Análisis, Diseño, Desarrollo E Implementación De Los Módulos De Créditos, Pagos Y Autenticación Para El Sistema De Créditos Inmobiliarios Solint.

Juan Jose Martinez Niño y Jhon Sneider Lopez Ruiz

Trabajo de Grado para Optar al Título de Ingeniero de Sistemas

Director

Jathinson Meneses Mendoza

Magister en Gestión, Aplicación y Desarrollo de Software

Tutor

Jhon Alexander Martinez Romero

Ingeniero de Sistemas

Práctica empresarial.

Infinity Prime S.A.S.

Universidad Industrial De Santander

Facultad De Ingenierías Fisicomecánicas

Escuela De Ingeniería De Sistemas E Informática

Bucaramanga

2023

Dedicatoria

En primer lugar, quiero expresar mi profundo agradecimiento a mis padres, quienes a lo largo de toda mi vida me han brindado amor incondicional y un apoyo inquebrantable. Su constante aliento y confianza en mí han sido mi motor para alcanzar este logro.

A mis queridos amigos, les debo un reconocimiento especial. Su amistad y apoyo a lo largo de este arduo camino han sido invaluables. Sus palabras de aliento y la voluntad de compartir su conocimiento han sido una fuente constante de inspiración.

No puedo pasar por alto a mi compañero, John López. Su presencia en mi vida ha sido un pilar fundamental en el desarrollo de este proyecto y a lo largo de toda nuestra carrera. Su inigualable conocimiento y sus cualidades excepcionales han sido un faro de orientación en momentos de desafío.

Por último, pero no menos importante, quiero expresar mi sincero agradecimiento a mi alma mater. Esta institución ha sido el crisol en el que he forjado mis conocimientos y habilidades, brindándome las herramientas necesarias para alcanzar mis metas académicas y profesionales.

A todos ustedes, mi gratitud es infinita. Este proyecto de grado no habría sido posible sin su amor, apoyo y orientación. Sus contribuciones han dado forma a mi viaje académico y personal, y espero que este trabajo sea un reflejo de la gratitud que siento hacia cada uno de ustedes.

Juan José Martínez Niño

Dedicatoria

Con profundo aprecio y gratitud, dedico este logro a mis padres, pilares incansables de amor y apoyo, quienes han sembrado en mí valores perdurables y una inquebrantable determinación.

Esta dedicación se extiende a mis familiares y amigos, que han iluminado mi camino con su confianza y ánimo, enriqueciendo cada paso de este viaje con su calidez y sabiduría.

Un reconocimiento especial a todas las personas que, de distintas maneras, han añadido valor y profundidad a mis experiencias. Sus consejos y alientos han construido una red de solidaridad que me ha sostenido hasta este momento significativo.

A Juan José, compañero inigualable de universidad y trabajo, cuya solidaridad y espíritu colaborativo han sido una fuente constante de inspiración y crecimiento. Tu apoyo y amistad son un regalo invaluable en este camino académico y profesional.

Finalmente, ofrezco un homenaje a mi alma mater, cuyas enseñanzas y momentos compartidos forman un precioso mosaico de recuerdos en mi corazón.

Gracias a cada uno de ustedes por ser parte integral de este importante capítulo de mi vida.

Jhon Sneider López Ruiz

Agradecimientos

Al alcanzar este hito importante, extendemos nuestra gratitud a quienes han sido soportes esenciales en nuestra travesía académica:

Al MSc Jathinson Meneses Mendoza, nuestro dedicado director de proyecto de grado, por su invalorable colaboración, tiempo y conocimiento brindados con generosidad y perspicacia durante cada fase de esta tesis. Su guía ha sido una luz brillante en nuestro camino hacia el éxito académico.

A la Ing Leidy Xiomara Medina, nuestra líder de desarrollo durante la práctica, cuya expertise y dedicación han enriquecido significativamente este proyecto. Sus insights y orientación han sido cruciales para alcanzar un nivel de excelencia y profundidad en nuestro trabajo.

A la Universidad Industrial de Santander y a la Escuela de Ingeniería de Sistemas, por ser el fértil terreno donde hemos podido sembrar y cultivar nuestras aspiraciones académicas. Extendemos un agradecimiento especial a cada uno de nuestros docentes, que con su compromiso y sabiduría, nos han brindado las herramientas para crecer tanto intelectual como personalmente.

Finalmente, a nuestras amadas familias y amigos, cuya constante presencia y apoyo han sido nuestro refugio y nuestra fortaleza en cada etapa de esta significativa travesía. Su fe en nosotros ha sido un baluarte inquebrantable, alentándonos a alcanzar nuevas alturas y a persistir en la búsqueda de nuestros sueños.

Tabla de Contenido

Lista de figuras

Lista de tablas

Resumen

Título: Análisis, diseño, desarrollo, e implementación de los módulos de créditos, pagos y autenticación para el sistema de créditos inmobiliarios SOLINT.*

Autores: Juan José Martínez Niño y Jhon Sneider López Ruiz**

Palabras Clave: Crédito, Tabla de amortización, Pago, Cuota, Base de datos, Sistema de información, Framework, Frontend, Backend.

Descripción: La empresa Empaques Cardenas ofrece el servicio de créditos inmobiliarios a personas naturales. En la actualidad esta compañía lleva los registros y la información de dichos créditos de forma digital usando un software antiguo que funciona de manera local en un único computador. Con un aumento significativo en la cantidad de créditos realizados la compañía se encuentra con un problema para almacenar y consultar la información de dichos créditos, pues el software que usan actualmente no puede cumplir de manera eficiente las necesidades de sus usuarios generando así retrasos en informes, realización de nuevos créditos y recepción de pagos. Por lo tanto, se busca diseñar e implementar un sistema de información que gestione y controle los procesos mencionados anteriormente y asegure la integridad de la información. El sistema de información será implementado utilizando la metodología Scrum y una arquitectura API REST. Para el desarrollo de este sistema utilizaremos el framework de frontend Vue.Js, el de backend Laravel y la base de datos relacional.

---

*Proyecto de grado. Práctica empresarial.

**Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática.

Director: MSc Jathinson Meneses Mendonza

Abstract

Title: Analysis, design, development, and implementation of the credits, payments, and authentication modules for the SOLINT real estate credit system.*

Authors: Juan José Martínez Niño and Jhon Sneider López Ruiz**

Keywords: Credit, Amortization table, Payment, Installment, Database, Information system, Framework, Frontend, Backend.

Description: The company Empaques Cardenas offers real estate credit services to individuals. Currently, this company manages the records and information of these credits digitally using an old software that operates locally on a single computer. With a significant increase in the number of credits issued, the company is facing a problem storing and consulting the information on these credits, as the software they are currently using cannot efficiently meet the needs of its users, thus generating delays in reports, issuing new credits, and receiving payments. Therefore, the goal is to design and implement an information system that manages and controls the previously mentioned processes and ensures the integrity of the information. The information system will be implemented using the Scrum methodology and a REST API architecture. For the development of this system, we will use the Vue.Js frontend framework, the Laravel backend framework, and the MySQL relational database.

---

*Degree project. Internship.

**Faculty of Physical-Mechanical Engineering. School of Systems and Computer Engineering.

Director: MSc Jathinson Meneses Mendonza

Introducción

Un sistema de información web es una tecnología que permite acceder, procesar, almacenar y distribuir información a través de internet, permitiendo a los usuarios acceder y compartir información desde cualquier lugar del mundo con acceso a internet. Estos sistemas se utilizan en una amplia gama de aplicaciones, como sitios web de comercio electrónico, portales de noticias, redes sociales, plataformas de aprendizaje en línea, aplicaciones de gestión de proyectos, entre otros (Concepto.de, s.f).

Sin una infraestructura tecnológica moderna como redes informáticas, almacenamiento de datos, protocolos de comunicación, software para gestión de recursos humanos, aplicaciones de planificación de recursos empresariales, etc., no solo sería difícil realizar tareas administrativas de rutina de manera eficiente, sino que los empleados podrían experimentar dificultades para comunicarse entre ellos y entre los departamentos de la empresa (Computer Weekly, s.f.).

En resumen, los sistemas de información son esenciales en la sociedad actual para mejorar la eficiencia, la productividad, la colaboración, la comunicación y la toma de decisiones.

Actualmente Empaques Cardenas hace uso de un sistema poco eficaz para manejar la información de sus créditos inmobiliarios el cual permite realizar los procesos y acciones necesarias para completar diferentes tareas sin embargo el sistema no logra ser eficiente pues cuenta con diferentes arquitecturas de software y el uso de tecnologías que al día de hoy se encuentran despreciadas, sus tiempos de respuesta son demorados y su interfaz gráfica no resulta intuitiva para el usuario.

Por los motivos mencionados anteriormente, la empresa Infinity Prime propone desarrollar un nuevo sistema de información para la gestión de los créditos inmobiliarios que se manejan en Empaques Cardenas. Este proyecto puntual consistirá en el desarrollo de tres

nuevos módulos: autenticación, pagos, créditos del sistema SOLINT, haciendo uso de arquitecturas limpias (Dominicode, s.f.) y patrones de diseño, de manera que prioricen la usabilidad, escalabilidad, robustez y soporte de los componentes de cada módulo.

1. Objetivos

1.1 Objetivo General

Analizar, diseñar y desarrollar los módulos de créditos, pagos y autenticación, para un sistema de gestión y control de créditos inmobiliarios, utilizando nuevas tecnologías tales como: los *frameworks* Laravel y VueJs, arquitectura limpia y patrones de diseño para garantizar la escalabilidad, el soporte de los componentes y la usabilidad del sistema.

1.2 Objetivos Específicos

- Establecer los requerimientos funcionales, no funcionales y casos de uso para los componentes pertenecientes a los módulos a desarrollar por medio de reuniones con el cliente.
- Determinar el diseño, la estructura de la base de datos y arquitectura para los diferentes módulos y sus componentes.
- Desarrollar componentes para los módulos mencionados en base a los diseños propuestos con el uso de los *frameworks:* Laravel y VueJs.
- Implementar pruebas de calidad para los componentes evaluando la eficacia del sistema orientado a usuarios finales.

2. Planteamiento Y Justificación Del Problema

El cliente: Empaques Cardenas, es una compañía que se dedica a brindar créditos inmobiliarios entre otras actividades. En la actualidad esta compañía lleva los registros y la

información de dichos créditos de forma digital usando un software antiguo que funciona de manera local en un único computador. Con un aumento significativo en la cantidad de créditos realizados la compañía se encuentra con un problema para almacenar y consultar la información de dichos créditos, pues el software que usan actualmente no puede cumplir de manera eficiente las necesidades de sus usuarios generando así retrasos en informes, realización de nuevos créditos y recepción de pagos.

La Empresa Infinity Prime como casa desarrolladora de soluciones software identifica la necesidad del cliente en centralizar y controlar la información de los créditos desde cualquier punto de acceso, por lo cual ha desarrollado el sistema SOLINT que le ofrece al cliente un sistema web de gestión y control de créditos inmobiliarios a la medida de sus necesidades.

El sistema contará con módulos para satisfacer las funcionalidades necesarias del cliente.

La principal intención del desarrollo de los módulos es contar con un mecanismo para el registro y control de créditos inmobiliarios que ofrece el cliente a quien se le desarrolla, también se desea controlar el pago de cuotas de los créditos, calcular el valor actual del crédito, generar el plan de amortización para el crédito, entre otros. También se requiere generar reportes de los pagos que realizan los deudores del crédito y demás soportes ligados al control e informe sobre el estado del crédito.

Por otro lado, el sistema facilitará la creación, modificación, eliminación y generación de reportes de créditos mediante el módulo de Créditos, también dispondrá de un módulo de pagos el cual se encargará de registrar los pagos a los créditos y de permitir abonos a capital, esta interacción con los módulos sólo se podrá llevar a cabo una vez el usuario valide su información de inicio con el módulo de autenticación.

3 Marco De Referencia

Para una mejor comprensión de las funcionalidades y el desarrollo del proyecto se deben tener en cuenta los siguientes conceptos:

3.1 Frameworks

Los frameworks o marcos de trabajo, en el contexto del desarrollo de software, son esencialmente estructuras predefinidas que sirven como una base conceptual y técnica para crear aplicaciones más estructuradas y cohesivas. Un framework proporciona una estructura y un conjunto de herramientas que facilitan el desarrollo de software, permitiendo a los desarrolladores centrarse más en las funcionalidades específicas de la aplicación que están creando, en lugar de los detalles de bajo nivel de cómo se implementan ciertas operaciones. A continuación, se explorarán dos frameworks populares: Laravel y Vue

3.1.1 Laravel

Laravel es un framework de desarrollo web basado en PHP, conocido por su elegante sintaxis y su enfoque en la escritura de código que es tanto robusto como mantenible. Laravel ofrece una rica gama de funcionalidades que incluyen herramientas para tareas comunes como autenticación, enrutamiento, y manejo de sesiones, así como herramientas más avanzadas como un ORM (Object-Relational Mapping) llamado Eloquent, y un preprocesador de consultas de base de datos llamado Query Builder. Su arquitectura sigue el patrón Modelo-Vista-Controlador (MVC), lo que facilita la organización del código y promueve la separación de responsabilidades, ayudando así a construir aplicaciones escalables y seguras.(Laravel, s.f.).

3.1.2 Vue

Vue, también conocido como Vue.js, es un framework progresivo para construir interfaces de usuario. Se caracteriza por ser reactivo, lo que significa que puede actualizar de manera dinámica la interfaz de usuario en respuesta a los cambios en los datos subyacentes (Vue.js, s.f.). Además, Vue permite una integración sencilla con proyectos existentes, así como una fácil adaptación y expansión. Ofrece una arquitectura basada en componentes que promueve la reutilización de código y la separación de responsabilidades, facilitando así el desarrollo de aplicaciones web de gran escala. También es conocido por su curva de aprendizaje suave, lo que lo hace accesible incluso para los desarrolladores menos experimentados, sin sacrificar la capacidad de construir aplicaciones complejas y robustas.

3.2 Aplicaciones De Servicios Rest

Las aplicaciones de servicios REST (Representational State Transfer) han ganado una gran popularidad en los últimos años debido a su capacidad para permitir la integración de sistemas y aplicaciones de una manera sencilla y eficiente. En lugar de utilizar protocolos complejos y pesados, como SOAP, las aplicaciones de servicios REST utilizan el protocolo HTTP y se basan en la representación de recursos para intercambiar datos (AWS, s.f.).

Esto significa que las aplicaciones de servicios REST son muy flexibles y pueden ser utilizadas en una amplia variedad de contextos, como aplicaciones móviles, sistemas de gestión de contenidos, sistemas de comercio electrónico y muchos otros sistemas que requieren el intercambio de datos y la integración con otros sistemas.

Las aplicaciones de servicios REST ofrecen varios beneficios en comparación con otros enfoques de intercambio de datos. Por ejemplo, son más simples y livianas, lo que las hace más fáciles de implementar y más rápidas de ejecutar. También son altamente escalables, lo que significa que pueden manejar grandes volúmenes de datos y usuarios.

Otro beneficio clave de las aplicaciones de servicios REST es su capacidad para permitir la interoperabilidad entre diferentes sistemas y plataformas. Esto significa que las aplicaciones de servicios REST pueden ser utilizadas por diferentes aplicaciones y sistemas, lo que facilita la integración y la comunicación entre ellos.

En resumen, las aplicaciones de servicios REST son una opción atractiva para aquellos que buscan una forma eficiente y escalable de intercambiar datos y facilitar la integración de sistemas y aplicaciones.

Las aplicaciones de servicios REST se utilizan ampliamente en muchos contextos diferentes. Aquí hay algunos ejemplos:

- Las aplicaciones móviles a menudo utilizan servicios REST para acceder a datos y recursos a través de la red. Esto puede incluir datos de usuario, como información de perfil y preferencias, así como datos de la aplicación, como el contenido y las actualizaciones de la aplicación.
- Los sistemas de gestión de contenidos, como WordPress y Drupal, a menudo utilizan servicios REST para permitir que diferentes aplicaciones y sistemas accedan a los datos y recursos almacenados en la plataforma. Esto puede incluir contenido de texto, imágenes, archivos y otros recursos que se utilizan en la plataforma.
- Los sistemas de comercio electrónico, como Shopify y Magento, utilizan servicios REST para permitir que diferentes aplicaciones y sistemas accedan a datos y recursos relacionados con la tienda, como productos, pedidos, clientes y transacciones.

3.3 Arquitectura De Software

La arquitectura de software representa la estructura, el conjunto de técnicas y patrones aplicados para desarrollar e integrar los componentes de un sistema de software. Es una representación conceptual del software, que ayuda en la comprensión de cómo se organiza el sistema, cómo interactúan las piezas individuales y cómo influye esto en su calidad y rendimiento. Su definición es esencial para coordinar las decisiones técnicas y asegurar que se cumplan los requerimientos tanto funcionales como no funcionales (Platzi, s.f.).

3.3.1 Arquitectura De Restful Api En Laravel

REST es un conjunto de principios arquitectónicos para la construcción de servicios web. La adopción de REST en Laravel es particularmente notable debido a cómo gestiona los recursos utilizando verbos HTTP como GET, POST, PUT, DELETE, entre otros. Laravel facilita la implementación de estos principios a través de sus herramientas integradas, como Eloquent para la gestión de recursos y la serialización de respuestas, permitiendo así que las aplicaciones cumplan con las convenciones REST de una manera natural y eficiente (Platzi, s.f.).

3.3.1.1 Estructura De Un Proyecto Laravel. Laravel presenta una estructura de carpetas bien definida:

- app/: Es el núcleo del código, subdividido en varias carpetas como Http (donde residen los controladores) y Providers (proveedores de servicios).
- config/: Contiene los archivos de configuración global.
- resources/: Guarda las vistas, archivos de idioma y LESS/SASS.
- storage/: Para logs, cache y archivos compilados.

Cada directorio está diseñado para facilitar la Modularidad, facilitando el mantenimiento del código y permitiendo la colaboración eficiente entre equipos(Laravel, s.f.).

3.3.1.2 Modelo De Datos Y Migraciones De Base De Datos. Laravel usa Eloquent, un ORM , que facilita la interacción con bases de datos. Las migraciones permiten llevar un registro versionado de los cambios en la estructura de la base de datos. Esto es vital para el mantenimiento y la escalabilidad del proyecto, permitiendo adaptaciones a los cambios de requerimientos sin comprometer los datos existentes(Laravel, s.f.).

3.3.1.3 Controladores Y Rutas De Api. Los controladores en Laravel manejan la lógica principal de la aplicación. Las rutas, por otro lado, actúan como un enrutador, dirigiendo las solicitudes HTTP a los controladores adecuados. Esta separación permite una mejor gestión del código y facilita la creación de pruebas unitarias y de integración(Laravel, s.f.).

3.3.1.4 Implementación De Autenticación Y Seguridad En Api. La seguridad es primordial en cualquier API. Laravel proporciona capacidades de autenticación out-of-the-box. Con herramientas como Laravel Sanctum y Passport, se facilita la autenticación y autorización, garantizando que los datos estén protegidos y solo sean accesibles para usuarios o servicios autorizados(Nigmacode, s.f.).

3.3.2 Arquitectura De La Spa En Vue

Vue.js, debido a su enfoque reactivo, posibilita el desarrollo de SPAs (Single Page Application) en las que el DOM (Document Object Model) se actualiza dinámicamente en función de las interacciones. Esta actualización dinámica proporciona una experiencia de usuario fluida, reduciendo la necesidad de recargas completas de la página (Vue.js, s.f.).

3.3.2.1 Estructura De Un Proyecto Vue.

- src/: Alberga el código fuente.
- store/: Para gestionar el estado de la aplicación mediante Vuex.
- router/: Define las rutas y la navegación con Vue Router.

Esta estructura modular es vital para mantener el código organizado, especialmente en aplicaciones grandes (Vue.js, s.f.).

3.3.2.2 Componentes De Vue Y Comunicación Con Api. Los componentes son el núcleo de las aplicaciones Vue. Permiten una estructura modular y reutilizable. Con bibliotecas como Axios, estos componentes pueden comunicarse fácilmente con APIs RESTful, recuperando o enviando datos según sea necesario (Vue.js, s.f.).

3.3.2.3 Implementación De Autenticación Y Seguridad En Spa. A diferencia de las aplicaciones tradicionales, las SPAs tienen el desafío de manejar la autenticación en el lado del cliente. Vue, junto con bibliotecas como Auth0 o herramientas de Laravel como Sanctum, puede proporcionar soluciones robustas para garantizar que la SPA esté protegida y solo sea accesible para usuarios autenticados, utilizando tokens y otros mecanismos de seguridad .

3.4 Bases De Datos

En teoría la definición de bases de datos incluye los principios formales para definir y manipular datos estructurados e interrelacionados. Para determinar los datos se utiliza un modelo de datos y para su manipulación un lenguaje. Diferentes modelos de datos se han propuesto buscando un mayor nivel expresivo para representar el mundo real. La potencia y limitaciones de cada modelo se pueden evaluar desde un punto de vista teórico y se evidencian desde un punto de vista práctico cuando se trata de implementarlos en aplicaciones tradicionales y modernas (Oracle, s.f.).

3.4.1 Estructura Y Relaciones De La Base De Datos

Las bases de datos relacionales se basan en la organización de la información en partes pequeñas que se integran mediante identificadores; a diferencia de las bases de datos no relacionales que, como su nombre lo indica, no tienen un identificador que sirva para relacionar dos o más conjuntos de datos.

Además, las bases de datos relacionales son más robustas, es decir, tienen mayor capacidad de almacenamiento, y son menos vulnerables ante fallas, estas son sus principales características.

Las relaciones se asocian con tablas nombradas cuyas columnas representan atributos que también pueden tener asociado un nombre. Las filas de las tablas son tuplas. Los valores que toman las tuplas se extraen de conjuntos de constantes llamados dominios (Barzana, s.f.).

3.5 Javascript

JavaScript es un lenguaje de programación esencial en la creación de páginas web dinámicas e interactivas. Se emplea extensamente para añadir diversas funcionalidades a las páginas web, desde refrescar contenido en plataformas de redes sociales hasta presentar animaciones y mapas interactuables, mejorando notablemente la interacción del usuario con el sitio web (Amazon Web Services, s.f.).

3.6 PHP

PHP, que inicialmente significaba Personal Home Page y ahora se conoce como PHP: Hypertext Preprocessor, es un popular lenguaje de programación de código abierto ampliamente utilizado para el desarrollo de sitios web y aplicaciones web dinámicas. Este lenguaje se integra fácilmente con varias bases de datos, siendo MySQL una de las más comunes. Es conocido por su flexibilidad y su capacidad para crear contenido web dinámico,

permitiendo una comunicación fluida entre el servidor y el cliente. Además, PHP tiene una gran comunidad de desarrolladores y una extensa biblioteca de recursos, lo que facilita el rápido desarrollo y la resolución de problemas (Group, 2021).

3.7 Implementación De Seguridad

Además de la autenticación, es fundamental implementar otras medidas de seguridad, como el cifrado de datos, el control de acceso basado en roles y la prevención de ataques, para garantizar la protección de los datos y la integridad del sistema.

3.8 Pruebas

Las pruebas en el desarrollo de software son una fase crucial que permite evaluar la funcionalidad, usabilidad y consistencia del sistema antes de su despliegue final. A través de las pruebas, se puede detectar y corregir errores, garantizar que el software cumple con los requisitos especificados y asegurar que funciona correctamente en todas las condiciones previstas. En este proyecto, se han realizado diversos tipos de pruebas, principalmente pruebas unitarias, pruebas de integración y pruebas E2E (End-to-End). A continuación, se detallan cada una de ellas (IBM, s.f.):

3.8.1 Pruebas Unitarias

Las pruebas unitarias se centran en verificar cada componente o unidad de software de manera individual, para asegurar que funciona correctamente y cumple con su especificación de diseño. Es el primer nivel de prueba y ayuda en la detección temprana de posibles fallos. Durante esta fase, se evalúan funciones, procedimientos y métodos individuales para garantizar que se produzcan los resultados esperados (KeepCoding, s.f.).

3.8.2 Pruebas De Integración

Las pruebas de integración se realizan después de las pruebas unitarias y tienen como objetivo verificar que los diferentes módulos o componentes del sistema funcionan correctamente cuando interactúan entre ellos. En este tipo de pruebas, se busca identificar problemas de interfaz, inconsistencias y fallos en las interacciones entre distintos segmentos del software (KeepCoding, s.f.).

3.8.3 Pruebas E2E (End-To-End)

Las pruebas E2E se realizan para verificar que el sistema como un todo funciona correctamente desde el inicio hasta el fin, emulando el comportamiento de un usuario real en un ambiente de producción. Esta fase de prueba asegura que toda la aplicación, en conjunto con cualquier sistema externo con el que interactúe, funcione de manera coherente y cumpla con los requisitos especificados. En este tipo de pruebas, se evalúan tanto las funcionalidades críticas del sistema como la integración y comunicación entre diferentes partes de la aplicación (QAlified, s.f.).

3.9 Postman

Postman es una herramienta visual que facilita el desarrollo y el uso de APIs. Esta plataforma agiliza todas las fases del ciclo de vida de la API, promoviendo una colaboración efectiva para diseñar APIs más eficientes y robustas. Su interfaz gráfica proporciona una forma intuitiva de construir y testear conexiones API, facilitando así la creación de soluciones más coherentes y optimizadas (OpenWebinars, s.f.).

3.10 Github

GitHub se destaca como un espacio colaborativo de desarrollo que facilita la creación y el hospedaje de repositorios en línea para almacenar proyectos, permitiendo que otros

usuarios los visualicen y accedan. En esta plataforma, es posible hospedar una diversidad de contenidos, desde programas individuales hasta sistemas operativos completos, fomentando la interacción de la comunidad mediante el acceso a códigos fuente y la posibilidad de proporcionar feedback. El nivel de interacción que pueden tener otros usuarios en un repositorio específico está determinado por los permisos otorgados por el propietario del repositorio, incluyendo la capacidad para realizar y registrar modificaciones, todo ello gestionado a través del eficiente sistema de control de versiones Git (Moreno, 2019).

3.11 Visual Studio Code

Visual Studio Code es un editor de código fuente avanzado que opera a nivel de escritorio y es compatible con Windows, macOS y Linux. Se caracteriza por su extenso ecosistema de extensiones, las cuales ofrecen soporte para una amplia gama de lenguajes de programación y entornos de ejecución, facilitando así un entorno de desarrollo altamente adaptable y versátil. Esta herramienta es una elección preferida entre los desarrolladores debido a su flexibilidad y a la amplia gama de funcionalidades que ofrece para facilitar el proceso de desarrollo de software (Microsoft, s.f.).

4. Antecedentes

La industria del crédito inmobiliario ha experimentado un rápido desarrollo y una considerable evolución en las últimas décadas, gracias en gran medida a los avances en las tecnologías de la información y la comunicación. Los sistemas modernos de gestión de créditos, especialmente en el sector inmobiliario, han pasado de ser sistemas tradicionales basados en papel a soluciones digitales completas que pueden manejar todos los aspectos de los préstamos, desde la originación hasta la gestión de los mismos.

Referente a este tema no se encontraron proyectos de grado realizados en universidades a nivel nacional, sin embargo se encontraron un par de ejemplos de sistemas similares usados en el mundo, estos son:

4.1 Temenos Transact

Es una plataforma bancaria digital que incluye funcionalidades para la gestión de créditos y préstamos. Permite a las instituciones financieras automatizar y optimizar los procesos de otorgamiento de crédito, evaluación de riesgo, seguimiento de pagos y gestión de cobranzas(Temenos, s.f).

4.2 Fiserv LoanServ

Es un sistema integral de administración de préstamos que ayuda a las instituciones financieras a gestionar y monitorear el ciclo de vida de los créditos. Ofrece funcionalidades para la originación, procesamiento, administración y cobranza de préstamos, así como para el seguimiento de pagos y la generación de informes (Fiserv, s.f).

5. Metodología

La metodología Scrum se utilizó en este proyecto de análisis, diseño, desarrollo e implementación de los módulos de autenticación, créditos y pagos para un sistema de información de créditos inmobiliarios. Scrum es una metodología ágil que se enfoca en la mejora continua y el trabajo en equipo para garantizar la entrega de productos de alta calidad de manera eficiente.

El proceso de desarrollo del proyecto se dividió en cinco fases principales las cuales se describen a continuación:

5.1 Primera Fase

La primera fase llamada: **DEFINICIÓN DE PROYECTO Y ANÁLISIS DE VIABILIDAD** se da inicio al proyecto planteando un problema inicial y los objetivos a cumplir para dar solución a dicho problema. Con los objetivos ya definidos se procede a evaluar el alcance y las limitaciones del proyecto para así desarrollar y entregar un producto que cumpla con las necesidades del cliente y este resulte satisfecho.

En esta fase también se desarrolla el marco teórico del proyecto, se definen conceptos que se usarán a lo largo del desarrollo del proyecto tales como la arquitectura y las herramientas que serán necesarias para completar el proyecto.

Por último se define un cronograma de actividades para trabajar en el proyecto, dividiéndolo en fases con tareas específicas para cada una, además se asignan actividades para realizar en cada semana del mes. De esta forma al finalizar cada mes, se evalúan las tareas realizadas en cada fase y se determina si se cumplieron de manera adecuada o no.

5.2 Segunda Fase

En la fase de **PLANEACIÓN** se definieron los requerimientos funcionales, requerimientos no funcionales y requerimientos de seguridad junto con diversos diagramas necesarios para la definición del proyecto tales como: diagramas de casos de uso, diagramas de secuencia, diagramas de actividad. También se definió el modelo entidad relación para la base de datos que se usará en el proyecto.

Las actividades presentes en esta fase se llevan a cabo en la primera semana de cada mes en la duración del proyecto, exceptuando la definición de arquitectura de software que se usará, esta actividad solo está presente la primera semana.

5.3 Tercera Fase

La fase de **EJECUCIÓN** se llevó a cabo en los cuatro meses de la duración del proyecto, pues en esta fase se desarrollaron los componentes del sistema tanto en el FrontEnd como en el BackEnd.

En el mes uno se desarrolla la arquitectura base del proyecto previamente definida en la fase de planeación, también se desarrolla el componente de autenticación con las funcionalidades de iniciar sesión, cerrar sesión, recuperar contraseña y la completa administración de los usuarios del sistema. Con este componente se garantiza que solo los usuarios autorizados ingresen a la información del sistema.

El segundo mes se destina para el desarrollo de los dos componentes principales del sistema: componente para administración de créditos y componente para la realización de pagos de cuotas. El primer componente tendrá las funciones de crear, editar, eliminar y mostrar la información de los créditos dentro del sistema, la información de estos incluye las cuotas y el estado de las mismas (creada, por pagar, en mora, pagada). El componente para el pago de cuotas se encarga de consultar el saldo de las cuotas pendientes y en mora a la fecha del pago y reducir de estas cuotas el valor pagado por el deudor del crédito.

Para el mes tres se desarrolla el componente para realizar abonos a capital el cual se encargará de recibir y manejar los pagos extraordinarios que realice el deudor del crédito, este componente debe reflejar el pago en el valor de las cuotas o en la cantidad de cuotas según sea el caso.

En el cuarto y último mes se desarrolla el componente para generación de recibos de pago y demás documentos, este componente consultara la información presente en la base de datos para generar al usuario documentos en formato PDF que sustenten los pagos de cuotas, abonos a capital y el estado del crédito de forma física.

5.4 Cuarta Fase

La fase de **SEGUIMIENTO Y CONTROL** al igual que las dos anteriores fases se lleva a cabo en los cuatro meses de duración del proyecto, las actividades en esta fase constan de pruebas sobre los componentes desarrollados en la fase de ejecución para verificar su correcto funcionamiento y en caso de encontrar errores o bugs reportarlos para corregirlos de manera óptima.

5.5 Quinta Fase

La fase final está destinada para la entrega de los componentes desarrollados totalmente funcionales.

La metodología Scrum permitió una gestión eficiente del proceso de desarrollo, ya que se pudo visualizar y priorizar el trabajo en tiempo real, garantizando la entrega de un prototipo de alta calidad en un plazo definido. Además, se fomentó la colaboración y el trabajo en equipo, lo que permitió la toma de decisiones más informadas y una mayor implicación en el proyecto por parte de todos los miembros del equipo.

6. Análisis

Durante la fase de análisis se realizaron diversas actividades con el fin de identificar los requerimientos funcionales y no funcionales que debían cumplir los módulos del sistema de información para garantizar su correcto desarrollo y construcción. Se llevaron a cabo diversas reuniones con el cliente (Empaques Cárdenas), se revisaron los procesos necesarios para la gestión de créditos y pagos y se evaluaron las tecnologías disponibles para la construcción del sistema.

Durante este proceso se identificaron las necesidades del cliente y se establecieron los requerimientos necesarios para el correcto funcionamiento del sistema. Se analizaron

diferentes opciones de arquitectura de software y se evaluaron las posibles soluciones de bases de datos que se adaptaran a los requerimientos propuestos.

Este proceso de análisis fue crucial para el éxito del proyecto, ya que permitió establecer una base sólida para el diseño, desarrollo e implementación del software. Se logró tener una comprensión profunda de las necesidades del cliente y se definieron los requerimientos para la construcción del sistema de información.

6.1 Requerimientos Del Proyecto

En este apartado se presenta el listado de requerimientos principales obtenidos al analizar el proceso de vida de un crédito y los pagos que se aplican sobre estos. Estos requerimientos se dividen en funcionales, los cuales enuncian los servicios que el sistema debe proveer, y no funcionales, los cuales abarcan las limitaciones sobre servicios o funciones que ofrece el sistema (Sommerville, 2011, pp. 84-85).

6.1.1 Requerimientos Funcionales

Tabla 1

Requerimientos funcionales

Requerimiento: RF01	Prioridad: Alta	Nombre: Iniciar Sesión	Rol: Administrador
Descripción: Los usuarios administradores registrados podrán iniciar sesión ingresando la siguiente información: usuario, contraseña			
Requerimiento: RF02	Prioridad: Alta	Nombre: Cerrar sesión	Rol: Administrador
Descripción: El usuario que haya iniciado sesión podrá cerrarla			
Requerimiento: RF03	Prioridad: Alta	Nombre: Listar créditos	Rol: Administrador
Descripción: El usuario podrá ver una lista de los créditos registrados en el sistema y su detalle.			

Requerimiento:	Prioridad:	Nombre:	Rol:
RF04	Alta	Crear crédito	Administrador
Descripción: El usuario podrá registrar un nuevo crédito en el sistema ingresando: el número de pagaré, el monto del crédito, la tasa de interés, la fecha de desembolso, el número de cuotas, el cliente al que pertenece el crédito y la fecha del crédito Opcionalmente podrá ingresar observaciones referentes a ese crédito.			
Requerimiento:	Prioridad:	Nombre:	Rol:
RF05	Alta	Editar crédito	Administrador
Descripción: El usuario podrá editar el campo de observaciones de un crédito.			
Requerimiento:	Prioridad:	Nombre:	Rol:
RF06	Alta	Desactivar crédito	Administrador
Descripción: El usuario podrá desactivar un crédito ingresando el motivo por el cual se desactivará.			
Requerimiento:	Prioridad:	Nombre:	Rol:
RF07	Alta	Mostrar tabla de amortización	Administrador
Descripción: El usuario podrá ver la tabla de amortización de cada crédito dentro del sistema.			
Requerimiento:	Prioridad:	Nombre: Listar pagos	Rol:
RF08	Alta		Administrador
Descripción: El usuario podrá ver una lista de los pagos registrados en el sistema.			
Requerimiento:	Prioridad:	Nombre:	Rol:
RF09	Alta	Registrar pago (cuota)	Administrador
Descripción: El usuario podrá registrar el pago de una cuota por parte del cliente ingresando la fecha del pago y el valor pagado.			
Requerimiento:	Prioridad:	Nombre:	Rol:
RF10	Alta	Registrar pago (abono a capital)	Administrador
Descripción: El usuario podrá registrar el pago de un abono a capital por parte del cliente ingresando la fecha del pago, el tipo de abono a capital y el valor del pago.			

6.1.2 Requerimientos No Funcionales

Tabla 2

Requerimientos no funcionales

Requerimiento:	Prioridad:	Nombre:
RNF01	Alta	Calcular tabla de amortización
Descripción: El sistema deberá calcular la tabla de amortización cuando se registre un nuevo crédito en el sistema.		
Requerimiento:	Prioridad:	Nombre:
RNF02	Alta	No exceder cupo máximo
Descripción: El sistema deberá validar que no se exceda el cupo máximo de un cliente en el momento en que el usuario quiera registrar un nuevo crédito para ese cliente.		
Requerimiento:	Prioridad:	Nombre: Recalcular cuotas con un abono a capital
RNF03	Alta	
Descripción: El sistema deberá recalcular la tabla de amortización de un crédito cuando el usuario registre un abono a capital, este cálculo debe tener en cuenta el tipo de abono a capital (reducir cuotas, reducir valor de la cuota)		
Requerimiento:	Prioridad:	Nombre: Aplicar excedente en pagos de cuotas
RNF04	Alta	
Descripción: El sistema deberá aplicar correctamente los pagos de cuotas que realicen los clientes si el valor que pagaron excede el valor mínimo de la cuota actual.		
Requerimiento:	Prioridad:	Nombre:
RNF05	Alta	Actualizar estado de cuotas
Descripción: El sistema deberá actualizar el estado de las cuotas (pendiente, en mora) de manera automática teniendo en cuenta la fecha de vencimiento de la cuota y los días de gracia del cliente.		

6.2 Análisis De Requerimientos

6.2.1 Casos De Uso

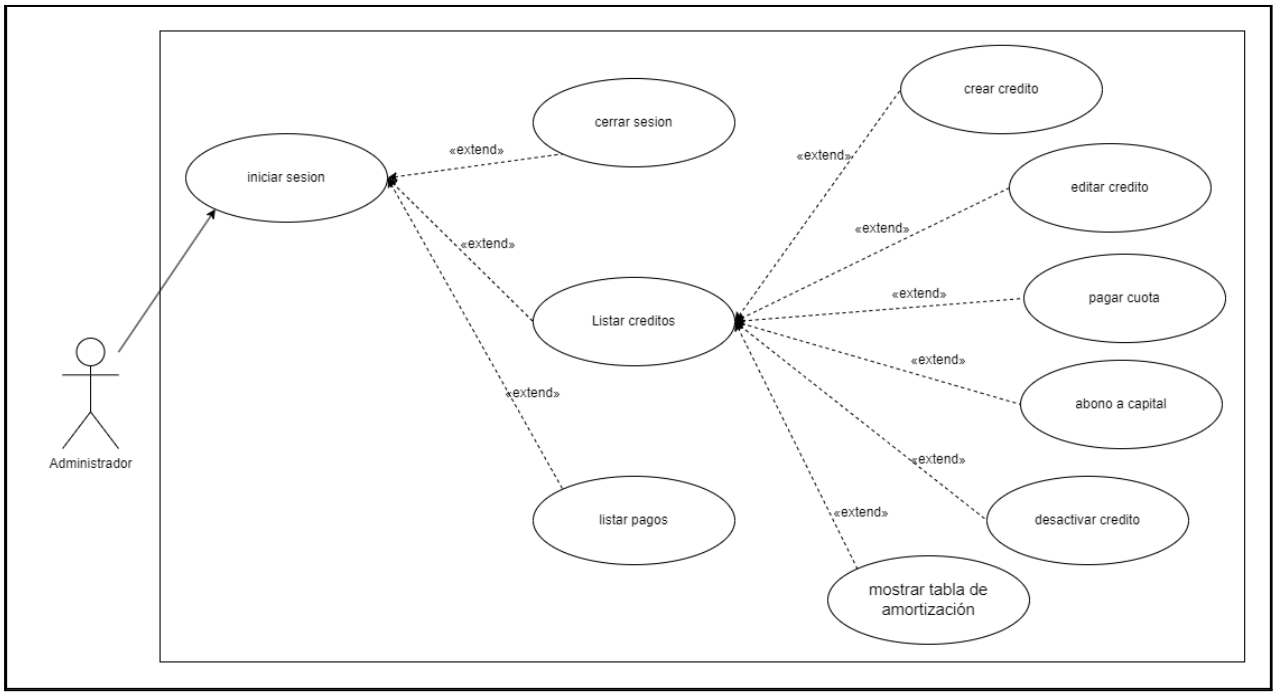
En el contexto de sistemas de información, los casos de uso son herramientas valiosas que facilitan la detallada representación de las funcionalidades del sistema, así como las interacciones existentes entre estas y los usuarios involucrados. Cada funcionalidad descrita en un caso de uso está vinculada a un actor específico, que, a través de la interacción con el sistema, puede desencadenar una serie de eventos que concretan dicha funcionalidad (IBM, s.f.).

En el proceso de desarrollo, los casos de uso actúan como un medio instrumental para clarificar y especificar cómo los requisitos establecidos serán implementados en el sistema. Al detallar las responsabilidades y acciones de los actores involucrados, estos casos se convierten en guías significativas para la estructuración adecuada y coherente de las operaciones del sistema, garantizando una ejecución fluida y eficiente de sus funciones.

De este modo, los casos de uso ayudan a estrechar la brecha entre los requisitos definidos y su implementación práctica, asegurando una alineación precisa con las expectativas y necesidades del usuario final.

Figura 1.

Diagrama de casos de uso.



Para facilitar la comprensión del diagrama de casos de uso presentado, a continuación, se detalla una lista que especifica los distintos requerimientos funcionales que están a cargo del único actor involucrado en este sistema: el administrador.

Tabla 3

Casos de uso del administrador

Actor	Administrador
Requerimientos funcionales	RF01 Iniciar sesión RF02 Cerrar sesión RF03 Listar créditos RF04 Crear crédito RF05 Editar crédito RF06 Desactivar crédito RF07 Mostrar tabla de amortización RF08 Listar pagos RF09 Registrar pago (cuota) RF10 Registrar pago (abono a capital)
Descripción	El actor "Administrador" es responsable de la gestión integral del sistema, incluyendo el monitoreo y la administración de las funcionalidades de créditos y pagos.

6.2.1.1 Diagrama De Actividades Para Crear Crédito

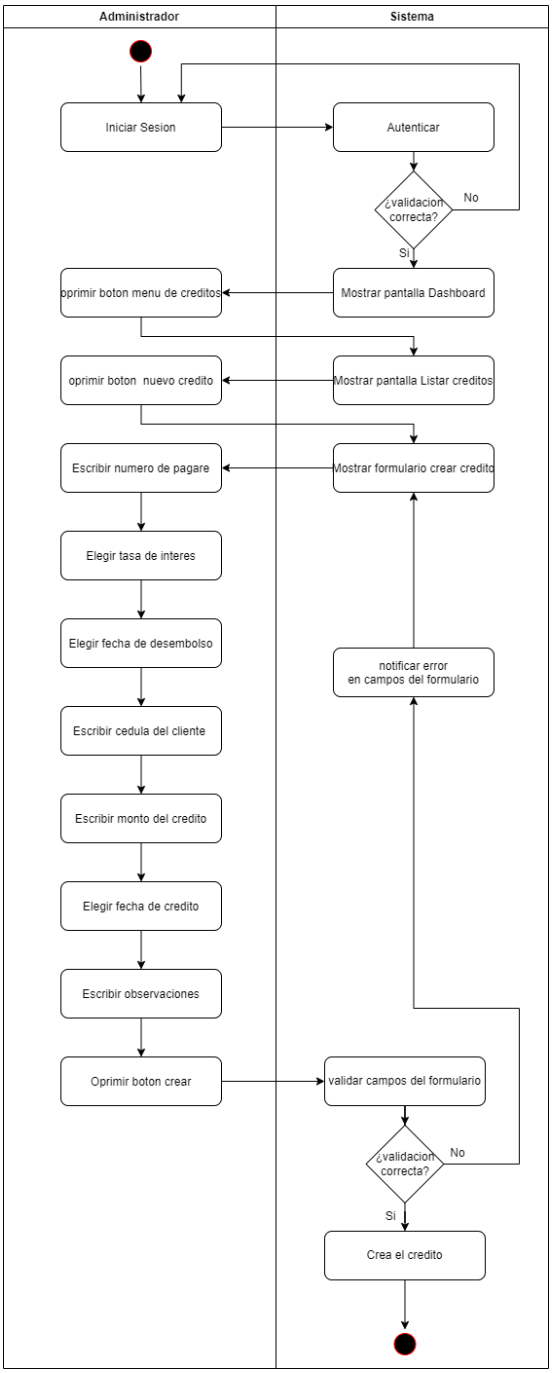
El proceso para crear un crédito en el sistema SOLINT comienza con el usuario iniciando sesión de manera exitosa y accediendo al panel principal o "dashboard". Desde aquí, el usuario selecciona la opción de menú de créditos, llevándolo a la pantalla donde puede visualizar una lista de los créditos existentes.

Para crear un nuevo crédito, el usuario selecciona la opción "Nuevo Crédito", que abre un formulario donde debe ingresar los detalles pertinentes del crédito. Esto incluye el número de pagaré, la tasa de interés, la fecha de desembolso, la cédula del cliente, la fecha del crédito, y cualquier observación relevante.

Una vez que se ha completado el formulario, el usuario procede a enviarlo para su validación. El sistema verifica la información proporcionada y, si es correcta, procede a crear el crédito en la base de datos. En caso de encontrar errores durante la validación, el sistema notifica al usuario para que realice las correcciones necesarias antes de reintentar el envío.

Figura 2.

Diagrama de actividades para crear un crédito.



6.2.1.2 Diagrama De Actividades Para Registrar Un Pago

El procedimiento para registrar un pago se inicia con la sesión del administrador en el sistema. Una vez autenticado exitosamente, el sistema le redirige a la pantalla principal o "dashboard".

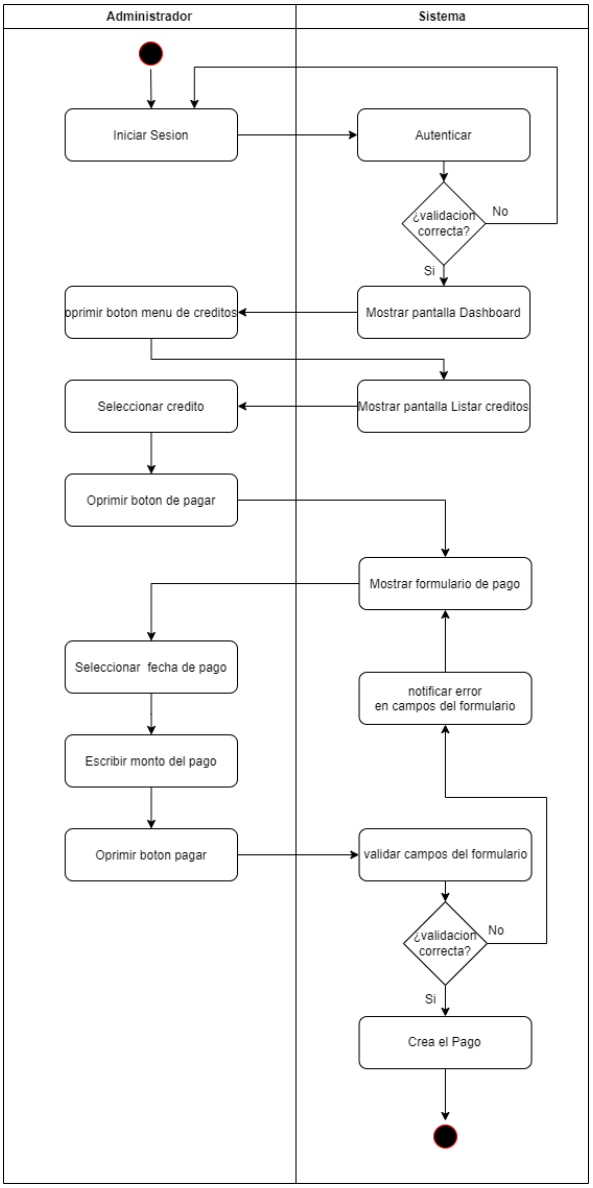
Posteriormente, el administrador accede a la sección de créditos pulsando sobre el botón "Menú de Créditos". Esto lleva a una pantalla donde se listan todos los créditos disponibles. Desde aquí, selecciona el crédito específico al cual desea registrar un pago, y procede a pulsar el botón "Pagar".

Esta acción abre el formulario de registro de pagos, en el cual el administrador indica la fecha de pago y el monto a pagar. Una vez ingresada esta información, envía el formulario a través del botón "Pagar".

A continuación, el sistema realiza una validación de los datos ingresados. En caso de detectar algún error, notifica al administrador para que realice las correcciones necesarias. Si la información ingresada es válida, el sistema registra exitosamente el pago en la base de datos, concluyendo así la operación de registro de pago.

Figura 3.

Diagrama de actividades para registrar un pago.



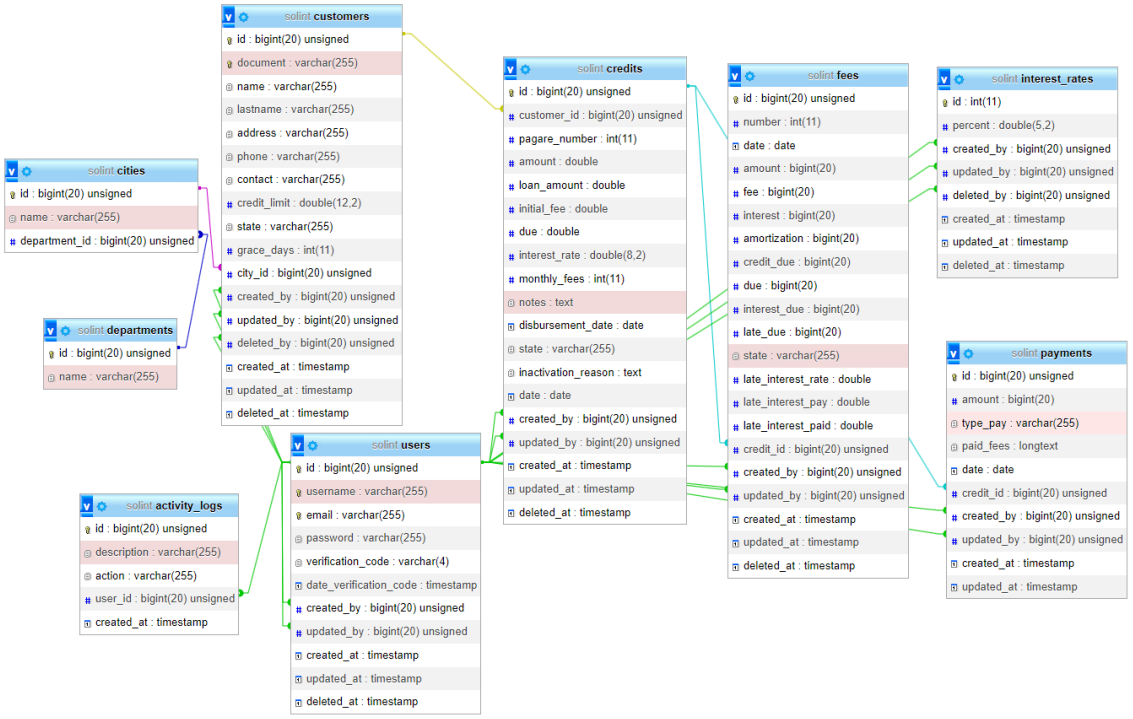
6.2.2 Diagrama Relacional

En el desarrollo de este sistema, se creó un diagrama relacional que representa la estructura de la base de datos utilizada en la implementación del mismo. Este diagrama muestra las tablas que conforman la base de datos, así como las relaciones entre ellas. Se puede observar cómo se modelaron las entidades involucradas en los procesos de los módulos

desarrollados, tales como el usuario, el cliente, el crédito, las cuotas, las tasas de interés y los pagos. Además, se establecieron las llaves primarias y foráneas necesarias para asegurar la integridad de la información. El diagrama relacional fue una herramienta fundamental en la fase de desarrollo de los módulos del sistema, ya que permitió visualizar la estructura de la base de datos y definir la forma en que se relacionan las distintas entidades.

Figura 4.

Modelo relacional.



6.2.3 Diseño De La Arquitectura

La arquitectura de software utilizada en este proyecto se basó en una API REST, la cual se compone de tres principales componentes: el servidor, el cliente y la base de datos.

El servidor fue desarrollado con Laravel, una plataforma de desarrollo en PHP que permite la construcción de servidores escalables y eficientes. El servidor se encarga de procesar y responder a las solicitudes del cliente, así como de interactuar con la base de datos.

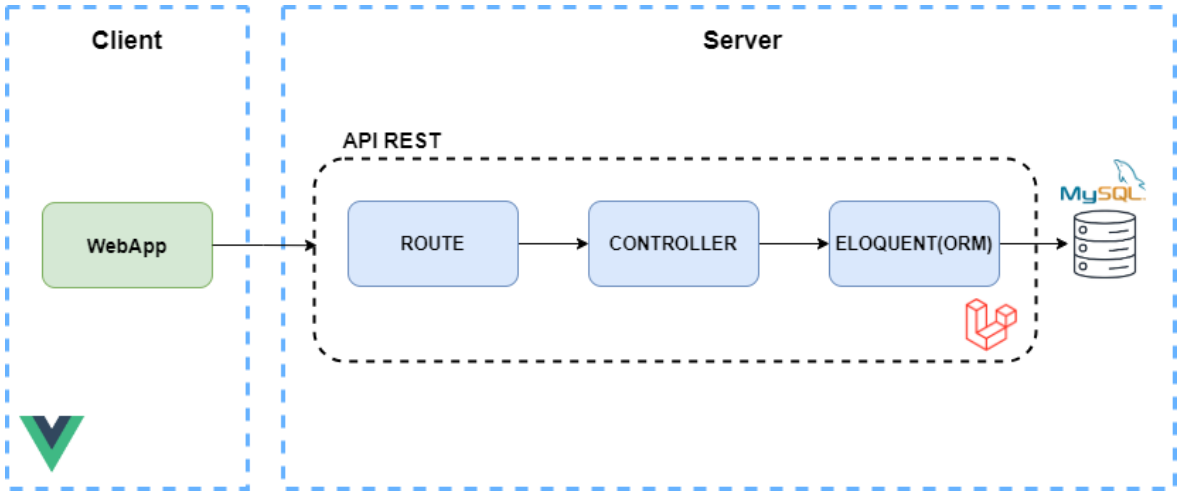
El cliente fue desarrollado en Vue.js, un framework de desarrollo de aplicaciones SPA multiplataforma que utiliza el lenguaje de programación JavaScript. El cliente se encarga de presentar la información al usuario final y de enviar solicitudes al servidor para obtener o enviar información.

La base de datos utilizada fue MySQL, un sistema de gestión de bases de datos relacional ampliamente utilizado en la industria. La base de datos se encarga de almacenar toda la información relevante del sistema, desde los usuarios hasta la información de los créditos y los pagos.

En conjunto, estos tres componentes permitieron la creación de una arquitectura escalable, robusta y eficiente, capaz de manejar grandes cantidades de datos y de responder a las solicitudes de los usuarios de manera rápida y eficiente. Además, la elección de tecnologías ampliamente utilizadas en la industria permitió una fácil integración con otras herramientas y sistemas, lo que permitió una mayor flexibilidad y capacidad de adaptación en el futuro.

Figura 5.

Arquitectura del software.



7. Diseño

7.1 Diseño De La Interfaz Del Usuario

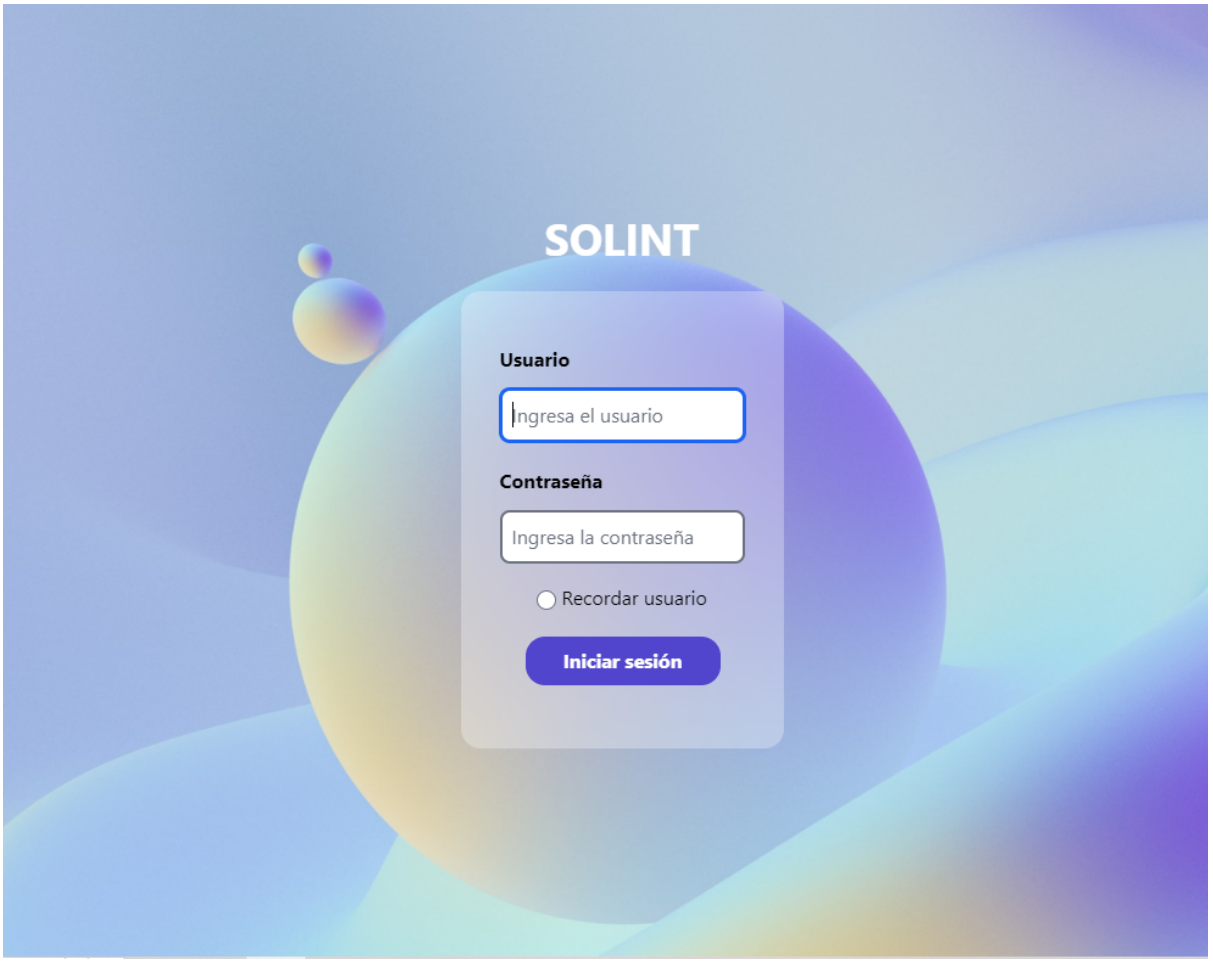
Para el inicio de este diseño se tuvo en cuenta la paleta de colores solicitada por el cliente. En este apartado podemos encontrar el diseño de las diferentes pantallas para los módulos desarrollados.

7.1.1 Diseño Del Inicio De Sesión

Se diseñó la pantalla de inicio de sesión teniendo en cuenta el formulario necesario para las credenciales del usuario y un check para recordar la información del usuario en el navegador.

Figura 6.

Bosquejo inicio de sesión.

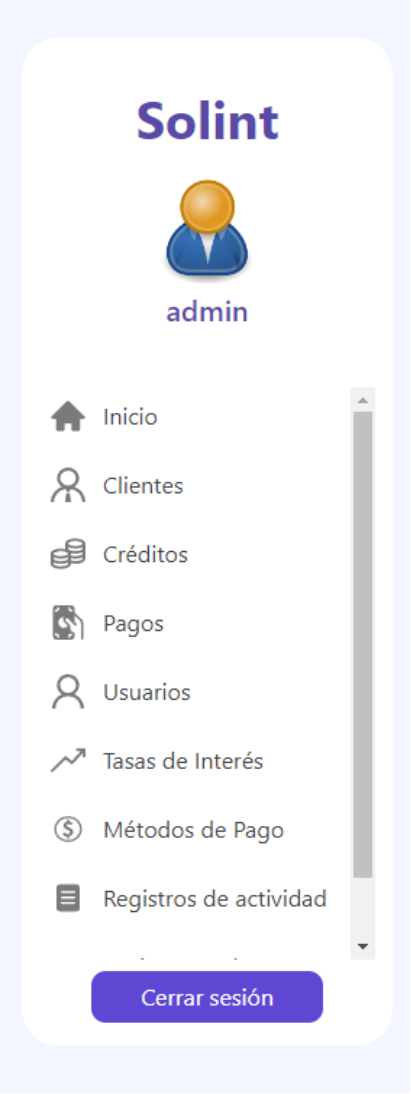


7.1.2 Diseño De La Barra De Navegación Para El Sistema

Se diseñó una barra de navegación que brinde acceso a todos los módulos del sistema, la cual estará integrada en todas las pantallas para brindar una mejor experiencia al usuario.

Figura 7.

Bosquejo barra de navegación.

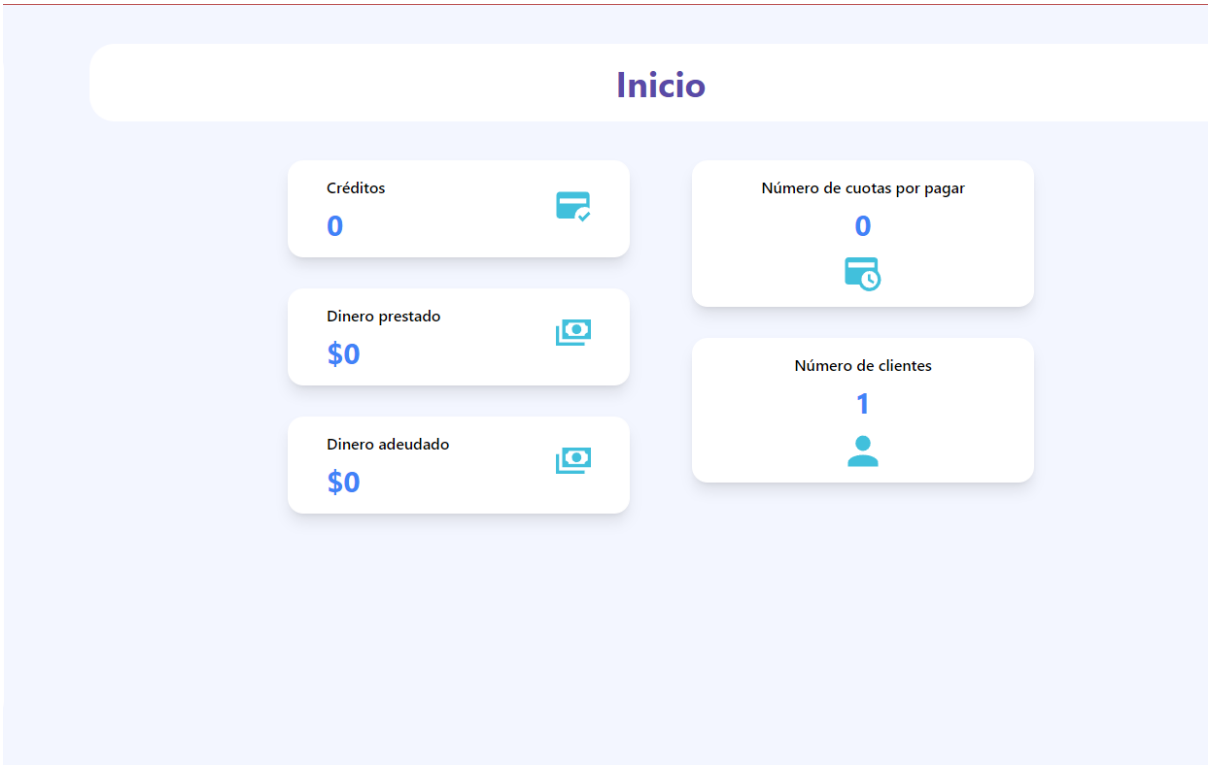


7.1.3 Diseño De La Pantalla De Inicio

Se diseñó la pantalla de inicio para que cumpla con la función de brindar un informe rápido y resumido del estado de los créditos y clientes del sistema.

Figura 8.

Bosquejo pantalla de inicio.

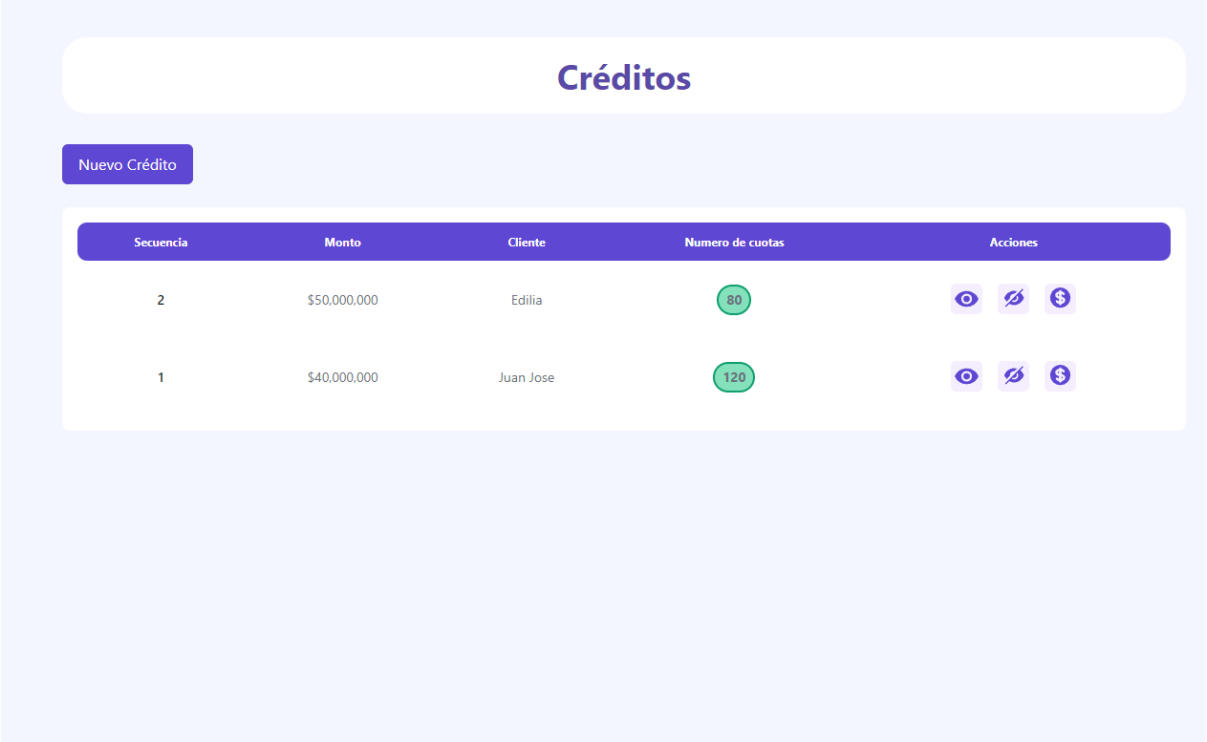


7.1.4 Diseño De La Pantalla Para Listar Los Créditos Del Sistema

Se diseñó la pantalla para mostrar una tabla ordenada al usuario con los créditos que existen hasta el momento dentro del sistema junto con botones para crear un nuevo crédito, acceder al detalle de un crédito y eliminarlo del sistema.

Figura 9.

Bosquejo pantalla de créditos dentro del sistema.

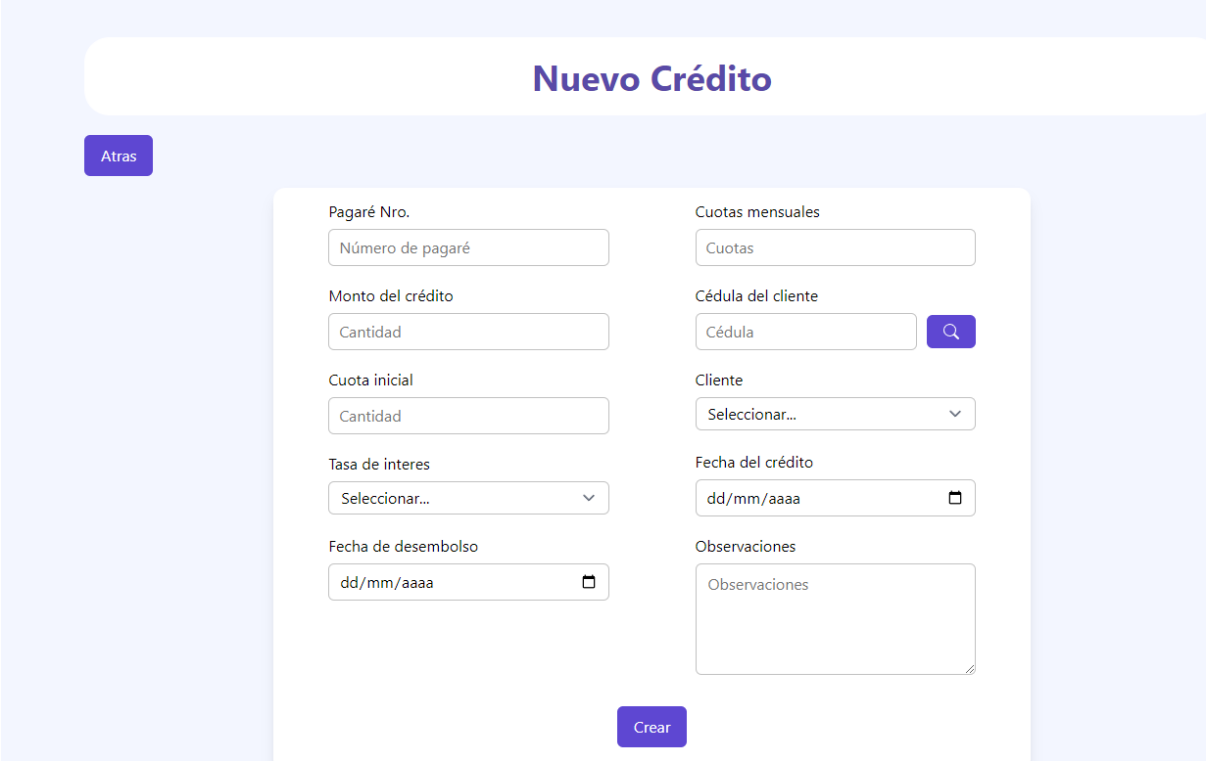


7.1.5 Diseño De La Pantalla Para Crear Un Nuevo Crédito

Se diseñó una pantalla para crear un nuevo crédito en el sistema con un formulario que solicite la información necesaria según los requerimientos del cliente.

Figura 10.

Bosquejo pantalla para crear un nuevo crédito.

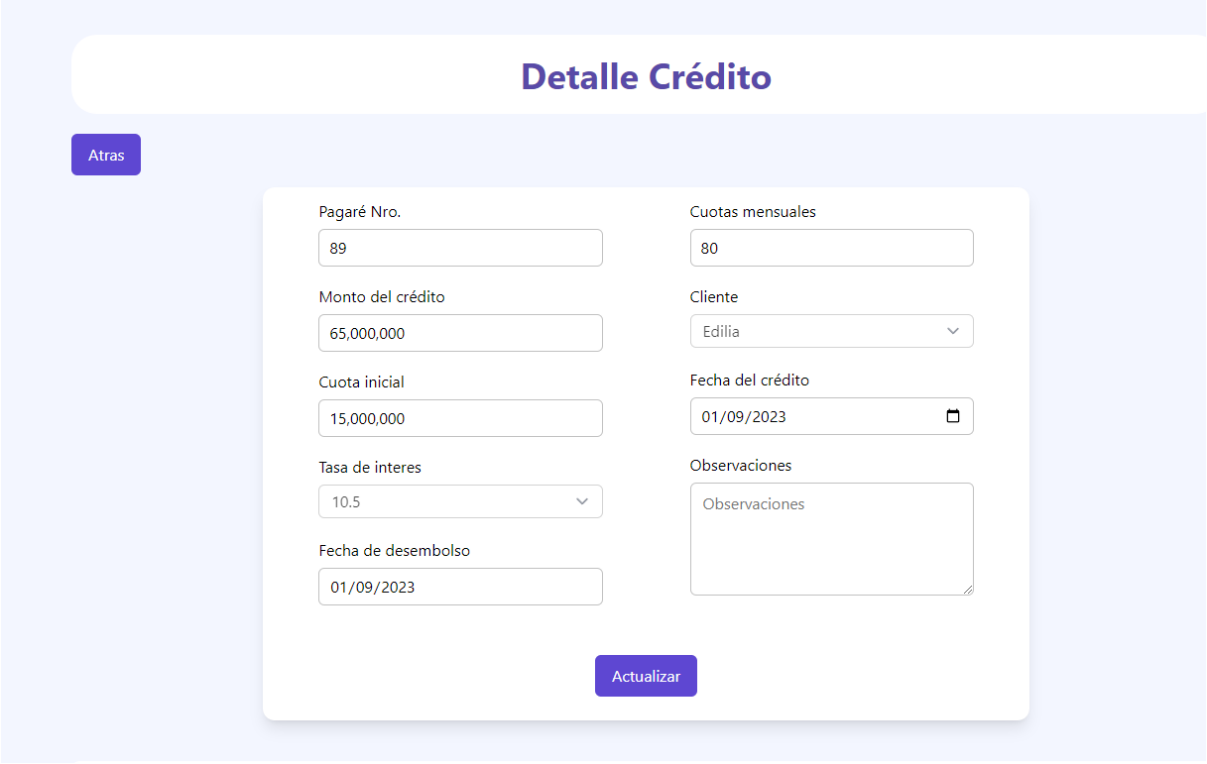


7.1.6 Diseño De La Pantalla Del Detalle De Un Crédito

Se diseñó una pantalla que muestre la información detallada de un crédito al usuario con un botón que permita actualizar solo el campo de observaciones.

Figura 11.

Bosquejo pantalla del detalle de un crédito.

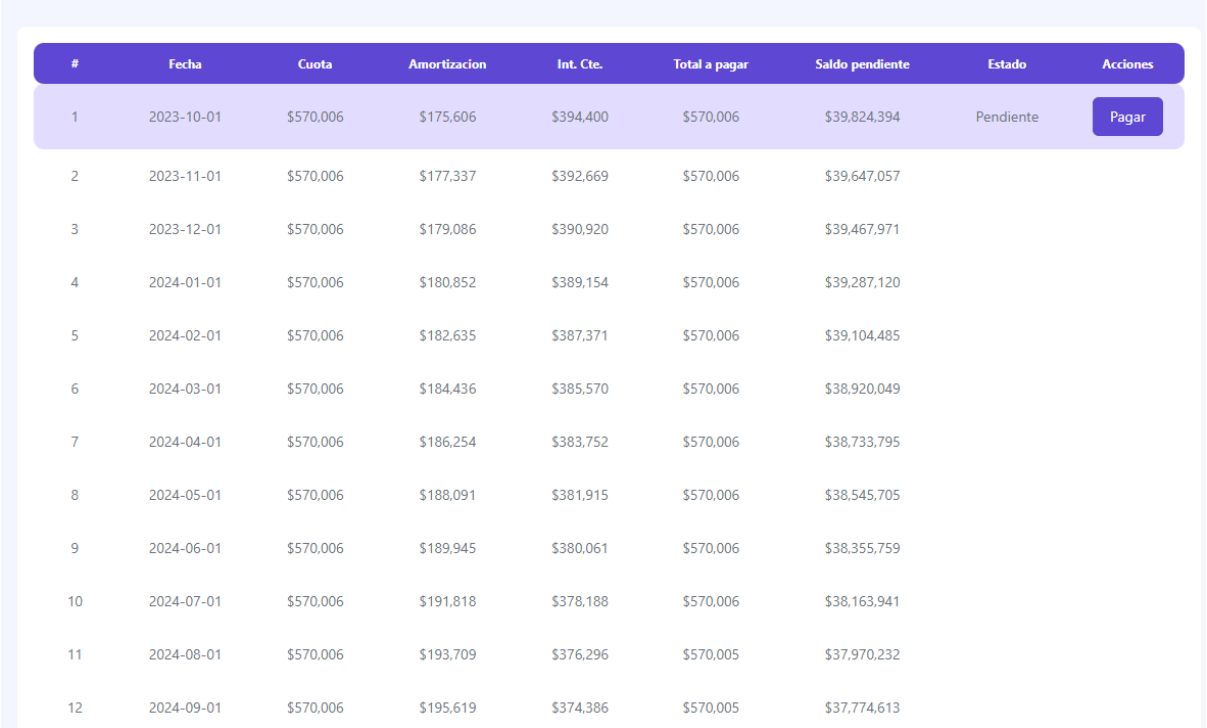


7.1.7 Diseño De La Tabla De Amortización De Un Crédito

Se diseñó una tabla que muestre la información detallada de las cuotas de un crédito con un botón para realizar el pago de la cuota pendiente más reciente según la fecha, esta tabla acompaña la pantalla de detalle de un crédito.

Figura 12.

Bosquejo tabla de amortización de un crédito.



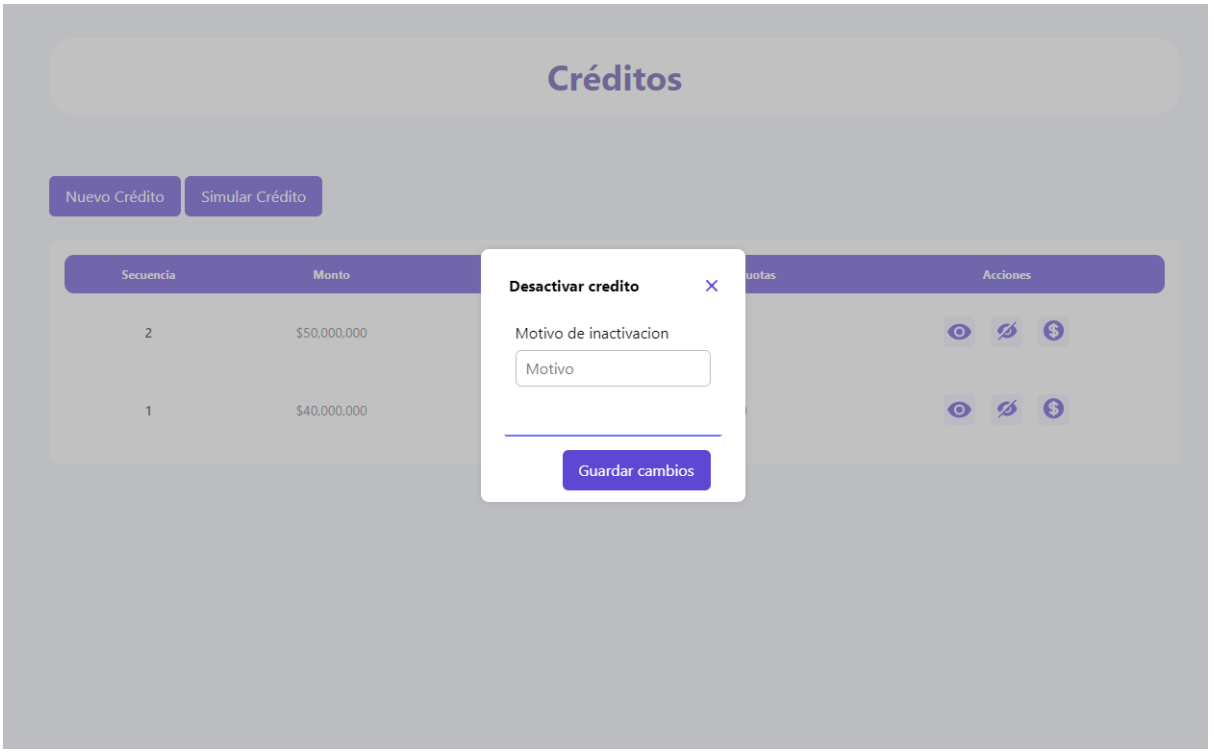
#	Fecha	Cuota	Amortizacion	Int. Cte.	Total a pagar	Saldo pendiente	Estado	Acciones
1	2023-10-01	$570,006	$175,606	$394,400	$570,006	$39,824,394	Pendiente	Pagar
2	2023-11-01	$570,006	$177,337	$392,669	$570,006	$39,647,057		
3	2023-12-01	$570,006	$179,086	$390,920	$570,006	$39,467,971		
4	2024-01-01	$570,006	$180,852	$389,154	$570,006	$39,287,120		
5	2024-02-01	$570,006	$182,635	$387,371	$570,006	$39,104,485		
6	2024-03-01	$570,006	$184,436	$385,570	$570,006	$38,920,049		
7	2024-04-01	$570,006	$186,254	$383,752	$570,006	$38,733,795		
8	2024-05-01	$570,006	$188,091	$381,915	$570,006	$38,545,705		
9	2024-06-01	$570,006	$189,945	$380,061	$570,006	$38,355,759		
10	2024-07-01	$570,006	$191,818	$378,188	$570,006	$38,163,941		
11	2024-08-01	$570,006	$193,709	$376,296	$570,005	$37,970,232		
12	2024-09-01	$570,006	$195,619	$374,386	$570,005	$37,774,613		

7.1.8 Diseño De La Pantalla Para Desactivar (Eliminar) Un Crédito

Se diseñó una pantalla que permite al usuario desactivar un crédito especificando el motivo por el cual se desactiva.

Figura 13.

Bosquejo pantalla para desactivar un crédito.

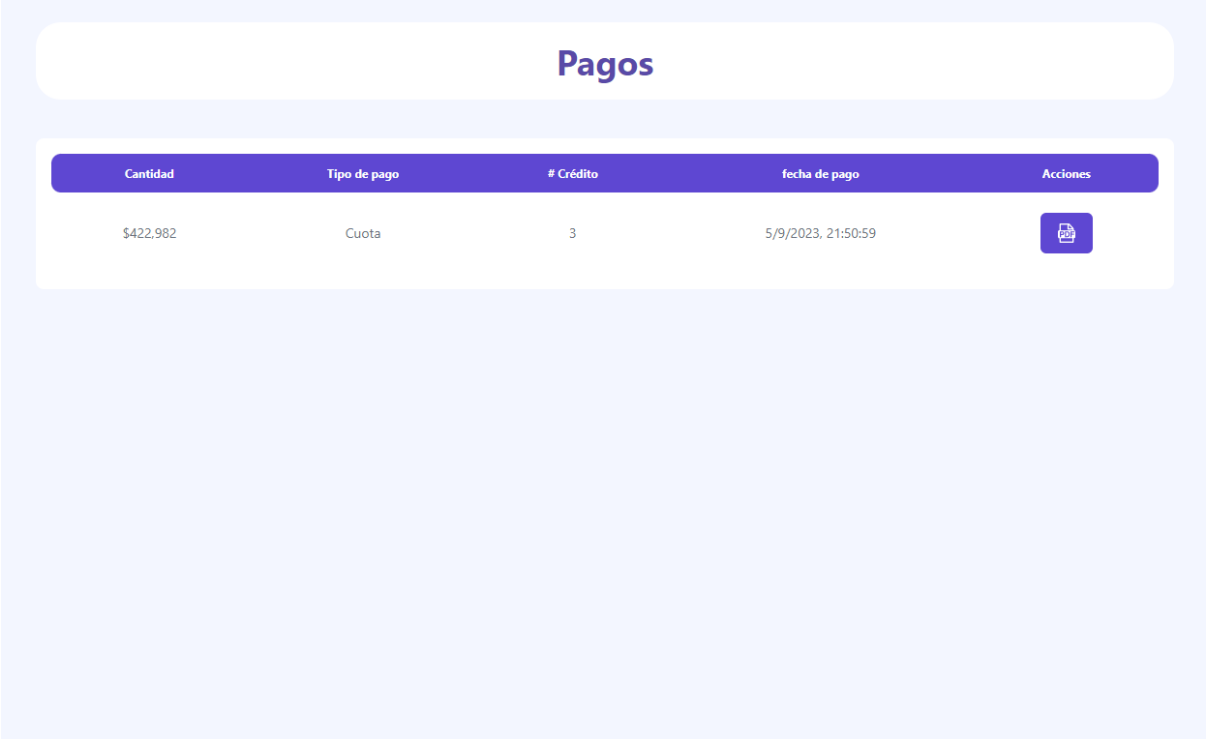


7.1.9 Diseño De La Pantalla Para Listar Los Pagos De Los Clientes

Se diseñó una pantalla para mostrar el historial de pagos registrados en el sistema dentro de una tabla.

Figura 14.

Bosquejo pantalla para listar pagos del sistema.

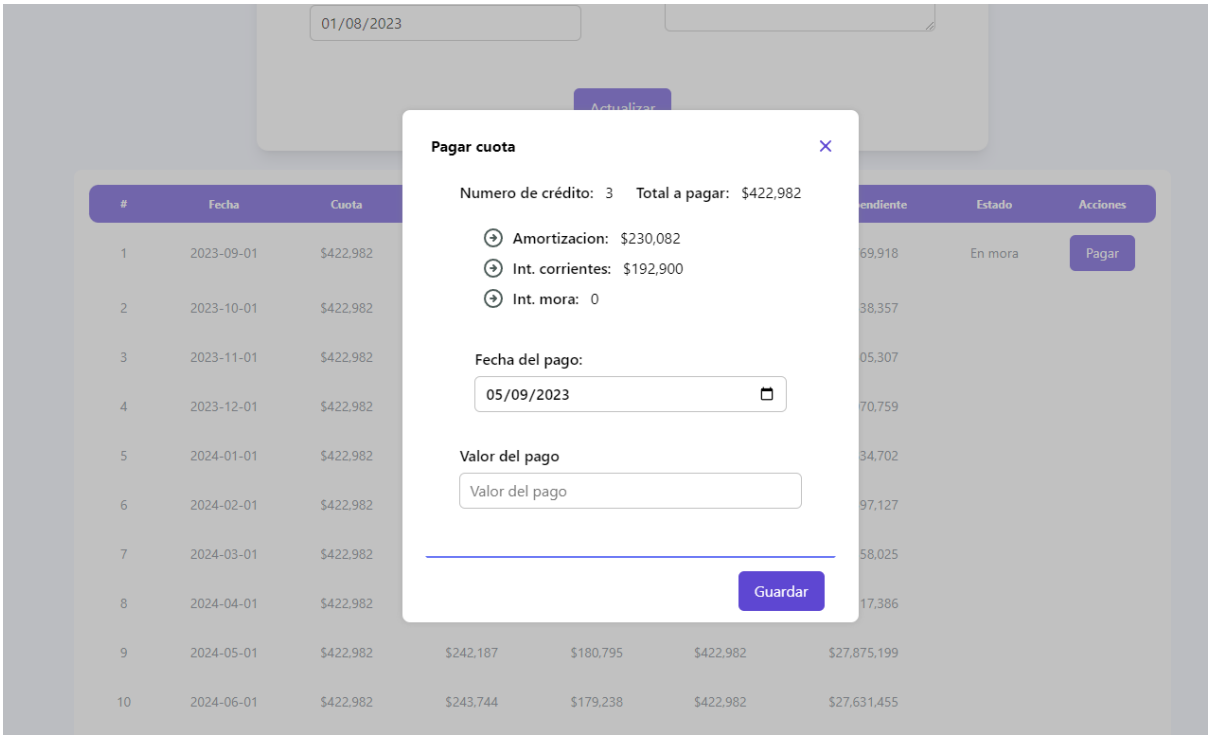


7.1.10 Diseño De La Pantalla Para Registrar El Pago De Una Cuota

Se diseñó una pantalla que permite al usuario registrar los pagos de cuotas (completas o parciales) por parte de los clientes.

Figura 15.

Bosquejo pantalla para registrar pago.

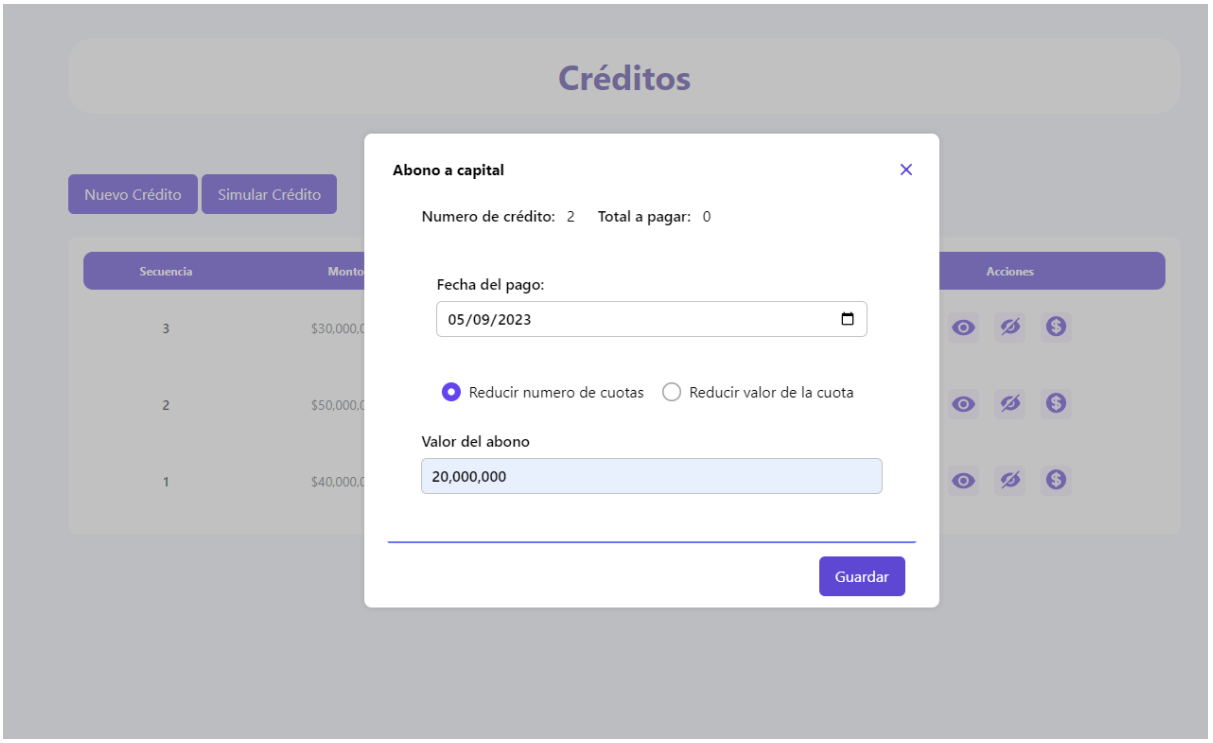


7.1.11 Diseño De La Pantalla Para Registrar Los Abonos A Capital De Los Clientes

Se diseñó una pantalla que permite al usuario registrar los pagos a capital por parte de los clientes, este le permite seleccionar el tipo de pago que desee realizar el cliente.

Figura 16.

Bosquejo pantalla para registrar abono a capital.



8. Desarrollo

El objetivo en esta etapa es concluir la funcionalidad de los módulos del sistema, cumpliendo con exactitud todos los requerimientos previamente establecidos. Dicha etapa transcurre de forma iterativa, ya que se encuentra en constante proceso de evaluación y deben ser corregidos los errores con el fin de entregar un sistema confiable.

8.1 Desarrollo De Software

Una vez definida la arquitectura y los casos de uso inicia esta fase, por tal razón fue necesario crear un flujo de trabajo por medio de un tablero virtual el cual ayudó a la gestión del proyecto, tablero conocido como JIRA, donde se crearon los apartados de tareas pendientes, tareas asignadas, tareas en desarrollo y tareas finalizadas, del desarrollo de frontend y backend de los módulos.

En la primera parte del proyecto, se desarrollaron interfaces iniciales para generar las vistas de los módulos, esto con el fin de agilizar y optimizar el proceso de desarrollo de la interfaz de usuario.

Seguido de esto se hizo la creación de un repositorio en GitHub el cual permite controlar las versiones y avances tanto en frontend como en backend, en secuencia a esto se empezó a desarrollar la base de datos relacional construida en MySql, una vez teniendo definida la base de datos se procede con la creación del Backend (creación del REST API) el cual fue construido con Laravel, encargado de proveer los servicios de los requerimientos preestablecidos.

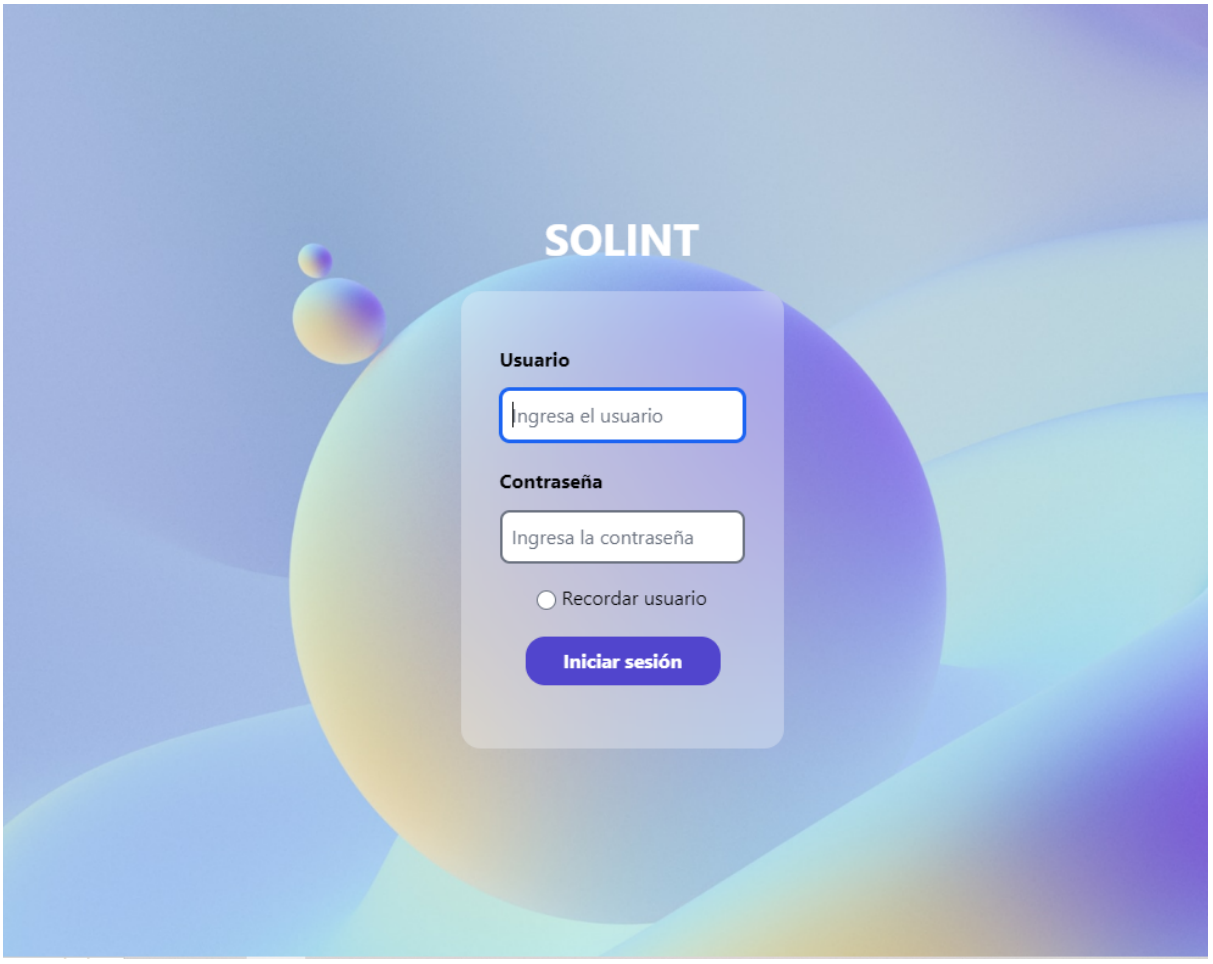
Una vez se finaliza la etapa de creación del REST API se procede a iniciar la fase de creación de interfaz de usuario usando el framework Vue.js, dicha interfaz se rige a las vistas previamente creadas, cumpliendo así con los requerimientos establecidos tales como:

8.1.1 Inicio De Sesión

La cual cumple con el requerimiento (RF01) teniendo los campos necesarios para la autenticación dentro del sistema.

Figura 17.

Vista inicio de sesión.

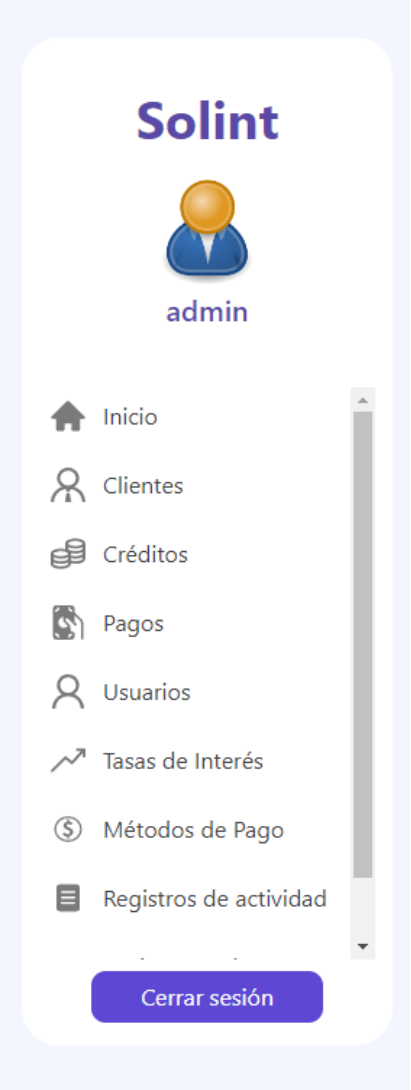


8.1.2 Cerrar Sesión

Con la vista de la barra de navegación la cual está integrada en las demás vistas dentro del sistema se cumple con el requerimiento (RF02) pues esta barra incluye un botón para cerrar sesión del sistema.

Figura 18.

Vista barra de navegación y cerrar sesión.

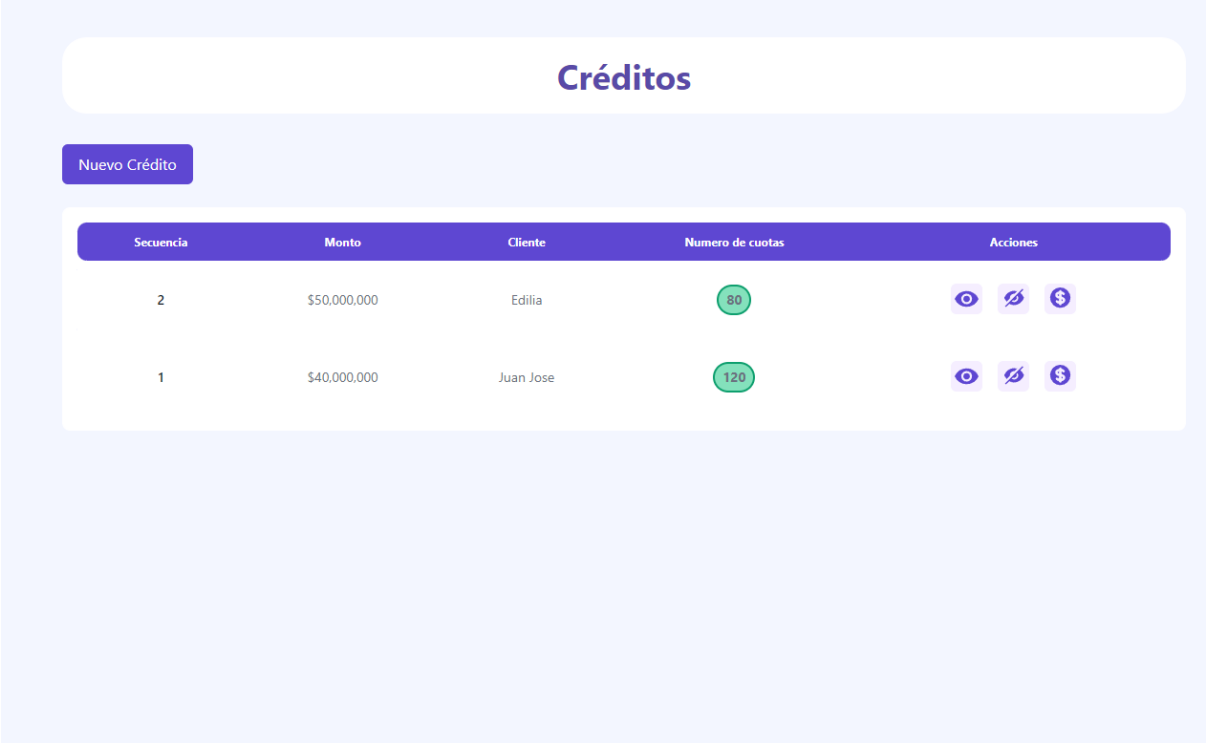


8.1.3 Listar Los Créditos Del Sistema

La siguiente vista cumple a cabalidad el requerimiento (RF03) pues lista todos los créditos presentes en el sistema y los muestra de manera ordenada en una tabla.

Figura 19.

Vista créditos del sistema.

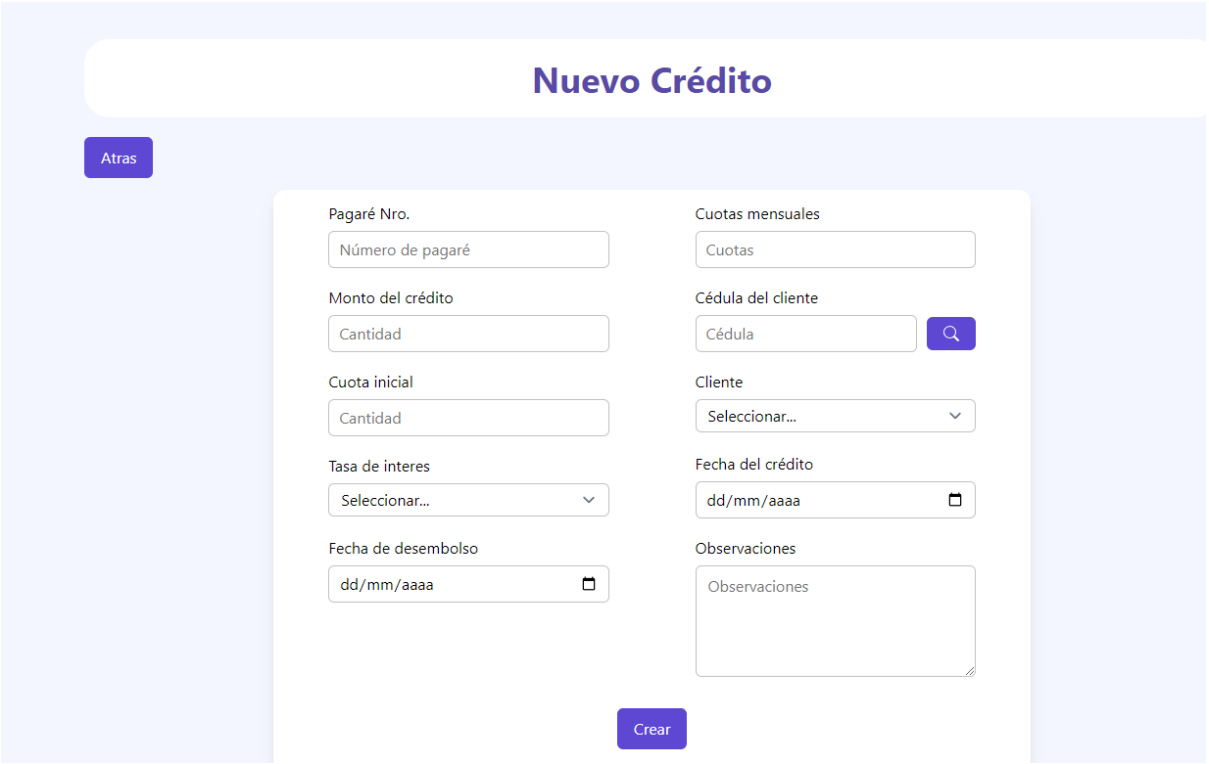


8.1.4 Crear Un Nuevo Crédito

Con esta vista se dan por completados los requerimientos (RF04 y RNF02) pues incluye los campos necesarios para la creación de un nuevo crédito en el sistema y a su vez valida que el valor del crédito no exceda el cupo máximo del cliente.

Figura 20.

Creación de créditos.

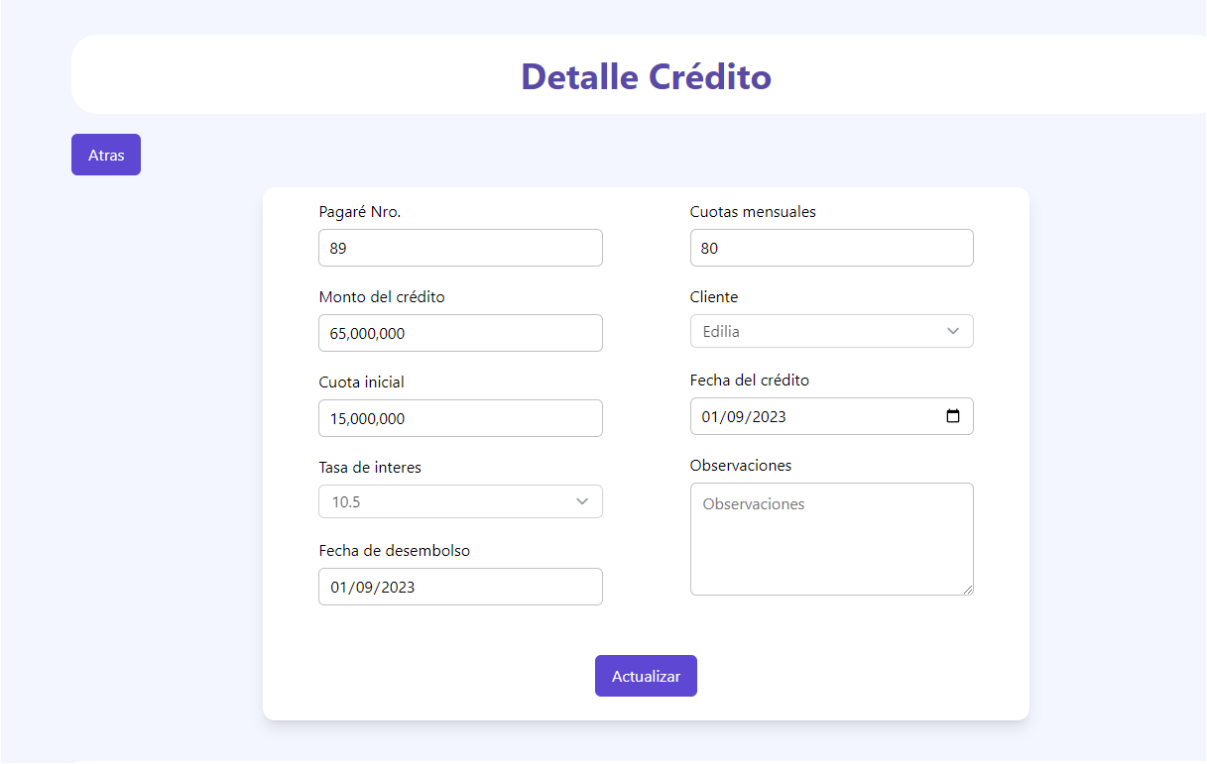


8.1.5 Editar Crédito

Con esta vista se cumple el requerimiento (RF05) para editar el campo de observaciones de un crédito.

Figura 21.

Detalle de crédito.

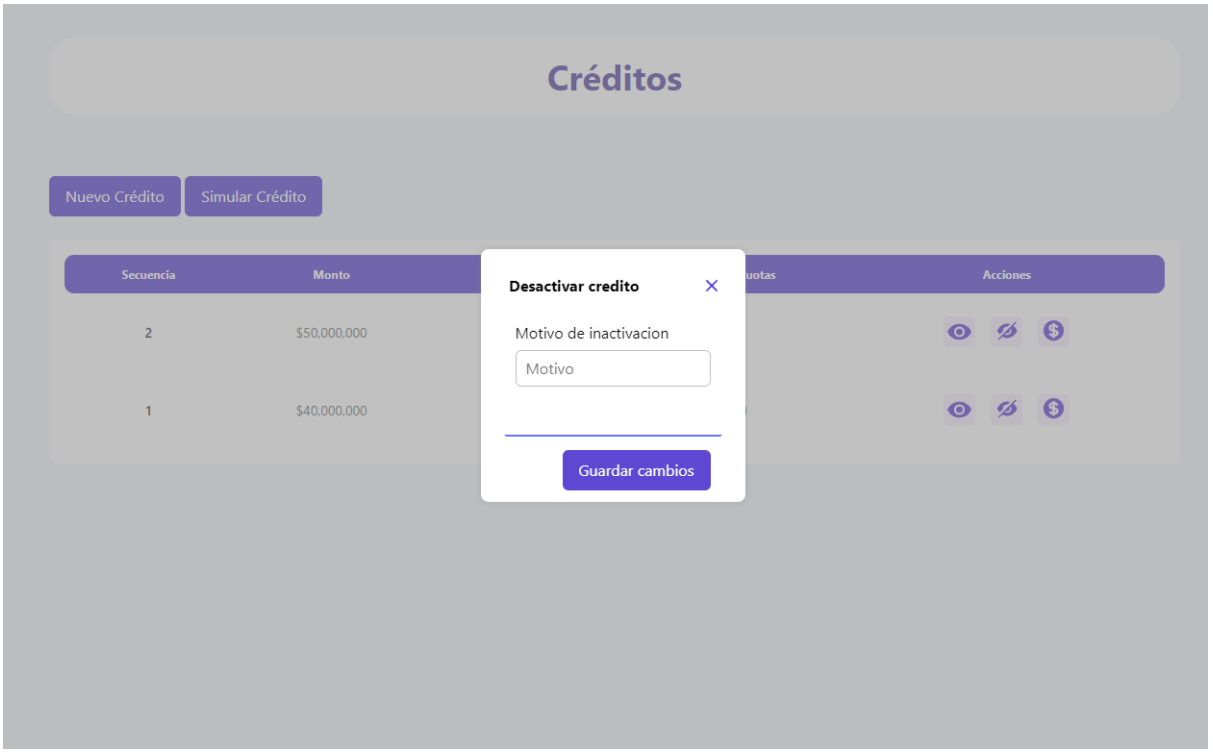


8.1.6 Desactivar (Eliminar) Un Crédito

Con la creación de este modal se da por completado el requerimiento (RF06) para desactivar un crédito detallando los motivos.

Figura 22.

Vista desactivar crédito.

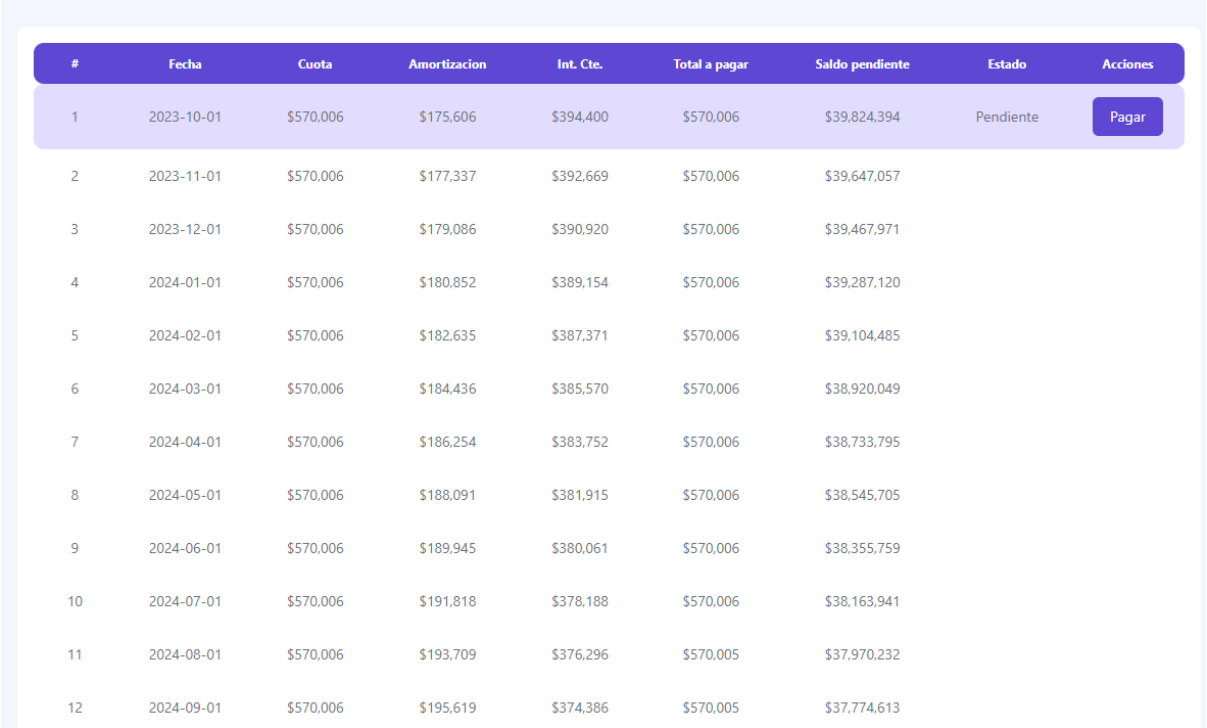


8.1.7 Calcular Y Mostrar Tabla De Amortización De Un Crédito

Con la construcción de esta tabla se dan por completados los requerimientos (RF07 y RNF01) pues se garantiza que el sistema está calculando las tablas de amortización de los créditos y el cliente muestra dicha información.

Figura 23.

Tabla de amortización.



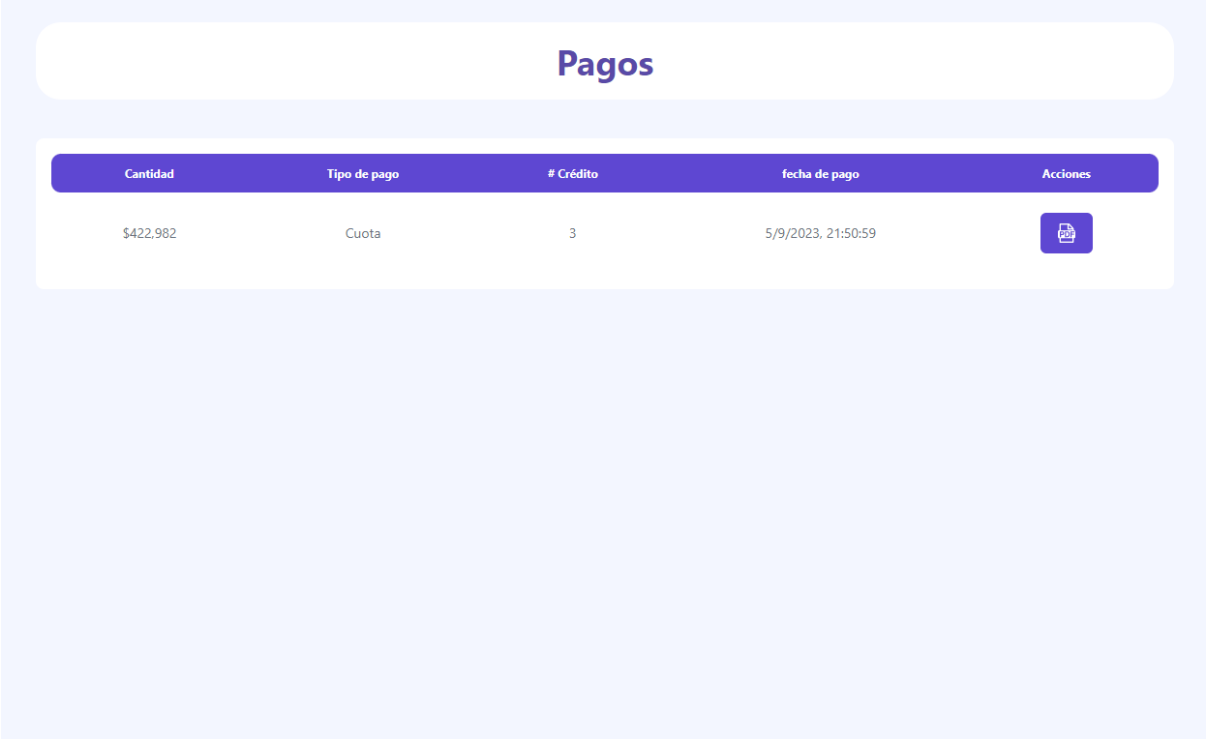
#	Fecha	Cuota	Amortizacion	Int. Cte.	Total a pagar	Saldo pendiente	Estado	Acciones
1	2023-10-01	$570,006	$175,606	$394,400	$570,006	$39,824,394	Pendiente	Pagar
2	2023-11-01	$570,006	$177,337	$392,669	$570,006	$39,647,057		
3	2023-12-01	$570,006	$179,086	$390,920	$570,006	$39,467,971		
4	2024-01-01	$570,006	$180,852	$389,154	$570,006	$39,287,120		
5	2024-02-01	$570,006	$182,635	$387,371	$570,006	$39,104,485		
6	2024-03-01	$570,006	$184,436	$385,570	$570,006	$38,920,049		
7	2024-04-01	$570,006	$186,254	$383,752	$570,006	$38,733,795		
8	2024-05-01	$570,006	$188,091	$381,915	$570,006	$38,545,705		
9	2024-06-01	$570,006	$189,945	$380,061	$570,006	$38,355,759		
10	2024-07-01	$570,006	$191,818	$378,188	$570,006	$38,163,941		
11	2024-08-01	$570,006	$193,709	$376,296	$570,005	$37,970,232		
12	2024-09-01	$570,006	$195,619	$374,386	$570,005	$37,774,613		

8.1.8 Listar Pagos

En esta vista se cumple el requerimiento (RF08) pues se muestra el historial de pagos registrados en el sistema.

Figura 24.

Vista listar pagos.

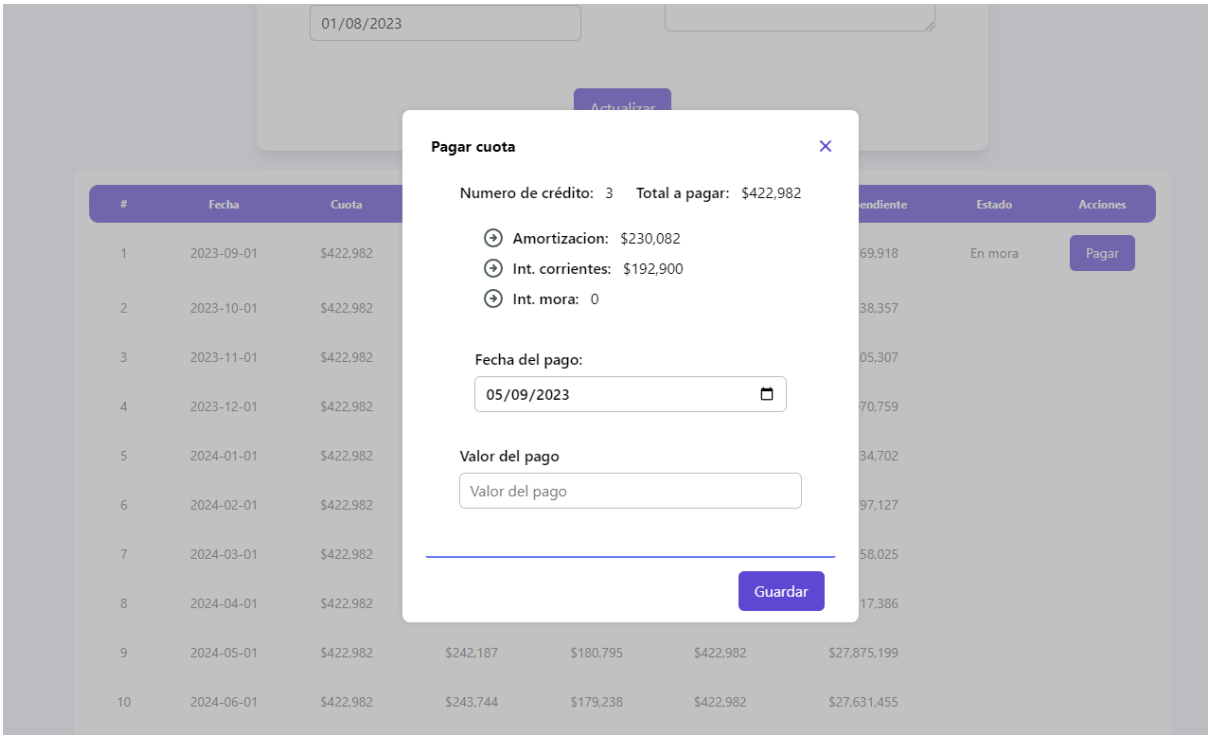


8.1.9 Registrar Pagos De Cuotas

Esta vista da por concluidos los requerimientos (RF09, RFN04 y RFN05) pues cumple no solo con la función de registrar los pagos si no calcular de manera correcta los valores que hay entre cuotas y deuda actual, además de actualizar el estado de las cuotas.

Figura 25.

Registrar pago de cuotas.

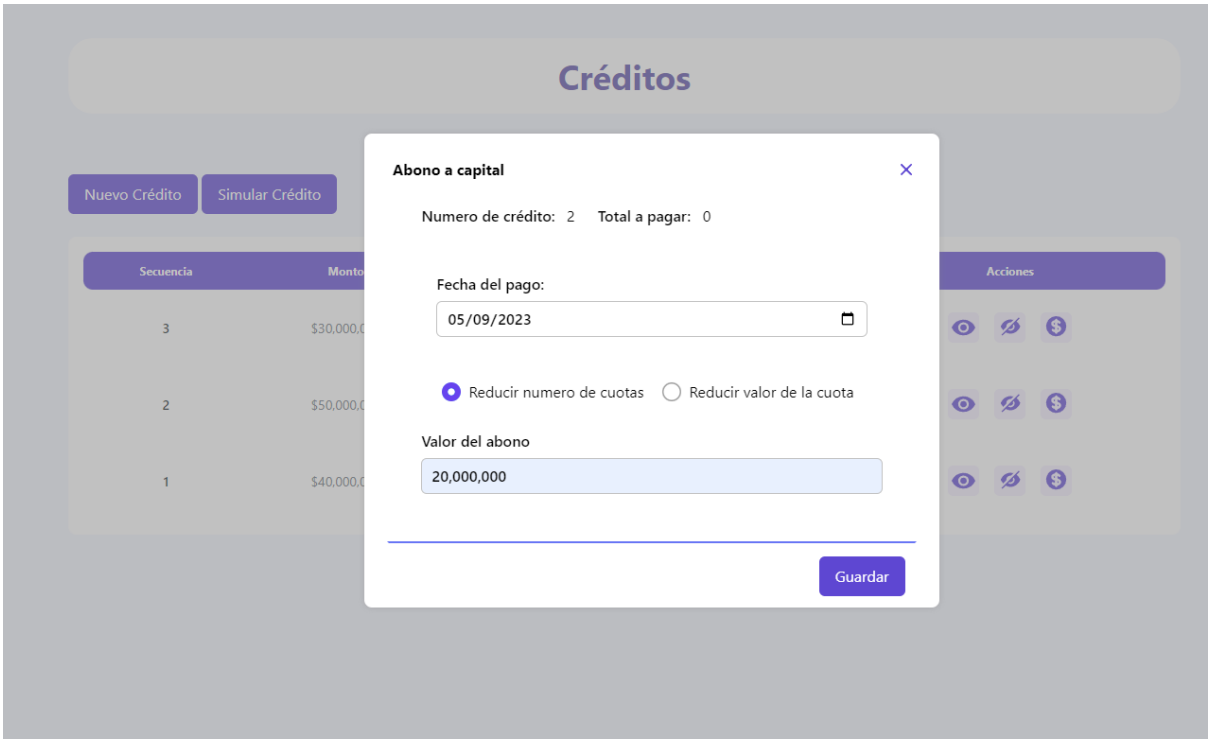


8.1.10 Registrar Pagos De Abonos A Capital

Esta vista da por concluidos los requerimientos (RF10 y RFN03) la cual cumple con la función de registrar y aplicar de manera correcta los abonos a capital.

Figura 26.

Registrar abono a capital.



9. Pruebas

Con el fin de garantizar un software de alta calidad y un funcionamiento óptimo de la aplicación desarrollada durante este proyecto, se implementó una serie rigurosa de pruebas técnicas. Estas pruebas, centrándose en diversos aspectos críticos del sistema, permitieron optimizar el rendimiento y garantizar una experiencia de usuario fluida y satisfactoria. A continuación se describen las distintas etapas de las pruebas realizadas:

9.1 Pruebas De Rendimiento Y Funcionales

La estructura de pruebas se diseñó con un enfoque centrado en garantizar la eficiencia y estabilidad de la aplicación, incluso bajo condiciones de alta demanda. A través de pruebas funcionales con carga, se midió la capacidad de respuesta del sistema ante un alto volumen de solicitudes simultáneas. A continuación, se detallan las distintas fases de estas pruebas:

9.1.1 Pruebas Unitarias

Durante esta fase, se efectuaron pruebas unitarias para validar la funcionalidad y la fiabilidad de los componentes individuales de la aplicación. Utilizando Laravel para el backend y Vue para el frontend, se garantizó que cada función y componente operará de manera óptima y conforme a los requerimientos especificados.

Backend (Laravel): Se enfocó en validar la lógica de negocio, garantizando que todas las operaciones relacionadas con la gestión de créditos, pagos y usuarios funcionaran de manera adecuada.

Frontend (Vue): Se concentró en verificar la correcta implementación de los componentes visuales y la interacción entre ellos, asegurando una navegación fluida y una interfaz de usuario eficiente.

9.1.2 Pruebas De Integración

Esta etapa involucró pruebas de integración centradas en evaluar la cohesión entre los distintos módulos y componentes de la aplicación. Fue fundamental garantizar una comunicación fluida y sin errores entre las funcionalidades proporcionadas por Laravel y Vue. Estas pruebas permitieron:

Verificar la correcta comunicación entre el servidor (Laravel) y el cliente (Vue), incluyendo la validación de las respuestas del API y la manipulación adecuada de los datos.

Garantizar que los flujos de trabajo que involucran múltiples componentes funcionen de manera integrada y sin errores.

9.1.3 Pruebas End To End (E2e)

Finalmente, se realizaron pruebas E2E para evaluar el sistema en un entorno que simula situaciones de uso real. Estas pruebas englobaron:

Verificación de flujos de trabajo completos, desde el inicio hasta la finalización, garantizando que todas las funcionalidades interactúan de manera cohesiva.

Simulación de interacciones del usuario real para garantizar que la aplicación ofrece una experiencia de usuario coherente y sin fricciones, desde el inicio de sesión hasta transacciones más complejas como la gestión de créditos y pagos.

Las pruebas E2E se ejecutaron en diferentes navegadores y dispositivos para garantizar una amplia cobertura y compatibilidad.

9.2 Resultado De Pruebas

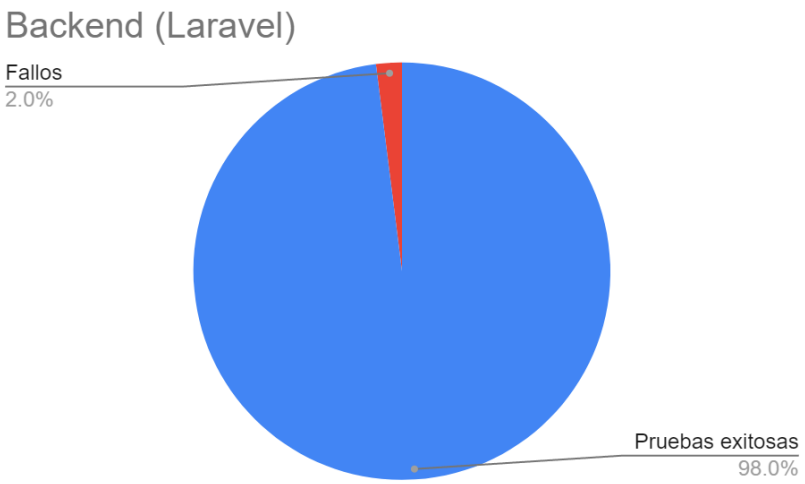
9.2.1 Resultados De Pruebas Unitarias

Backend (Laravel)

Durante la fase de pruebas unitarias en el backend desarrollado con Laravel, se llevaron a cabo un total de 50 pruebas meticulosas diseñadas para evaluar la funcionalidad y fiabilidad de los componentes individuales del sistema. De estas, 49 resultaron exitosas, evidenciando la alta estabilidad y eficiencia del backend. A pesar de identificar un fallo, este fue atendido y resuelto de manera oportuna, permitiendo una optimización continua del sistema. Este riguroso proceso de pruebas permitió alcanzar una cobertura de código del 96%, lo que no solo asegura una alta calidad en la base de código, sino también promete una experiencia de usuario consistente y satisfactoria.

Figura 27.

Gráfica pruebas BackEnd.

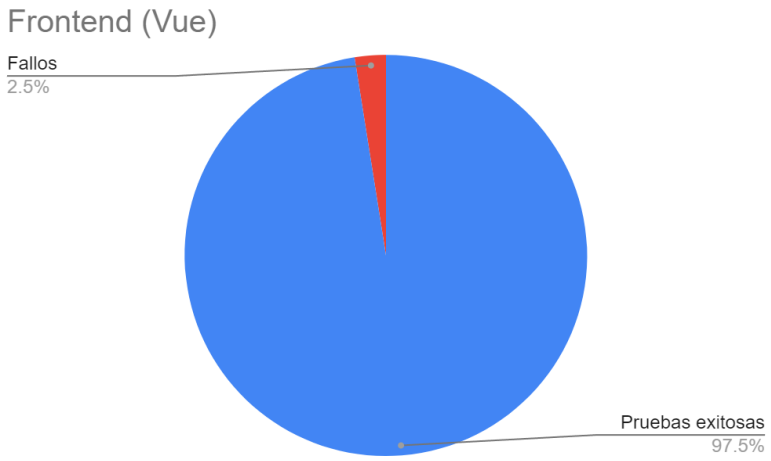


Frontend (Vue)

En la parte del frontend, construida utilizando el framework Vue.js, se implementó una serie rigurosa de pruebas unitarias para asegurar la eficacia y la interactividad del interfaz de usuario. Se realizaron 40 pruebas detalladas, de las cuales 39 fueron completadas con éxito, demostrando así la solidez y la fiabilidad del frontend. A pesar de haber encontrado un pequeño fallo durante las pruebas, se tomaron las medidas adecuadas para corregirlo prontamente, asegurando una experiencia de usuario óptima. Este conjunto de pruebas permitió lograr una notable cobertura de código del 95%, garantizando una implementación limpia y eficiente que satisface las expectativas de los usuarios.

Figura 28.

Gráfica pruebas FrontEnd.



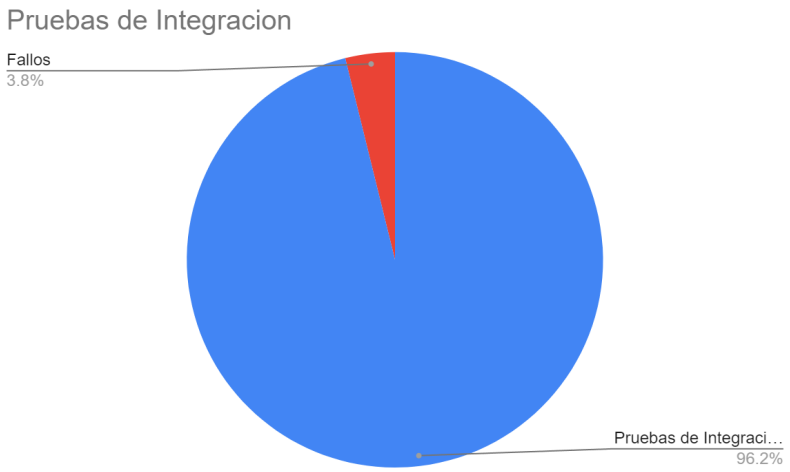
9.2.2 Resultados De Pruebas De Integración

En la fase de pruebas de integración, se efectuaron 26 pruebas críticas, diseñadas para evaluar la cohesión y la interoperabilidad entre los distintos módulos y componentes de la aplicación. De estas, 25 se ejecutaron con éxito, lo que indica una integración armoniosa y efectiva de las distintas partes del sistema. Se identificó un único fallo, el cual fue prontamente abordado y corregido, garantizando así una comunicación fluida y sin errores

entre las funcionalidades proporcionadas por Laravel y Vue.js. Este proceso de pruebas de integración no sólo asegura una transición suave entre los distintos componentes, sino que también refuerza la robustez y la fiabilidad del sistema como un todo.

Figura 29.

Gráfica pruebas de integración.



Casos de prueba específicos centrados en la comunicación entre Laravel y Vue:

- Validación de Respuestas de API

Objetivo: Asegurar una comunicación fluida y precisa entre el frontend y el backend.

Descripción: Se realizó una serie de pruebas donde se verificó que todas las respuestas del API de Laravel fueran recibidas y manejadas correctamente por Vue, garantizando así una interacción sin problemas entre las dos capas.

Resultado: Exitoso.

- Manejo de Errores de API

Objetivo: Prevenir y gestionar de manera efectiva posibles errores que puedan surgir durante las interacciones API.

Descripción: Se implementaron pruebas para asegurar que los errores de API, especialmente los códigos de estado comunes como 400 y 500, fueran manejados adecuadamente y mostrados en el frontend, facilitando así una rápida identificación y resolución de problemas.

Resultado: Exitoso.

- Autenticación y Autorización

Objetivo: Mantener la seguridad e integridad de los datos del usuario.

Descripción: Se realizaron pruebas específicas para verificar que los tokens JWT se utilizaran correctamente para la autenticación y la autorización entre Vue y Laravel, asegurando una gestión segura de las sesiones de usuario.

Resultado: Exitoso.

- Transmisión de Datos de Formularios

Objetivo: Garantizar una transferencia de datos precisa y segura entre el cliente y el servidor.

Descripción: Se llevó a cabo una serie de pruebas para confirmar que los datos de los formularios en Vue fueran transmitidos y procesados de manera precisa por Laravel, permitiendo así transacciones de datos sin errores.

Resultado: Exitoso.

- Sincronización de Estados de la Aplicación

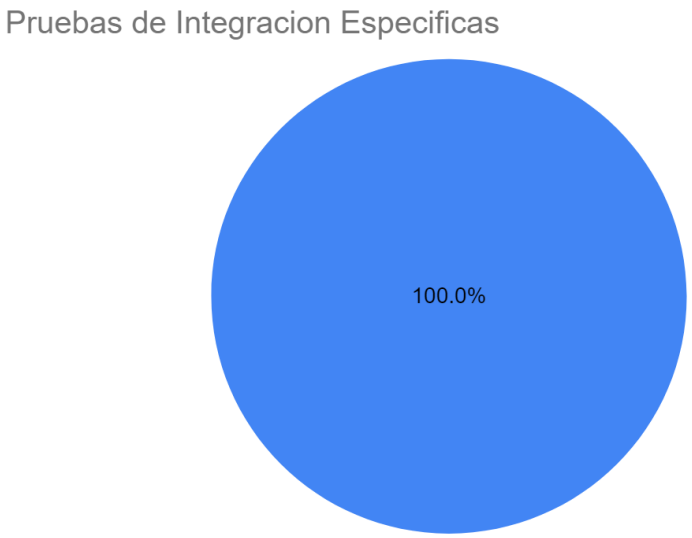
Objetivo: Asegurar una experiencia de usuario coherente a través de todas las sesiones y transacciones.

Descripción: Se condujeron pruebas para verificar que los estados de la aplicación, como el estado de sesión del usuario, se sincronizaran correctamente entre Vue y Laravel, garantizando una experiencia de usuario uniforme y fluida.

Resultado: Exitoso.

Figura 30.

Gráfica pruebas de integración específicas.

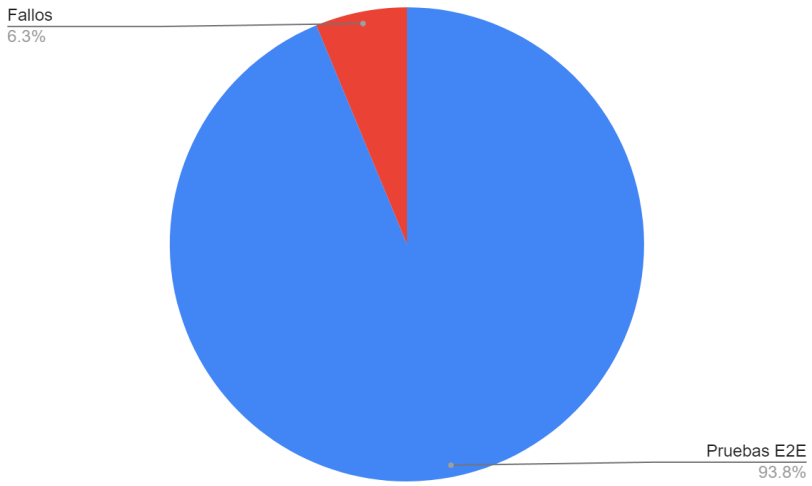


9.2.3 Resultados De Pruebas End To End (E2e)

En la etapa final de la evaluación, se realizaron pruebas End-to-End (E2E) cruciales para asegurar que la aplicación funciona de manera cohesiva y fluida, simulando escenarios de uso real. Se ejecutaron 16 pruebas E2E exhaustivas, que abarcaron una amplia gama de flujos de trabajo y procesos en la aplicación. De estas, 15 pruebas fueron completadas con éxito, señalando una integración efectiva y una interacción armónica entre los diversos componentes del sistema. A pesar de un fallo identificado, se tomó nota para futuras mejoras y optimizaciones. Adicionalmente, se llevó a cabo una evaluación multiplataforma, probando la aplicación en diferentes navegadores y dispositivos, lo que resultó en un notable 96% de éxito. Sin embargo, se identificó un problema menor en Internet Explorer, que será abordado para garantizar una compatibilidad más amplia y una experiencia de usuario superior.

Figura 31.

Gráfica pruebas E2E.



9.3 Conclusiones Y Optimizaciones Post-Pruebas

Después de completar una serie rigurosa de pruebas, que abarcaban desde pruebas unitarias hasta pruebas E2E, se identificaron varias áreas de oportunidad para optimizar aún más la aplicación. Este proceso crítico permitió no solo identificar y corregir fallos menores, sino también destacar la eficacia de la integración entre las tecnologías de Vue y Laravel.

A través de las pruebas, se logró una significativa mejora en la eficiencia y estabilidad del sistema. Se hicieron ajustes finos, especialmente en la comunicación entre el frontend y el backend, asegurando una transición suave y una interacción cohesiva. Además, se llevó a cabo una revisión exhaustiva para garantizar que la aplicación cumplía con todos los requerimientos funcionales y no funcionales establecidos, ofreciendo así una experiencia de usuario que no solo es satisfactoria sino altamente profesional.

Esta etapa de pruebas también sirvió como una validación de la robustez de la arquitectura implementada, demostrando que el sistema está bien equipado para manejar diferentes escenarios y cargas de trabajo con una alta tasa de éxito. En general, las

optimizaciones post-pruebas han elevado la calidad del software, preparándolo para una implementación exitosa en el entorno de producción.

10. Conclusiones

Se ha logrado cumplir con el objetivo general de analizar, diseñar y desarrollar los módulos de autenticación, créditos y pagos de un sistema de información para el control de créditos inmobiliarios. A su vez, los objetivos específicos planteados han sido cumplidos de manera satisfactoria, ya que se identificaron los requerimientos funcionales y no funcionales necesarios para el desarrollo de la aplicación, se diseñó una arquitectura de software que se adaptó a dichos requerimientos, se implementaron los módulos del sistema y finalmente, se validó su funcionamiento mediante un conjunto de pruebas de software.

El desarrollo de estos módulos dentro del sistema permitirá que funcione a totalidad, facilitando el acceso a la información de los créditos inmobiliarios que tenga la empresa y los que realice a futuro, también permitirá llevar un control unificado de los pagos efectuados por los clientes, además el sistema garantiza la seguridad de la información y el acceso limitado a esta solo a las personas autorizadas.

Además, el desarrollo de esta aplicación ha permitido aplicar conocimientos teóricos y prácticos en el área de desarrollo de software, así como también en el área de gestión de proyectos, al seguir una metodología adecuada y cumplir con los plazos establecidos para la entrega del proyecto.

En cuanto a la validación del prototipo, se puede concluir que el conjunto de pruebas implementadas posibilitó comprobar el correcto funcionamiento del sistema, identificando oportunidades de mejora, lo que permitió realizar ajustes en el sistema para mejorar su experiencia de uso.

11. Trabajo A Futuro

Para futuros trabajos en el desarrollo del sistema de información de créditos inmobiliarios se recomienda implementar nuevos módulos y funcionalidades con el fin de mejorar significativamente la usabilidad y eficiencia del sistema. En primer lugar se recomienda crear un módulo orientado hacia los clientes que solicitan los créditos, con funcionalidades que les permitan visualizar la información de sus respectivos créditos, su historial de pagos y estado actual de las cuotas, todo esto teniendo en cuenta la seguridad e integridad de la información.

Además, se recomienda la implementación de una pasarela de pagos que conecte con todas las entidades bancarias necesarias para que los clientes puedan realizar sus pagos de manera virtual en el momento que así lo deseen, es imperativo que esta pasarela de pagos cumpla con las normas de seguridad y fiabilidad establecidas para garantizar a los usuarios una buena experiencia con el uso del sistema. Se recomienda el uso de pasarelas de pago populares y confiables a nivel internacional como por ejemplo: Wompi.

Por último, se recomienda diseñar e implementar vistas responsivas de todo el sistema para las diferentes resoluciones de pantallas disponibles en todo el mercado, garantizando así la portabilidad y accesibilidad del mismo.

Referencias Bibliográficas

ADM Cloud Services. (s.f.). ¿Qué es un framework? Tipos de frameworks y sus características. Retrieved September 11, 2023, from https://admcloudservices.com/blog/que-es-un-framework-tipos-de-frameworks.html

Amazon Web Services. (s.f.). ¿Qué es JavaScript? Recuperado de https://aws.amazon.com/es/what-is/javascript/

AWS. (s.f.). ¿Qué es una API RESTful? Retrieved September 12, 2023, from https://aws.amazon.com/es/what-is/restful-api/

Barzana, A. (s.f.). Bases de datos relacionales. Recuperado de https://dis.um.es/~barzana/Informatica/IAGP/IAGP_BD_Relacional.html

IBM. (s.f.). Defining use cases. In IBM Engineering Lifecycle Management (Version 6.0.3). Retrieved September 12, 2023, from https://www.ibm.com/docs/es/engineering-lifecycle-management-suite/lifecycle-management/6.0.3?topic=requirements-defining-use-cases

IBM. (s.f.). Software testing. Recuperado de https://www.ibm.com/mx-es/topics/software-testing

KeepCoding. (s.f.). ¿Qué son las pruebas de integración? Recuperado de https://keepcoding.io/blog/que-son-las-pruebas-de-integracion/

KeepCoding. (s.f.). ¿Qué son las pruebas unitarias de software? Recuperado de https://keepcoding.io/blog/que-son-las-pruebas-unitarias-de-software/

Laravel. (s.f.). Installation: Why Laravel. Retrieved September 11, 2023, from https://laravel.com/docs/10.x/installation#why-laravel

Microsoft. (s.f.). Visual Studio Code. Recuperado de https://visualstudio.microsoft.com/es/

Moreno, A. (2019, Febrero 28). ¿Qué es GitHub y qué le ofrece a los desarrolladores? Xataka. Recuperado de https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores

Nigmacode. (s.f.). JWT en Laravel: Autenticación API. Retrieved September 12, 2023, from https://www.nigmacode.com/laravel/jwt-en-laravel/

OpenWebinars. (s.f.). ¿Qué es Postman? Recuperado de https://openwebinars.net/blog/que-es-postman/

Oracle. (s.f.). ¿Qué es una base de datos relacional? Recuperado de https://www.oracle.com/co/database/what-is-a-relational-database/

PHP Group. (2021). ¿Qué es PHP? Recuperado de https://www.php.net/manual/es/intro-whatis.php

Platzi. (s.f.). La arquitectura de Laravel y su ecosistema. Retrieved September 12, 2023,from https://platzi.com/blog/arquitectura-laravel/#:~:text=La%20arquitectura%20de%20Laravel%20es,con%20la%20base%20de%20datos.

QAlified. (s.f.). End-to-End Testing. Recuperado de https://qalified.com/es/blog/end-to-end-testing/#:~:text=Las%20pruebas%20end%2Dto%2Dend%20se%20pueden%20definir%20simplemente%20como,espera%20analizando%20todos%20sus%20componentes.

Vue.js. (s.f.). Introduction to Vue.js. Retrieved September 11, 2023, from https://vuejs.org/guide/introduction.html