



Análisis, diseño e implementación de la plataforma Endabank

Juan Esteban Gutiérrez Zuluaga

Sebastián Gómez Ramírez

Informe de práctica presentado para optar al título de Ingeniero de Sistemas

Asesores

Asesora interna: Deisy Yuried Loaiza Berrío, Ingeniera de Sistemas

Asesor externo: Jairo Alonso Valderrama Gallego, Ingeniero de Sistemas

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería de Sistemas
Medellín, Antioquia, Colombia

2023

Referencia

- [1] J. E. Gutiérrez Zuluaga y S. Gómez Ramírez, “Análisis, diseño e implementación de la plataforma Endabank”, Semestre de industria, Ingeniería de Sistemas, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2023.

Estilo IEEE (2020)



Centro de Documentación Ingeniería (CENDOI)

Repositorio Institucional: <http://bibliotecadigital.udea.edu.co>

Universidad de Antioquia - www.udea.edu.co

Rector: John Jairo Arboleda Céspedes

Decano/Director: Julio César Saldarriaga Molina

Jefe departamento: Diego José Luis Botía Valderrama

El contenido de esta obra corresponde al derecho de expresión de los autores y no compromete el pensamiento institucional de la Universidad de Antioquia ni desata su responsabilidad frente a terceros. Los autores asumen la responsabilidad por los derechos de autor y conexos.

Agradecimientos

A la Universidad de Antioquia, por brindarnos las herramientas necesarias para afrontar retos de la industria, hemos sido testigos del compromiso y la excelencia que caracterizan a la UdeA, y estamos profundamente orgullosos de haber formado parte de esta comunidad académica.

A Endava S.A.S, por proveernos un constante acompañamiento durante todo el proceso de prácticas profesionales. Siempre contábamos con personas que con su experiencia, consejos y retroalimentación constructiva fueron fundamentales para el desarrollo del proyecto.

TABLA DE CONTENIDO

RESUMEN	8
ABSTRACT	9
I. INTRODUCCIÓN	10
II. OBJETIVOS	11
A. Objetivo general	11
B. Objetivos específicos	11
III. MARCO TEÓRICO	12
A. Marco Contextual Bancario: Fundamentos	12
B. Tecnologías	14
IV. METODOLOGÍA	18
A. Marco de trabajo empleado	18
B. Ejecución del proyecto con el fin de lograr los objetivos planteados	20
1) Capacitación en la vertical de pagos	20
2) Visioning	21
3) Mapeo de historias de usuario (User Story Mapping)	21
4) Priorización de historias de usuario	23
5) Estimación de HU	24
6) Selección de tecnologías	24
7) Flujo de usuario	25
8) Wireframes del aplicativo	25
9) Realizar guía de estilos del aplicativo	25
10) Prototipos de alta fidelidad del aplicativo	26
11) Desarrollo de módulos de la plataforma web	29
12) Definición de sistema de gestión de base de datos	30
13) Arquitectura de la Rest API	30
14) Desarrollo de pruebas unitarias	32
15) Ejecución de pruebas manuales	32
16) Demos con el PO	33
17) Validación de que el Frontend del aplicativo corresponda con el diseño propuesto	33
F. Funcionalidades del aplicativo	44
1) Flujo de usuario: Agrupación de algunas funcionalidades de clientes de Endabank.	44
a) Inicio de sesión	44
b) Registro	45
c) Home del usuario	46
d) Creación de productos	47
e) Creación de producto personal	48
f) Creación de producto empresarial	49
g) Lista de productos	50

h) Detalles de un producto personal	51
i) Detalles de un producto empresarial	52
j) Servicio de depósito	53
k) Servicio de transacción	54
l) Resultado de una transacción	55
m) Solicitudes de producto empresarial del usuario	56
n) Detalles de una solicitud de producto empresarial del usuario	57
o) Conexión con e-commerce	58
p) Correo electrónico de notificación de una transacción	61
2) Flujo de administrador: Agrupación de algunas funcionalidades de administradores de Endabank.	62
a) Home del administrador	62
b) Lista de solicitudes del administrador	63
c) Detalles de una solicitud de producto empresarial, vista del administrador	64
d) Lista de usuarios	65
e) Detalles del producto empresarial de un usuario, vista de administrador	67
G. Infraestructura de la plataforma web	68
H. Análisis estático del código con SonarQube	70
I. Métricas de QA	71
VI. ANÁLISIS	72
VII. CONCLUSIONES	74
REFERENCIAS	77

LISTA DE FIGURAS

Fig. 1. Backlog de Endabank en el Sprint 8	19
Fig. 2. Ciclo del marco de trabajo SCRUM	21
Fig. 3. Muestra de visioning con el BA y el equipo de trabajo	22
Fig. 4. Mapeo de historias de usuario	23
Fig. 5. Comparación entre Vertical Slicing y Horizontal Slicing	24
Fig. 6. Tipografía, colores e íconos en Figma	28
Fig. 7. Atomic Design en Figma	29
Fig. 8. Prototipos de alta fidelidad en Figma	30
Fig. 9. Package by layer vs Package by feature	32
Fig. 10. Package by feature/Component	32
Fig. 11. Ejemplo de casos de prueba en una HU	33
Fig. 12. Diagrama entidad-relación en alto nivel	35
Fig. 13. Muestra de endpoints para la gestión de usuarios con OpenAPI 3	39
Fig. 14. Muestra de payload empleado para los errores	38
Fig. 15. Estructura de backend	41
Fig. 16. Endpoints de servicio de notificaciones de transacciones	42
Fig. 17. Estructura del frontend	43
Fig. 18. Pantalla de inicio de sesión	45
Fig. 19. Pantalla de registro	46
Fig. 20. Pantalla de homepage	47
Fig. 21. Pantalla de creación de productos	48
Fig. 22. Pantalla de creación de producto personal	49
Fig. 23. Pantalla de creación de producto empresarial	50
Fig. 24. Pantalla de lista de productos	51
Fig. 25. Pantalla de detalles de un producto personal	52
Fig. 26. Pantalla de detalles de un producto empresarial	53
Fig. 27. Pantalla de servicio de depósito	54
Fig. 28. Pantalla de servicio de transacción	55
Fig. 29. Pantalla de resultado de una transacción	56

Fig. 30. Pantalla de solicitudes de producto empresarial del usuario	57
Fig. 31. Pantalla de detalles de una solicitud de producto empresarial del usuario	58
Fig. 32. Pantalla de conexión con e-commerce	59
Fig. 33. Pantalla de pago, paso 1	60
Fig. 34. Pantalla de pago, paso 2	61
Fig. 35. Pantalla de pago, paso 3	62
Fig. 36. Notificación de pago desde e-commerce	63
Fig. 37. Pantalla de homepage del administrador	64
Fig. 38. Pantalla de lista de solicitudes del administrador	65
Fig. 39. Pantalla de detalles de una solicitud de producto empresarial, vista del administrador	66
Fig. 40. Pantalla de lista de usuarios	67
Fig. 41. Pantalla de detalles del producto personal de un usuario, vista de administrador	68
Fig. 42. Pantalla de detalles del producto empresarial de un usuario, vista de administrador	69
Fig. 43. Infraestructura de Endabank	70
Fig. 44. Métricas e indicadores con SonarQube en el backend	72
Fig. 45. Métricas e indicadores con SonarQube en el frontend	73
Fig. 46. Ejemplo de métricas de testing (Sprint 7)	73

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

ISO	International Organization for Standardization
MVP	Minimum Viable Product
ORM	Object Relational Mapping
AAA	Arrange, Act and Assert
CI	Continuous Integration
CD	Continuous Delivery
UdeA	Universidad de Antioquia
HU	Historia de usuario
PO	Product Owner
QA	Quality Assurance
BA	Business Analyst

RESUMEN

La multinacional de desarrollo de software Endava SAS llevó a cabo el proyecto interno "Endabank" con el objetivo de diseñar e implementar un aplicativo bancario. Este permite funcionalidades como la apertura de productos bancarios y transacciones entre cuentas, así como la conexión con plataformas de comercio electrónico para funcionar como método de pago. Para garantizar un desarrollo eficiente, se empleó el marco de trabajo Scrum y se formó un equipo con roles claramente definidos, incluyendo Scrum Master y Product Owner. El desarrollo del proyecto fue un gran desafío, debido a la incertidumbre técnica y de negocio que se tenía inicialmente, pero gracias a las capacitaciones y la cultura de aprendizaje continuo que existe dentro de Endava, se logró completar satisfactoriamente el MVP (Producto Mínimo Viable) de Endabank, ofreciendo una versión inicial y funcional del sistema. Además, se identificaron oportunidades de mejora para futuras versiones, incluyendo la adición de nuevas funcionalidades y la mejora de implementaciones a nivel de código.

Palabras clave — Sistema bancario, E-commerce, Scrum, Desarrollo de software, Sistemas de información

ABSTRACT

The multinational software development company Endava SAS carried out the internal project "Endabank" with the objective of designing and implementing a banking application. This allows functionalities such as the opening of banking products and transactions between accounts, as well as the connection with e-commerce platforms to function as a payment method. To ensure an efficient development, the Scrum framework was used and a team was formed with clearly defined roles, including Scrum Master and Product Owner. The development of the project was a great challenge, due to the technical and business uncertainty that initially existed, but thanks to the training and the continuous learning culture that exists within Endava, the MVP (Minimum Viable Product) of Endabank was successfully completed, offering an initial and functional version of the system. In addition, improvement opportunities were identified for future versions, including the addition of new functionalities and the improvement of code-level implementations.

Keywords — **Banking system, E-commerce, Scrum, Software development, Information Systems**

I. INTRODUCCIÓN

La multinacional de desarrollo de software Endava SAS cuenta con una amplia gama de verticales, entre ellas la vertical de Payments, con la cual se busca diseñar, construir y operar las soluciones de pago que ayudan a los negocios a seguir siendo competitivos, mejorar la velocidad de comercialización y generar lealtad a través de una experiencia de cliente efectiva. Como parte del aprendizaje de nuevos conceptos y la contribución a soluciones innovadoras de la vertical de pagos, la compañía crea la iniciativa “Endabank”, un proyecto interno (bench) que tiene como objetivo que sean colaboradores o practicantes quienes diseñen e implementen un sistema asemejado a un sistema bancario, que permitiera funcionalidades como la apertura de productos bancarios y transacciones entre cuentas. Además, se buscaba permitir la conexión del aplicativo bancario con cualquier plataforma de e-commerce y de esta forma funcionar como un método de pago por lo que pueden optar los usuarios de “Endabank”.

Como base teórica fundamental del proyecto, se tuvieron en cuenta los modelos de tres y cuatro partes utilizados en la vertical de pagos. Estos modelos proporcionan un enfoque integral para habilitar transacciones que no involucran dinero físico. Específicamente, se empleó el conocido Third-Party Model, el cual representa el ciclo de procesamiento de pagos y las entidades clave involucradas.

Para lograr un desarrollo eficiente y ágil del proyecto "Endabank", se utilizó el marco de trabajo Scrum y se formó un equipo con roles claramente definidos, como el Scrum Master y el Product Owner, para garantizar una colaboración efectiva, una entrega continua de valor y de esta forma alcanzar cada uno de los objetivos específicos planteados.

Como resultado, se logra completar satisfactoriamente el MVP planteado, el cual es una versión inicial y funcional de Endabank y además, se logró sentar las bases para posibles mejoras en futuras versiones del aplicativo, las cuales incluyen la adición de nuevas funcionalidades y la mejora de algunas implementaciones hechas a nivel de código.

II. OBJETIVOS

A. Objetivo general

Construir el aplicativo web “Endabank” que permita la transferencia de dinero entre usuarios registrados en el sistema bancario, además de posibilitar la integración con cualquier plataforma de comercio electrónico.

B. Objetivos específicos

- Levantar los requerimientos de “Endabank” y proponer el diseño de la plataforma web de modo que sea intuitiva para el usuario.
- Desarrollar el frontend y el backend del aplicativo.
- Realizar pruebas unitarias del código.
- Validar el funcionamiento del sistema de forma que este cumpla con lo esperado.

III. MARCO TEÓRICO

A. Marco Contextual Bancario: Fundamentos

El surgimiento del comercio electrónico, tal como menciona [1], ha creado nuevas necesidades financieras que en muchos casos no pueden ser satisfechas por los sistemas de pago tradicionales y como consecuencia la industria de los pagos ha evolucionado a medida que se han ido introduciendo nuevas tecnologías y formas de pago. El término pago electrónico incluye cualquier tipo de pago a negocios, bancos o servicios públicos de ciudadanos o negocios, los cuales se ejecutan a través de una red de telecomunicaciones o redes electrónicas que utilizan tecnología moderna y permite que el pago sea ejecutado por el comprador mismo, sin necesidad de presencia física y sin el uso de dinero en efectivo. [1]

Para permitir pagos que no involucren dinero, múltiples agentes son requeridos. Es por esto que el ecosistema de los pagos electrónicos está soportado en un modelo, siendo el más comúnmente usado el conocido como Four-Party Model, el cual representa el ciclo de proceso de los pagos y las entidades involucradas, que son básicamente 4: el portador de la tarjeta / cliente (cardholder), el vendedor (merchant), el banco del vendedor (acquirer) y el banco del portador de la tarjeta / cliente (issuer).

- **Cardholder:** la persona o entidad que tiene un acuerdo con un emisor para obtener una tarjeta de pago. Mediante este acuerdo, se autoriza al titular a utilizar la tarjeta para los fines previstos. [2]
- **Merchant:** Es la parte de venta y aquí es donde terminará el pago después del procesamiento.
- **Acquirer bank:** El banco o institución que tiene una relación comercial contractual con el vendedor y factura los datos de transacción de la tarjeta enviados por el vendedor al emisor respectivo a través del esquema de pago correspondiente. [2]
- **Issuer:** Banco o institución que emite tarjetas a sus clientes (titulares de tarjetas). Gestiona las cuentas de tarjetas de sus clientes, autoriza transacciones con tarjetas y garantiza la liquidación de pagos de transacciones con tarjetas válidas para el adquirente. [2]

El flujo que sucede al realizarse una compra es el siguiente, tal como lo describe [3]: “en una transacción, (1) el cliente pasa la tarjeta y autentica el pago, tras lo cual (2) el comercio envía la transacción al banco del vendedor, (3) que a su vez procesa la transacción transmitiéndola a la red de pagos pertinente (por ejemplo, Visa o Mastercard), y la red, que también establece las normas generales del sistema de pagos, realiza comprobaciones automáticas de fraude y remite la transacción al banco emisor para su autorización. Si el banco emisor autoriza la transacción, (4) realiza un cargo en la cuenta del cliente y (5) liquida el pago al banco del vendedor, menos una tasa de intercambio. Por último, (6) el banco del vendedor paga al vendedor, menos una comisión de descuento al vendedor, que cubre los costes de adquisición, incluido el intercambio, la depreciación del terminal, el riesgo, el servicio al vendedor, los gastos de explotación y un cierto margen de beneficio para el propio banco del vendedor.”

De igual forma, también existe un modelo de 3 partes, o Three-Party Model y su peculiaridad radica en que tanto el banco del cliente (issuer bank) y el banco del vendedor (acquirer bank) son la misma institución. Debido a esto, también hay ciertas diferencias a la hora de procesar un pago, siendo el procesamiento y validación de la transacción más fácil al tratarse de una *On-Us transaction*, que quiere decir que el banco realiza las dos partes de la transacción: el retiro y el depósito en las dos cuentas, evitándose el paso por la red de pagos. Para el desarrollo de este proyecto se ha elegido hacer uso del modelo de 3 partes, debido al alcance y complejidad del mismo.

En el ámbito de las instituciones financieras, el intercambio de información masiva tanto entre ellas como con sus clientes es fundamental. Sin embargo, para que estos intercambios sean efectivos, es necesario contar con un entendimiento común entre el remitente y el receptor sobre cómo interpretar la información transmitida. Para abordar esta necesidad, la industria bancaria ha desarrollado definiciones de mensajes, que consisten en formas estructuradas de organizar los datos que se desean intercambiar, abarcando tanto su sintaxis (estructura) como su semántica (significado). El objetivo principal de estas definiciones de mensajes es eliminar la intervención humana en la interpretación de los datos, permitiendo un intercambio eficiente y preciso.

Según lo mencionado en “ISO 20022 for dummies” [4], se observa una variedad de estándares en el sector financiero, como el SWIFT MT103 y el Fedwire Proprietary Message. Además, algunas entidades han desarrollado estándares personalizados para su uso interno o con

sus clientes. Sin embargo, esta proliferación de estándares de mensajería asociados a pagos plantea desafíos en la automatización de las cadenas de valor, especialmente considerando que los instrumentos financieros suelen abarcar múltiples áreas geográficas.

En 2004, se introdujo por primera vez ISO 20022 como una solución para abordar el desafío de los múltiples estándares. ISO 20022 se estableció como la metodología acordada utilizada por la industria financiera para crear estándares de mensajes coherentes en todos los procesos comerciales del sector. J.P. Morgan, una de las principales empresas financieras a nivel mundial, ha descrito ISO 20022 como una herramienta fundamental en este sentido [5]. La adopción de ISO 20022 ofrece varias ventajas significativas. En primer lugar, permite la fácil vinculación de mensajes a los procesos comerciales, lo que facilita la interoperabilidad y la comunicación eficiente entre las diferentes entidades financieras. Además, ISO 20022 fomenta la reutilización de componentes, lo que reduce la redundancia y promueve la eficiencia en el intercambio de información.

Una característica importante de ISO 20022 es su estructura bien definida, que proporciona una base sólida para la comunicación a través de esquemas XML y JSON, incluyendo el uso de API RESTful para soluciones de mensajería. Esto facilita la integración de ISO 20022 en diversos sistemas y tecnologías, promoviendo la estandarización y la automatización en los procesos comerciales de la industria financiera. En el caso específico de Endabank, se tiene como base una comunicación que cumple con los parámetros establecidos por ISO 20022.

B. Tecnologías

Para el desarrollo del MVP de Endabank, se emplean las tecnologías y herramientas descritas a continuación:

- **JavaScript:** es un lenguaje de programación basado en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (por ejemplo programación funcional). Ampliamente usado para controlar el comportamiento de las páginas web [6]

-
- **HTML:** HTML o HyperText Markup Language, es usado para definir la estructura y significado del contenido de una página web. Generalmente es utilizado con otras tecnologías como JavaScript y CSS. [7]
 - **CSS:** CSS o Cascading Style Sheets, es un lenguaje de estilos que se utiliza para describir la presentación y cómo deben ser renderizados los documentos HTML. [8]
 - **SASS:** Es un preprocesador de CSS. Es una herramienta utilizada para facilitar el desarrollo de los estilos debido a las diversas funcionalidades que provee, haciéndolo más potente que CSS [9].
 - **React:** es una biblioteca gratuita y open-source para interfaces de usuario basadas en componentes [10]. El objetivo de React es ayudar a construir aplicaciones que utilizan datos que cambian constantemente. Para el desarrollo del proyecto se utilizó la versión 18.2.0, se utilizaron paquetes como react-router-dom en su versión 6.8.2, debido a que en el proyecto se manejó la navegación en el lado del cliente; **axios**, el cual es un cliente HTTP basado en promesas que puede ejecutarse en el navegador y brinda la habilidad de hacer peticiones HTTP y manejar la transformación de los datos de petición y respuesta. Este último se utilizó para el consumo de todos los servicios expuestos por el aplicativo backend.
 - **npm:** por sus siglas Node Package Manager, es el sistema de gestión de paquetes por defecto para Node.js. Se utilizó para la instalación y manejo de dependencias existentes en el proyecto.
 - **JsDoc 3:** JSDoc 3 es un generador de documentación API para JavaScript, similar a Javadoc. Fue utilizado para documentar gran parte del código desarrollado.
 - **Vite:** es una herramienta de compilación cuyo objetivo es proporcionar una experiencia de desarrollo más rápida y ágil para los proyectos web modernos [11]. Se utilizó la versión 4.1.2 y se escogió esta tecnología debida su popularidad y a algunas ventajas que brinda como: server para desarrollo altamente optimizado que permite que los cambios realizados se vean reflejados rápidamente sin necesidad de recargar la página web, arquitectura escalable, compatibilidad inmediata con funcionalidades modernas y compatibilidad con preprocesadores como SASS.

- **Vitest:** es un framework de pruebas unitarias. Provee un set de funcionalidades que facilitan la escritura y ejecución de pruebas unitarias y permite la prueba de componentes de React [12]. Se utilizó la versión 0.29.2
- **Eslint:** herramienta de análisis de código estático utilizado para encontrar problemas rápidamente en el código JavaScript [13]. Eslint se puede configurar y definir diversas reglas, pero para el desarrollo del proyecto se utilizó específicamente la configuración de Airbnb, la cual contiene una serie de reglas pre-establecidas para JavaScript y React.
- **Java:** Lenguaje de programación de alto nivel y orientado a objetos muy utilizado en el desarrollo de aplicaciones empresariales. Es altamente tipado lo cual trae muchas ventajas en el proceso de desarrollo de software.
- **Spring Boot:** Es un framework de desarrollo basado en Spring que simplifica la creación, configuración y despliegue de aplicaciones Java. Este agrega características adicionales para facilitar la creación de aplicaciones autocontenidas, con incorporación de servidores web integrados y configuración automática basada en convenciones y anotaciones útiles, el proyecto puede ser inicializado fácilmente como se evidencia en [14]. Para el desarrollo del backend de Endabank se utiliza Spring Boot 3 con la versión 17 de Java y dependencias como Spring Security con JWT, Lombok y Spring Data JPA/Hibernate como ORM para la interacción con la base de datos.
- **Open API 3:** Especificación de lenguaje que documenta las API de manera clara y consistente. A través de esta herramienta se documentan todos los servicios del backend de Endabank.
- **JUnit/Mockito:** Librerías de Java para el desarrollo de pruebas unitarias. Por un lado, JUnit 5 es un framework de pruebas unitarias que provee un set de funcionalidades y anotaciones que facilitan la escritura y ejecución de pruebas unitarias [15]. Por otra parte, Mockito es un framework de Java que ayuda a la creación de mocks (objetos simulados) para la creación de pruebas unitarias con JUnit.

- **Jira:** Plataforma de gestión de proyectos que permite llevar un seguimiento de estos de una forma sencilla. Se utiliza comúnmente en SCRUM (marco de trabajo empleado en el desarrollo del MVP de Endabank.)
- **Confluence:** Software de colaboración empleado por Endava para la documentación de distintos procesos y proyectos de la compañía. Para el caso de Endabank, se realizó la documentación con esta herramienta.
- **Sonarqube:** Herramienta de revisión de código automática y autoadministrada que permite sistemáticamente la entrega de un código limpio, esto se logra a través de un análisis estático del código por medio de métricas [16]. Endava define sus propias reglas para el tratamiento de los code smells y deuda técnica.
- **Azure:** Plataforma de computación en la nube creada por Microsoft. Toda la infraestructura de Endabank está sobre este proveedor de servicios de nube.
- **PostgreSQL:** Sistema de gestión de base de datos relacional usado para el almacenamiento de datos en Endabank.
- **Bitbucket:** Plataforma de almacenamiento de repositorios de control de versiones como Git empleada en Endabank.
- **Jenkins:** Servidor de automatización de código abierto que permite llevar a cabo el proceso de CI/CD en Endabank.
- **Node.js:** Entorno de ejecución Open Source que permite la ejecución de código JavaScript en el lado del servidor [17]. Es usado para la creación del servicio de emails en Endabank, en conjunto con Sendgrid.
- **SendGrid:** Plataforma para el envío de correos electrónicos que puede ser gestionada en la nube. Provee una API que es consumida desde Node.js para el envío de correos de transacciones en Endabank.
- **Flyway:** Herramienta de migración de bases de datos que permite llevar un registro y versión de los cambios en la base de datos ya sea a nivel de estructura o de datos. Su fácil integración con Spring Boot y PostgreSQL fue muy útil para su inclusión en Endabank.

IV. METODOLOGÍA

A. Marco de trabajo empleado

El marco de trabajo utilizado para el desarrollo del MVP de Endabank fue Scrum, un marco de trabajo iterativo e incremental para el desarrollo de proyectos, productos y aplicaciones [18]. Se emplearon varios componentes clave:

- **Sprints:** Se utilizaron Sprints con una duración de 10 días hábiles. Durante este período, se llevaba a cabo el trabajo del equipo. Es posible evidenciar un ejemplo de uno de los Sprints de Endabank (**Fig. 1**).

The image shows a Jira Backlog for 'EndaBank Sprint 8'. The sprint is active and has 13 issues. The issues are listed in a table-like format with columns for issue ID, description, category, and assignee. The issue 'MEDDV002-5157 US-3 | Login to Endabank from ecommerce' is highlighted in yellow.

Issue ID	Description	Category	Assignee
MEDDV002-5123	US-9 View transaction history of a product	Product Management	Assignee 1
MEDDV002-5118	US-7 View transaction details	Product Management	Assignee 2
MEDDV002-5782	Create an E-commerce		Assignee 3
MEDDV002-5157	US-3 Login to Endabank from ecommerce	Account Management	Assignee 4
MEDDV002-5136	US-27 Check API Key	Product Management	Assignee 5
MEDDV002-5218	US-30 Internal transaction result	Money Management	Assignee 6
MEDDV002-5095	US-16 Receive a transaction confirmation e-mail	Money Management	Assignee 7
MEDDV002-5226	US-19 View the detail of my business product request	Business Product Re...	Assignee 8
MEDDV002-5237	US-31 View my business products requests		Assignee 9
MEDDV002-5889	The dates when a transaction is approved or rejected is in a different format	Endabank2023	Assignee 10
MEDDV002-5928	DevOps Test and modify pipeline for mail sarvice	Architecture	Assignee 11
MEDDV002-5929	DevOps IaC FE automation	Architecture	Assignee 12
MEDDV002-5959	Transactions of less than \$1 can be made through the API.	Endabank2023	Assignee 13

Fig. 1. Backlog de Endabank en el Sprint 8

- **Roles:** El Product Owner (PO) fue el líder técnico de Endava encargado de garantizar el valor del producto. El Scrum Master desempeñó el papel de facilitador y gestor del proceso Scrum, asegurándose de evitar bloqueos e impedimentos. Además, un analista QA desempeñó un papel fundamental al validar el producto de cada Sprint mediante un riguroso plan de pruebas. El devops se encargó de la infraestructura del aplicativo y, finalmente, cinco desarrolladores (tres frontends y dos backends) completaron el equipo, aportando sus habilidades técnicas para el desarrollo del proyecto.
- **Daily Scrum:** Todos los días laborables se realizaban reuniones con el equipo que duraban aproximadamente 15 minutos. Estas reuniones servían para compartir el progreso, informar sobre pendientes y discutir cualquier bloqueo.
- **Sprint Retrospective:** Al finalizar cada Sprint, se llevaba a cabo una reunión de aproximadamente 30 minutos con el equipo. Durante esta reunión, se respondían preguntas como: ¿Qué nos ayudó a avanzar?, ¿Qué obstáculos encontramos?, ¿Cómo podríamos mejorar nuestra forma de trabajar en el futuro?, ¿Qué debemos hacer en el próximo Sprint? El objetivo era mejorar el trabajo en equipo en cada iteración.
- **Sprint Planning:** Antes de comenzar cada Sprint, se realizaba una reunión de planificación. Durante esta reunión, se inspeccionaba el backlog y se determinaban las Historias de Usuario (HU) que se abordarían en el siguiente Sprint.
- **Sprint Refinement:** A mitad de cada Sprint, se realizaba una reunión de refinamiento. Durante esta reunión, se priorizaban las HU y se actualizaba el Product Backlog en caso de ser necesario.

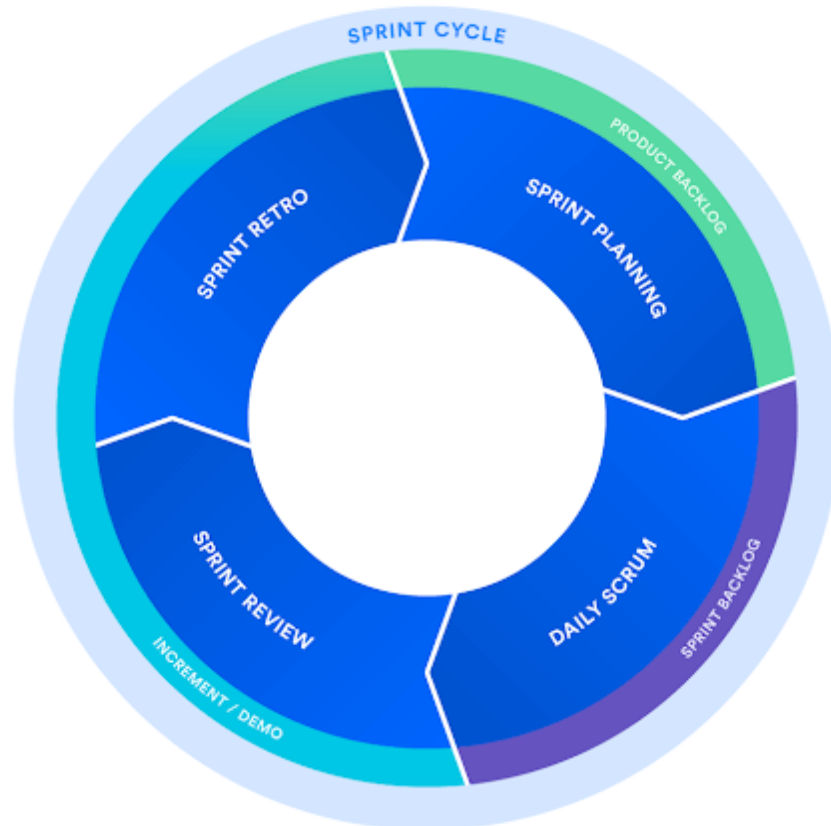


Fig. 2. Ciclo del marco de trabajo SCRUM.

Nota: fuente <https://www.atlassian.com/es/agile/scrum/sprints>

B. Ejecución del proyecto con el fin de lograr los objetivos planteados

1) Capacitación en la vertical de pagos

Antes de comenzar a recopilar los requisitos de Endabank, se llevó a cabo una capacitación para el equipo de desarrollo sobre la vertical de Pagos de Endava. El objetivo era contextualizar al equipo y adquirir los conocimientos necesarios para el desarrollo del proyecto. La capacitación se realizó durante dos semanas y consistió en la visualización de vídeos que abarcaban varios módulos como *Payments Overview (Acquiring)*, *Payment Instruments*, *Card Types & Mediums*, *Payment Players*, *Effective Authorisation*, entre otros. Durante la capacitación, surgieron preguntas relacionadas con los diversos módulos, las cuales se resolvían en las daily.

2) Visioning

Se realizaron reuniones de visioning entre la Business Analyst y los miembros del equipo para determinar la visión del producto, es decir, la motivación para la creación de Endabank y el impacto positivo que debería provocar. Además, se discutió el público objetivo al que se dirige el producto, identificando el segmento de mercado al que se pretende llegar. También las necesidades que satisface el producto y los beneficios que ofrece, asimismo qué hace destacar al producto y por último, los objetivos de negocio.



Fig. 3. Muestra de visioning con el BA y el equipo de trabajo

3) Mapeo de historias de usuario (User Story Mapping)

Posterior a las reuniones de visioning, teniendo mucho más clara la visión del producto y el objetivo del proyecto, se realizó un mapeo de historias de usuario, el cual establece una visión compartida de un producto digital de manera visual y facilita en gran medida la especificación de un MVP [19]. En esta etapa se llevó a cabo una identificación de las historias de usuario que generarían valor para Endabank.

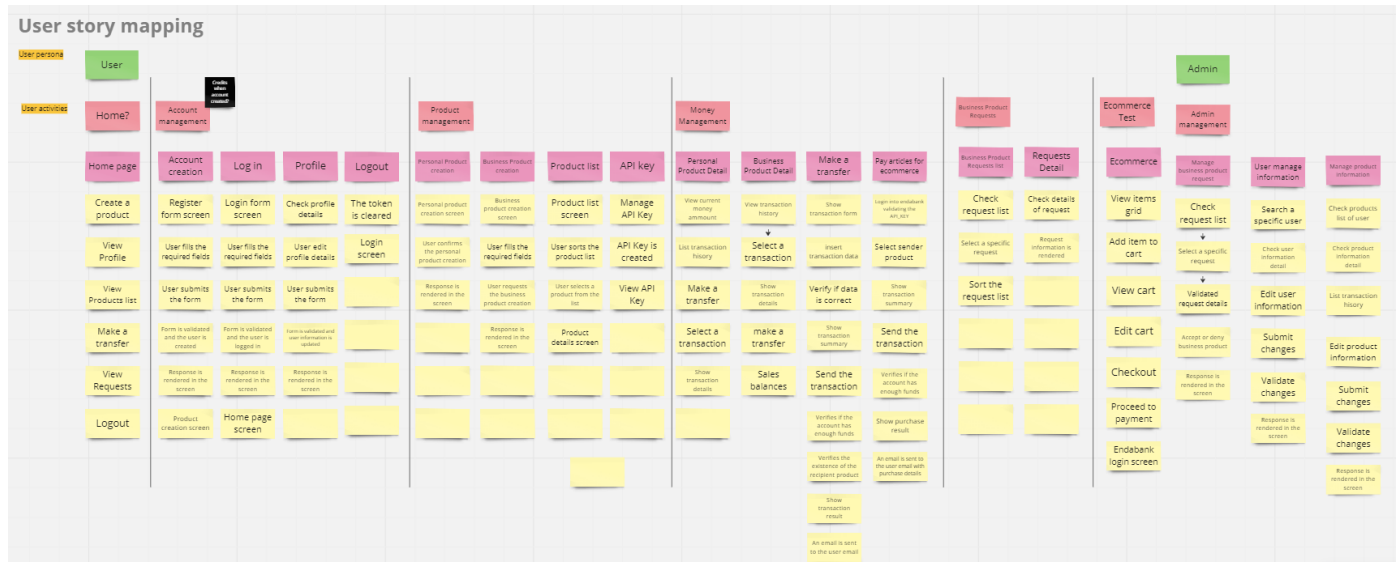


Fig. 4. Mapeo de historias de usuario

Después de mapear las historias de usuario, se redactaron utilizando la estructura "As an/I need to/So that I can" y los criterios de aceptación con la forma semiestructurada "Given/When/Then". Durante la redacción de las historias de usuario, surgieron dudas en el equipo sobre cómo diferenciar las tareas de frontend y backend, ya que era posible crear historias de usuario específicas para cada uno. Después de investigar y explorar diferentes estrategias de escritura de historias de usuario, se descubrió el enfoque de Vertical Slicing. Este enfoque consiste en dividir una característica en partes funcionales en todas las capas de la aplicación (UI, lógica de negocio y capa de datos) [20]. Esto permite obtener incrementos completamente funcionales que brindan retroalimentación en etapas tempranas del desarrollo con la metodología Scrum. En consecuencia, las historias de usuario contenían subtareas tanto de frontend como de backend.

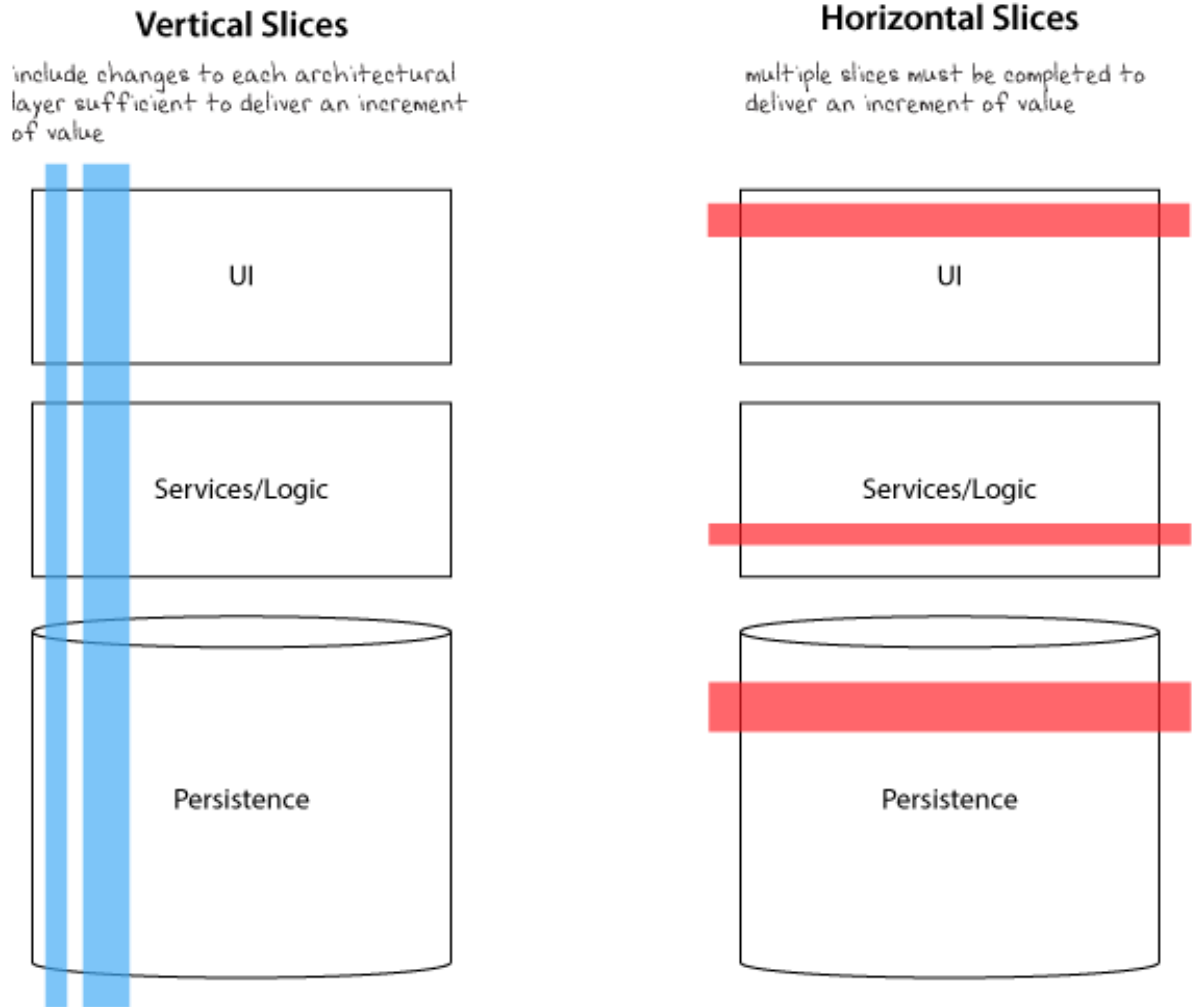


Fig. 5. Comparación entre Vertical Slicing y Horizontal Slicing.

Nota: fuente <https://www.humanizingwork.com/the-humanizing-work-guide-to-splitting-user-stories/#vertical-slices>

4) *Priorización de historias de usuario*

Tras contar con las HU identificadas y redactadas, se realizó una priorización de estas con la técnica MoSCoW [21], la cual se encarga de segmentar las HU en 4 tipos:

- **Must have:** HU que provee una funcionalidad utilizable mínima para el MVP.
- **Should have:** HU importante pero no esencial.
- **Could have:** HU que se puede realizar, si queda tiempo.
- **Won't have this time:** HU que se decide desplazar a un futuro release.

5) Estimación de HU

Se llevó a cabo la estimación de las distintas historias de usuario empleando el método Fibonacci, asignando números de la secuencia de Fibonacci (1, 2, 3, 5, 8, 13, 21, 34, 55 ... 89) a las HU. Cada número representaba la cantidad de días de trabajo estimados para completar una HU.

6) Selección de tecnologías

Para el desarrollo del frontend de la aplicación se optó por usar la librería React, ya que se considera una sólida elección por diversas razones. En primer lugar, por su rendimiento y eficiencia, ya que al utilizar una tecnología llamada “virtual DOM” , permite realizar actualizaciones rápidas y eficientes de la interfaz de usuario, mejorando la velocidad y respuesta de la aplicación. En segundo lugar, React fomenta la creación de componentes reutilizables y modulares, lo cual facilita la reutilización de código, mantenibilidad, legibilidad, separación de responsabilidades y la escalabilidad. Además, React cuenta con una gran y activa comunidad de desarrolladores, por lo que hay una amplia cantidad de recursos en línea disponibles como tutoriales, libros y cursos; lo cual facilita y optimiza el aprendizaje. Estos beneficios finalmente contribuyen a un desarrollo más eficiente.

Para el desarrollo del backend se optó por el uso Spring Boot ya que se basa en Java, un lenguaje de programación robusto, orientado a objetos y altamente tipado, lo cual ofrece muchas ventajas durante el proceso de desarrollo de software. Por otra parte Spring Boot proporciona una amplia variedad de librerías y componentes preconfigurados que simplifican en gran medida el desarrollo de backends para apps empresariales. Estos elementos cubren cosas como el acceso a bases de datos, la seguridad y la integración con servicios web, lo que permite implementar rápidamente funcionalidades avanzadas sin tener que configurar todo manualmente. Además, Spring Boot cuenta con una comunidad de desarrollo de software sumamente activa la cual ofrece recursos útiles, tutoriales y documentación, lo que facilita el aprendizaje y la resolución de problemas durante el desarrollo. Por último, Spring Boot simplifica la realización de pruebas unitarias proporcionando herramientas y anotaciones específicas para el desarrollo de pruebas y

se integra sin ningún inconveniente con JUnit y Mockito, dos librerías utilizadas en el proyecto para el desarrollo de pruebas unitarias.

7) *Flujo de usuario*

Se tuvieron en cuenta los tipos de usuarios y los objetivos de cada uno, para así idear la arquitectura de información de la aplicación y analizar cómo interactuaría el usuario con el flujo de la aplicación.

8) *Wireframes del aplicativo*

Se utilizó el enfoque de mobile first (estrategia de diseño y desarrollo web que prioriza el diseño y la optimización para dispositivos móviles antes que para computadoras de escritorio), se buscó que todos los diseños fueran responsivos para asegurar que las interfaces desarrolladas posteriormente se adaptaran y se ajustaran automáticamente a diferentes tamaños de pantalla. Finalmente, se incluyeron diversos UI Patterns (patrones de interfaz de usuario) para mejorar la experiencia de usuario, como lo son modales para mostrar información adicional y la confirmación de acciones, un navbar (barra de navegación) para una navegación intuitiva y un sidebar (barra lateral) para acceder rápidamente a diferentes secciones de la aplicación.

9) *Realizar guía de estilos del aplicativo*

Se construyó un sistema de diseño utilizando la metodología llamada *Atomic Design*, ideada por Brad Frost [22], el cual nos permitió crear una biblioteca de componentes y patrones reutilizables, organizados en una estructura jerárquica basada en átomos, moléculas, organismos y plantillas. Adicionalmente, se crearon los lineamientos de diseño que buscaban alinearse con el brand guide de Endava. La estandarización de los lineamientos de diseño y componentes permitieron un desarrollo más escalable y desacoplado.

10) Prototipos de alta fidelidad del aplicativo

Se utilizó la herramienta Figma para generar versiones previas al desarrollo de las interfaces de usuario, para compartirlas con el equipo para recibir retroalimentación y su posterior validación con el PO y de esta forma definir las pantallas finales para ser base para el desarrollo. También se realizaron prototipos tanto para mobile como para desktop. Adicionalmente se utilizó Figma para el manejo de assets como íconos e imágenes.



Fig. 6. Tipografía, colores e íconos en Figma



Fig. 7. Atomic Design en Figma

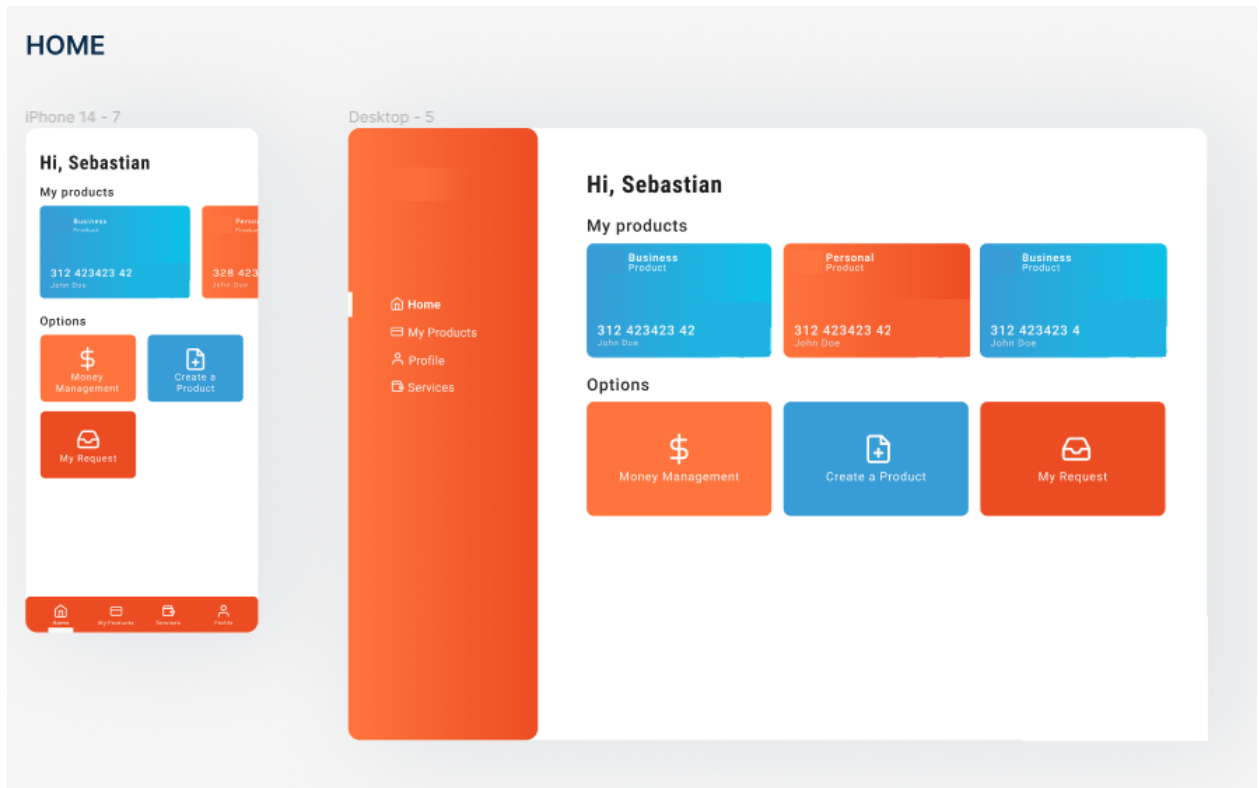


Fig. 8. Prototipos de alta fidelidad en Figma

11) Desarrollo de módulos de la plataforma web

Se llevaron a cabo una serie de acciones para la implementación de los módulos de gestión de usuarios, gestión de productos, gestión de transacciones y administración:

- **Análisis de requisitos:** Identificación de las funcionalidades necesarias para cada módulo. Esto implicó identificar las necesidades de los usuarios y establecer los objetivos específicos de cada módulo.
- **Diseño de las interfaces de usuario:** Creación de las pantallas y componentes necesarios, teniendo en cuenta las mejores prácticas de usabilidad y experiencia de usuario.
- **Implementación de la lógica de negocio:** Se desarrollaron las funcionalidades necesarias para implementar la lógica de negocio de cada módulo. Esto implicó la creación de o componentes de software que permitieran realizar las acciones específicas relacionadas con la gestión de usuarios, productos, transacciones y administración.

- **Documentación:** Se generó una documentación que describe el diseño y la implementación de cada módulo, esto empleando la plataforma Confluence.

12) Definición de sistema de gestión de base de datos

En el desarrollo de sistemas bancarios como Endabank, es preferible utilizar una base de datos relacional debido a su estructura organizada y su capacidad para garantizar la integridad de los datos, por ejemplo, mediante la definición de constraints. Además, las bases de datos relacionales permiten realizar consultas complejas de manera eficiente, lo cual es fundamental en un entorno bancario con grandes volúmenes de datos. Por estas razones, se opta por utilizar una base de datos relacional, específicamente el sistema de gestión de base de datos PostgreSQL. Para el diseño del sistema, se realiza un modelado de datos utilizando un diagrama entidad-relación, el cual se encuentra disponible en la sección V. RESULTADOS.

13) Arquitectura de la Rest API

Después de comprender los requisitos del MVP de Endabank y tener un dominio bien definido, se establece la arquitectura del backend. Con el objetivo de crear un código más comprensible y menos propenso a errores, se adopta el enfoque *Package by Feature/Component*, propuesto por el arquitecto de software Simon Brown. Tradicionalmente, se ha utilizado el enfoque *Package by Layer*, que es adecuado para proyectos pequeños donde realizar cambios en el código no es costoso. Sin embargo, al aplicar este enfoque en un proyecto con una gran cantidad de características, puede resultar en un código más complejo. Además, al abordar un nuevo proyecto, generalmente tenemos en mente un dominio o característica específica que se necesita cambiar, lo que hace que la implementación de *Package by Feature/Component* sea más conveniente. Este enfoque proporciona una mejor visibilidad y una visión general más clara desde la perspectiva del dominio, lo que nos permite acercarnos a una base de código con un objetivo comercial en mente [23]. Por otro lado, el código se vuelve más simple, ya que al tener la necesidad de agregar o modificar una característica solo se necesita manejar un caso de uso específico. Esto implica un código más fácil de entender y que puede evolucionar con el tiempo, de hecho una futura migración a microservicios se facilitaría en gran medida ya que se tienen

identificados los dominios del aplicativo y se cuenta con un alto nivel de independencia entre paquetes.

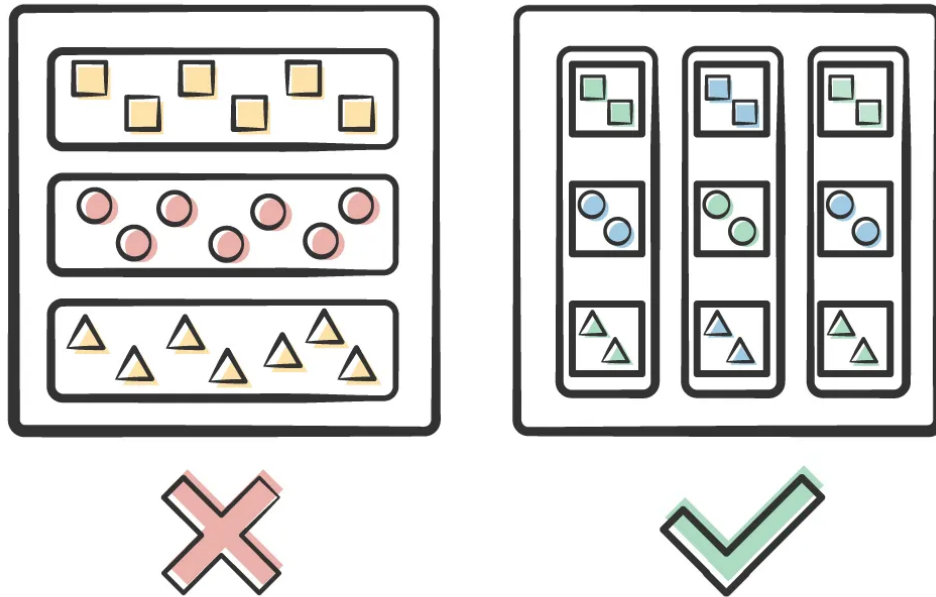


Fig. 9. Package by layer vs Package by feature

Nota: fuente <https://medium.com/sahibinden-technology/package-by-layer-vs-package-by-feature-7e89cde2ae3a>

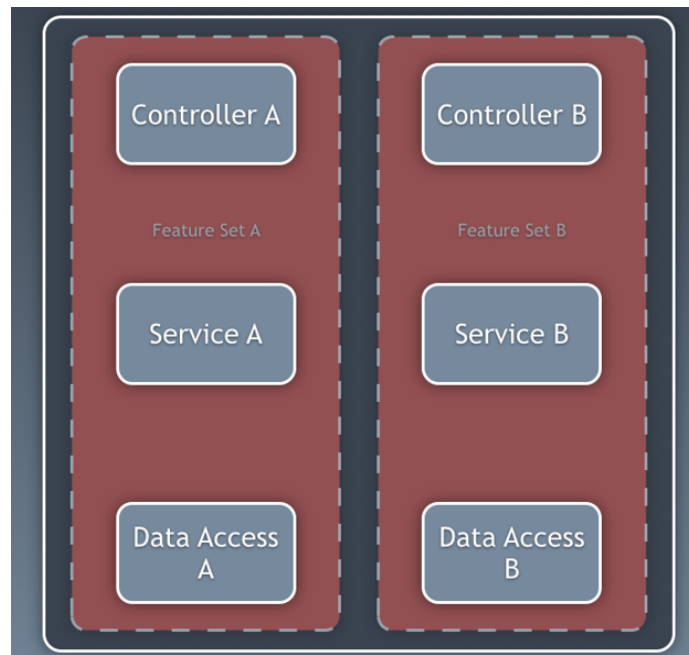


Fig. 10. Package by feature/Component

Nota: fuente <https://herbertograca.com/2017/11/16/explicit-architecture-01-ddd-hexagonal-onion-clean-cqrs-how-i-put-it-all-together/>

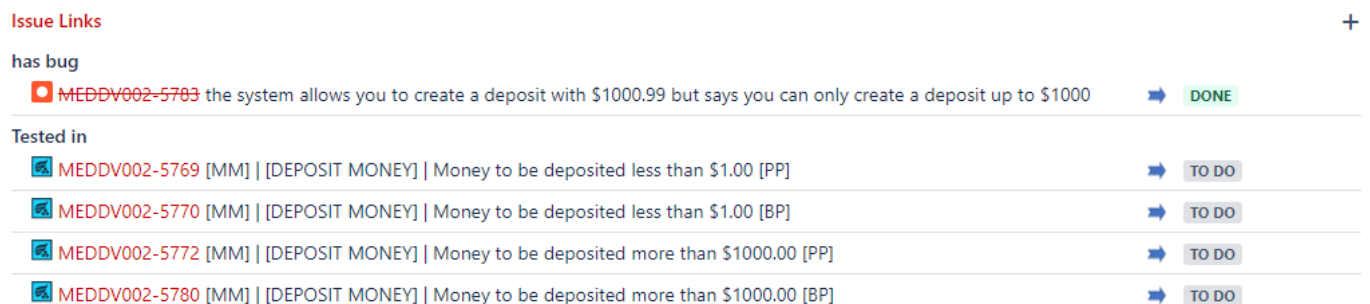
14) Desarrollo de pruebas unitarias

Tanto en el backend como en el frontend se desarrollaron las pruebas unitarias siguiendo el patrón AAA (Arrange, Act and Assert) y se definió una cobertura de código mayor o igual a 75%. En el backend se utilizaron las librerías JUnit 5 y Mockito y en el Frontend se utilizó el framework Vitest.

15) Ejecución de pruebas manuales

Para el aseguramiento de la calidad del proyecto, el equipo de QA creó ciclos de pruebas a partir del sprint #2. Estos ciclos de pruebas contenían los casos de prueba (**Fig. 11**) pertinentes que aseguraban que las funcionalidades tuvieran el comportamiento esperado. Además, cada ciclo de pruebas contenía un balance de cuántos casos de prueba fallaron o fueron exitosos. La elaboración de los casos de pruebas se hacía aplicando técnicas de reducción de casos pruebas, como lo son: particiones de equivalencia y análisis de valores al límite.

Las pruebas del frontend del aplicativo se realizaban en un ambiente de pruebas, las del backend a través de la herramienta Postman y las pruebas a la base de datos a través de la herramienta DBeaver. Al momento de ser encontrado un bug, este era agregado al backlog del proyecto y una vez solucionado, se realizaban pruebas de regresión, es decir, se ejecutaban todos los casos de prueba relacionados a la funcionalidad y algunos casos de prueba adicionales para asegurar que la solución no introdujera nuevos bugs.



The screenshot shows a Jira issue page. At the top, there is a section titled "Issue Links" with a plus sign on the right. Below it, the issue is identified as "has bug" with ID "MEDDV002-5783". The description of the bug is "the system allows you to create a deposit with \$1000.99 but says you can only create a deposit up to \$1000". To the right of the description is a status indicator "DONE" with a green arrow pointing right. Below the issue description is a section titled "Tested in" which lists four test cases. Each test case has a status indicator "TO DO" with a grey arrow pointing right.

Issue ID	Test Case Description	Status
MEDDV002-5783	the system allows you to create a deposit with \$1000.99 but says you can only create a deposit up to \$1000	DONE
MEDDV002-5769 [MM] [DEPOSIT MONEY] Money to be deposited less than \$1.00 [PP]		TO DO
MEDDV002-5770 [MM] [DEPOSIT MONEY] Money to be deposited less than \$1.00 [BP]		TO DO
MEDDV002-5772 [MM] [DEPOSIT MONEY] Money to be deposited more than \$1000.00 [PP]		TO DO
MEDDV002-5780 [MM] [DEPOSIT MONEY] Money to be deposited more than \$1000.00 [BP]		TO DO

Fig. 11. Ejemplo de casos de prueba en una HU

16) Demos con el PO

Con el fin de aplicar correctamente el marco de trabajo Scrum, era fundamental la realización de entregas continuas a través de Sprint Demos. Estos consistían en reuniones empleadas para exponer al Product Owner los avances de los Sprints, buscándose mostrar el incremento a nivel de funcionalidad. Además, cada área (Frontend, Backend, Devops y Testing) exhibía aspectos técnicos importantes para el desarrollo del proyecto. En estas demostraciones se recibía un feedback por parte del PO y los asistentes a la reunión que por lo general eran mentores, business analysts y desarrolladores de la empresa.

17) Validación de que el Frontend del aplicativo corresponda con el diseño propuesto

Se realizó una evaluación a partir de la guía de estilos planteada, donde se verificó que los elementos visuales correspondan con la definición planteada en los diferentes tamaños de pantalla en el diseño final, como: fuentes, colores y espaciado.

V. RESULTADOS

A. Diagrama entidad-relación

El diagrama entidad-relación en alto nivel que se muestra a continuación (**Fig. 12**) proporciona una visualización de las entidades de Endabank junto con sus atributos y las relaciones existentes sin incluir detalles específicos de implementación como lo son el tamaño de caracteres o los tipos de datos específicos de PostgreSQL. A través de la representación gráfica se pueden comprender las entidades involucradas y cómo se relacionan entre sí mediante la cardinalidad mostrada.

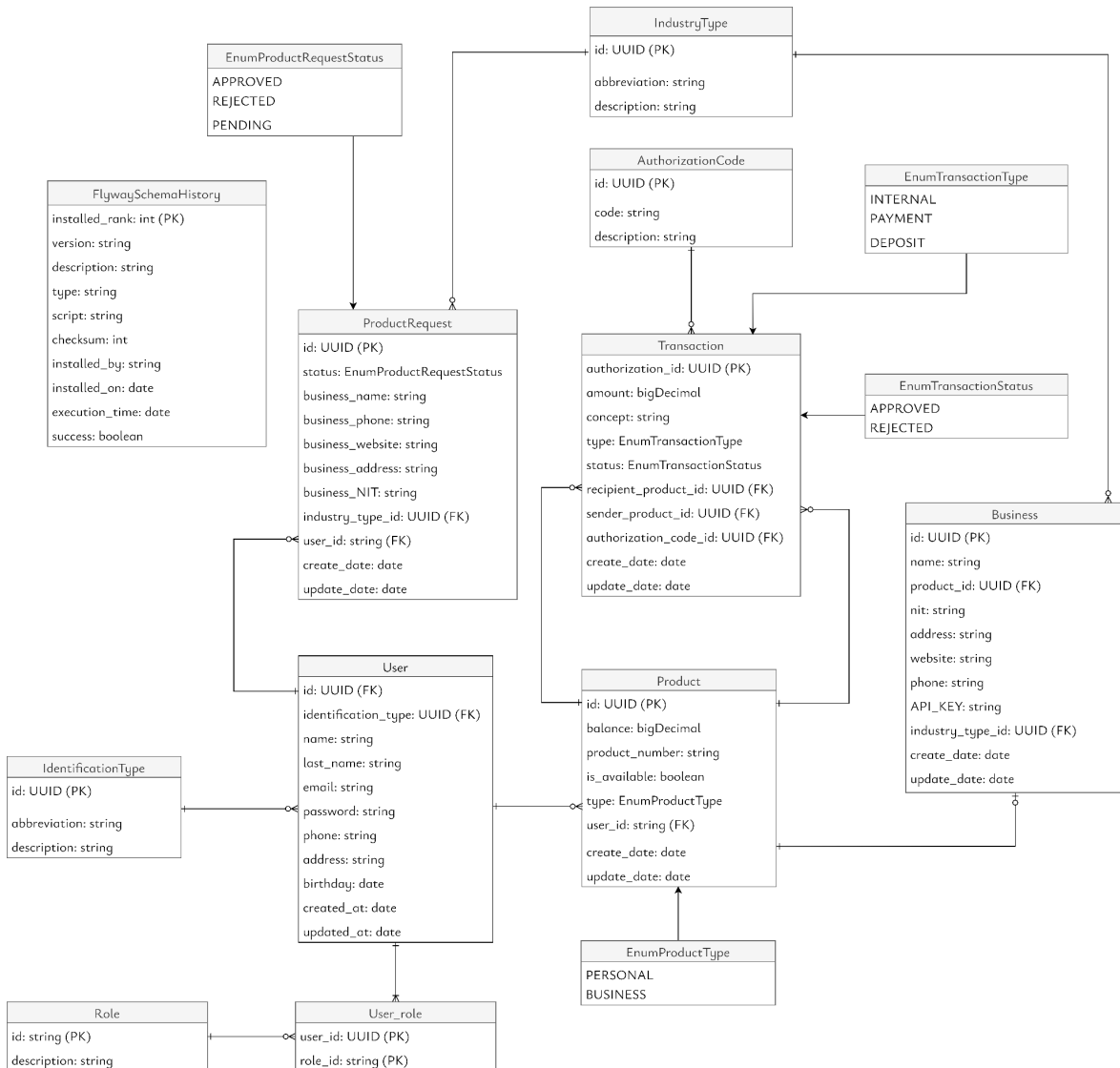


Fig. 12. Diagrama entidad-relación en alto nivel

El diagrama brinda una idea general de la estructura y las interacciones principales de la base de datos de Endabank, de lo cual es posible enunciar lo siguiente:

- Un usuario de Endabank puede tener varios roles (relación muchos a muchos), lo cual requiere la creación de una tabla intermedia. Los roles actuales son usuario regular y administrador. Además, el usuario puede iniciar sesión utilizando su correo electrónico y contraseña.
- Un usuario puede tener cero o varios productos (cuentas bancarias), tanto personales como como de negocio (de ahora en adelante, producto empresarial). Los productos empresariales están destinados a recibir pagos desde un e-commerce. Cada producto está asociado a un solo usuario.
- Antes de la creación de un producto empresarial, un usuario debe realizar una solicitud para dicho producto, por medio de la cual se ingresa la información asociada a la empresa que se quiere asociar al producto empresarial. Esta relación implica que un usuario puede tener cero o muchas solicitudes de productos, y cada solicitud debe estar asociada a un usuario. Las solicitudes de productos pueden tener los estados aprobada, rechazada o pendiente.
- En el caso de los productos empresariales, existe una asociación con la entidad negocio, que tiene atributos como nombre, NIT, dirección, sitio web, teléfono, tipo de industria y una API KEY utilizada para la conexión con el e-commerce.
- Cada producto tiene una relación doble con la entidad transacción. La primera relación representa el producto de origen y la segunda representa el producto de destino. Por lo tanto, existen dos relaciones de "uno a cero o muchos" desde producto a transacción. Cada transacción debe estar asociada con dos productos.

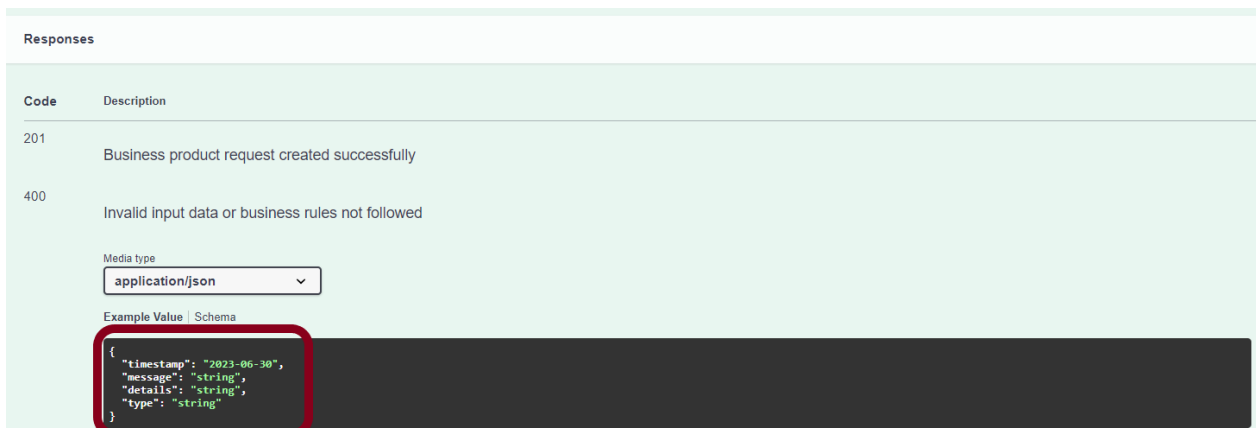
- De acuerdo con el estándar ISO 20022, se utiliza un código de autorización para identificar si una transacción ha sido exitosa o no. En caso de que la transacción sea rechazada, se utilizan códigos específicos para identificar la razón del rechazo, como por ejemplo fondos insuficientes o cuenta bancaria de destino inexistente. Las transacciones pueden tener los estados rechazada o aprobada y pueden ser un depósito (útil para agregar dinero a un producto), una transacción interna (transacción de un producto de origen a un producto de destino desde el aplicativo Endabank) y un pago (transacción proveniente de un pago desde un e-commerce).
- Se hace uso de UUID como tipo de dato para los identificadores únicos ya que estos son prácticamente imposibles de predecir a diferencia de los id 's incrementales que siguen un patrón muy predecible además que proveen información de la cantidad de registros de la base de datos. Se emplea UUID para evitar vulnerabilidades en términos de seguridad.

B. Buenas prácticas en el diseño de la Rest API

Con el fin de contar con una REST API escalable a lo largo del tiempo, es necesario aplicar un conjunto de buenas prácticas en su diseño. Algunas de las buenas prácticas se pueden observar en la **Fig. 13**:

1. Adherencia a los principios de diseño REST, lo cual implica utilizar los métodos HTTP adecuados para las diferentes operaciones CRUD. Es decir, utilizar GET para obtener recursos, POST para crear nuevos recursos, PUT para actualizar recursos existentes, PATCH para actualizar parcialmente recursos y, finalmente, DELETE para eliminar recursos.
2. Utilización de sustantivos en el nombramiento de los recursos en vez de verbos. Por ejemplo, en lugar de definir el recurso `"/getUsers"`, es mejor definirlo `"/users"`.
3. Nombramiento de recursos en plural. Por ejemplo, en lugar de definir el recurso `/user` es preferible utilizar `"/users"`.
4. Versionamiento de la API REST. Es sumamente recomendable incluir una versión en la URI para poder realizar cambios en el futuro sin afectar las versiones anteriores. Como muestra de esto, es posible definir el endpoint como `"api/v1/users"`.

5. Uso de rutas de forma jerárquica. Por ejemplo, definir una ruta como `"/users/{userId}/products"` para obtener los productos asociados a un usuario específico.
6. Uso de códigos de estado HTTP correctos. Por ejemplo, utilizar el código de estado 200 para una respuesta exitosa, 201 para la creación de recursos, 400 para errores del cliente, 401 cuando se requiere autenticación previa, 403 cuando el acceso no está permitido, 404 cuando no se encuentra un recurso, 500 para errores del servidor, entre otros. Además, es importante tener un payload definido para los errores, el cual retorna una respuesta clara al cliente (**Fig. 14**).



The screenshot displays a 'Responses' section in a REST client. It lists two status codes: 201 (Business product request created successfully) and 400 (Invalid input data or business rules not followed). The 400 response is selected, and its details are shown below. The 'Media type' is set to 'application/json'. Under the 'Example Value' tab, a JSON payload is displayed, which is highlighted with a red box. The JSON payload is:

```
{
  "timestamp": "2023-06-30",
  "message": "string",
  "details": "string",
  "type": "string"
}
```

Fig. 14. Muestra de payload empleado para los errores

7. Uso de paginación. Con el fin de mejorar la eficiencia en la transferencia de datos y maximizar el rendimiento de la API REST, se implementa la paginación en los servicios que lo requieran. Esto se logra mediante los parámetros "page" (comúnmente denominado "offset", que establece la posición desde donde comienza una lista) y "size" (comúnmente denominado "limit", que establece el número de elementos de la lista a retornar a partir de "page"). Un ejemplo de un endpoint con paginación es `"/api/v1/users?page=0&size=12"`.
8. Uso de filtros y ordenación. Con el objetivo de reducir significativamente los datos retornados al cliente y facilitar la ordenación de los resultados, se implementan los parámetros "sort" (para ordenar por un campo, ya sea de forma ascendente o descendente) y la capacidad de filtrar por campos de forma dinámica. Como ejemplo de esto, un endpoint en el cual se visualice el filtrado y la ordenación es `"/api/v1/users?sort=createDate,asc&identification=1152225144&email=user@test.com"`.
9. La API se protege mediante autenticación y autorización basada en roles con JWT.
10. Uso de DTO's tanto para las peticiones como para las respuestas de los controladores. Por ejemplo, para la creación de un usuario se tiene un DTO `UserSaveRequest` y otro `UserSaveResponse`, de esta manera en la respuesta solo es retornada la data necesaria del usuario creado.

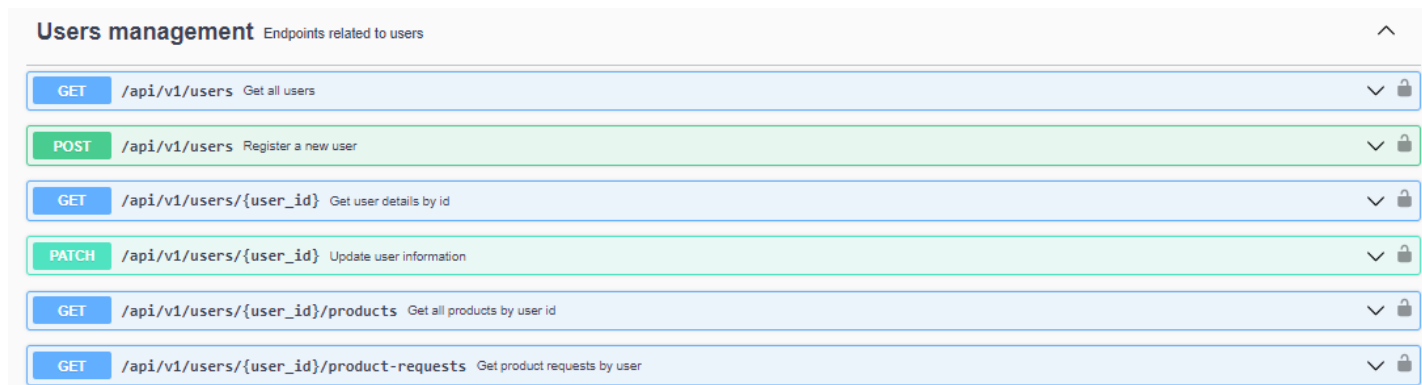


Fig. 13. Muestra de endpoints para la gestión de usuarios con OpenAPI 3

C. Estructura del backend

Para la definición del backend, se identifican los siguientes componentes o características que hacen parte del dominio de la aplicación:

- **Product Management:** Se agrupan los servicios asociados a los productos (cuentas bancarias).
- **Product Request handling:** Agrupación de servicios relacionados con la gestión de las solicitudes de apertura de productos asociados a un negocio en particular.
- **Transaction Management:** Se agrupan todos los servicios que tienen relación con el manejo de transacciones.
- **User Management:** Agrupación de servicios para la gestión de usuarios, es decir, los clientes de Endabank.
- **Authentication:** Servicios con relación a la autenticación y autorización en Endabank.
- **Notifications Management:** Servicios encargados de la comunicación con el microservicio de envío de correos electrónicos en Endabank.

Cada uno de las características mencionadas cuenta con una estructura bien definida por medio de los componentes:

- **Infraestructura:** La infraestructura abarca las diferentes partes de la aplicación, como los controladores REST o SOAP, los repositorios de bases de datos, los DAO, entre otros. En el caso de Endabank, se organiza en una carpeta llamada "web", que contiene el controlador asociado al componente y los modelos de datos utilizados para las solicitudes y respuestas de los servicios, siguiendo el patrón DTO. Además, se encuentran los repositorios de base de datos, donde se implementa la interfaz JpaRepository debido al uso de PostgreSQL. Por último, se encuentra la carpeta "gateway", que sirve como intermediario entre el servicio y el repositorio que está vinculado a una base de datos específica.
- **Modelo:** Se definen los modelos asociados al componente/feature.
- **Servicio:** En el servicio se define la lógica de negocio de Endabank. Hay una carpeta modelo que tiene definidos los comandos, los cuales incluyen la información requerida para la ejecución de una acción o la realización de una operación particular en el backend. Además de contener la lógica de negocio de los casos de uso asociados al componente, se cuenta con las distintas abstracciones o adaptadores (interfaces).

- **Shared:** Recursos compartidos entre los componentes. Por ejemplo las distintas excepciones del aplicativo.
- **Config:** Configuraciones generales del proyecto, por ejemplo documentación con Open API 3, seguridad con JWT, etc...

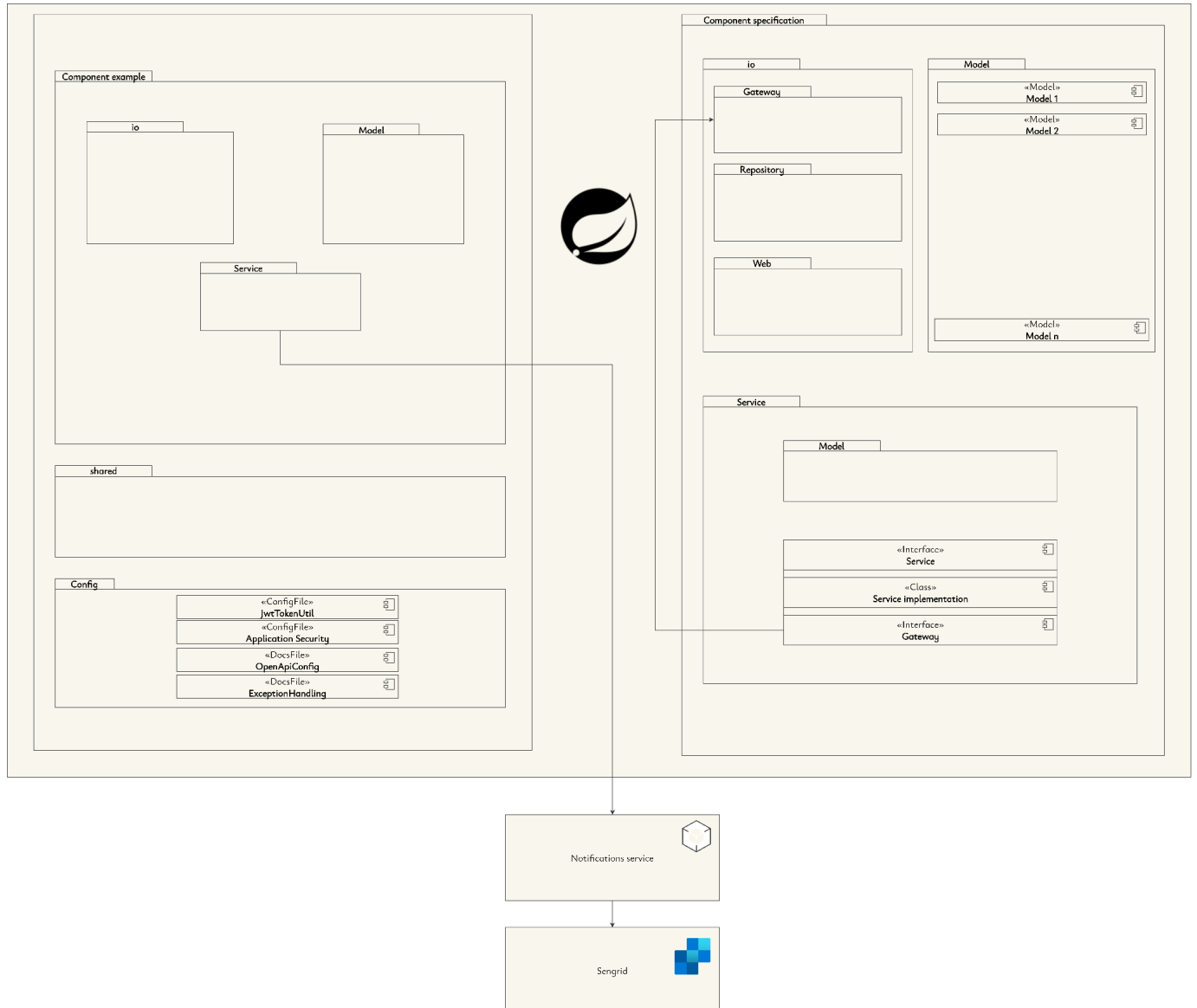


Fig. 15. Estructura de backend

D. Servicio de email con Node.js

Con respecto al envío de los emails de notificación de transacciones, se ha desarrollado un microservicio específico para esta tarea utilizando Node.js. Dado que se utilizan plantillas HTML dinámicas que varían según el tipo de transacción y si el destinatario del correo es el titular de la cuenta de origen o de destino, se ha empleado la biblioteca EJS (Embedded JavaScript). EJS permite la inclusión de lógica de programación con una sintaxis muy similar a la de JavaScript.

Por otra parte, para el envío de los correos electrónicos se ha optado por utilizar SendGrid como plataforma, la cual proporciona una API que se integra de manera sencilla con Node.js. SendGrid garantiza una alta tasa de entrega de correos electrónicos y evita que los mensajes sean clasificados como emails no deseados [24]. El microservicio se comunica de forma síncrona con el backend de Endabank para enviar los correos electrónicos.

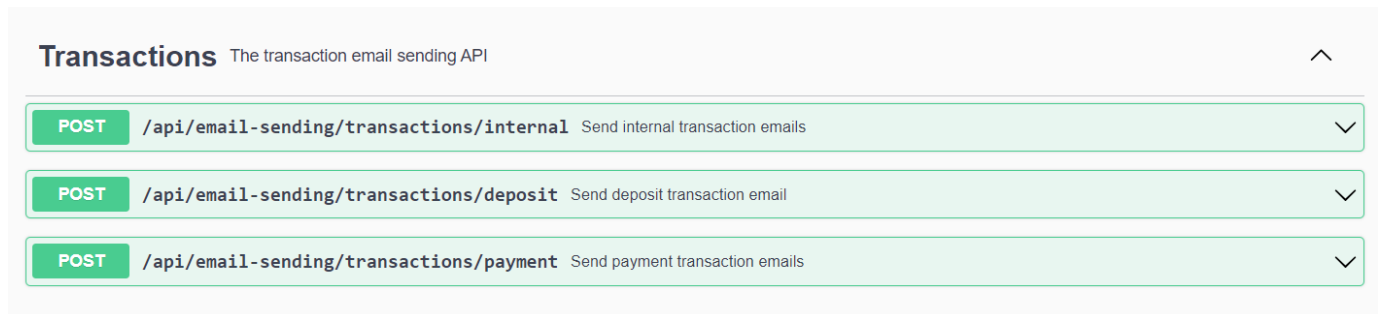


Fig. 16. Endpoints de servicio de notificaciones de transacciones

E. Estructura del frontend

Una vez comprendidos los requisitos del MVP de Endabank, se estableció una estructura de carpetas que favorezca el orden y permita estructurar el código de manera coherente y escalable. La estructura de carpetas elegida es una comunmente usada en proyectos de React y fue la siguiente:

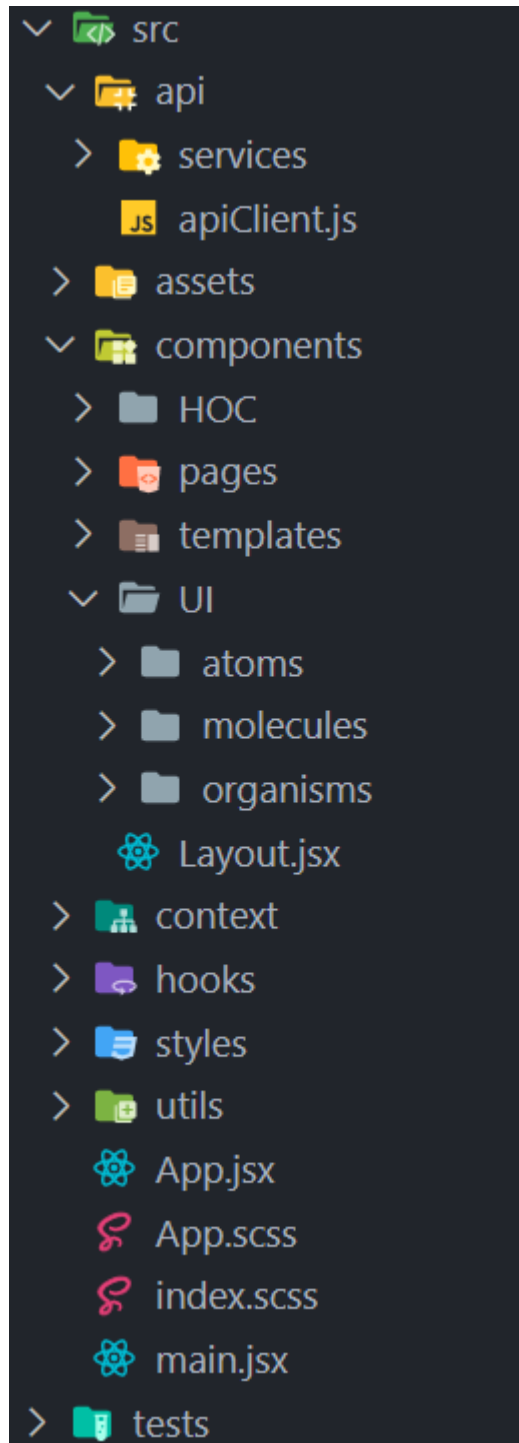


Fig. 17. Estructura del frontend

- **Carpeta api:** Esta carpeta contiene archivos relacionados a llamadas a API o servicios externos. Se tiene una carpeta que contiene todos los servicios y un archivo donde se configura el cliente de API.

- **Carpeta assets:** Esta carpeta contiene todos los recursos estáticos usados en la aplicación como imágenes e íconos.
- **Carpeta components:** Siguiendo el enfoque de Atomic Design, esta carpeta contiene subcarpetas como atoms, molecules, organisms, templates y pages. En cada una de estas carpetas se encuentran los componentes de React organizados según su nivel de abstracción y complejidad.
- **Carpeta context:** En esta carpeta se definieron los contextos de React, que permiten el intercambio de datos entre componentes sin necesidad de pasar props (o argumentos) en cascada a través de múltiples componentes.
- **Carpeta hooks:** Esta carpeta contiene los llamados custom hooks, que son funciones especiales de React que permiten utilizar características como el estado y los ciclos de vida en componentes funcionales. Haciendo uso de estas ventajas, se crearon algunos custom hooks, como por ejemplo el custom hook que permitía hacer peticiones POST y a su vez manejar los estados carga, error y respuesta de la petición.
- **Carpeta styles:** Siguiendo los lineamientos de la guía de estilo y del sistema de diseño, se guardaron en esta carpeta estilos globales y reutilizables como lo son la paleta de colores utilizados en la aplicación, filtros utilizados, espaciado utilizado para la disposición de elementos y componentes, las tipografías utilizadas, y estilos que se utilizan para restablecer los estilos predeterminados de los elementos HTML.
- **Carpeta utils:** En esta carpeta se almacenan funcionalidades comunes utilizadas en diferentes partes de la aplicación, como lo son funciones de formateo para fechas, cadenas de texto y números.
- **Carpeta tests:** En esta carpeta se almacenaron todos los archivos relacionados con las pruebas unitarias del proyecto. Dentro de esta carpeta se replica la estructura de carpetas del proyecto para mantener una coherencia y consistencia en la organización de archivos y facilitar la navegación entre el código fuente y las pruebas correspondientes.

F. Funcionalidades del aplicativo

A continuación se muestran algunas de las pantallas de las funcionalidades del aplicativo más relevantes, las cuales se dividen en el flujo de usuario y de administrador.

1) *Flujo de usuario*: Agrupación de algunas funcionalidades de clientes de Endabank.

a) Inicio de sesión

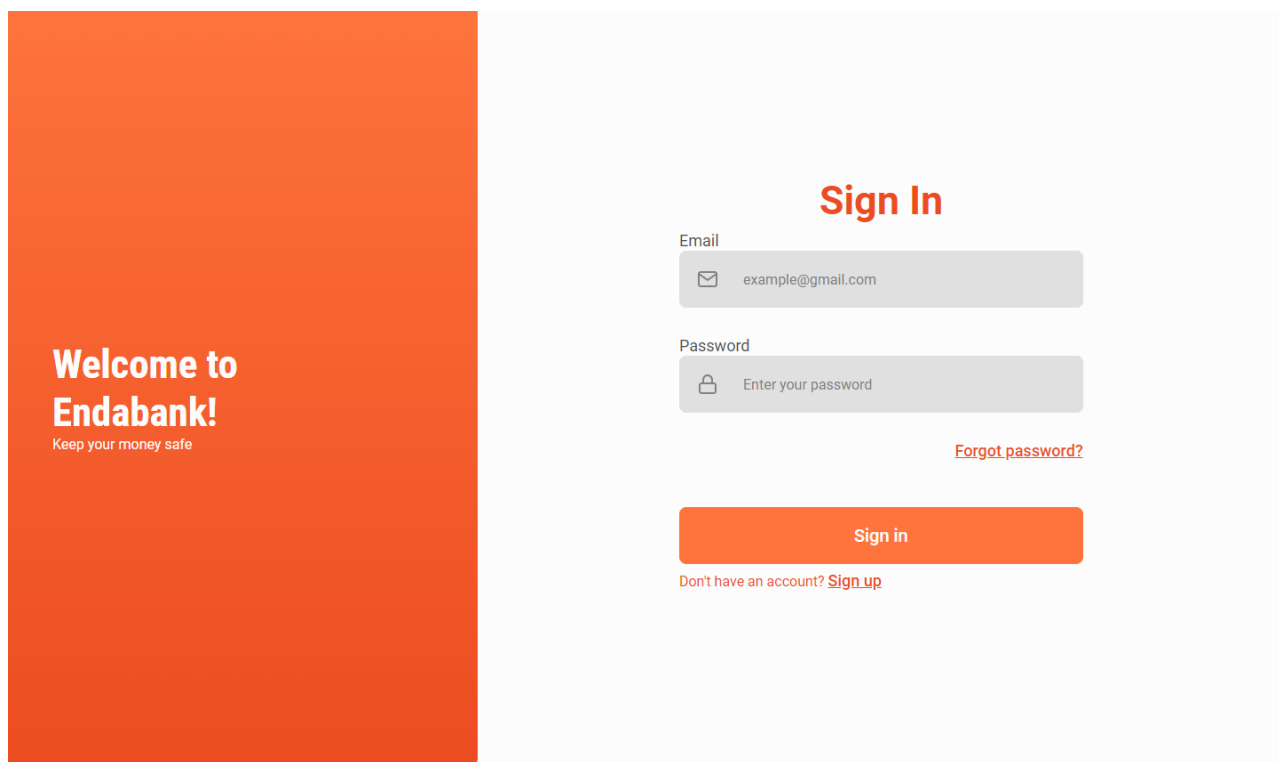


Fig. 18. Pantalla de inicio de sesión

En esta pantalla el usuario puede iniciar sesión en Endabank.

b) Registro

The image shows a web registration form titled "Create Account". The form consists of several input fields with icons indicating their purpose: a dropdown for "Identification type" (with "Select a value"), a text field for "Identification number" (with "Identification"), a date picker for "Birthday" (with "07/04/2005"), an email field for "Email" (with "johndoe@gmail.com"), a password field for "Password", a confirm password field for "Confirm Password", a name field for "Name" (with "John"), a last name field for "Last name" (with "Doe"), a phone number field for "Phone number" (with "3444444444"), and an address field for "Address" (with "Enter your address"). A "Register" button is located at the bottom of the form. Below the button, there is a link: "Already have an account? [Log In](#)". To the right of the form is a large orange vertical banner with the text "Register now!" and "Thanks for keeping your money safe with us!" below it.

Fig. 19. Pantalla de registro

En esta pantalla el usuario puede registrarse en Endabank, al introducir una serie de datos como lo son identificación, fecha de nacimiento, correo electrónico, contraseña, nombre, apellidos, número de teléfono y dirección.

c) *Home del usuario*

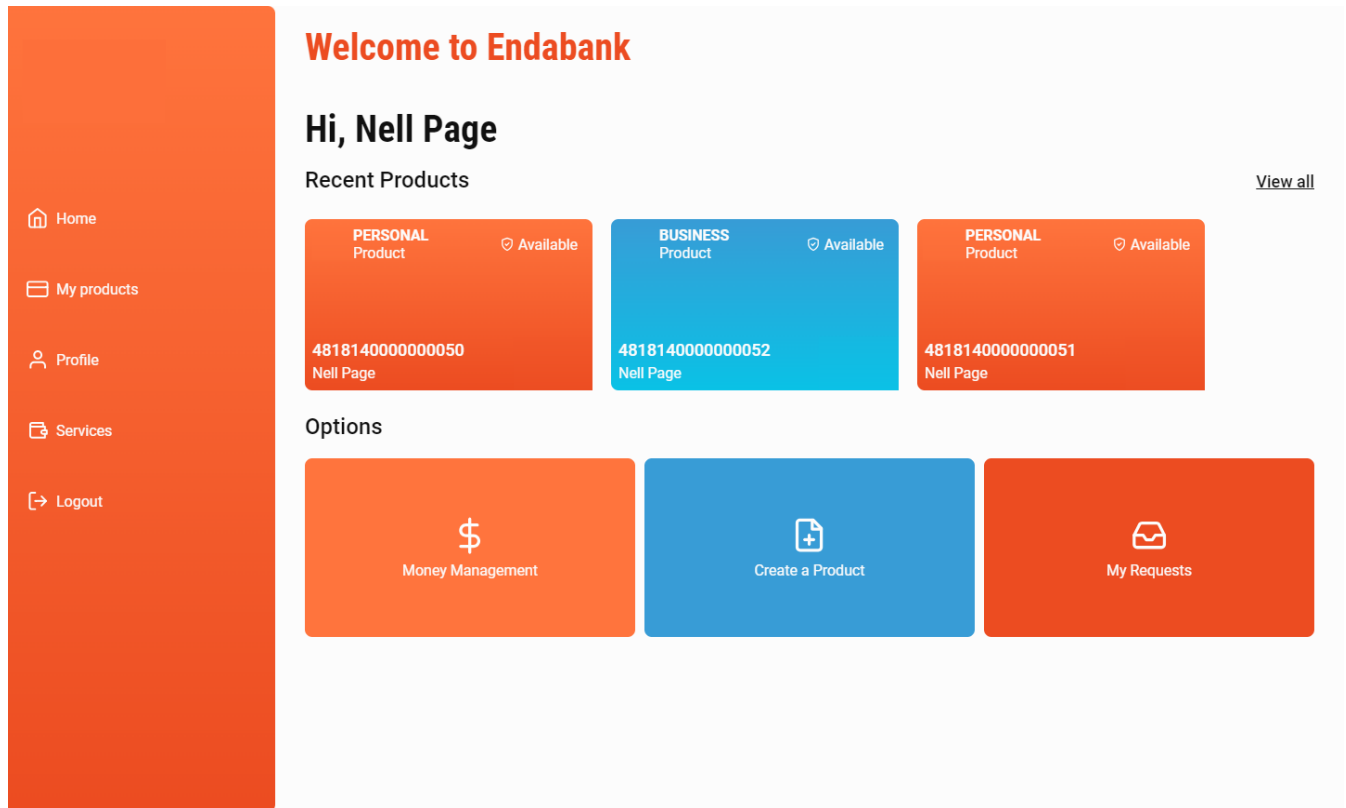


Fig. 20. Pantalla de homepage

Esta es la pantalla principal de Endabank, donde el usuario puede visualizar rápidamente cuáles son los últimos productos con los cuales realizó transacciones y también tiene acceso a los distintos servicios de Endabank. De igual manera en el lado izquierdo de la pantalla se encuentra una barra lateral que proporciona un acceso rápido a pantallas como el home, productos del usuario, etc, la cual estará disponible a lo largo de todas las pantallas del aplicativo.

d) *Creación de productos*

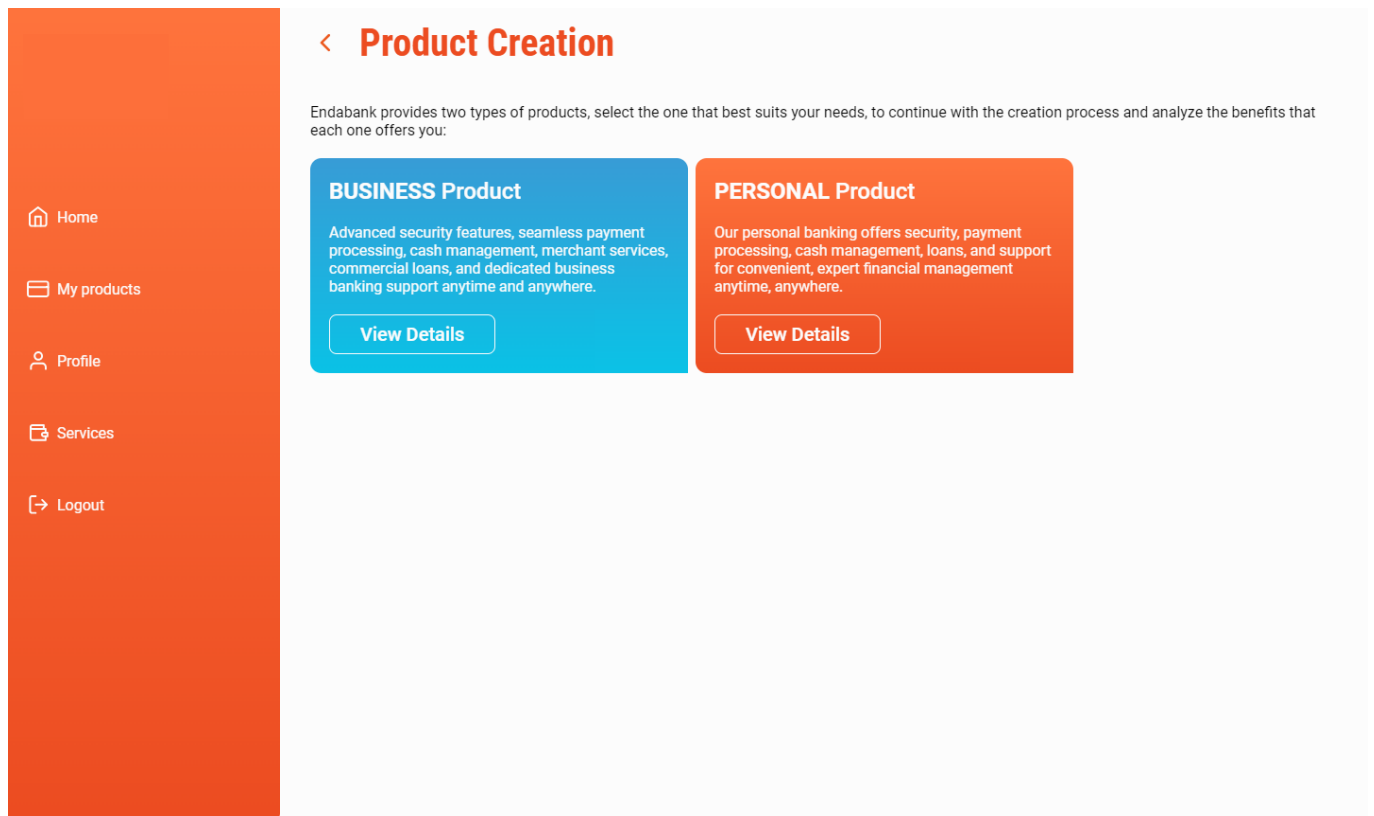


Fig. 21. Pantalla de creación de productos

En esta pantalla se le da al usuario una descripción de los distintos tipos de productos bancarios que ofrece Endabank y se le da la opción de seleccionar el tipo de producto deseado y proceder a su creación.

e) *Creación de producto personal*

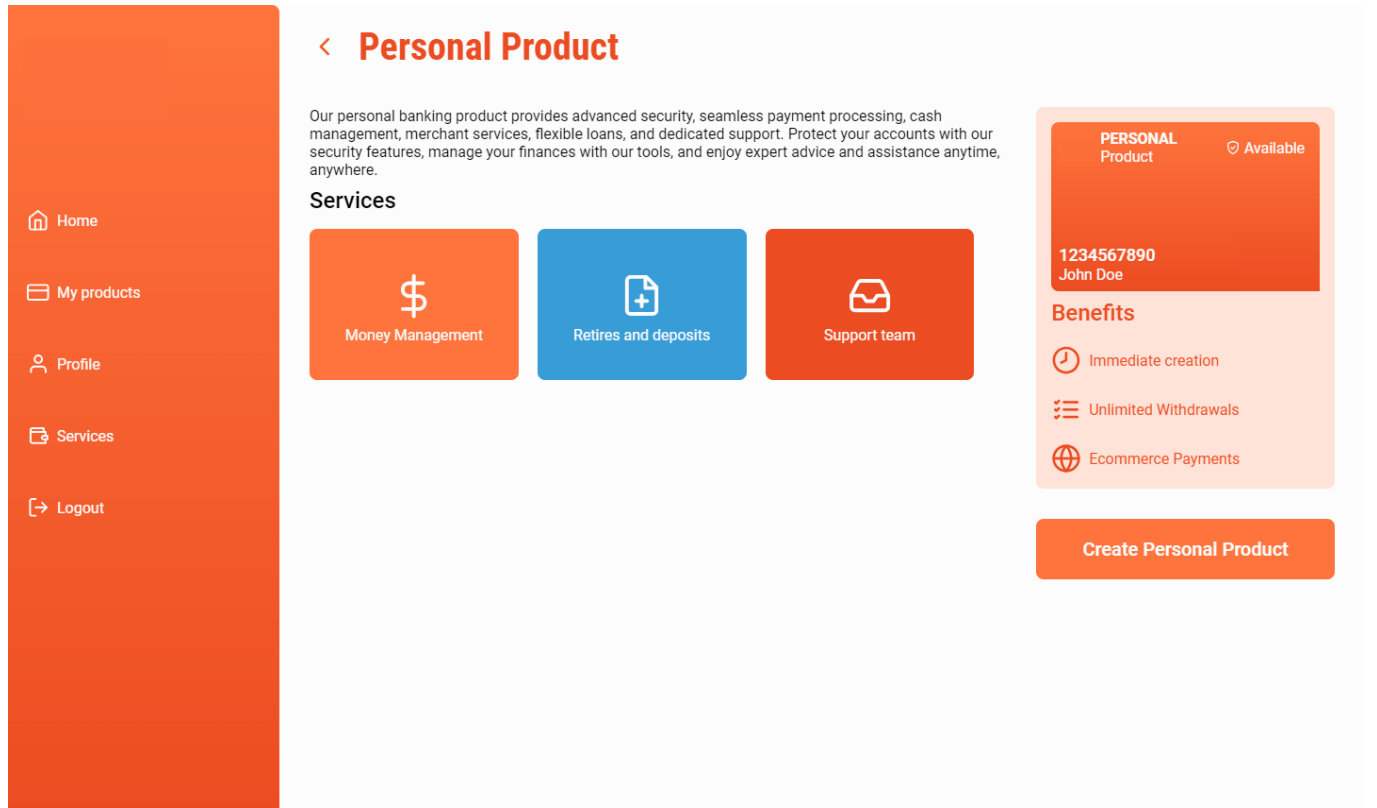


Fig. 22. Pantalla de creación de producto personal

En esta pantalla se provee una descripción del producto personal, los servicios y beneficios de los cuales dispone un usuario con este producto, y se permite la creación del mismo.

f) Creación de producto empresarial

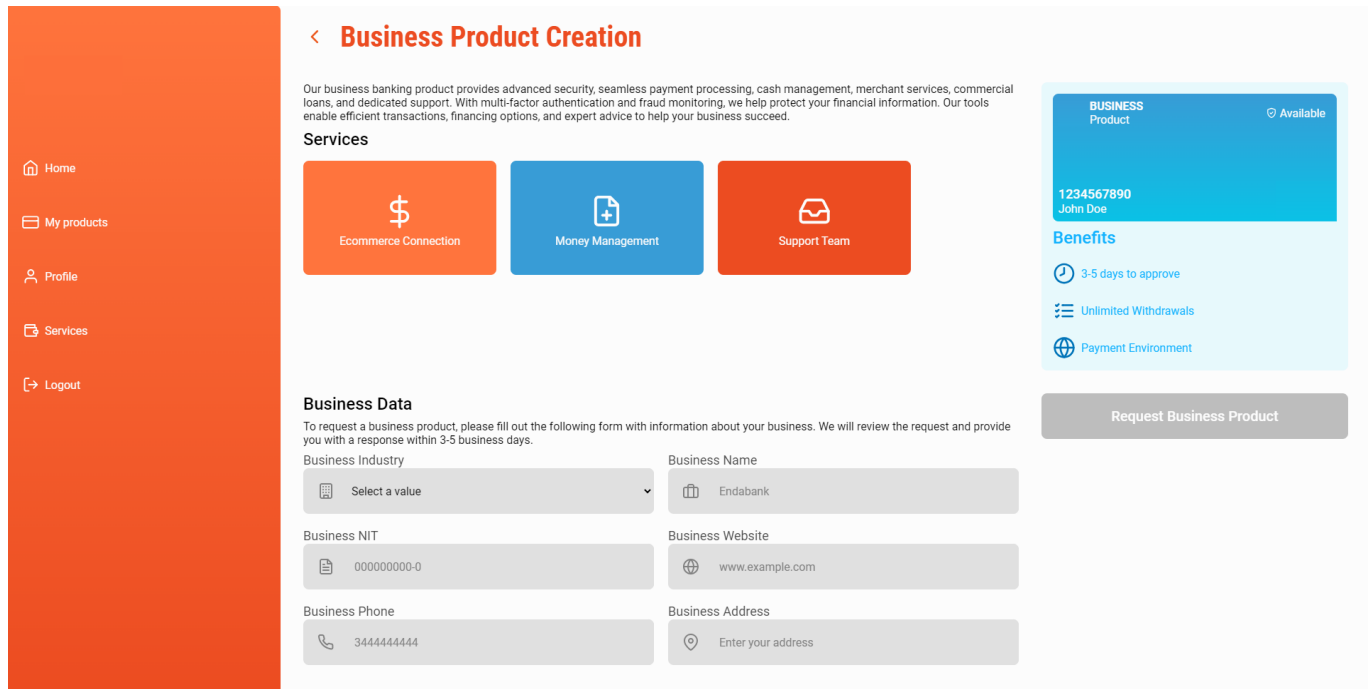


Fig. 23. Pantalla de creación de producto empresarial

En esta pantalla se provee una descripción del producto empresarial, los servicios y beneficios de los cuales dispone un usuario con este producto, y se permite la creación del mismo. Para la creación de este producto es necesario que el usuario ingrese una serie de datos del negocio que estará asociado al producto, como lo son el tipo de industria, nombre, NIT, website, teléfono y dirección.

g) Lista de productos



Fig. 24. Pantalla de lista de productos

Aquí el usuario puede ver una lista de los productos con los que dispone y puede seleccionar uno en particular para acceder a más información y a los distintos servicios que provee Endabank.

h) Detalles de un producto personal

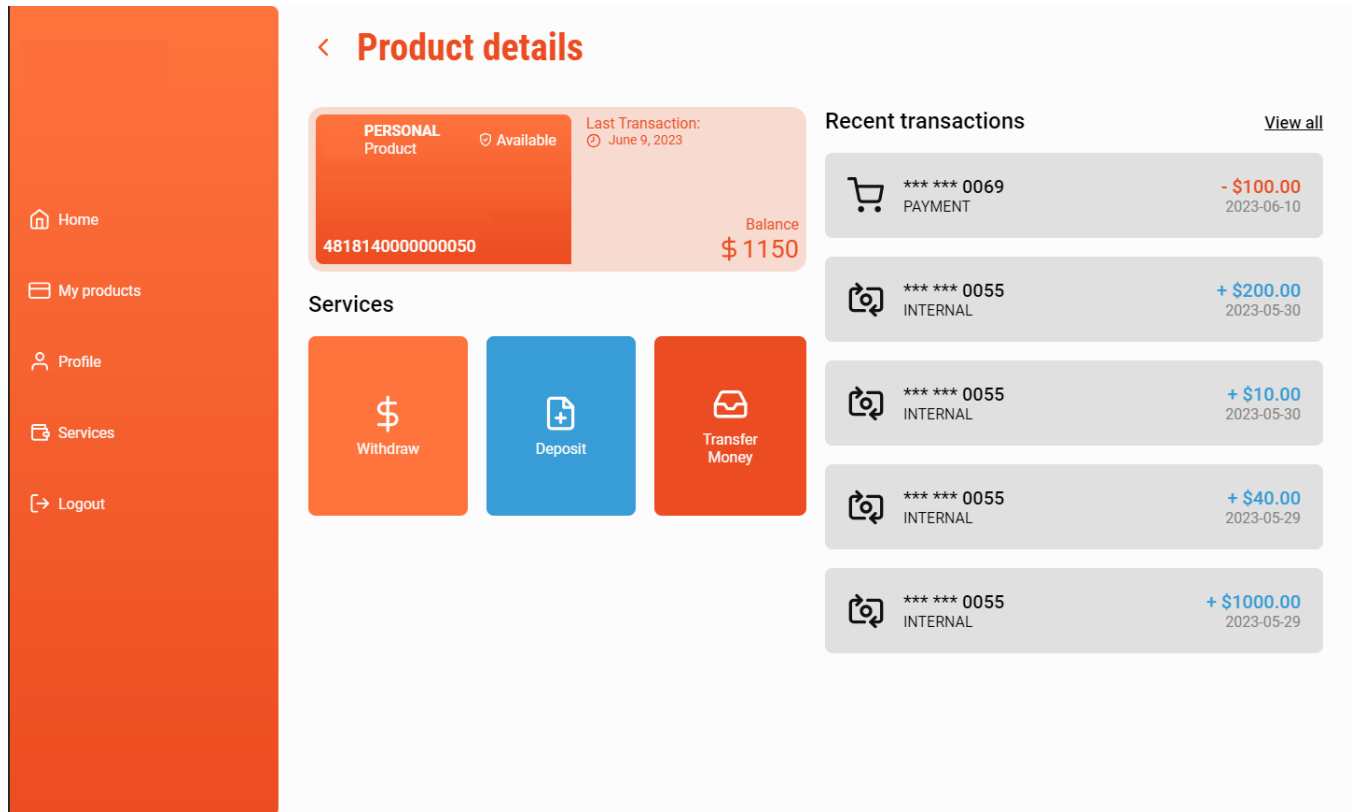


Fig. 25. Pantalla de detalles de un producto personal

En esta página el usuario puede ver información de su producto como lo son el número, estado del producto (si está disponible o no), fecha de la última transacción y el balance del mismo. Adicionalmente se provee una lista de las últimas transacciones relacionadas con el producto seleccionado y los servicios de los que puede hacer uso el usuario.

i) *Detalles de un producto empresarial*

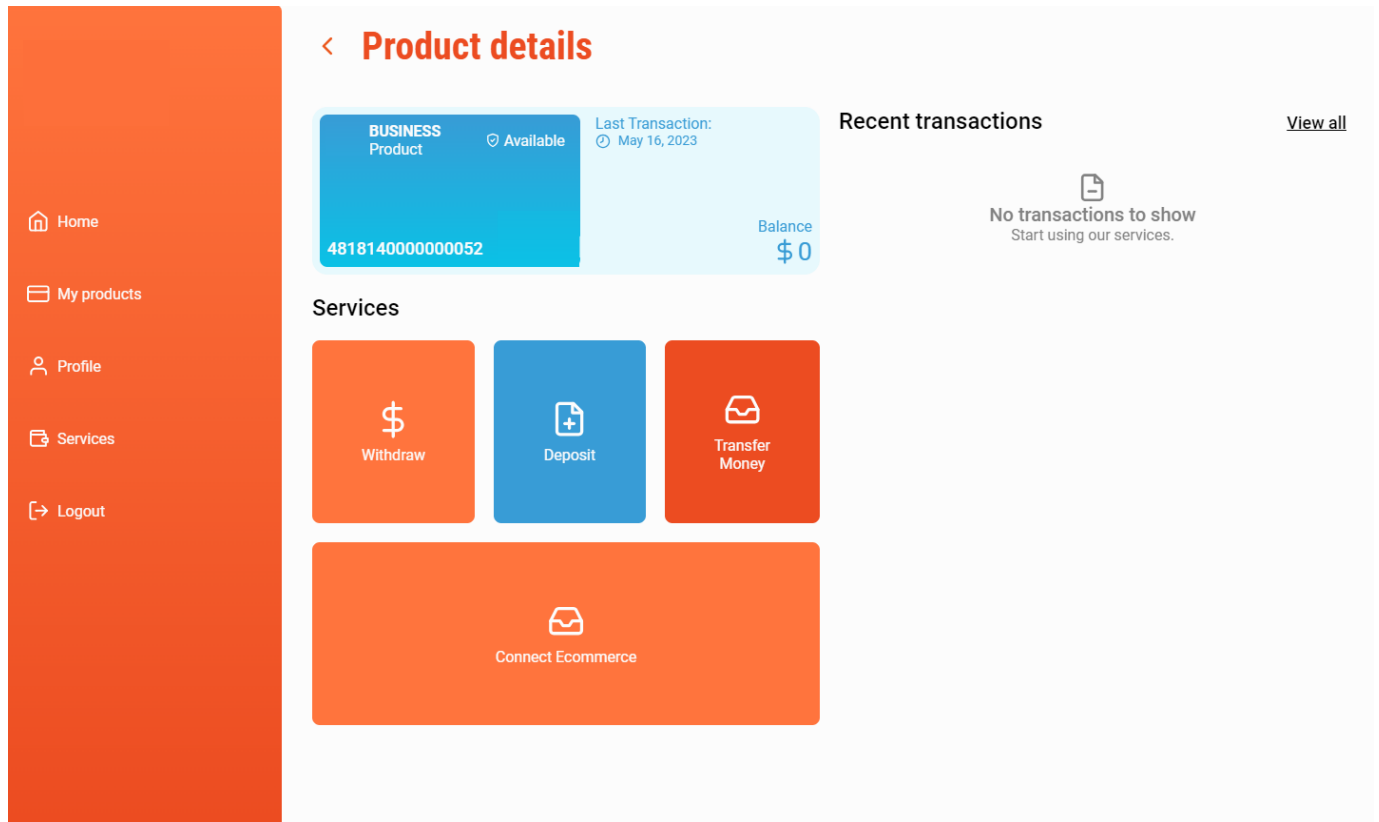


Fig. 26. Pantalla de detalles de un producto empresarial

En esta página el usuario puede ver información de su producto como lo son el número, estado del producto (si está disponible o no), fecha de la última transacción y el balance del mismo. Adicionalmente se provee una lista de las últimas transacciones relacionadas con el producto seleccionado y los servicios de los que puede hacer uso el usuario. Hay un servicio adicional en el caso de los productos empresariales, que es la opción para realizar la conexión con su e-commerce.

j) Servicio de depósito

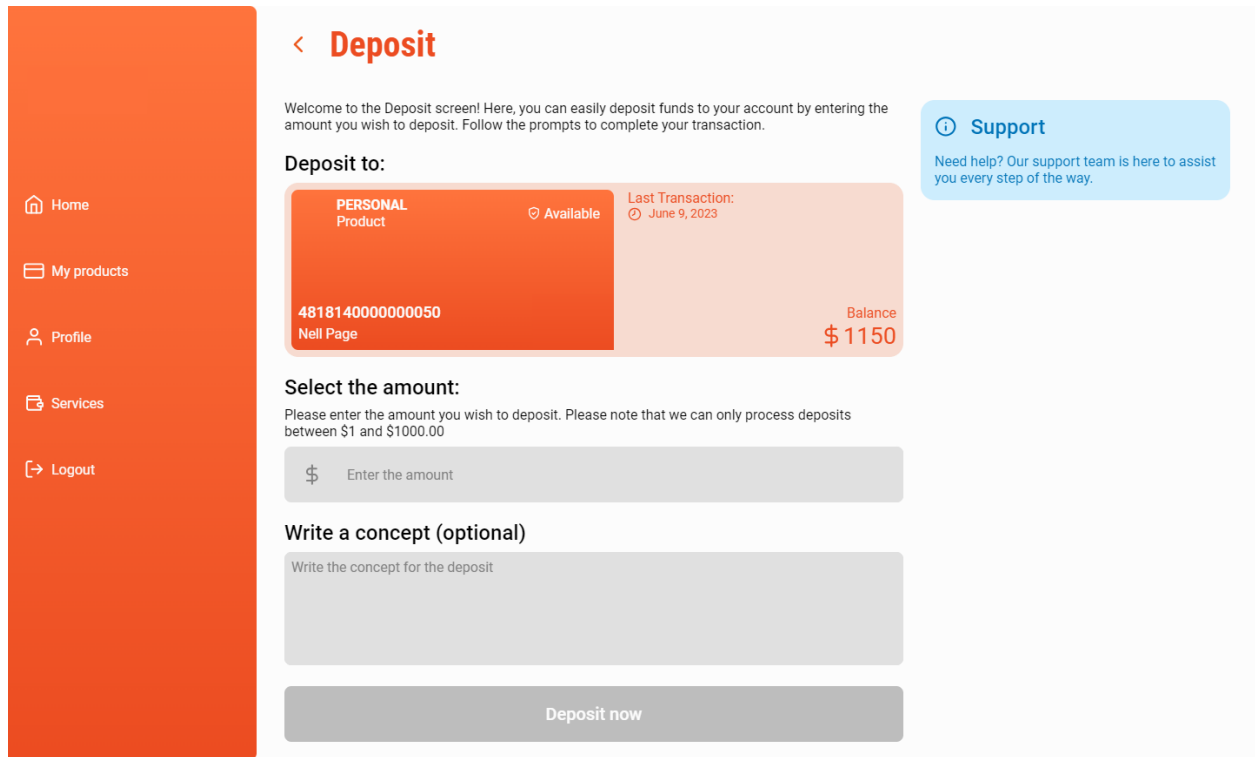


Fig. 27. Pantalla de servicio de depósito

Uno de los servicios con los cuales cuentan los usuarios de Endabank, es el de depósito, a través del cual los usuarios pueden depositar dinero en sus cuentas bancarias. En esta pantalla se le muestra al usuario el producto al cual se le deposita el dinero y se le solicita que ingrese la cantidad a depositar y un concepto, el cual es opcional.

k) Servicio de transacción

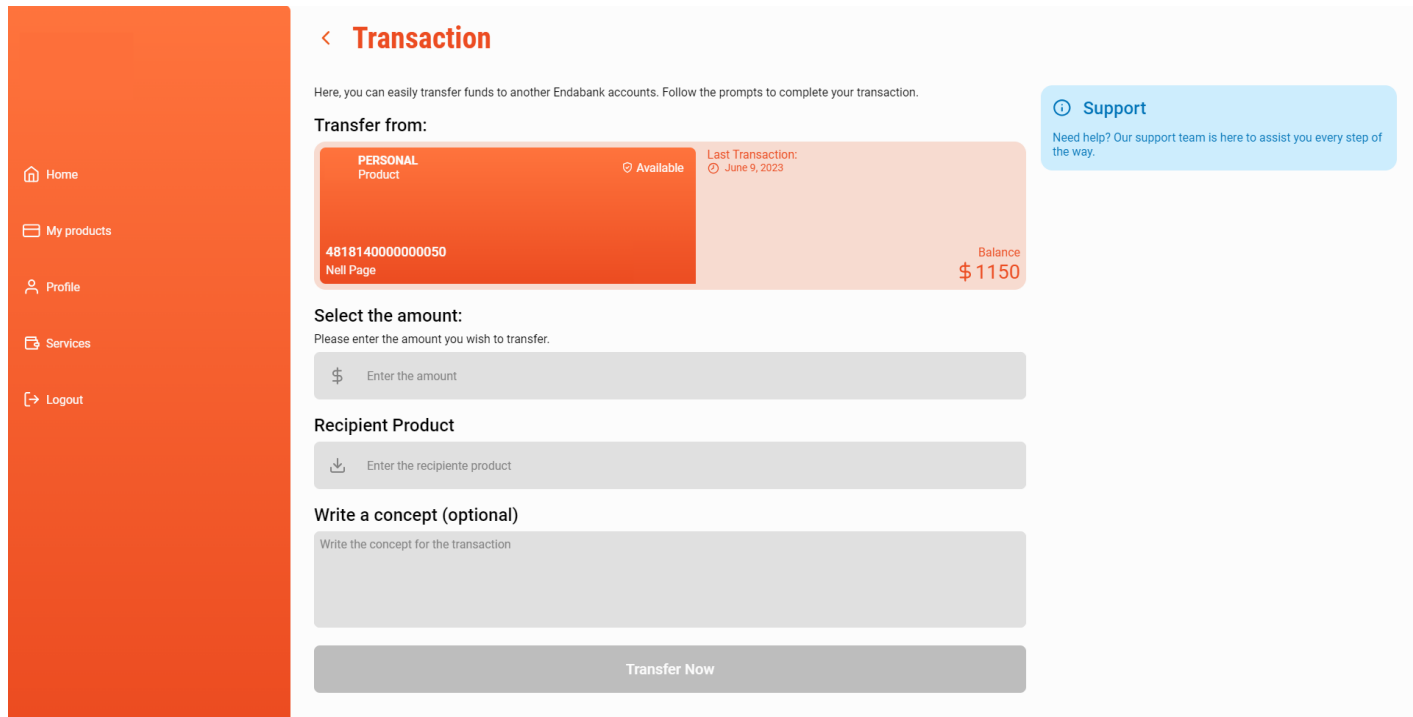


Fig. 28. Pantalla de servicio de transacción

Uno de los servicios con los cuales cuentan los usuarios de Endabank, es el de transacción, a través del cual los usuarios pueden enviar dinero a otras cuentas de Endabank. En esta pantalla se le solicita que ingrese el producto destino, la cantidad de dinero a enviar y un concepto, el cual es opcional.

l) Resultado de una transacción

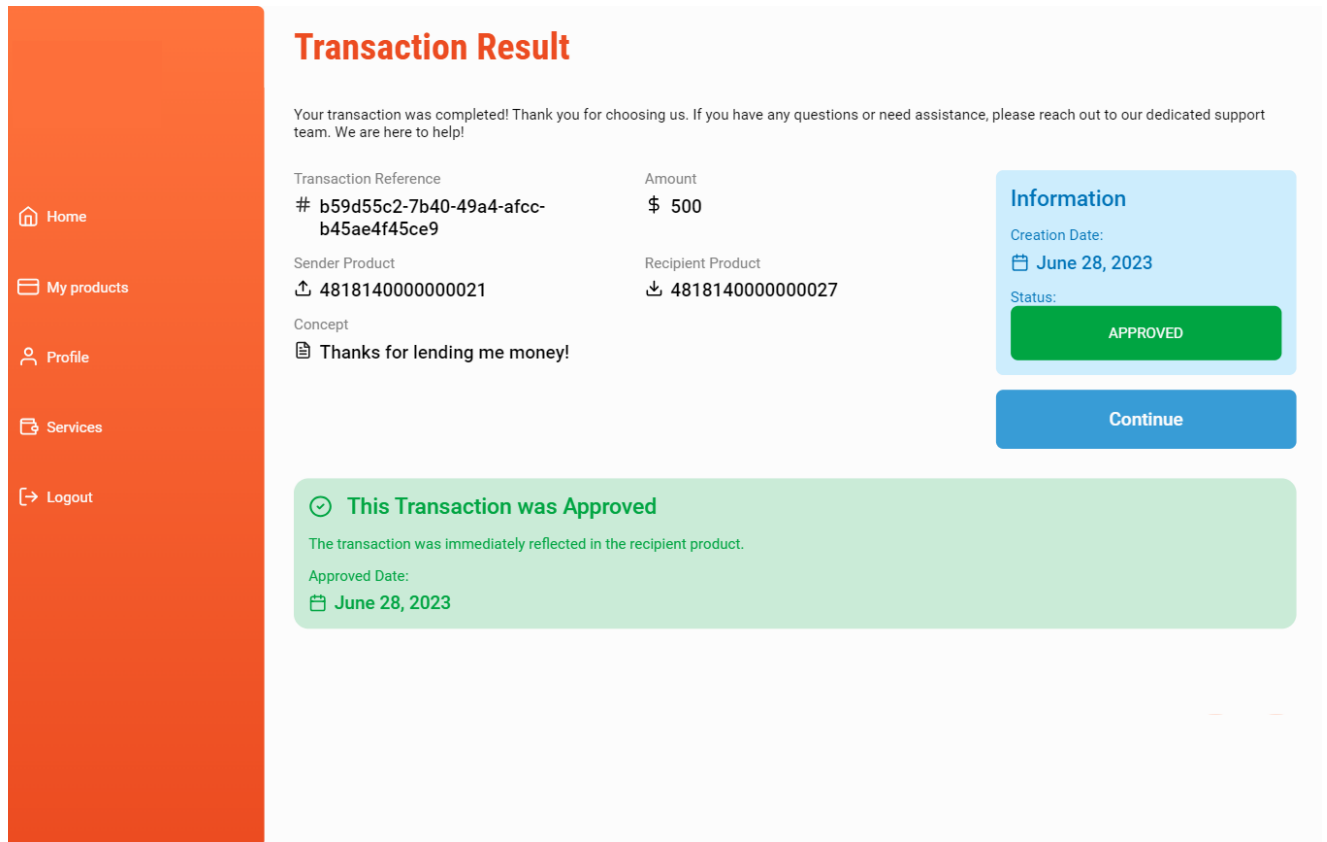


Fig. 29. Pantalla de resultado de una transacción

Una vez diligenciada la información en el formulario de transacción, se le muestra al usuario el resultado de la misma, en caso de que sea exitosa o fallida y se le da la opción de continuar usando el aplicativo.

m) Solicitudes de producto empresarial del usuario

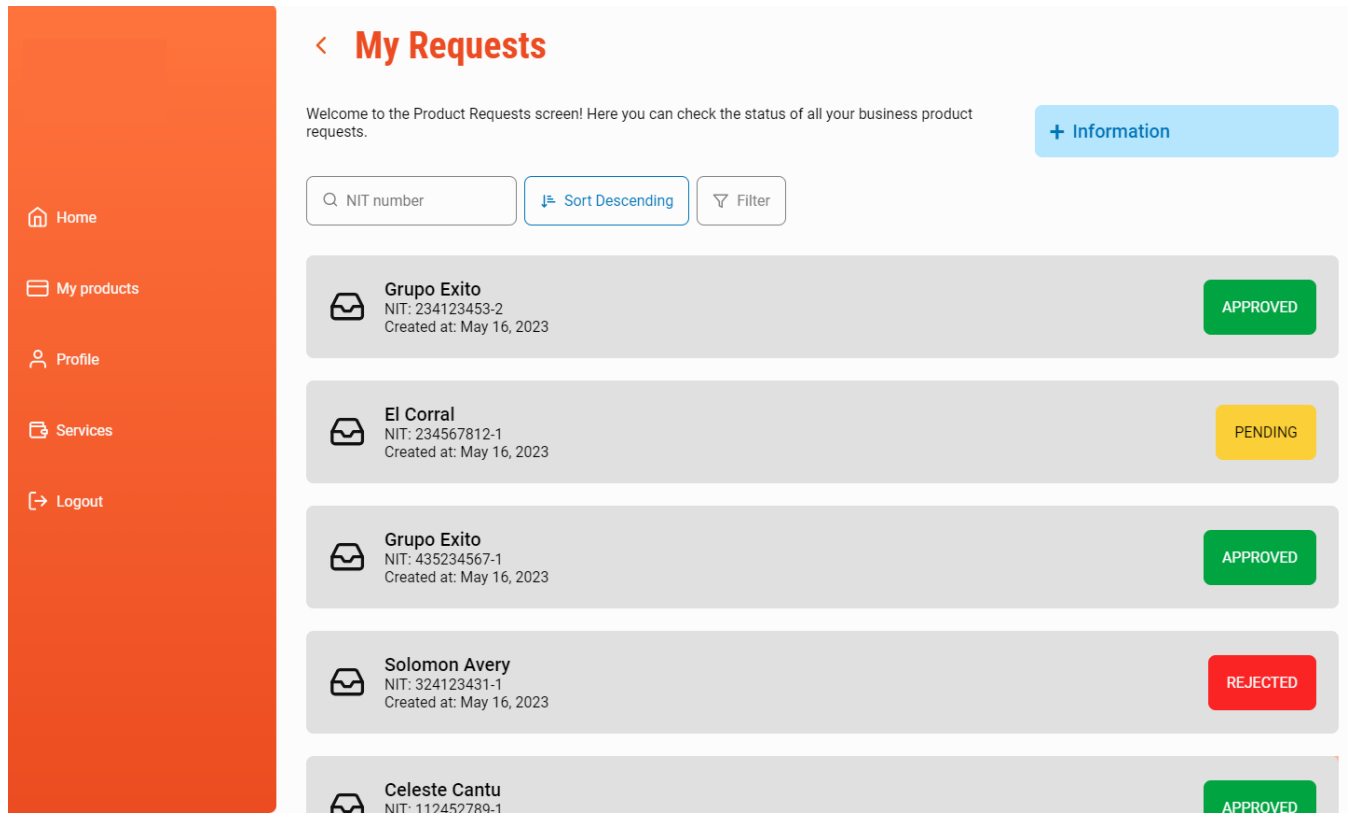


Fig. 30. Pantalla de solicitudes de producto empresarial del usuario

Cuando se diligencia el formulario de creación de productos, se crea una solicitud la cual debe ser revisada y aprobada/rechazada por un administrador de Endabank. Debido a esto, el usuario puede acceder a una lista de las solicitudes que ha hecho y así revisar su estado y los detalles de la misma.

n) *Detalles de una solicitud de producto empresarial del usuario*

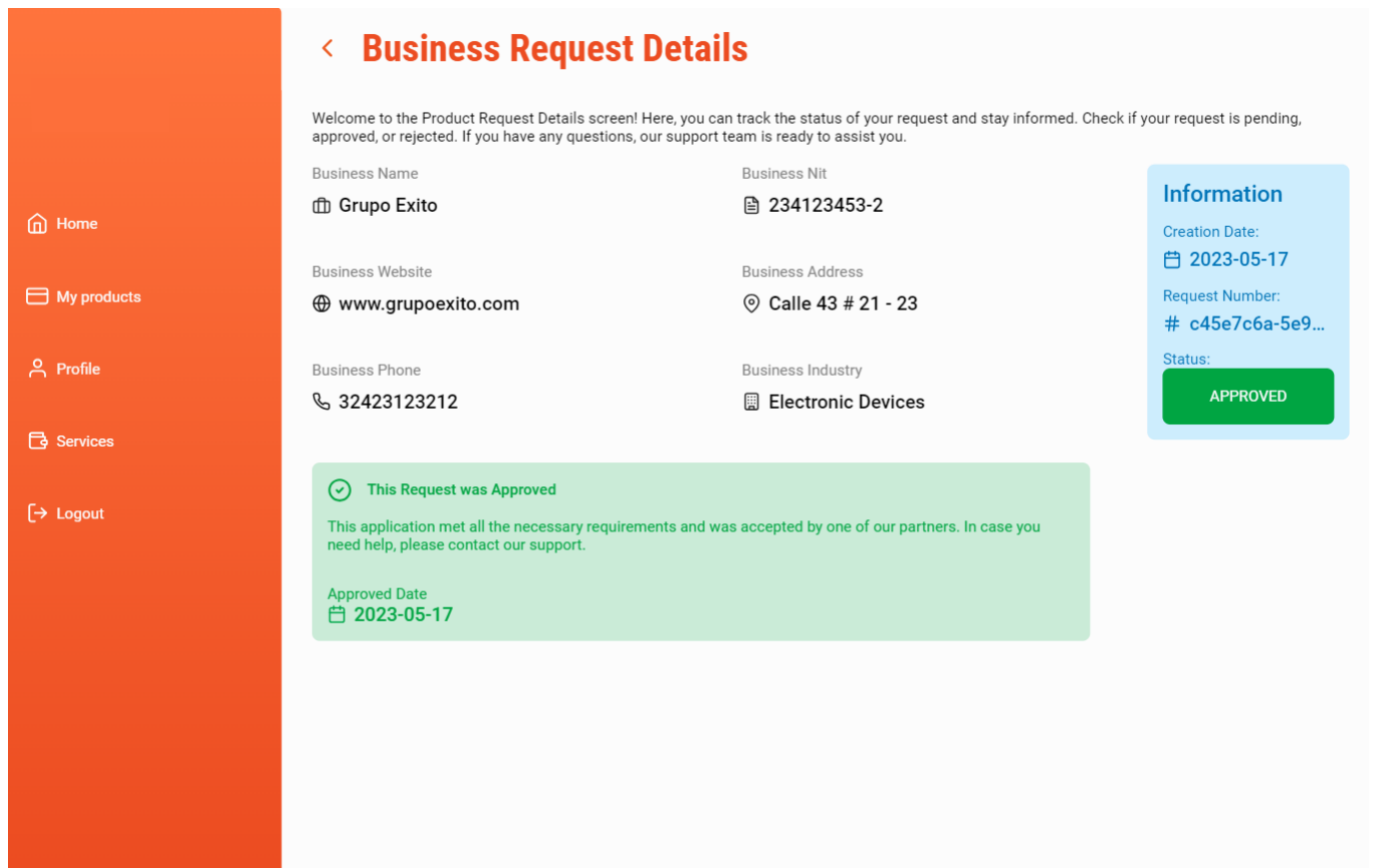


Fig. 31. Pantalla de detalles de una solicitud de producto empresarial del usuario

Al seleccionar una solicitud de la lista, el usuario puede revisar los datos suministrados y conocer el estado de la solicitud, siendo los estados posibles aprobada, pendiente o rechazada.

o) *Conexión con e-commerce*

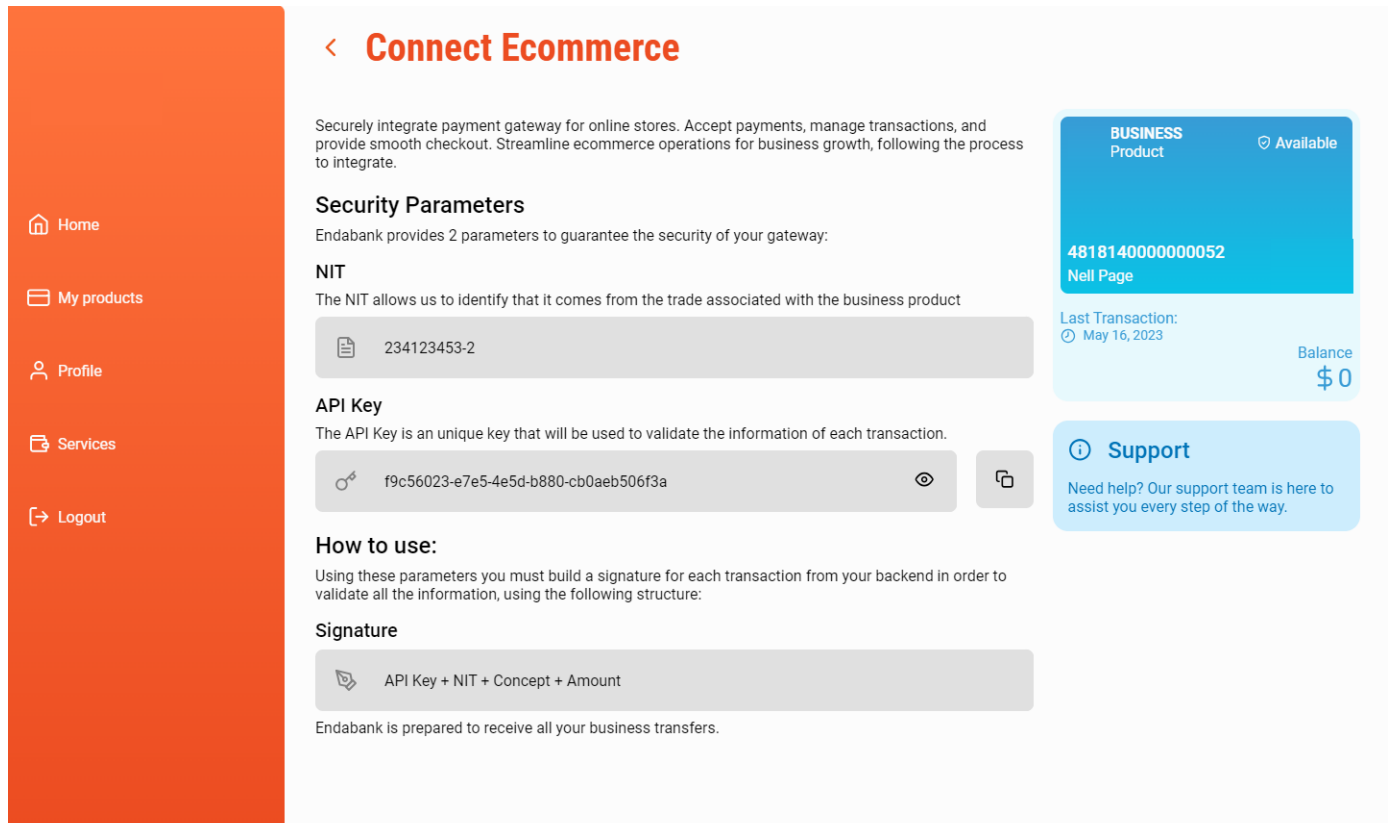


Fig. 32. Pantalla de conexión con e-commerce

Todo producto empresarial cuenta con la opción de permitir su conexión con un e-commerce. En esta pantalla se le indica al usuario la información y pasos necesarios para realizar la implementación de la conexión en su e-commerce.

Para poder probar la conexión entre un e-commerce y Endabank, se diseñó un e-commerce de prueba. Para realizar esta conexión se tuvieron en cuenta algunas consideraciones y decisiones de diseño:

- Para la conexión de un negocio con Endabank, tanto el NIT de la empresa y un API Key son requeridos. Este API Key es generado cuando un producto empresarial es creado y puede ser verificado/obtenido cuando el dueño del negocio ingresa a Endabank y de esta manera se provee una capa de seguridad adicional.

- Se utiliza el algoritmo de hash de una sola vía SHA-256 para la creación de la signature que se ensambla con lo siguiente: ApiKey + NIT + Concepto + Monto de la transacción. En el backend de Endabank se ensambla el signature nuevamente y se recibe el signature enviado en la petición, los cuales son comparados para verificar la veracidad de la información. Esto será útil en futuras versiones para prevenir que una persona altere la información del pago desde el e-commerce.
- Cualquier e-commerce que quiera usar Endabank como método de pago, debería tener un backend dedicado, desde el cual la comunicación con Endabank sea realizada de forma segura.
- Actualmente por términos prácticos, la conexión entre el e-commerce y Endabank se está realizando de forma síncrona, esto es que ambos actores se comunican en real time, esperando confirmaciones y respuestas inmediatas para avanzar en el proceso de pago teniendo como método de pago Endabank.

Endabank Payment

Follow the next steps to complete your payment

1 Product Selection — **2** Confirm Transaction — **3** Transaction Result

Select a product
The product you select will be the one used to pay for the ecommerce purchase

Product	Last Transaction	Balance
PERSONAL Product Available 48181400000 Jason Johnson	June 14, 2023	\$ 800000
PERSONAL Product Available 48181400000 Jason Johnson	June 14, 2023	\$ 501022.14
PERSONAL Product Available 48181400000 Jason Johnson	June 27, 2023	\$ 9133.39
PERSONAL Product Available 48181400000 Jason Johnson	June 2, 2023	\$ 4000
PERSONAL Product Available 48181400000 Jason Johnson	June 27, 2023	
PERSONAL Product Available 48181400000 Jason Johnson	May 30, 2023	

Payment Information

Business: Jabra
Concept: JBass Pro
Total Amount: \$ 299.99

Continue

Fig. 33. Pantalla de pago, paso 1

Cuando el usuario es redirigido a Endabank para proceder con el pago de su compra, este será llevado a un formulario de 3 pasos, donde primero debe seleccionar el producto con el cual desea hacer el pago.

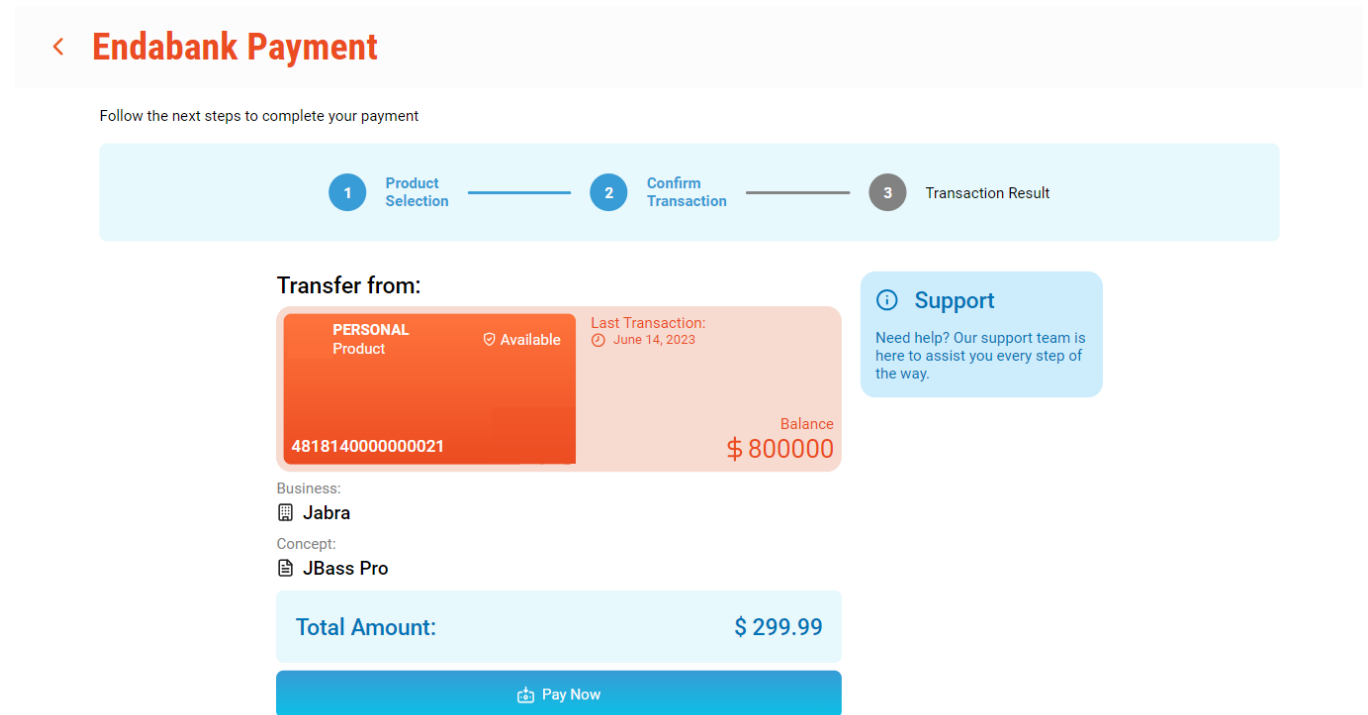


Fig. 34. Pantalla de pago, paso 2

Una vez seleccionado el producto, se le muestra un pequeño resumen del pago que está por realizar para que confirme toda la información y proceder al pago.

Endabank Payment

Follow the next steps to complete your payment

- 1 Product Selection
- 2 Confirm Transaction
- 3 Transaction Result

Your transaction was completed! Thank you for choosing us. If you have any questions or need assistance, please reach out to our dedicated support team. We are here to help!

Transaction Reference # 957b5585-1543-42a4-a545-a4614c17ebfe	Amount \$ 299.99
Sender Product 📦 4818140000000021	Recipient Product 📦 Jabra
Concept 📦 JBass Pro	

Information

Creation Date:
📅 June 28, 2023

Status:
APPROVED

[Back to Commerce](#)

👍 **This Transaction was Approved**

The transaction was immediately reflected in the recipient product.

Approved Date:
📅 June 28, 2023

Fig. 35. Pantalla de pago, paso 3

Una vez pagado, se le muestra al usuario el resultado de la transacción con toda la información relacionada al pago y se le da la opción de volver al e-commerce.

p) Correo electrónico de notificación de una transacción

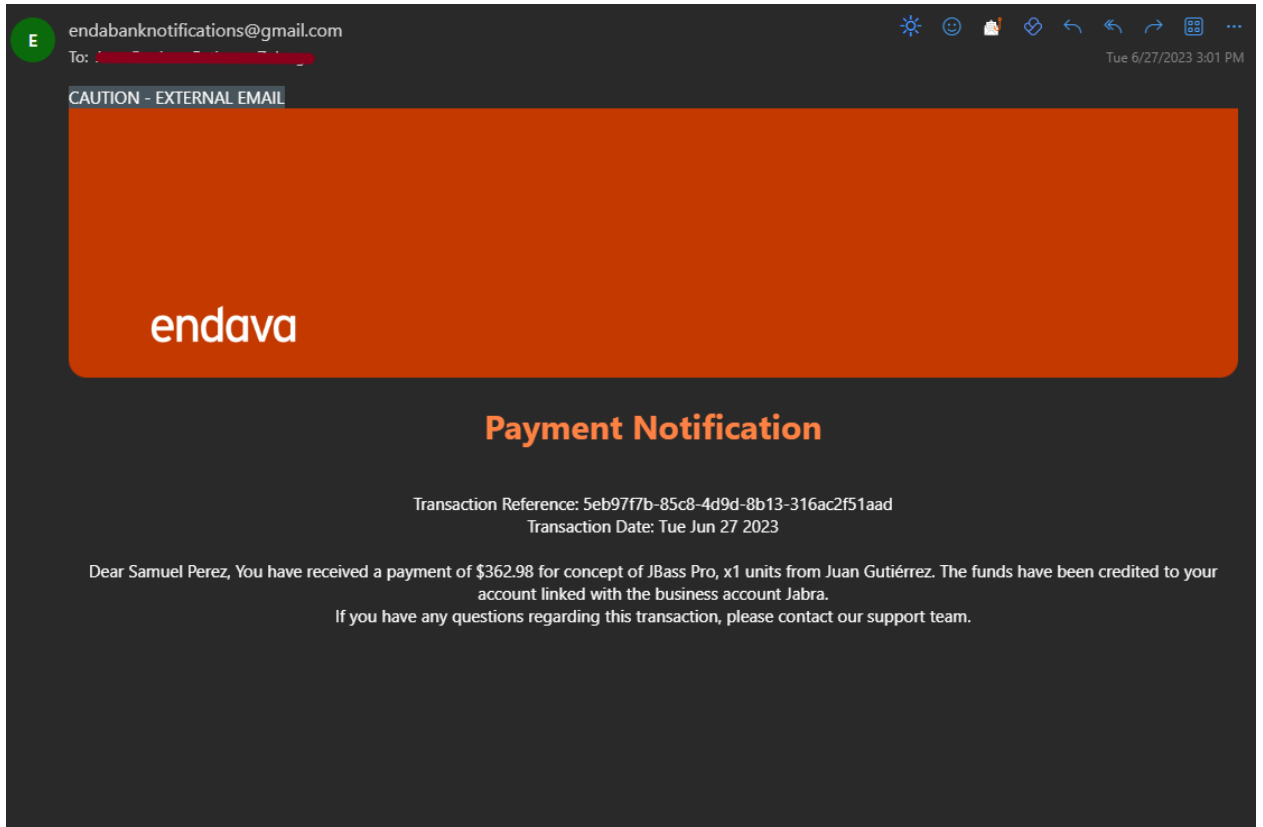


Fig. 36. Notificación de pago desde e-commerce

Una vez creada la transacción, ya sea un depósito, una transacción interna o un pago desde un e-commerce, se envía un correo electrónico tanto al dueño del producto de origen (en depósitos no aplica) como al dueño del producto de destino. En la **Fig. 36** se muestra un ejemplo de un correo remitido tras una compra proveniente del comercio electrónico.

2) *Flujo de administrador*: Agrupación de algunas funcionalidades de administradores de Endabank.

a) *Home del administrador*

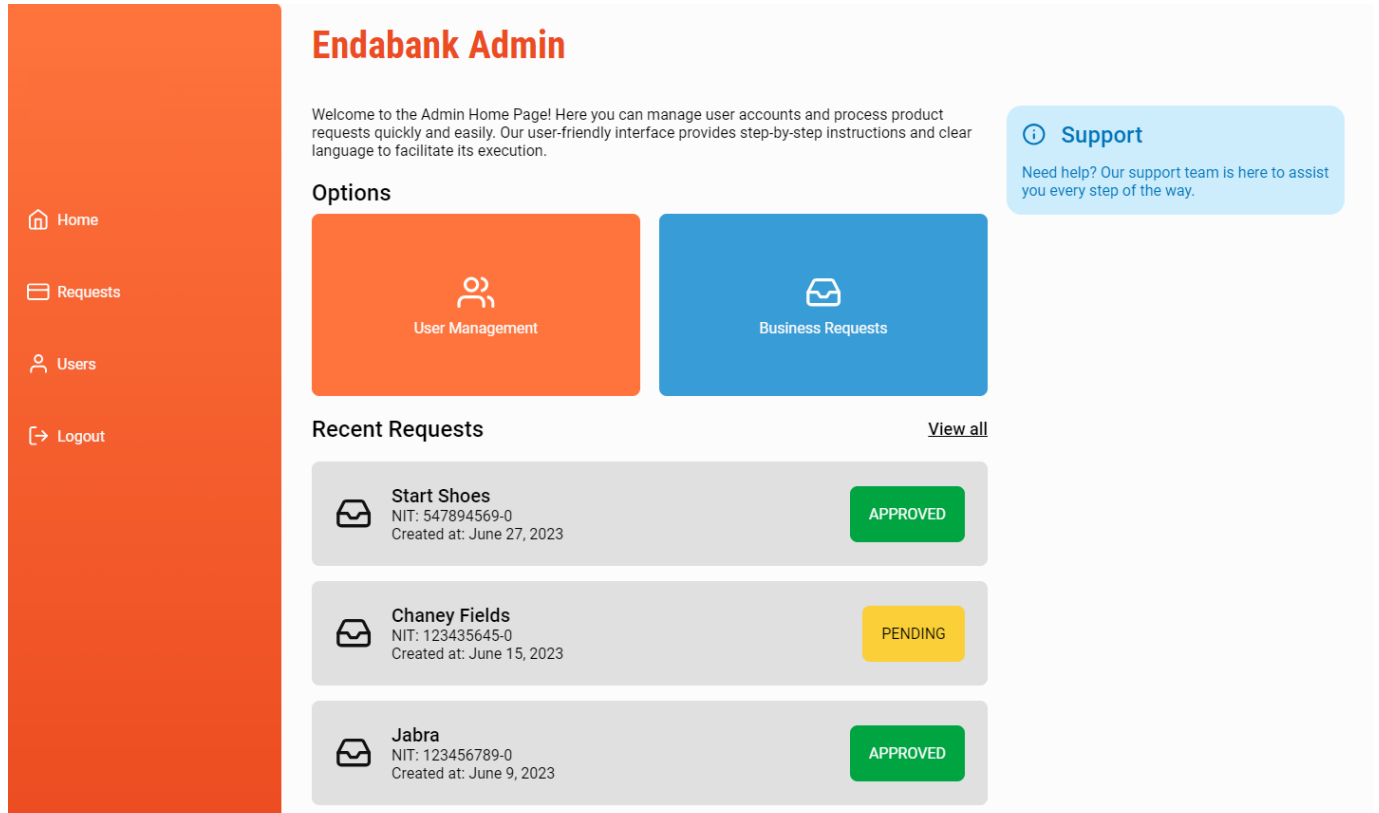


Fig. 37. Pantalla de homepage del administrador

Esta es la pantalla principal de Endabank para un administrador, donde el mismo puede visualizar rápidamente cuáles son las últimas transacciones creadas y también tiene acceso a los distintos servicios de administración. De igual manera en el lado izquierdo de la pantalla se encuentra una barra lateral que proporciona un acceso rápido a pantallas como el home, solicitudes, usuarios etc, la cual estará disponible a lo largo de todas las pantallas del aplicativo.

b) *Lista de solicitudes del administrador*

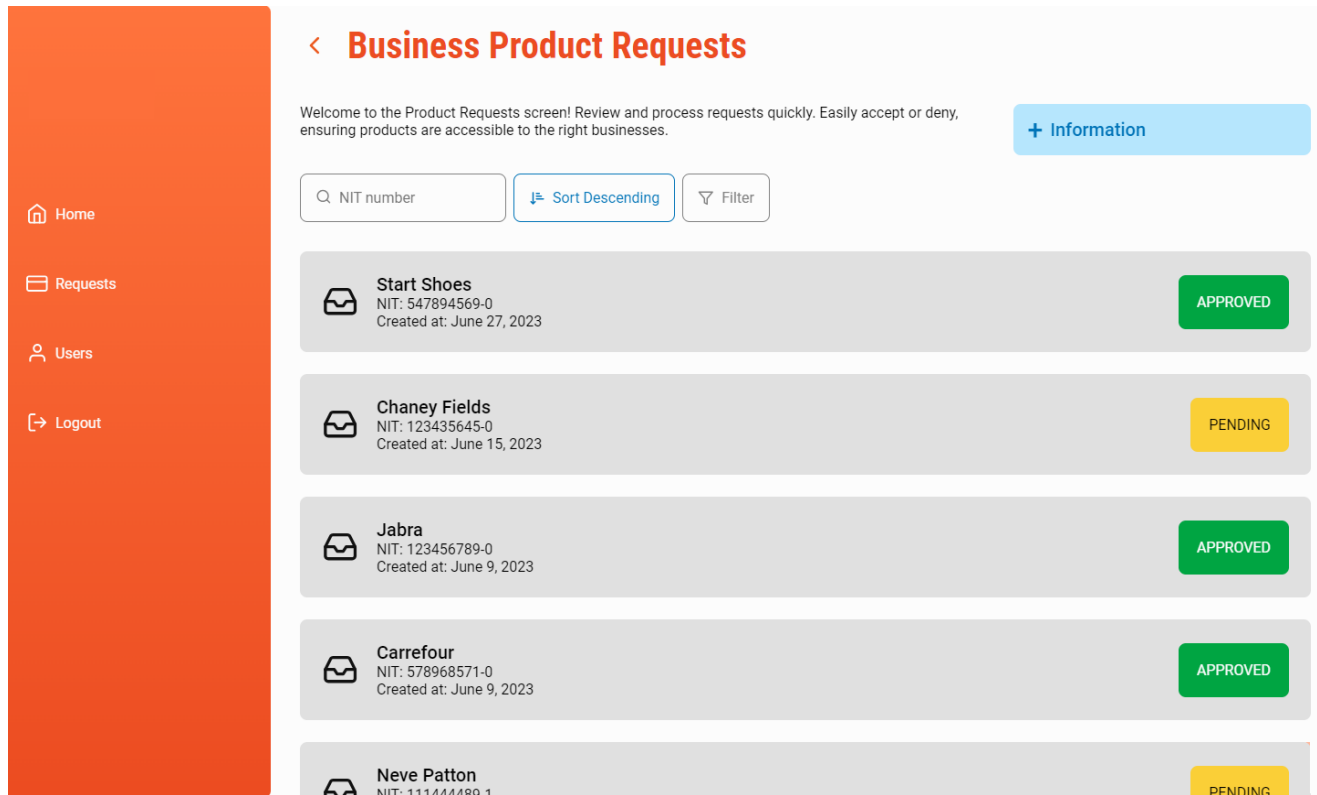


Fig. 38. Pantalla de lista de solicitudes del administrador

En esta pantalla el administrador puede visualizar una lista de todas las solicitudes generadas y podrá filtrar por NIT de la empresa, organizarlas ascendente o descendientemente y filtrar por los estados posibles de las solicitudes, que son: aprobado, pendiente y rechazado. De igual forma, puede seleccionar una en particular para ver los detalles y proceder a su evaluación.

c) *Detalles de una solicitud de producto empresarial, vista del administrador*

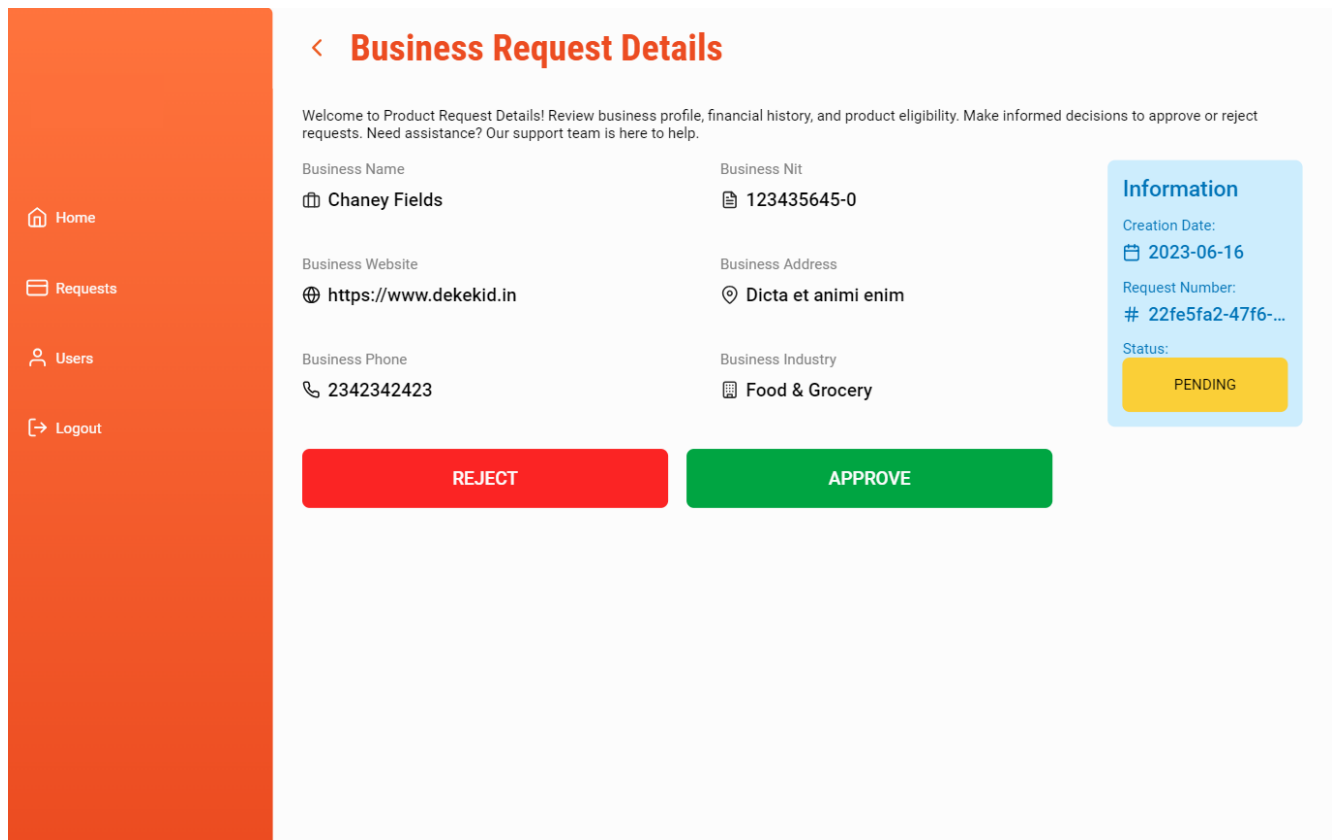


Fig. 39. Pantalla de detalles de una solicitud de producto empresarial, vista del administrador

Al seleccionar una solicitud de la lista, el administrador puede ver toda la información del negocio relacionado con la solicitud y en caso de que esta esté en el estado de “pendiente”, puede aprobarla o rechazarla.

d) Lista de usuarios

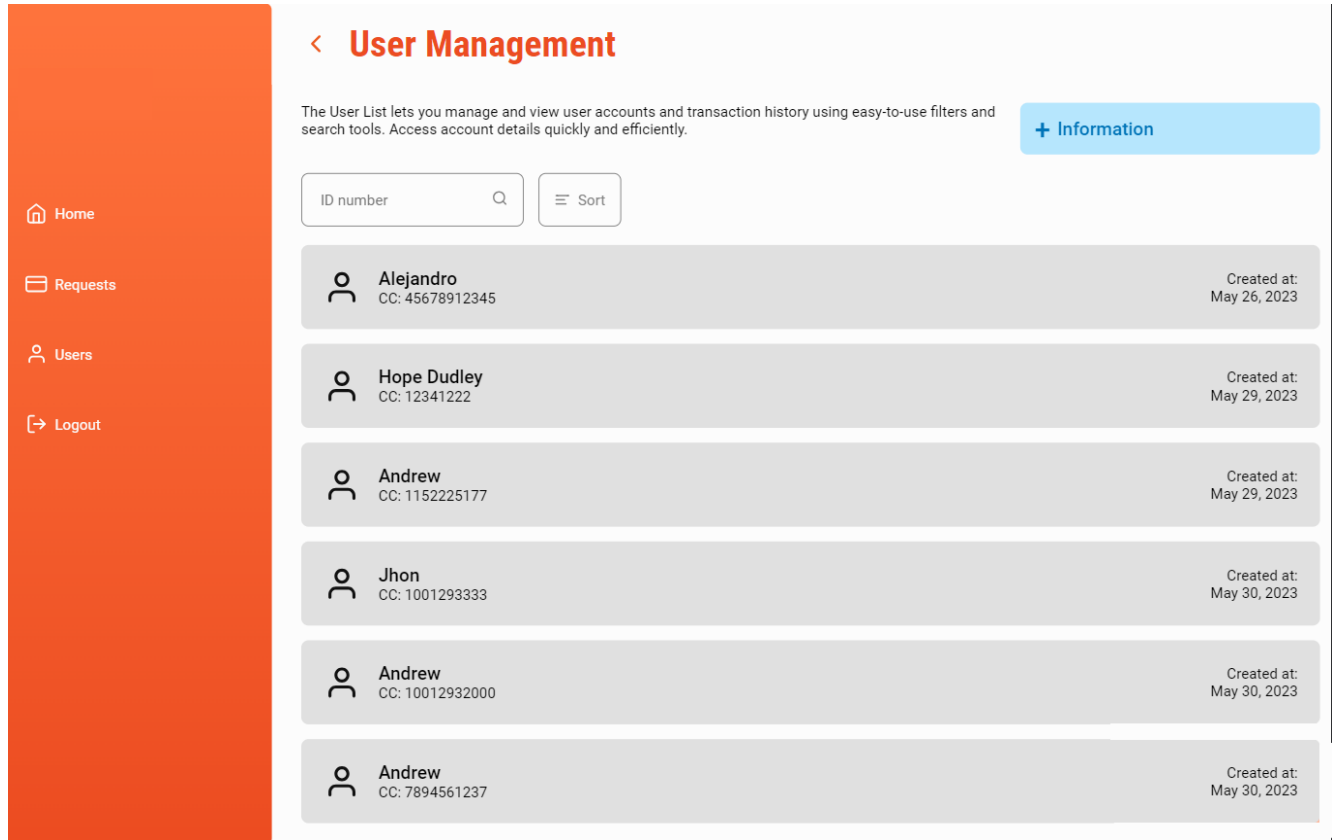


Fig. 40. Pantalla de lista de usuarios

En esta pantalla el administrador puede hacer manejo de los usuarios del sistema y podrá buscarlos por identificación y ordenarlos por fecha de creación o identificación, ya sea ascendente o descendente.

e) *Detalles del producto personal de un usuario, vista de administrador*

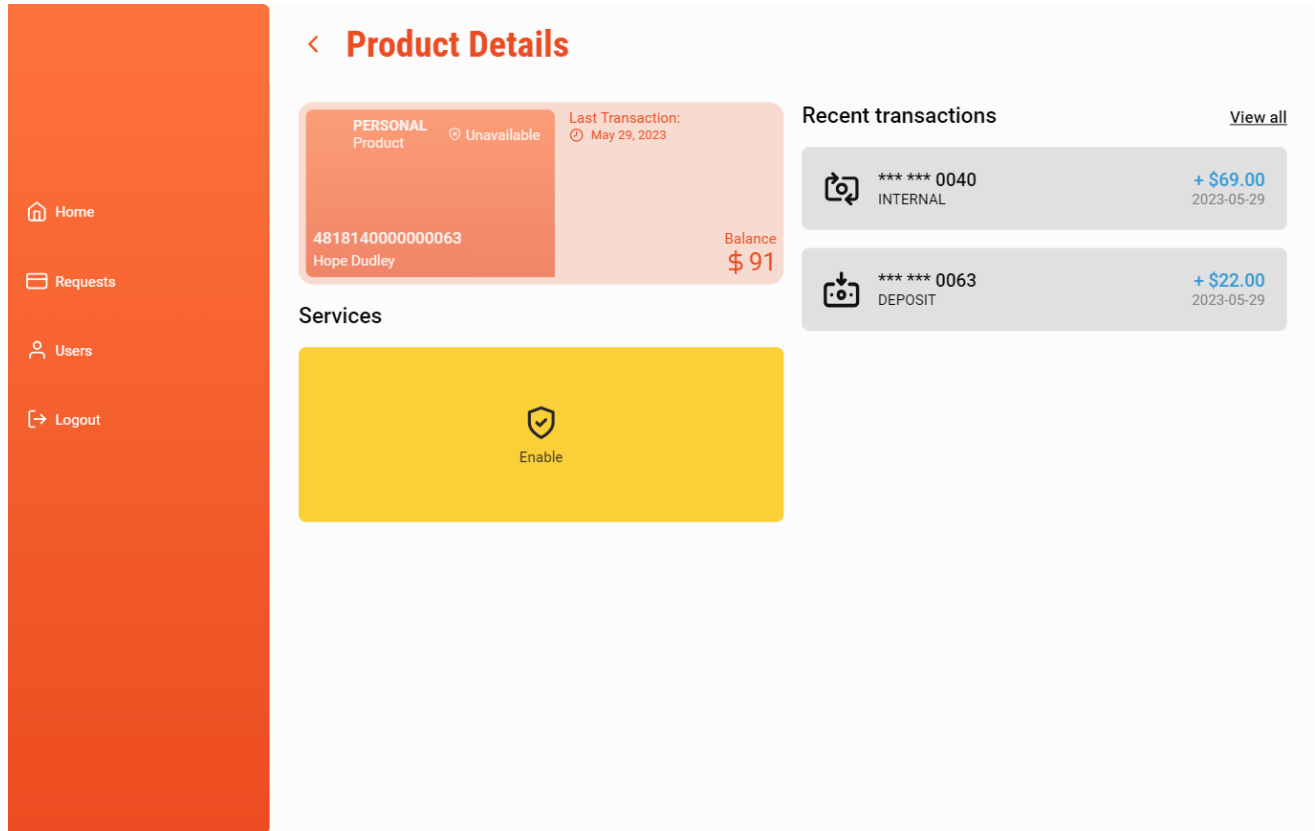


Fig. 41. Pantalla de detalles del producto personal de un usuario, vista de administrador

Al seleccionar un producto de un usuario, el administrador puede ver detalles del producto, como a quién pertenece, el número del producto, fecha de la última transacción, el balance, un listado de las últimas transacciones y podrá habilitarlo/deshabilitarlo en caso de ser necesario.

f) *Detalles del producto empresarial de un usuario, vista de administrador*

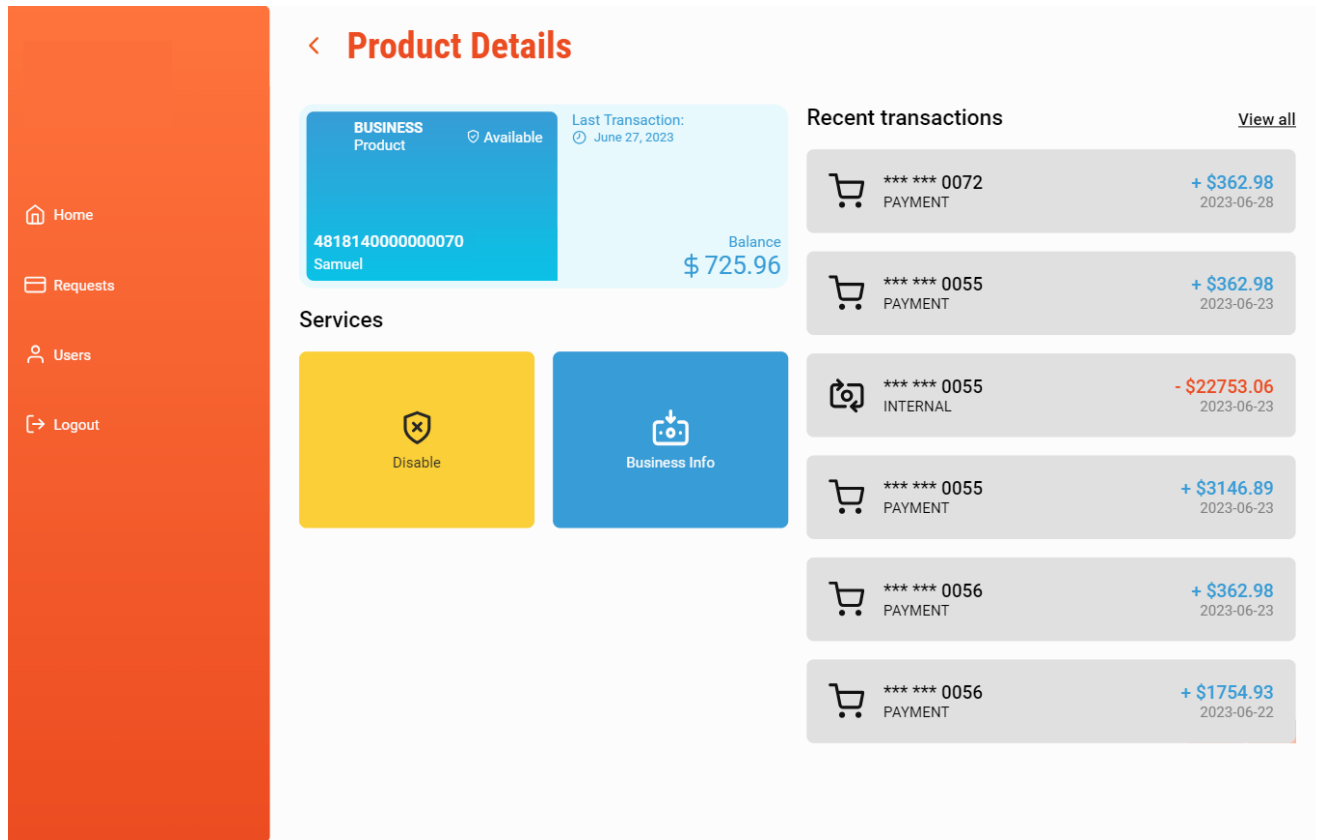


Fig. 42. Pantalla de detalles del producto empresarial de un usuario, vista de administrador

Al seleccionar un producto de un usuario, el administrador puede ver detalles del producto, como a quién pertenece, el número del producto, fecha de la última transacción, el balance, un listado de las últimas transacciones y podrá habilitarlo/deshabilitarlo en caso de ser necesario. En el caso de un producto empresarial, también podrá visualizar la información del negocio asociado al producto.

G. Infraestructura de la plataforma web

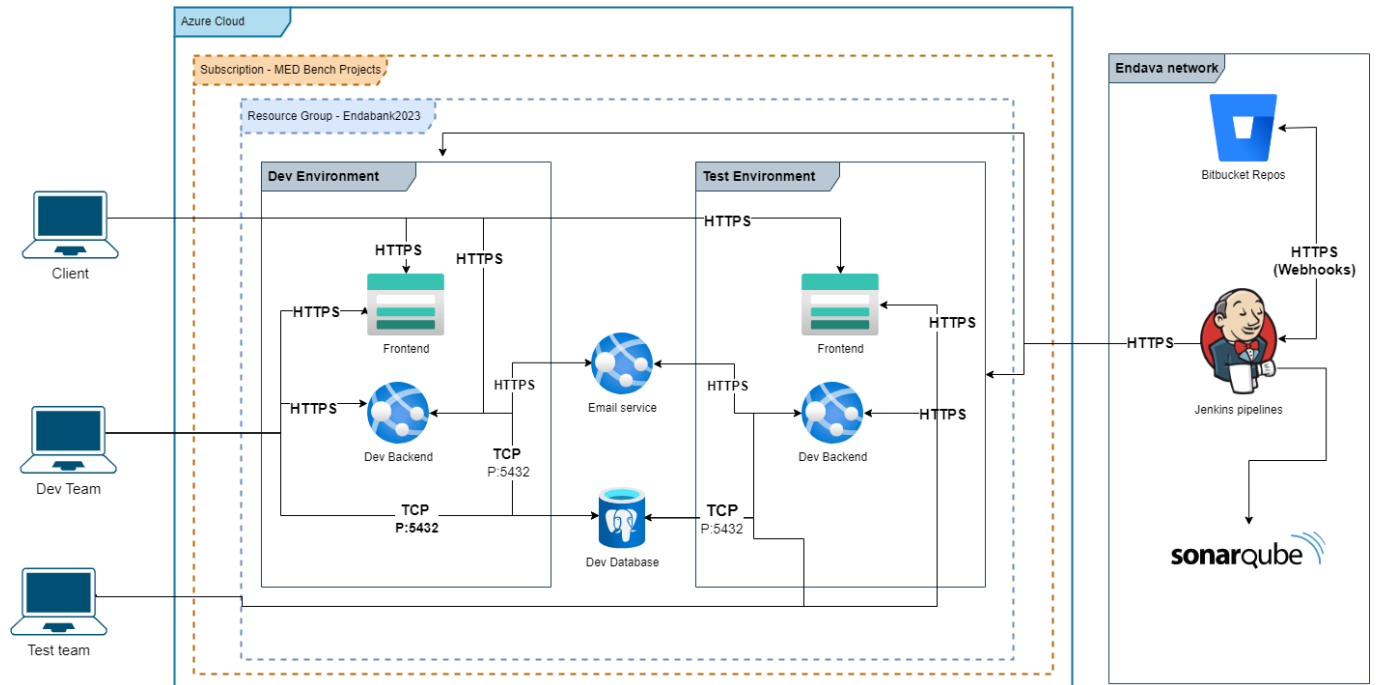


Fig. 43. Infraestructura de Endabank

En cuanto a la infraestructura del proyecto hay que tener en cuenta dos proveedores, un proveedor es Azure y el otro es Endava, porque gran parte de los recursos utilizados ya los tenía Endava dentro de sus portafolios, como Bitbucket, Jenkins y SonarQube, por lo que ya no era necesario consumirlos desde el proveedor directamente.

Como se observa en la **Fig. 43**, en la nube de Azure se crea una suscripción llamada MED Bench Projects que contiene un grupo de recursos llamado Endabank2023 donde se albergan los ambientes y servicios desplegados. Debido al presupuesto y al alcance del proyecto sólo se requería un servidor para alojar la base de datos de desarrollo y de pruebas, al igual que el servicio de correo electrónico. Una vez se implementaron los pipelines se creó la lógica para poder hacer despliegues al ambiente de desarrollo y también al ambiente de pruebas, de acuerdo a las reglas que se definieron dentro del proyecto.

Del lado de los servicios de Endava, inicialmente se configuró un servicio de Bitbucket para alojar el código dentro de la misma estancia de Endava, luego se hizo el pipeline en Jenkins y posteriormente la implementación de Sonarqube. Su funcionamiento es el siguiente: al ser una

instancia de Endava con una configuración en particular, siempre que se sube algo a Bitbucket se envía una señal a Jenkins notificando que hay cambios en el repositorio y si hay configurado un pipeline para ese repositorio, se inicia el proceso de CI/CD. Dentro de cada repositorio hay una carpeta llamada .cicd, la cual contiene toda la configuración de los pipelines y todo lo que sería integración continua y despliegue continuo.

Los servicios requeridos por el proyecto en términos de frontend son un proyecto en React con la versión 18 y la herramienta de compilación vite para lo cual fue utilizado un Blob storage habilitado para servir el artefacto estático disponible almacenado generado tras una compilación del proyecto en React. En cuanto al backend, un proyecto monolítico desarrollado con Spring Boot con la versión 17 de Java y el gestor de dependencias Maven, el cual se desplegó utilizando el servicio App Service de Azure al igual que el servicio de envío de correos electrónicos en Node.js. Para el despliegue de la base de datos se utilizó el servidor de base de datos de bajos recursos de Azure para PostgreSQL llamado Flexible Server.

H. Análisis estático del código con SonarQube

El análisis estático del código fuente del backend de Endabank a través del servidor SonarQube de Endava (**Fig. 44**) que incluye reglas definidas por la compañía, presenta los siguientes resultados:

1. No se presenta código duplicado.
2. No se presentan bugs, consecuentemente se tiene una confiabilidad con rating A.
3. No se presentan vulnerabilidades, consecuentemente la seguridad tiene rating A.
4. No se presentan puntos de seguridad críticos, consecuentemente la revisión de seguridad tiene un rating en A.
5. La deuda técnica son 8 minutos y existen 4 code smells, consecuentemente, se obtiene un rating de A en mantenibilidad.
6. La cobertura de código es de un 95.6%, superando el 75% definido por el equipo de QA.

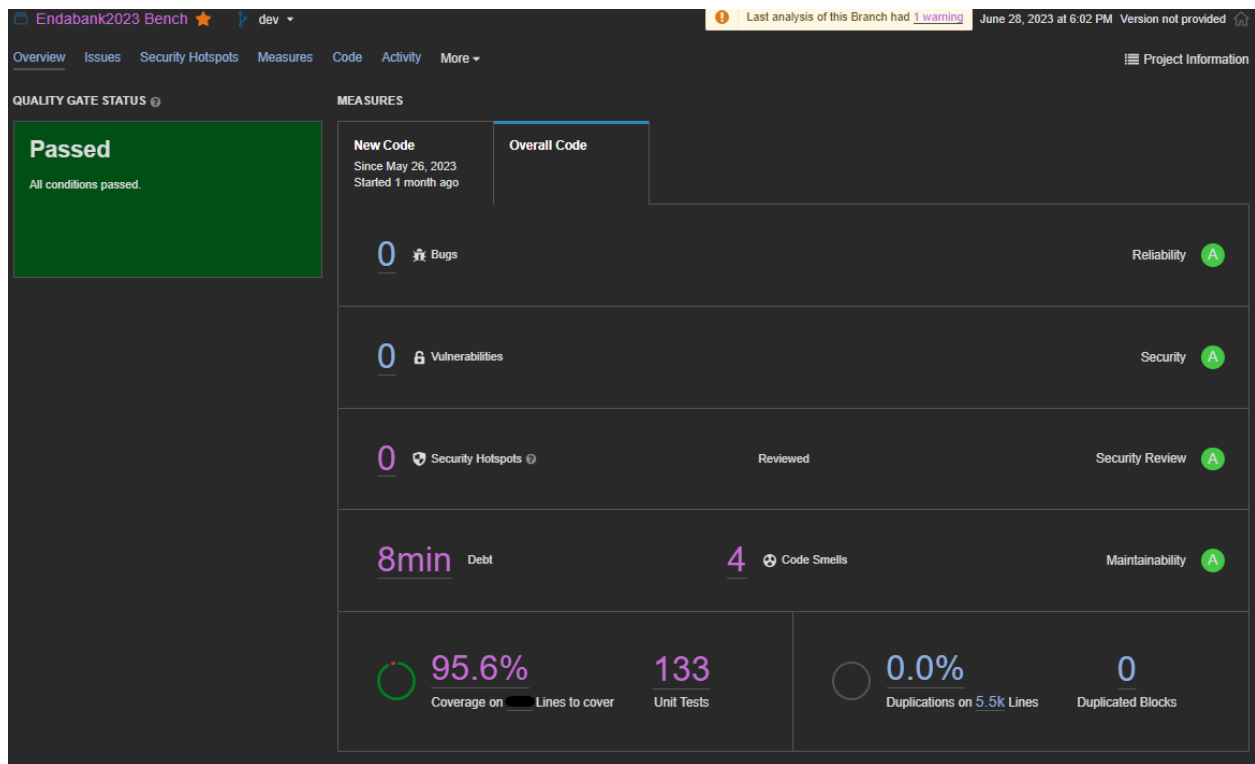


Fig. 44. Métricas e indicadores con SonarQube en el backend

En el caso del frontend (**Fig. 45**), se presentan los siguientes resultados:

1. Se presenta un porcentaje bajo de código duplicado
2. Se presenta 1 bug, el cual no está relacionado a ninguna vulnerabilidad crítica al ser una regla de internacionalización que sugiere que el título de la aplicación debería estar definido en un resource bundle. Consecuentemente se tiene una confiabilidad con rating B.
3. No se presentan vulnerabilidades, consecuentemente la seguridad tiene rating A.
4. No se presentan puntos de seguridad críticos, consecuentemente la revisión de seguridad tiene un rating en A.
5. La deuda técnica es de 1 hora y 47 minutos y existen 19 code smells, consecuentemente, se obtiene un rating de A en mantenibilidad.
6. La cobertura de código es de un 82.2%, superando el 75% definido por el equipo de QA.

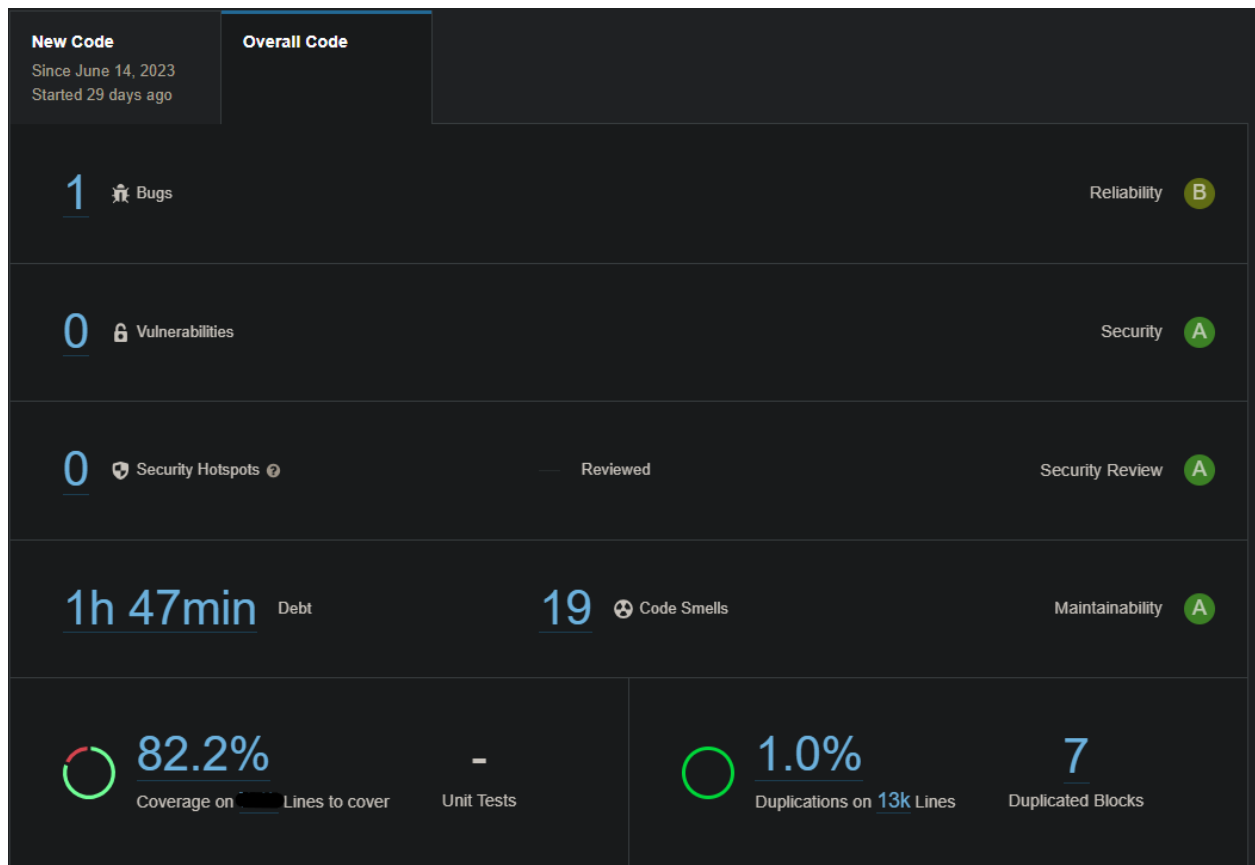


Fig. 45. Métricas e indicadores con SonarQube en el frontend

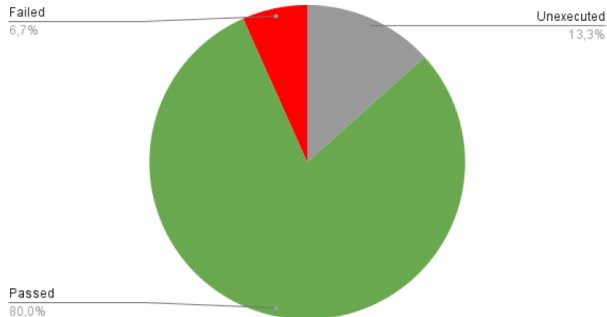
I. Métricas de QA

Total No of test from previous sprint	Total No of test planned this sprint	Total No of test cases planned	Total No of test cases executed	Total No of passed test cases	Total No of failed test cases	Total of not executed test cases	Total of bugs reported	Total of bugs solved
11	19	30	26	24	2	4	8	6

- Test pass rate = 92%
- Test execution rate = 86,7%
- Bug detection rate = 30,7%

- Manual test = 16
- Database test = 1
- API test = 8
- Automated test = 5

State of test cases



Type of test case planned

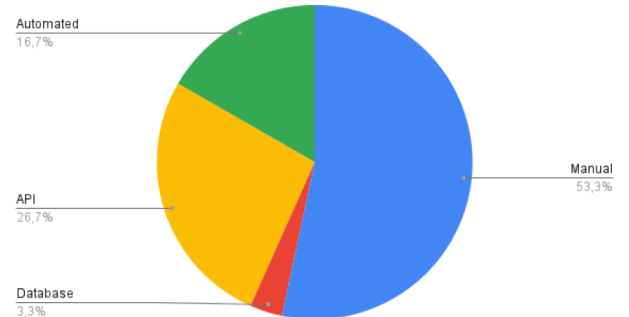


Fig. 46. Ejemplo de métricas de testing (Sprint 7)

Al finalizar cada sprint, el equipo de QA realizaba un reporte con métricas de testing del sprint (**Fig. 46**), que incluía el número de tests del sprint previo, el número de tests del sprint actual, el número de casos de prueba planeados, el número de casos de prueba ejecutados, número de casos de prueba exitosos, el número de casos de prueba que fallaron, el número de casos de prueba que no se ejecutaron, el total de bugs reportados y el total de bugs solucionados.

VI. ANÁLISIS

El análisis, diseño e implementación de aplicaciones bancarias conlleva un trabajo arduo el cual tiene una serie de implicaciones en términos de seguridad, definición de lógica de negocio y experiencia de usuario. Debido a esto, fue necesario desarrollar una gran cantidad de funcionalidades y servicios para satisfacer las necesidades del cliente (en nuestro caso el PO) y los usuarios finales. A pesar de esto, se logró completar satisfactoriamente el MVP, que abarcaba los módulos de gestión de usuarios, gestión de productos, gestión de transacciones y administración, siendo algunas de las funcionalidades principales: registro e inicio de sesión, creación de productos personales, solicitud y visualización de apertura de productos empresariales, visualización de productos, transacciones y pagos, historial de transacciones, notificaciones vía email, entre otras.

De igual forma, a lo largo de los sprints se evidenció una mejoría en el equipo, lo cual se concluye a partir de los reportes de las métricas de QA, ya que con el pasar de los sprints disminuían el número de casos de prueba fallidos. Adicionalmente, en los últimos sprints el tipo de bugs reportados tenían una prioridad menor, ya que no eran bugs sobre funcionalidades, sino detalles (errores ortográficos, visuales).

Debido a la naturaleza del proyecto de aprendizaje y el alcance definido, existen oportunidades de mejora, una de las cuales es el servicio de email desarrollado. El objetivo inicial era construir un microservicio para la comunicación con el backend de Endabank. Sin embargo, este servicio se implementó utilizando Azure App Service, que no es la opción más eficiente debido al consumo de recursos. Como trabajo futuro, se propone desplegar el servicio utilizando Azure Functions, que actúa como un trigger HTTP con un modelo de cobro basado en el consumo. Esto podría resultar más económico en comparación con el uso de App Service. Por último, en cuanto al envío de notificaciones por correo electrónico de transacciones, debido al alcance del proyecto, se implementó una comunicación síncrona. Sin embargo, en una etapa más avanzada del proyecto, sería posible considerar la implementación de una comunicación asíncrona utilizando colas de mensajería, lo cual proporciona una opción más escalable.

Por otra parte, como trabajo futuro, se propone mejorar la implementación del e-commerce desarrollado para probar la conexión con Endabank. Una opción es desarrollar un

backend dedicado para todas las funcionalidades del e-commerce, o bien, utilizar herramientas como el plugin WooCommerce de Wordpress o VTEX. Para lograr esto, es necesario explorar otros enfoques.

El primer enfoque consiste en utilizar una callback url PUSH, donde el servidor (en este caso, Endabank) enviaría una notificación a un callback url expuesta por el e-commerce a través de una solicitud HTTP POST. Esta notificación se enviaría cuando la transacción cambie de un estado pendiente a un estado aprobado. El segundo enfoque es el uso de una callback url PULL, donde el cliente (en este caso, el e-commerce) realizaría solicitudes HTTP GET periódicas a Endabank utilizando un cronjob o una tarea programada. Esto permitiría al e-commerce verificar constantemente si hay actualizaciones o eventos nuevos, y así comprobar el estado de la solicitud de pago.

Una última oportunidad de mejora, sería cambiar el enfoque adoptado al inicio del proyecto y permitir que cada negocio asociado a un producto empresarial tenga su propio correo electrónico y este pueda ser usado para iniciar sesión en el e-commerce, lo cual no es posible actualmente; ya que todos los productos empresariales de un usuario están asociados a un único email.

VII. CONCLUSIONES

- La adopción del marco de trabajo ágil Scrum en el desarrollo de software ofrece numerosos beneficios. Permite una fácil adaptación a los cambios, entrega resultados incrementales de forma temprana y fomenta la comunicación y colaboración continua dentro del equipo a través de distintas ceremonias (Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective y Sprint Refinement). En proyectos de software donde los requerimientos no están completamente definidos, Scrum aumenta la probabilidad de éxito, como sucedió en este proyecto donde el producto evolucionó constantemente hasta cumplir con lo acordado en el MVP dentro del tiempo establecido.
- La realización de flujos de usuario, wireframes y prototipos antes de la programación da una serie de ventajas de cara al desarrollo, ya que si bien puede llegar a tomar bastante tiempo la realización de estos insumos, es más barato en términos de tiempo y dinero la elaboración y edición de estos (usando herramientas como Figma), que primero desarrollar toda una funcionalidad completamente y luego validar con el PO y con el equipo. La adopción de este enfoque nos dio un marco de orientación de hacia dónde queremos llegar y nos permitió probar y validar cosas de cara al usuario, y así enriquecer e ir iterando sobre el producto que se va a programar posteriormente y de esta forma desarrollar un producto más usable. El hecho de haber construido un sistema de diseño nos permitió fijarnos en cuáles van a ser los patrones de cara a la interfaz de usuario que se pueden llegar a repetir y así generar módulos estandarizados, escalables y reutilizables (todo esto alineándose con el enfoque de Atomic Design utilizado). Si no se hubiera adoptado este enfoque antes de programar, es muy probable que los desarrollos de los distintos miembros del equipo no estuvieran alineados y pudieran variar. En conclusión, nos permitió establecer un camino a seguir desde el inicio del desarrollo y así facilitar la programación como tal.
- Es esencial la aplicación de buenas prácticas en el diseño de una REST API puesto que así se puede asegurar una consistencia y estandarización en el diseño de estas, mejorando la escalabilidad a lo largo del tiempo y permitiendo una facilidad en el mantenimiento del sistema.

-
- El uso e implementación de herramientas de análisis de código estático como SonarQube y ESLint proporcionó una serie de beneficios, incluyendo la detección temprana de problemas de calidad del código, mejora de la calidad y la seguridad del código, cumplimiento de estándares de codificación y aumento de la confianza en los cambios de código. Estos beneficios nos permitieron finalmente mantener un desarrollo más eficiente y confiable.
 - Sin duda uno de los factores más determinantes en el éxito del proyecto fue el buen trabajo en equipo que existió a lo largo de todo el proceso de prácticas, ya que al ser un equipo relativamente grande (7 personas), la sinergia y comunicación fueron vitales tanto en la fase de diseño como en la de desarrollo. De igual manera, el buen acoplamiento que tuvo el equipo permitió la implementación de una cultura de retroalimentación constructiva y aprendizaje continuo, la cual fue beneficiosa para la identificación y resolución de problemas de manera oportuna. Todo este buen trabajo se ve reflejado en el desarrollo exitoso del MVP planteado.
 - A pesar de que la conexión de Endabank con un e-commerce se realizó de forma síncrona debido al alcance y tiempo del proyecto además de lo previamente definido en el MVP, llevamos a cabo una exhaustiva búsqueda sobre implementaciones utilizadas en pasarelas de pago como PayU, que incluyen sistemas que utilizan las callback url tanto en un enfoque PUSH (webhook que notifica al e-commerce) como en un enfoque PULL (verificación constante del estado del pago). Estos conocimientos resultaron muy útiles para nuestro aprendizaje como practicantes y nos acompañarán a lo largo de nuestra vida profesional. Lo mismo se aplica a las capacitaciones recibidas en la vertical de pagos, las cuales fueron muy enriquecedoras y nos brindaron una comprensión más profunda sobre cómo funcionan los sistemas de pago y su interacción con otros subsistemas. En una segunda versión del MVP sería ideal la inclusión de un backend dedicado y un cambio de enfoque empleando callback url.
 - La experiencia vivida durante las prácticas académicas no solo fue beneficiosa para adquirir habilidades técnicas relevantes para la vida profesional, sino que también resultó sumamente valiosa para familiarizarnos con un entorno laboral real y una cultura empresarial amena en Endava. En Endava, no se consideran los "errores" como tal, sino como oportunidades de mejora, las cuales fueron identificadas a través de los constantes

feedbacks brindados por mentores y compañeros de equipo. Durante el período de prácticas, se pudo observar mejoras significativas en habilidades blandas como adaptabilidad, asertividad e inteligencia emocional, gracias a los consejos de los mentores, el diálogo abierto, el constante feedback y los entrenamientos programados en habilidades blandas proporcionados por la compañía. Estas mejoras han permitido un crecimiento impresionante tanto a nivel personal como profesional.

- El buen ambiente en el que se desarrolló esta práctica sin duda produjo gran motivación y satisfacción laboral, ya que todos los integrantes del equipo siempre tuvieron gran compromiso con el proyecto y sin duda, una gran calidad humana que hizo muy ameno todo el proceso.
- El desarrollo de este proyecto fue sumamente enriquecedor debido a que a nivel de Frontend no había conocimiento previo en el desarrollo de pruebas unitarias y de igual forma, el conocimiento que se tenía de React era muy escaso; por lo que el proyecto permitió ampliar el conocimiento como desarrollador al proporcionar más habilidades, estar envuelto en un ambiente de aprendizaje continuo y finalmente lograr una gran satisfacción a nivel personal.
- Las pruebas unitarias a nivel de backend fueron un gran desafío, dado que la experiencia en el desarrollo de las mismas era casi nula. Esto inicialmente complicó en gran medida el desarrollo del proyecto. Sin embargo, contamos con un inmenso apoyo técnico por parte de los mentores, además de las capacitaciones internas de Endava llamadas "Pass It On". En general, Endava es el lugar ideal para que cualquier estudiante pueda realizar sus prácticas académicas.

REFERENCIAS

- [1] S. Singh, "Emergence of Payment Systems in the Age of Electronic Commerce: The State of Art", Global Journal of International Business Research. 2009. [En línea]. Disponible en: <https://ssrn.com/abstract=1536620>.
- [2] Banco Central Europeo, "Oversight Framework for Card Payments Systems", Banco Central Europeo, 2008. [En línea]. Disponible en: <https://www.ecb.europa.eu/pub/pdf/other/oversightfwcardpaymentss200801en.pdf>.
- [3] CGAP, "Acquiring Models", 2019. [En línea]. Disponible en: [https://www.cgap.org/research/publication/acquiring-models#:~:text=In%20the%20four%20party%20model,%3B%20\(iii\)%20the%20merchant%20accepting](https://www.cgap.org/research/publication/acquiring-models#:~:text=In%20the%20four%20party%20model,%3B%20(iii)%20the%20merchant%20accepting).
- [4] John Wiley & Sons, Inc, "ISO20022 For Dummies", 6th Limited Edition, 2022, pp. 5–20.
- [5] Morgan, J. P. "ISO 20022 — El Lenguaje Universal Para El Futuro de Pagos.", JPMorgan Chase Bank, 2022, [En línea]. Disponible en: <https://www.jpmorgan.com/content/dam/jpm/global/documents/iso-20022-spanish-white-paper-ada-compliant.pdf>
- [6] Mozilla Developer Network. "JavaScript | MDN." [En línea]. Disponible en: <https://developer.mozilla.org/es/docs/Web/JavaScript>.
- [7] Mozilla Developer Network. "HTML | MDN." [En línea]. Disponible en: <https://developer.mozilla.org/es/docs/Web/HTML>.
- [8] Mozilla Developer Network. "CSS | MDN." [En línea]. Disponible en: <https://developer.mozilla.org/es/docs/Web/CSS>.
- [9] Sass. "Sass: Syntactically Awesome Style Sheets." [En línea]. Disponible en: <https://sass-lang.com/>.
- [10] React. "Documentación de React." [En línea]. Disponible en: <https://es.react.dev/>.
- [11] Vite. "Vite | Next Generation Frontend Tooling." [En línea]. Disponible en: <https://vitejs.dev/guide/>.
- [12] Vite. "Why Vite? | Vite." [En línea]. Disponible en: <https://vitest.dev/guide/why.html>.
- [13] ESLint. "ESLint - Pluggable JavaScript linter." [En línea]. Disponible en: <https://eslint.org/>.
- [14] Spring, "Spring Quickstart Guide" 2023, [En línea]. Disponible en: <https://spring.io/quickstart>.
- [15] JUnit, "JUnit 5 User Guide" 2023, [En línea]. Disponible en: <https://junit.org/junit5/docs/current/user-guide/>.
- [16] Sonarqube, "SonarQube Documentation" 2023, [En línea]. Disponible en: <https://docs.sonarqube.org/latest/>.
- [17] Nodejs, "Acerca de Node.js" 2023, [En línea]. Disponible en: <https://nodejs.org/es/about>.

-
- [18] P. Deemer, G. Benefield, C. Larman, and B. Vodde, “Información básica de SCRUM.”, California: Scrum Training Institute, 2009. [En línea]. Disponible en: http://libroslibres.uls.edu.sv/informatica/informacion_basica_scrum.pdf
- [19] Oficina Nacional de Tecnologías de Información, “Mapa de Historia de Usuario”. Secretaría de Modernización - Presidencia de la Nación. [En línea]. Disponible en: https://www.argentina.gob.ar/sites/default/files/onti/onti/1_mapa_de_historia_de_usuarios.pdf
- [20] I. Ratner and J. Harvey, *Vertical slicing: Smaller is better* [conferencia]. 2011 Agile Conference, Salt Lake City, EE.UU, 2011.
- [21] M. Cohn, “User stories applied: For agile software development”, Addison-Wesley Professional, 2004, pp. 98–99.
- [22] Brad Frost, "Atomic Web Design", bradfrost.com, [En línea]. Disponible en: <https://bradfrost.com/blog/post/atomic-web-design/>
- [23] P. Hauer, “Package by Feature.”, 2020 [En línea]. Disponible en: <https://phauer.com/2020/package-by-feature/>.
- [24] F. García, “¿Qué es SendGrid y cómo funciona?.”, 2023 [En línea]. Disponible en: <https://blog.cliengo.com/que-es-sendgrid/>