Authorization* para generar una *Cookie* de sesión con expiración. Esta *cookie* permitía al usuario salir y volver a la aplicación sin tener que reingresar sus credenciales, siempre y cuando la *cookie* siguiera activa. El segundo *endpoint* se encargaba de la

verificación de sesión, validando la existencia de una sesión válida en la *cookie* y que no estuviera expirada para mantener la sesión iniciada. Finalmente, el tercer *endpoint* gestionaba el cierre de sesión, eliminando la *cookie* para evitar el acceso automático en futuras visitas.

Por la parte de *Firestore*, este servicio almacenaba los datos de las solicitudes de contratación de productos financieros creadas a partir de las búsquedas por inteligencia artificial. Además, estas solicitudes activaban *Cloud Functions* cuando cambiaban de estado, por ejemplo, al crearse se ejecutaba código para notificar al asesor, o cuando la solicitud requería documentos por parte del usuario. Por último, el servicio de *Cloud Storage* se implementó para guardar los archivos multimedia de la aplicación, como fotos de perfil o documentos de los usuarios (identificaciones o requerimientos específicos para la contratación de productos financieros).

La incorporación de *Google Firebase* eliminó la necesidad de desarrollar y mantener un *backend* complejo para la mayoría de las funciones de la aplicación, lo que se tradujo en una aceleración considerable del proceso de desarrollo.

2.3.4. Conexión con backend de Inteligencia Artificial

Como mencioné anteriormente, mi equipo estuvo encargado exclusivamente del desarrollo del *frontend* de la aplicación, por lo que el desarrollo del *backend* de Inteligencia Artificial, incluyendo los modelos y obtención de datos, no estuvo entre mis actividades. Sin embargo, en esta sección comentaré cómo se realizó la conexión entre el *backend* y *frontend*.

Se decidió implementar un sistema de formularios estáticos (es decir, no cambiaban) para cada tipo de solicitud de búsqueda de productos financieros, de esta forma el *backend* de IA se limitaba a solamente obtener los resultados de las búsquedas. Estos formularios fueron diseñados entre el equipo del cliente y el equipo de inteligencia artificial, los cuales se diseñaron de forma que no fueran muy tediosos para responder por los usuarios y fueran suficientes para obtener resultados precisos por el modelo de IA.

Los formularios fueron proporcionados a nuestro equipo, y nosotros nos encargamos de estandarizarlos en un formato mediante archivos JSON, lo que facilitó su modificación y mantenimiento. Esta decisión de diseño resultó en la creación de un componente dinámico que interpretaba estos archivos JSON para renderizar el formulario específico que cada tipo de búsqueda requería. Esta aproximación proporcionó flexibilidad para adaptar los formularios según las necesidades cambiantes del sistema.

Una vez que el usuario completaba el formulario, los datos se enviaban al *backend* mediante una *API REST* proporcionada, utilizando un sistema de Cifrado para garantizar la seguridad de la información. El *backend* procesaba esta información a través del modelo de IA y retornaba un conjunto de productos financieros que mejor se adaptaban a las necesidades y perfil del usuario, ordenados según su relevancia.

Los resultados se presentaban en una interfaz intuitiva que permitía a los usuarios comparar las diferentes opciones y explorar los detalles de cada producto. El *backend* de IA también almacenaba la información multimedia de las opciones (imágenes, etc.) mediante un *endpoint*. Se decidió realizar todas las peticiones antes de mostrar la información al cliente para evitar tiempos de espera después de obtener los resultados, aunque esto aumentara un poco más el tiempo de la solicitud, por ello, durante este proceso se implementó una animación que indica al usuario que la operación está en curso, para que el usuario estuviera al tanto que su información seguía en camino y tuviera una buena experiencia al realizar sus consultas.

2.4. Etapa de pruebas

Durante la etapa de pruebas del proyecto, se adoptó un enfoque principalmente centrado en pruebas manuales y de integración, en lugar de pruebas unitarias exhaustivas. Esta decisión se fundamentó en varios factores estratégicos:

En primer lugar, al ser una aplicación *frontend* con un fuerte componente de interacción usuario-interfaz, las pruebas manuales permitían evaluar de manera más efectiva la experiencia del usuario y el flujo de la aplicación. El equipo realizaba sesiones dedicadas donde se intentaba generar errores en el flujo normal de la aplicación, probando diferentes escenarios y casos de uso no convencionales.

También un aspecto fundamental de las pruebas consistía en la comparación meticulosa entre la implementación final y los diseños originales en *Figma*. Esto aseguraba que la aplicación no solo funcionara correctamente, sino que también mantuviera la fidelidad visual y de experiencia usuario planificada en la etapa de diseño.

Cuando se detectaban errores, inconsistencias o comportamientos inesperados durante estas sesiones de prueba, se seguía un proceso estructurado:

1. Documentación detallada del error encontrado
2. Creación de una tarea específica en *Notion* con toda la información relevante
3. Asignación de prioridad según el impacto en la funcionalidad de la aplicación
4. Seguimiento del proceso de corrección mediante el sistema de gestión de tareas

Si bien se realizaron menos pruebas unitarias de las que podrían considerarse ideales en otros contextos —como en el desarrollo de sistemas críticos, aplicaciones bancarias o plataformas de alta concurrencia— esta decisión se debió principalmente al limitado tiempo de desarrollo con el que contábamos al llegar a esta etapa. Además, el equipo no estaba muy familiarizado con estas prácticas, lo que habría requerido tiempo adicional de capacitación. Sin embargo, esta decisión se compensó con un riguroso proceso de pruebas manuales y una estrecha colaboración entre los miembros del equipo para identificar y resolver problemas. Este enfoque resultó ser efectivo para

nuestro caso específico, permitiendo mantener un alto nivel de calidad en la aplicación mientras se optimizaban los recursos y tiempos disponibles.

Para asegurar que la aplicación respondiera de manera eficiente bajo distintas condiciones de carga, se implementó un proceso de pruebas de rendimiento utilizando la herramienta *Lighthouse*, la cual está integrada en navegadores basados en *Chromium*. Los resultados obtenidos para la aplicación web se muestran en la figura 2.9. Esta herramienta evalúa diversos aspectos como la velocidad de carga, la optimización de imágenes, la accesibilidad, el puntaje SEO y el rendimiento general de la aplicación. Las pruebas se complementaron con evaluaciones manuales en dispositivos físicos utilizando diferentes navegadores. El objetivo principal fue alcanzar el mayor puntaje posible en SEO —lo cual mejora la visibilidad en motores de búsqueda como *Google*— y en los demás indicadores evaluados por la herramienta.

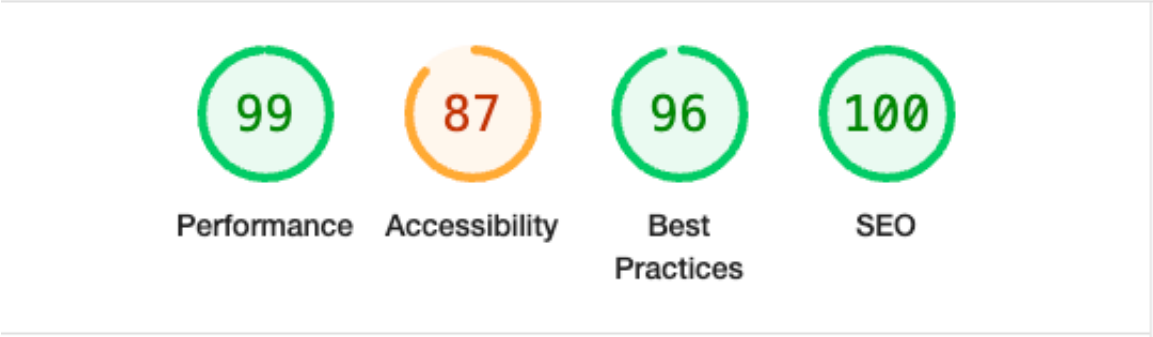

Figura 2.9: Resultados finales de las pruebas de Lighthouse en la aplicación.

2.5. Despliegue de la aplicación

Para el despliegue de la aplicación *frontend* se utilizó la plataforma *Vercel*, como se mencionó en la sección del diseño de la arquitectura. Esta elección se basó en que, además de ofrecer la ventaja de realizar despliegues de forma sencilla, automatizada y eficiente, es la plataforma mejor optimizada para alojar aplicaciones de *Next.js*, principalmente porque *Vercel* es la empresa creadora de este *framework*.

Para integrar *Vercel* al proyecto se conecto el repositorio alojado en GitHub con la plataforma, además de configurar la rama principal "main" como producción. Además se conecto el Dominio adquirido para tener una *URL* personalizada con el nombre del proyecto, para esto basto con colocar en el registrador de dominio las *DNS* que *Vercel* indicaba en su plataforma. De igual forma para que la plataforma pudiera generar una versión que funcionara correctamente se agregaron las variables de entorno con los *Tokens* y valores secretos que la aplicación requería para funcionar.

Con esas configuraciones básicas *Vercel* ya es capaz de generar un *deployment* para la versión publicada en la rama "main" en el dominio que se configuró, además, por

defecto, *Vercel* crea una versión por cada *commit* realizado en cualquiera de las ramas. Lo cual es muy útil para probar tanto la versión de una rama en desarrollo en web u otros dispositivos fuera de un ambiente local, como para probar si existen errores en las compilaciones del proyecto, por ejemplo, si existía un error de tipado en alguna variable. En este caso, *Vercel* no compilaba la versión y enviaba un mensaje por medio de su plataforma enviando un error.

Además de esto, la integración de *Vercel* dentro de nuestro proyecto nos añadió un tipo de Integración Continua (CI) al momento de generar *Pull Requests* para combinar los cambios de una característica a cualquiera de las ramas, ya que *Vercel* generaba una versión a partir del PR y enviaba un mensaje en forma de comentario por medio de *GitHub* si la versión se había generado y desplegado correctamente o si había algún error que se tenía que arreglar antes de mezclar la versión.

Sin embargo, para completar el *deployment*, fue necesario configurar el CMS y las métricas. Para el CMS, se requirió acceder al administrador de *Sanity* y establecer el dominio personalizado como dominio seguro, permitiendo así que los usuarios administradores accedieran con sus credenciales y modificaran el contenido. Además, para finalizar la configuración del CMS en producción, se implementó un *webhook* en *Vercel* que desencadena una reconstrucción del sitio cuando recibe una petición. En el administrador de *Sanity*, se configuró un "GROQ-powered" *webhook* que, al detectar cualquier modificación en los contenidos, inicia una reconstrucción completa del sitio. Aunque el administrador permite seleccionar contenidos específicos para las reconstrucciones, en este caso particular necesitábamos abarcar todos los contenidos, ya que el sitio se construye de forma estática (Sanity, s.f.).

Por otro lado, para la configuración de las analíticas se utilizó la plataforma de *Google Analytics*, la cual tiene una configuración sencilla que consiste en incluir un identificador en las Metaetiquetas del sitio, permitiendo que la plataforma obtenga las estadísticas y las muestre en su administrador.

Finalmente, con todo configurado, se publicó el sitio y como último paso, se probaron las metaetiquetas agregadas mediante código. Estas debían reflejar correctamente la información en redes sociales, creando tarjetas de contenido y mostrando vistas previas de la imagen del sitio web compartido. Estas pruebas no eran posibles en un entorno de desarrollo, ya que al ser utilizadas por sitios web externos (como *Facebook* o *Twitter*), estos obtienen la información accediendo directamente al enlace proporcionado, lo cual no es posible cuando el sitio está en un entorno local. Mediante estas pruebas verificamos si era necesario realizar ajustes en la generación de las metaetiquetas.

Capítulo 3

Resultados

El resultado principal del trabajo es la plataforma de asesoría financiera basada en inteligencia artificial. Esta aplicación web permite a los usuarios registrados obtener recomendaciones personalizadas de productos financieros y si así lo desea, contratarlos con el apoyo de un asesor financiero, todo desde una misma plataforma. Además, los usuarios pueden aprender sobre los productos financieros listados que les interesen, convirtiendo a la plataforma no solo en un buscador de productos financieros, sino también en un recurso de educación financiera.

Las características principales de la aplicación son:

- La aplicación web utiliza el popular *framework Next.js*, facilitando el desarrollo de una aplicación web moderna y eficiente con renderizado del lado del servidor.
- Se implementó *TypeScript* como lenguaje de programación principal, lo que permitió un desarrollo más seguro y mantenible gracias a su sistema de tipado.
- Se cuenta con un sistema de autenticación basado en *Firebase Authentication* que permite tener diferentes tipos de usuarios en la plataforma (personas físicas, morales, asesores y administradores), garantizando un acceso seguro y personalizado.
- Se implementó un *CMS* basado en *Sanity* para facilitar la gestión del contenido informativo (estático) sin necesidad de conocimientos técnicos, permitiendo actualizaciones rápidas y eficientes.
- La aplicación cuenta con una interfaz intuitiva y amigable que sigue las pautas de diseño de la marca.
- Se integró exitosamente el *backend* del sistema de búsqueda de productos financieros basado en Inteligencia Artificial, proporcionando recomendaciones personalizadas a partir de la información del usuario.
- El rendimiento y tiempos de carga de la aplicación se optimizaron resultando en una experiencia fluida para el usuario.

- La aplicación web alcanzó un puntaje alto en *SEO* mediante la generación de páginas estáticas, implementación de buenas prácticas de accesibilidad y optimización de metaetiquetas, lo que provoca que se encuentre entre los primeros resultados de los motores de búsqueda.
- Se logró un eficiente proceso de desarrollo mediante la creación de componentes reutilizables y la implementación de una arquitectura modular.
- Se utilizó *Notion* para la documentación y *Scrum* como metodología ágil, facilitando la organización y seguimiento del proyecto.
- Se implementó un flujo de trabajo profesional utilizando *GitHub*, incluyendo control de versiones, ramas de desarrollo e integración continua (CI).
- Se estableció una conexión robusta con *Firebase* para el almacenamiento de datos y ejecución de código en la nube, garantizando escalabilidad y seguridad.
- Se implementó un sistema de *webhooks* para actualizaciones automáticas del sitio web, permitiendo mantener el contenido siempre actualizado.
- Se implemento una sección de *blog* configurada para tener actualizaciones constantes y optimizada para compartir en redes sociales con las meta etiquetas adecuadas.
- Se implementó *Google Analytics* para el seguimiento detallado del comportamiento de los usuarios, permitiendo analizar patrones de uso, tasas de conversión y métricas clave para la optimización continua de la plataforma.

El desarrollo de la versión *MVP* del proyecto se completó en un periodo de 5 meses, durante los cuales asumí la responsabilidad de liderar tanto el diseño como el desarrollo web de la aplicación. Este tiempo permitió implementar todas las funcionalidades esenciales planificadas, establecer una base sólida para la arquitectura del sistema, y realizar las pruebas necesarias para garantizar la calidad del producto final. La fase de desarrollo incluyó la creación de prototipos, múltiples iteraciones de diseño, y la implementación progresiva de las características principales de la plataforma.

Además, a lo largo del desarrollo, se mantuvo una comunicación constante con los *stakeholders*, lo que permitió realizar ajustes y mejoras basados en retroalimentación real. La implementación de un proceso iterativo de desarrollo facilitó la adaptación a cambios en los requerimientos y la incorporación de nuevas funcionalidades según las necesidades identificadas. Este enfoque ágil y colaborativo fue fundamental para asegurar que el producto final cumpliera con las expectativas y necesidades de los usuarios objetivo.

Capítulo 4

Conclusiones

Se logró cumplir con el objetivo de la aplicación, terminando en tiempo y forma con la versión inicial de la aplicación que proporciona un servicio de valor para los usuarios finales. La plataforma cumple con los requerimientos establecidos inicialmente, demostrando la viabilidad de crear una solución tecnológica que democratice el acceso a asesoría financiera de calidad ofreciendo una experiencia fluida y profesional que facilita la búsqueda y contratación de productos financieros. El desarrollo se realizó siguiendo las mejores prácticas de la industria, lo que resultó en un producto robusto y escalable.

Este desarrollo fue un gran reto para mi, ya que era la primera vez que lideraba un equipo profesionalmente lo que me permitió desarrollar habilidades importantes de liderazgo y gestión de proyectos. Aprendí a manejar efectivamente las expectativas de los *stakeholders*, coordinar tareas y recursos, y mantener al equipo motivado y enfocado en los objetivos. Esta experiencia también me ayudó a mejorar mis habilidades de comunicación y toma de decisiones bajo presión.

Las principales conclusiones que destaco del proyecto son:

- La elección de *Next.js* como *framework* principal demostró ser acertada, proporcionando una base sólida para el desarrollo y facilitando la implementación de características avanzadas.
- La arquitectura basada en servicios (*Firebase, Sanity, Vercel*) facilitó un desarrollo ágil y simplificó la implementación y despliegue de la aplicación, mientras mantuvo los costos de mantenimiento optimizados.
- La metodología *Scrum* adaptada a las necesidades del proyecto facilitó una gestión efectiva y permitió mantener un ritmo de desarrollo constante.
- La aplicación logra mantener un buen nivel de participación de los usuarios mediante blogs con contenido financiero actualizado y relevante.
- La plataforma logra simplificar el proceso de búsqueda, aprendizaje y selección de productos financieros mediante los apartados informativos de la misma.

- El proceso de búsqueda atrae efectivamente a los usuarios hacia los servicios de contratación y asesoría de nuestra plataforma, lo que permite la monetización del sistema.
- El sistema de recomendación basado en IA proporciona sugerencias personalizadas que ayudan a los usuarios a tomar decisiones informadas.
- La implementación del *CMS* permite a administradores no especializados en el desarrollo de *software* mantener el contenido actualizado y relevante para los usuarios.
- La implementación de *Google Analytics* proporciona información valiosa sobre el comportamiento de los usuarios, lo que permite optimizar la plataforma y obtener conclusiones basadas en datos reales de uso.

Sin embargo, durante el desarrollo se identificaron áreas que podrían mejorar y funciones que quedaron pendientes para futuras iteraciones, algunas de ellas son:

- Implementación de pruebas unitarias más exhaustivas para garantizar la calidad del código.
- Desarrollo de una aplicación móvil nativa para mejorar la experiencia en dispositivos móviles.
- Expansión de las capacidades del sistema de IA para incluir más tipos de productos financieros y análisis más detallados, además de optimizarlo para funcionar de una forma más rápida.
- Implementación de un sistema de mensajería instantánea entre usuarios y asesores, incluyendo notificaciones push en tiempo real para mantener informados a los usuarios sobre actualizaciones importantes, nuevas oportunidades de inversión, cambios en el estado de sus solicitudes y recordatorios personalizados para optimizar su experiencia en la plataforma.
- Integración con más proveedores de productos financieros para expandir la oferta de productos disponibles en la plataforma.
- Implementación de un sistema de newsletter a los correos electrónicos para mantener a los usuarios informados sobre nuevos productos financieros, contenido educativo y actualizaciones de la plataforma, incluyendo segmentación personalizada basada en intereses y comportamiento del usuario.
- Implementación de un sistema de historial detallado que permita a los usuarios y asesores ver el registro completo de las búsquedas de productos realizados con anterioridad, permitiendo al usuario contratar resultados obtenidos con anterioridad.
- Implementación de inicio de sesión con redes sociales (*Google, Facebook, Apple*) para simplificar el proceso de registro y acceso a la plataforma, reduciendo la fricción en la experiencia del usuario y aumentando las tasas de conversión.

Finalmente, este proyecto me permitió aplicar y demostrar los conocimientos adquiridos durante mi formación como Ingeniero en Computación, especialmente en el área de Ingeniería de Software. A lo largo del desarrollo, implementé conceptos fundamentales como:

- El ciclo de vida completo del desarrollo de *software*, desde la conceptualización y análisis de requerimientos hasta la implementación y mantenimiento del sistema.
- Metodologías ágiles y gestión de proyectos que fueron parte fundamental de mi formación académica.
- Conocimientos en bases de datos, seguridad informática y desarrollo web adquiridos durante la carrera.
- Habilidades de análisis y resolución de problemas desarrolladas a lo largo de diferentes cursos de la ingeniería.
- Conceptos de desarrollo de aplicaciones web y negocios electrónicos, incluyendo arquitectura de sistemas web, estrategias de monetización y análisis de métricas de negocio.

Además de poder demostrar conocimientos adquiridos mediante cursos externos —por ejemplo, mi formación en diseño de experiencia de usuario e interfaces de usuario— me permitió aplicar principios de diseño centrado en el usuario y optimizar la experiencia, los cuales fueron esenciales para crear una interfaz intuitiva y accesible.

Este proyecto no solo representa la culminación de mi formación académica, sino también demuestra mi capacidad para aplicar estos conocimientos en un entorno profesional real, creando una solución tecnológica que aporta valor tangible a sus usuarios. La combinación de fundamentos teóricos con experiencia práctica me permitió desarrollar una plataforma robusta y escalable que cumple con los estándares de la industria.

Las bases sólidas en ingeniería de software adquiridas durante mi carrera fueron fundamentales para tomar decisiones técnicas informadas, implementar buenas prácticas de desarrollo y mantener una visión holística del proyecto. Esta experiencia ha validado la importancia de una formación integral en ingeniería en computación y ha demostrado cómo los conocimientos académicos se traducen en soluciones prácticas y efectivas en el mundo real.

Glosario

API Acrónimo de Application Programming Interface o Interfaz de Programación de Aplicaciones en español, es una interfaz que permite la comunicación entre aplicaciones de software.. 4

API REST Interfaz de Programación de Aplicaciones que utiliza el protocolo HTTP para realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) en un servidor.. 22

Aprendizaje Profundo Tipo de Aprendizaje Automático enfocado en aprender de los datos similar a los humanos utilizando redes neuronales con múltiples capas de procesamiento para reconocer patrones.. 1

Aprendizaje Automático Rama de la Inteligencia Artificial donde un sistema aprende automáticamente a partir de la experiencia.. 1

Backlog Lista de tareas pendientes de un proyecto.. 6

Blog Sitio web que publica artículos sobre diversos temas de manera periódica.. 13

Chatbot La palabra proviene de la combinación de las palabras en inglés chat y bot, la cual hace referencia a una Inteligencia Artificial que responde una conversación con personas a través de mensajes de texto o voz.. 1

Cifrado Proceso de convertir información mediante algoritmos en un formato ilegible para proteger la confidencialidad de los datos.. 42

Cookie Pequeño archivo de datos que los sitios web almacenan en el navegador del usuario con información de la sesión del usuario.. 41

CSS Modules Sistema de formato de archivos de estilos CSS que permite encapsularlos en un componente en lugar de afectar a toda la aplicación.. 37

Deepfake Técnica de uso de Inteligencia Artificial para crear contenido audiovisual falso con la apariencia de un ser real.. 2

DNS Acrónimo de Domain Name System o Sistema de Nombres de Dominio en español, el cual permite traducir nombres de dominio de las URL en direcciones IP.. 44

Dominio Nombre único que identifica a una IP en internet.. 44

Endpoint Punto al que se realiza una petición, usualmente siendo URLs para consumir una API.. 36

Enrutamiento Proceso de dirigir al usuario a la vista deseada utilizando direcciones web (URLs).. 35

Framework Entorno de trabajo que proporciona una estructura y bibliotecas de programación para facilitar el desarrollo de aplicaciones.. 4

Fullstack Desarrollador que tiene habilidades tanto en el desarrollo de la parte del cliente como en la del servidor, es decir del stack tecnológico completo.. 3

INE Acrónimo de Instituto Nacional Electoral, el INE es el organismo encargado de organizar las elecciones en México. Además, es responsable de emitir la credencial para votar, comúnmente conocida como "INE".. 31

Inteligencia Artificial Generativa Uso de la Inteligencia Artificial para crear contenido nuevo y original como texto, imágenes, entre otros.. 1

Inteligencia Artificial Conjunto de tecnologías para simular la inteligencia humana en las maquinas.. VI

Intellisense Función creada por Microsoft disponible en los editores de código que proporciona sugerencias y completado automático de código.. 34

JSON Acrónimo de JavaScript Object Notation o Notación de Objetos de JavaScript en español, es un formato de texto que permite intercambiar datos de manera ligera y fácil de leer, incluso entre diferentes lenguajes de programación.. 36

Metaetiquetas Etiquetas HTML que proporcionan información sobre la página web para ser interpretada por navegadores web.. 45

Middleware Software que actúa como intermediario entre dos aplicaciones o sistemas para facilitar la gestión de datos.. 36

NoSQL Base de datos que no utiliza el modelo relacional ni el lenguaje SQL para almacenar y gestionar la información.. 4

Pair Programming Práctica de programación en la que dos (o en ocasiones más) programadores trabajan a la par resolviendo un modulo de una aplicación a la vez.. 4

Procesamiento de Lenguaje Natural (PNL) Rama de la Inteligencia Artificial enfocada en entender y generar lenguaje natural, este es muy utilizado en asistentes inteligentes para generar una interacción sencilla entre los humanos y las máquinas.. 1

Productos Financieros Instrumento ofrecido por una institución financiera con el objetivo de ahorrar, invertir o gestionar el dinero.. vi

Pull Request (PR) Solicitud de cambios en un repositorio Git de código fuente.. 40

RFC Acrónimo de Registro Federal de Contribuyentes, el cual es un identificador único que el Servicio de Administración Tributaria (SAT) asigna a las personas físicas y morales en México.. 31

Scrum Metodología ágil de desarrollo que se basa principalmente en la organizar el trabajo en ciclos llamados Sprints.. 6

Serverless Modelo de computación en la nube en el que el desarrollador no gestiona los servidores; en su lugar, el proveedor de servicios se encarga de administrar la infraestructura necesaria para ejecutar código del lado del servidor.. 35

Singleton Patrón de diseño que garantiza que una clase o tipo tenga una única instancia.. 39

SQL Acrónimo de Structured Query Language o Lenguaje de Consulta Estructurado en español, este lenguaje se utiliza para gestionar y manipular bases de datos relacionales.. 4

Stack Tecnológico Conjunto de tecnologías, herramientas y lenguajes de programación utilizados en el desarrollo de un proyecto.. 3

Stakeholders Personas o grupos interesados en el desarrollo de un proyecto.. 4

Startup Empresa emergente que busca un modelo de negocio escalable.. 8

Tiempo de Ejecución Período durante el cual un programa de computadora se ejecuta.. 21

Token Unidad de información que representa de forma segura datos sensibles.. 44

Trade-off Situación en la que se debe elegir entre dos opciones que son mutuamente excluyentes, es decir, se obtiene una ventaja sacrificando algo.. 39

UX/UI Acrónimo en inglés que reúne dos conceptos clave en el diseño de productos digitales: la Experiencia de Usuario (User Experience, UX) y el Diseño de Interfaz de Usuario (User Interface, UI).. 4

Visión Computacional Campo de la Inteligencia Artificial que interpreta y entiende de entradas visuales (imágenes).. 1

WebHook Método que permite a una aplicación enviar información en tiempo real a otra aplicación a través de una URL específica, generalmente en respuesta a un evento.. 39

Web Scraping Técnica de recopilación de datos que consiste en extraer información de sitios web mediante procesos automatizados.. 22

Bibliografía

Al Prestamo. (2024). *Preguntas Frecuentes Alprestamo*. Consultado el 22 de octubre de 2024, desde https://alprestamo.mx/faqs

Baláz Orbán, D. d. O. (2022). Next.js 13. Consultado el 7 de enero de 2025, desde https://nextjs.org/blog/next-13

Beautiful Soup : ¿cómo aprender a hacer web scraping en Python? (2022). Consultado el 14 de noviembre de 2024, desde https://www.selenium.dev/about/

Cataldi, Z., Lage, F., Pessacq, R., & García Martínez, R. (1999). Ingeniería de software educativo. *Proceedings del V Congreso Internacional de Ingeniería Informática*, 185-199.

da Silva, D. (2021). *Qué es MVP: concepto, importancia y ejemplos. — zendesk.com.mx*. Consultado el 11 de mayo de 2021, desde https://www.zendesk.com.mx/blog/que-es-mvp/

de Mendieta, A. (2022). Funciones del Director de Tecnología o CTO en la actualidad. Consultado el 30 de enero de 2025, desde https://openwebinars.net/blog/funciones-del-director-de-tecnologia-en-la-actualidad/

Desarrollo de Soluciones con Inteligencia Artificial México. (2023). Consultado el 7 de octubre de 2024, desde https://brita.mx/brita-inteligencia-artificial-mexico/

Destácame. (2024a). *Destácame*. Consultado el 19 de octubre de 2024, desde https://www.destacame.com.mx/

Destácame. (2024b). *Destácame Pro*. Consultado el 19 de octubre de 2024, desde https://www.destacame.com.mx/pro?

educaopen. (2024). *Librería o biblioteca en programación: para que sirven y tipos*. Consultado el 5 de noviembre de 2024, desde https://www.educaopen.com/digital-lab/metaterminos/l/libreria

Fernández, L. F. (2006). Arquitectura de software. *Software Guru*, *2*(3), 40-45.

Figma. (2024). *Herramienta de diseño gratuita para sitios web, diseño de productos y más | Figma*. Consultado el 21 de noviembre de 2024, desde https://www.figma.com/es-la/design/

GBM. (2024). *La importancia de la asesoría financiera en México*. Consultado el 17 de octubre de 2024, desde https://gbm.com/media/the-academy/la-importancia-de-la-asesoria-financiera-en-mexico/

GitHub. (2024). *Acerca de GitHub y Git - Documentación de GitHub*. Consultado el 14 de noviembre de 2024, desde https://docs.github.com/es/get-started/start-your-journey/about-github-and-git

Google. (s.f.). *Solutions for app development challenges | Firebase*. Consultado el 9 de enero de 2025, desde https://firebase.google.com/solutions

Google. (2024). *Firebase | Google's Mobile and Web app development Platform*. Consultado el 14 de noviembre de 2024, desde https://firebase.google.com/?hl=es-419

Hack a Boss. (2024). *Lead Developer: qué es y qué hace*. Consultado el 10 de octubre de 2024, desde https://www.hackaboss.com/blog/lead-developer

Hireline. (2024). *Desarrollador Fullstack*. Consultado el 10 de octubre de 2024, desde https://hireline.io/mx/enciclopedia-de-perfiles-de-tecnologia/desarrollador-fullstack

iKiwi. (2024). *iKiwi*. Consultado el 20 de octubre de 2024, desde https://ikiwi.com.mx/

Instituto Técnico Superior, T. (2024). *Conocé los lenguajes de backend más demandados*. Consultado el 5 de noviembre de 2024, desde https://teclab.edu.ar/tecnologia-y-desarrollo/lenguajes-de-backend-mas-demandados/

Lounn. (2022). *Lounn: La plataforma de financiamento empresarial*. Consultado el 22 de octubre de 2024, desde https://lounn.mx/

Martins, J. (2024). *Scrum: conceptos clave y cómo se aplica en la gestión de proyectos*. Consultado el 4 de enero de 2025, desde https://asana.com/es/resources/what-is-scrum

Microsoft. (2024). *TypeScript: JavaScript with sintax for types*. Consultado el 14 de noviembre de 2024, desde https://www.typescriptlang.org/

Mousinho, A. (2020). SEO: la guía completa para que conquistes la cima de Google en el 2022. Consultado el 7 de enero de 2025, desde https://rockcontent.com/es/blog/que-es-seo

Mozilla MDN. (2024a). *CSS*. Consultado el 4 de noviembre de 2024, desde https://developer.mozilla.org/es/docs/Web/CSS

Mozilla MDN. (2024b). *HTML: Lenguaje de etiquetas de hipertexto*. Consultado el 4 de noviembre de 2024, desde https://developer.mozilla.org/es/docs/Web/HTML

Mozilla MDN. (2024c). *JavaScript*. Consultado el 4 de noviembre de 2024, desde https://developer.mozilla.org/es/docs/Web/JavaScript

Muente, G. (2020). Guía completa del Framework: qué es, cuáles tipos existen y por qué es importante en Internet. *Rock Content*. Consultado el 5 de noviembre de 2024, desde https://rockcontent.com/es/blog/framework/

Notion. (2024). *Tu espacio de trabajo que conecta tus wikis, documentos y proyectos | Notion*. Consultado el 21 de noviembre de 2024, desde https://www.notion.com/es

NumFOCUS, Inc. (2024). *pandas - Python Data Analysis Library*. Consultado el 14 de noviembre de 2024, desde https://pandas.pydata.org/about/

NumPy team. (2024). *NumPy - About Us*. Consultado el 14 de noviembre de 2024, desde https://numpy.org/about/

Pérez Ibarra, S. G., Quispe, J. R., Mullicundo, F. F., & Lamas, D. A. (2021). Herramientas y tecnologías para el desarrollo web desde el FrontEnd al BackEnd.

XXIII Workshop de Investigadores en Ciencias de la Computación (WICC 2021, Chilecito, La Rioja).

Qiip. (2023). *Qiip*. Consultado el 22 de octubre de 2024, desde https://qiip.com.mx/es/

Ramírez-Acosta, K. (2017). Interfaz y experiencia de usuario: parámetros importantes para un diseño efectivo. *Revista tecnología en marcha*, *30*, 49-54.

Ricardo Quiroga. (2023). *Por elecciones abundarán deepfakes políticos, advierte Brita IA*. Consultado el 9 de octubre de 2024, desde https://www.eleconomista.com.mx/tecnologia/Por-elecciones-abundaran-deepfakes-politicos-advierte-Brita-IA-20230924-0059.html

Rouhiainen, L. (2018). Inteligencia artificial. *Madrid: Alienta Editorial*, 20-21.

Sanity. (s.f.). *GROQ-powered webhooks | Content Lake*. Consultado el 9 de enero de 2025, desde https://www.sanity.io/docs/webhooks

Sanity. (2024a). *The Composable Content Cloud | Sanity.io*. Consultado el 7 de enero de 2025, desde https://www.sanity.io/

Sanity. (2024b). *Query Language (GROQ)*. Consultado el 8 de enero de 2025, desde https://www.sanity.io/docs/groq

Sanity. (2025). *Repositorio Open Source de Sanity Studio*. Consultado el 9 de enero de 2025, desde https://github.com/sanity-io/sanity

Schwaber, K., & Sutherland, J. (2013). La guía de Scrum. *Scrumguides. Org*, *1*, 21.

Software Freedom Conservancy. (2024). *About Selenium | Selenium*. Consultado el 14 de noviembre de 2024, desde https://www.selenium.dev/about/

Sommerville, I. (2005). *Ingeniería del software*. Pearson.

UNIR La Universidad en Internet. (s.f.). *¿Qué es el CEO de una empresa? Estas son sus habilidades y funciones*. Consultado el 30 de enero de 2025, desde https://colombia.unir.net/actualidad-unir/que-es-ceo-empresa/

Vercel. (2024a). *Next.js by Vercel*. Consultado el 14 de noviembre de 2024, desde https://nextjs.org/

Vercel. (2024b). *Vercel: Build and deploy the best web experiences with the Frontend Cloud*. Consultado el 14 de noviembre de 2024, desde https://vercel.com/

Zenfi. (2024). *Plataforma de salud financiera | Zenfi*. Consultado el 22 de octubre de 2024, desde https://www.zenfi.mx/