UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Ingeniería en Sistemas Computacionales

ESTUDIO COMPARATIVO DE LOS FRAMEWORKS PHP: SYMFONY 3 Y LARAVEL COMO BACKEND, PARA EL DESARROLLO DE UN SISTEMA DE GESTIÓN DE COBROS EN LA EMPRESA ANTENAWIFI S.A.S, IMPLEMENTADO CON ANGULAR 4 Y UN SERVIDOR LOCAL APACHE XAMPP.

Trabajo de grado previo a la obtención del título de Ingeniero en Sistemas Computacionales

Autor:

Darwin Arturo Túquerres Ipiales

Director:

MSc. Diego Javier Trejo España

Ibarra – Ecuador

2025

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1003228309		
APELLIDOS Y NOMBRES:	TUQUERRES IPIALES DARWIN ARTURO		
DIRECCIÓN:	CARANQUI, COMUNIDAD NARANJITO		
EMAIL:	datuquerresi@utn.edu.ec / darwin.a.99@gmail.com		
TELÉFONO FIJO:		**TELÉFONO MÓVIL:**	0992417570

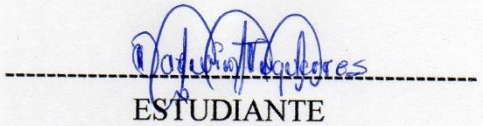
DATOS DE LA OBRA	
TÍTULO:	ESTUDIO COMPARATIVO DE LOS FRAMEWORKS PHP: SYMFONY 3 Y LARAVEL COMO BACKEND, PARA EL DESARROLLO DE UN SISTEMA DE GESTIÓN DE COBROS EN LA EMPRESA ANTENAWIFI S.A.S, IMPLEMENTADO CON ANGULAR 4 Y UN SERVIDOR LOCAL APACHE XAMPP.
AUTOR (ES):	TUQUERRES IPIALES DARWIN ARTURO
FECHA: DD/MM/AAAA	02 / 09 / 2025
PROGRAMA:	■ **PREGRADO** □ **POSGRADO**
TITULO POR EL QUE OPTA:	INGENIERO EN SISTEMAS COMPUTACIONALES
ASESOR /DIRECTOR:	MSC. DIEGO JAVIER TREJO ESPAÑA

2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 02 días del mes de septiembre del 2025

EL AUTOR:



ESTUDIANTE
Darwin Arturo Túquerres Ipiales
C.I 1003228309

CERTIFICACIÓN DIRECTOR

Ibarra 04 de agosto del 2025

CERTIFICACIÓN DIRECTOR DEL TRABAJO DE TITULACIÓN

Por medio del presente yo Ing. MSc. Diego Trejo, certifico que el Sr. Darwin Arturo Túquerres Ipiales portador de la cedula de ciudadanía número 1003228309, ha trabajado en el desarrollo del proyecto de grado **"ESTUDIO COMPARATIVO DE LOS FRAMEWORKS PHP: SYMFONY 3 Y LARAVEL COMO BACKEND, PARA EL DESARROLLO DE UN SISTEMA DE GESTIÓN DE COBROS EN LA EMPRESA ANTENAWIFI S.A.S, IMPLEMENTADO CON ANGULAR 4 Y UN SERVIDOR LOCAL APACHE XAMPP."**, previo a la obtención del Título de Ingeniero en Sistemas Computacionales realizado con interés profesional y responsabilidad que certifico con honor de verdad.

Es todo en cuanto puedo certificar a la verdad

Atentamente.

Ing. MSc. Diego Trejo

DIRECTOR DE TRABAJO DE GRADO

Ibarra, 01 de agosto del 2025

CERTIFICADO DE IMPLEMENTACIÓN

Mediante el presente certifico que el Sr. DARWIN ARTURO TÚQUERRES IPIALES portador de la cedula de identidad N° 100322830 – 9, estudiante de la Carrera de Ingeniería en Sistemas Computacionales de la Universidad Técnica del Norte, implemento el proyecto titulado: "**ESTUDIO COMPARATIVO DE LOS FRAMEWORKS PHP: SYMFONY 3 Y LARAVEL COMO BACKEND, PARA EL DESARROLLO DE UN SISTEMA DE GESTIÓN DE COBROS EN LA EMPRESA ANTENAWIFI S.A.S, IMPLEMENTADO CON ANGULAR 4 Y UN SERVIDOR LOCAL APACHE XAMPP**" como parte de la automatización del proceso.

Adicionalmente informo que el sistema ha sido desarrollado en su totalidad cumpliendo todos los requerimientos solicitados, de esta manera se recibe el proyecto como culminado e implementado con satisfacción.

El Sr. DARWIN ARTURO TÚQUERRES IPIALES, puede hacer uso de este documento para los fines pertinentes que lo amerite.

Atentamente.

@ntena RUC 1003685219001 tu punto de conexión

Sr. Juan Andrés Chávez Realpe

Gerente Empresa

ANTENAWIFI S.A.S

DEDICATORIA

Con gratitud y emoción, dedico este trabajo a todas las personas que, de una u otra manera, hicieron posible la culminación de esta etapa.

A mis padres, mi esposa y mis hijas, por su amor incondicional, su apoyo constante y su ejemplo de esfuerzo y perseverancia. Gracias por ser mi mayor inspiración y por enseñarme que el conocimiento es el mejor legado.

A mis docentes y asesores, por su guía, paciencia y enseñanzas que me han permitido crecer académica y personalmente. Sus consejos fueron fundamentales para la construcción de este trabajo.

A ti, mi estrellita en el cielo “Selene”. Gracias por tu amor inmenso, por enseñarme con tu vida lo que significa la fuerza, la bondad y la esperanza. Aunque tus pasos ya no caminen junto a los míos, siento tu presencia en cada logro, en cada momento difícil, y en cada decisión que me trajo hasta aquí. Esta tesis es también tuya, porque parte de mi camino lo recorrí con tu luz guiándome desde allá arriba.

Y, sobre todo, a mí mismo, por la dedicación, la disciplina y la pasión que me han permitido llegar hasta aquí. Este logro es el resultado de años de esfuerzo y sacrificio, y hoy lo celebro con orgullo.

Darwin Arturo Túquerres Ipiales

AGRADECIMIENTO

Al finalizar esta etapa tan significativa de mi vida, quiero expresar mi más sincero agradecimiento a todas las personas que hicieron posible la culminación de este trabajo.

En primer lugar, a Dios, por darme la fortaleza, la sabiduría y la perseverancia necesarias para superar cada desafío.

A mi familia, especialmente a mis padres, mi esposa y mis hijas, por su amor incondicional, su apoyo inagotable y por ser mi mayor fuente de inspiración. Gracias por su confianza en mí y por motivarme a seguir adelante aún en los momentos difíciles.

A mis profesores y asesores, por su invaluable guía, paciencia y conocimientos compartidos a lo largo de este proceso. Sus enseñanzas han sido fundamentales para mi formación y para la realización de este trabajo.

A todas aquellas personas que, de una u otra manera, contribuyeron a la realización de esta tesis, ya sea brindándome su tiempo, conocimientos o palabras de ánimo.

Este logro es el reflejo del esfuerzo, la dedicación y el apoyo recibido de muchas personas. A todos, gracias de corazón.

Darwin Arturo Túquerres Ipiales

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente documento se encuentra conformado por tres capítulos, en el cual se detalla todo el proceso para realizar el Trabajo de Grado: "ESTUDIO COMPARATIVO DE LOS FRAMEWORKS PHP: SYMFONY 3 Y LARAVEL COMO BACKEND, PARA EL DESARROLLO DE UN SISTEMA DE GESTIÓN DE COBROS EN LA EMPRESA ANTENAWIFI S.A.S, IMPLEMENTADO CON ANGULAR 4 Y UN SERVIDOR LOCAL APACHE XAMPP."

En el capítulo 1, se presenta todo el marco teórico, se establece el fundamento conceptual del estudio, abordando los principios fundamentales de los frameworks PHP y su evolución. Se examinan en profundidad Symfony 3 y Laravel, destacando su arquitectura, componentes clave, ventajas y limitaciones en el contexto del desarrollo backend. Además, se contextualiza el papel de estos frameworks en la construcción de sistemas de información web, detallando aspectos como el patrón Modelo-Vista-Controlador (MVC), el enfoque basado en microservicios, y la gestión de bases de datos.

En el capítulo 2, se detalla la planificación del proyecto de investigación, se describe el proceso práctico de implementación del proyecto dentro de la empresa ANTENAWIFI S.A.S. Se detalla la organización del equipo SCRUM, la definición del backlog del producto, y la planificación de los sprints. Cada sprint aborda el desarrollo progresivo del sistema: desde la comparación de los frameworks, configuración del entorno y modelado de la base de datos, hasta la implementación de los módulos funcionales.

En el capítulo 3, se detallan la parte del análisis de resultados y la interpretación de resultados obtenidos mediante el Modelo de Aceptación de Tecnología (TAM). Se explican sus fundamentos teóricos y su adaptación al contexto del estudio, seguido del diseño y validación del instrumento de recolección de datos aplicado a la población objetivo. Se presentan análisis descriptivos e inferenciales (correlación y regresión), los cuales evidencian el grado de aceptación y usabilidad percibida del sistema implementado.

Palabras clave: Frameworks PHP, Symfony 3, Laravel, Backend, Sistema de Gestión de Cobros, ANTENAWIFI S.A.S., SCRUM, Modelo TAM.

ABSTRACT

This document is made up of three chapters, in which the entire process to carry out the Degree Project is detailed: “ESTUDIO COMPARATIVO DE LOS FRAMEWORKS PHP: SYMFONY 3 Y LARAVEL COMO BACKEND, PARA EL DESARROLLO DE UN SISTEMA DE GESTIÓN DE COBROS EN LA EMPRESA ANTENAWIFI S.A.S, IMPLEMENTADO CON ANGULAR 4 Y UN SERVIDOR LOCAL APACHE XAMPP.”.

In Chapter 1, the entire theoretical framework is presented, establishing the conceptual foundation of the study by addressing the fundamental principles of PHP frameworks and their evolution. Symfony 3 and Laravel are examined in depth, highlighting their architecture, key components, advantages and limitations in the context of backend development. Additionally, the role of these frameworks in building web information systems is contextualized, detailing aspects such as the Model-View-Controller (MVC) pattern, the microservices-based approach, and database management.

In Chapter 2, the research project planning is detailed, describing the practical implementation process of the project within the company ANTENAWIFI S.A.S. The organization of the SCRUM team, the definition of the product backlog, and sprint planning are detailed. Each sprint addresses the progressive development of the system: from framework comparison, environment configuration and database modeling, to the implementation of functional modules.

In Chapter 3, the results analysis section and the interpretation of results obtained through the Technology Acceptance Model (TAM) are detailed. Its theoretical foundations and adaptation to the study context are explained, followed by the design and validation of the data collection instrument applied to the target population. Descriptive and inferential analyses (correlation and regression) are presented, which demonstrate the degree of acceptance and perceived usability of the implemented system.

Keywords: PHP Frameworks, Symfony 3, Laravel, Backend, Billing Management System, ANTENAWIFI S.A.S., SCRUM, TAM Model.

INTRODUCCIÓN

Tema

Estudio comparativo de los frameworks php: symfony 3 y laravel como backend, para el desarrollo de un sistema de gestión de cobros en la empresa ANTENAWIFI S.A.S, implementado con angular 4 y un servidor local apache XAMPP.

Problema

Antecedentes

En el ámbito del desarrollo de software, la evolución de los frameworks PHP ha sido notable durante la última década, impulsada por la necesidad creciente de crear aplicaciones web robustas, seguras y escalables. Entre estos frameworks, Symfony y Laravel han emergido como dos de las opciones más destacadas y populares dentro de la comunidad de desarrolladores. Cada uno de ellos ofrece un conjunto de características y herramientas que facilitan el desarrollo de aplicaciones complejas, promoviendo buenas prácticas y una estructura de código bien organizada (Parada, Zamora, & Trujillo, 2019).

Symfony, lanzado en 2005, ha sido reconocido por su flexibilidad, modularidad y su enfoque en la reutilización de componentes. Desde sus primeras versiones, Symfony ha sido adoptado por grandes empresas y proyectos debido a su capacidad para adaptarse a diversas necesidades y su compatibilidad con otras tecnologías. Symfony 3, en particular, introdujo mejoras significativas en cuanto a rendimiento, facilidad de uso y una mejor integración con el ecosistema PHP. El uso de bundles y componentes desacoplados permite a los desarrolladores construir aplicaciones altamente personalizables y escalables, lo cual es esencial para sistemas de gestión de cobros que requieren una gran precisión y confiabilidad (Uranga, Correoso, & Tomacen, 2021).

Por otro lado, Laravel, lanzado en 2011, ha ganado rápidamente popularidad gracias a su enfoque en la simplicidad y elegancia del código. Laravel se basa en algunos componentes de Symfony, pero ofrece una experiencia de desarrollo más amigable y accesible, especialmente para aquellos que son nuevos en PHP o en el desarrollo de frameworks en general. Laravel incluye herramientas poderosas como Eloquent ORM, Blade Template Engine y una estructura de comandos Artisan, que simplifican tareas comunes y aceleran el proceso de desarrollo. Estas características hacen de Laravel una opción atractiva para proyectos que requieren una rápida implementación sin sacrificar la calidad del código (Parada, Zamora, & Trujillo, 2019).

En el contexto de los sistemas de gestión de cobros, donde la precisión, la seguridad y la eficiencia son primordiales, la elección del framework de backend adecuado puede tener un impacto significativo en el éxito del proyecto. Los sistemas de gestión de cobros no solo deben manejar grandes volúmenes de datos y transacciones de manera eficiente, sino también garantizar la seguridad de la información financiera y personal de los usuarios. Symfony y Laravel, con sus respectivas fortalezas, ofrecen soluciones viables, pero cada uno presenta diferentes ventajas y desafíos que deben ser evaluados cuidadosamente.

Este estudio comparativo se realiza en un momento en el que la empresa busca continuamente mejorar sus sistemas de gestión para optimizar los procesos financieros y ofrecer mejores servicios a sus clientes. Al entender las diferencias y similitudes entre Symfony 3 y Laravel, este trabajo proporcionará una base sólida para tomar decisiones informadas sobre qué framework puede cumplir mejor con los requisitos específicos de un sistema de gestión de cobros.

Situación Actual

En el entorno actual del desarrollo de software, la elección de un framework adecuado es un factor determinante para el éxito de cualquier proyecto de desarrollo web. Con la creciente demanda de aplicaciones robustas, eficientes y seguras, especialmente en el ámbito financiero, los desarrolladores y las empresas deben evaluar cuidadosamente sus opciones para asegurarse de que están utilizando las herramientas más adecuadas para sus necesidades (González, Sanoja, & Rivas, 2020).

Los sistemas de gestión de cobros representan una categoría crítica de aplicaciones que requieren un manejo preciso y seguro de transacciones financieras. Estos sistemas no solo deben ser capaces de procesar grandes volúmenes de datos en tiempo real, sino que también deben cumplir con estrictos estándares de seguridad y regulación para proteger la información sensible de los usuarios (Rochina, Guashpa, & Coloma, 2022).

Actualmente, en la empresa ANTENAWIFI S.A.S, todos los procesos se realizan de manera manual utilizando el programa Excel para registrar y facturar los servicios de telecomunicaciones prestados a los clientes. Aunque este método ha sido funcional hasta ahora, la empresa reconoce la necesidad imperiosa de implementar una aplicación de software. Esta transición no solo mejorará la eficiencia y el rendimiento del personal, sino que también optimizará los tiempos de procesamiento y la precisión en la gestión de datos.

En este sentido, el desarrollo de un sistema para la gestión de cobros requiere una evaluación cuidadosa de las herramientas disponibles. Symfony 3 y Laravel, como dos de los frameworks PHP más destacados, presentan diferentes fortalezas y desafíos. Este estudio tiene como objetivo proporcionar una guía clara y detallada para ayudar a los desarrolladores a seleccionar el framework más adecuado que permita optimizar los procesos de desarrollo del sistema de cobros, garantizando la calidad y seguridad del producto implementado.

Prospectiva

La industria del software está en constante evolución, impulsada por la necesidad de desarrollar aplicaciones más eficientes, seguras y escalables. En este contexto, los frameworks PHP como Symfony 3 y Laravel desempeñan un papel trascendental, especialmente en el desarrollo de sistemas de gestión de cobros que demandan altos niveles de precisión y confiabilidad. Mirando hacia el futuro, se espera que ambos frameworks continúen evolucionando y adaptándose a las nuevas tecnologías y prácticas de desarrollo emergentes.

Tanto Symfony como Laravel han demostrado un compromiso sólido con la innovación y la mejora continua. Symfony, conocido por su estabilidad y arquitectura modular, probablemente seguirá refinando sus componentes y facilitando la integración con tecnologías emergentes, como microservicios y arquitecturas sin servidor, esto permitirá a los desarrolladores crear sistemas de gestión de cobros aún más robustos y escalables. La seguridad seguirá siendo una prioridad clave, especialmente en sistemas que manejan datos financieros sensibles. Se espera que Symfony y Laravel continúen fortaleciendo sus mecanismos de seguridad para proteger contra amenazas emergentes. Además, con la creciente regulación en torno a la protección de datos (como el GDPR en Europa), ambos frameworks probablemente introducirán características que faciliten el cumplimiento normativo, proporcionando herramientas para el manejo adecuado de datos y la generación de auditorías (Díaz, Acosta, Checa, & León, 2023).

La comunidad de desarrolladores que respalda tanto a Symfony como a Laravel es un factor crítico para su evolución. Se prevé que estas comunidades sigan creciendo y ofreciendo recursos valiosos, como paquetes, plugins y extensiones que amplíen las capacidades de los frameworks. La colaboración y el intercambio de conocimientos entre los desarrolladores seguirán impulsando la innovación y mejorando las prácticas de desarrollo.

En este sentido, el presente estudio no solo es relevante en el contexto actual, sino que también ofrece una visión a futuro de cómo estas herramientas pueden evolucionar para

enfrentar nuevos desafíos. La elección del framework adecuado dependerá de las necesidades específicas del proyecto, considerando que los dos analizados se encuentran bien posicionados para adaptarse a las tendencias emergentes y continuar proporcionando soluciones robustas y eficientes. Este análisis prospectivo subraya la importancia de mantenerse informado sobre las actualizaciones y mejoras en estos frameworks, garantizando así la implementación de sistemas de gestión de cobros que no solo satisfacen las necesidades presentes, sino que también están preparados para el futuro.

Planteamiento del problema

En la actualidad, la empresa ANTENAWIFI S.A.S. gestiona todos sus procesos de registro y facturación de servicios de telecomunicaciones utilizando hojas de cálculo en Excel. Aunque este método ha sido funcional hasta cierto punto, presenta múltiples limitaciones que afectan la eficiencia operativa y la precisión de los datos. La gestión manual en Excel no está diseñada para manejar grandes volúmenes de información de manera eficiente, lo que resulta en tiempos de procesamiento prolongados y un mayor riesgo de errores humanos. Esta situación se vuelve especialmente crítica a medida que la base de clientes de la empresa crece, aumentando la complejidad y la carga de trabajo administrativo.

Además, la seguridad de los datos es una preocupación importante. Las hojas de cálculo de Excel no proporcionan mecanismos robustos de protección de datos, lo que expone la información financiera y personal de los clientes a posibles brechas de seguridad. En un entorno donde la protección de datos es clave, la falta de medidas de seguridad adecuadas puede tener consecuencias graves tanto para la empresa como para sus clientes. La colaboración y el acceso simultáneo también se ven obstaculizados, ya que múltiples usuarios no pueden trabajar eficientemente en los mismos archivos sin el riesgo de sobrescribir datos, lo que ralentiza los procesos y reduce la productividad.

La falta de automatización es otra limitación significativa del uso de Excel. Muchas tareas, como la generación de facturas, el seguimiento de pagos y la actualización de registros de clientes, deben realizarse manualmente. Este enfoque manual consume tiempo y recursos valiosos, lo que podría ser mejor aprovechado en otras áreas estratégicas de la empresa. La ineficiencia operativa resultante no solo afecta la capacidad de respuesta de la empresa, sino que también puede impactar negativamente en la satisfacción del cliente.

Ante estas limitaciones, ANTENAWIFI S.A.S. se ve en la necesidad de implementar un sistema de gestión de cobros más eficiente y seguro. La transición a un sistema basado en un

framework de backend robusto puede resolver muchos de los problemas actuales, mejorando la precisión, la seguridad y la eficiencia operativa. Para tomar una decisión informada se propone realizar una comparación basada en varios criterios, incluyendo la escalabilidad, la seguridad, la facilidad de uso, la capacidad de automatización y la integración con otras tecnologías. A través de este análisis detallado, ANTENAWIFI S.A.S. podrá identificar la mejor solución tecnológica que satisfaga sus necesidades y supere las limitaciones actuales impuestas por el uso de Excel.

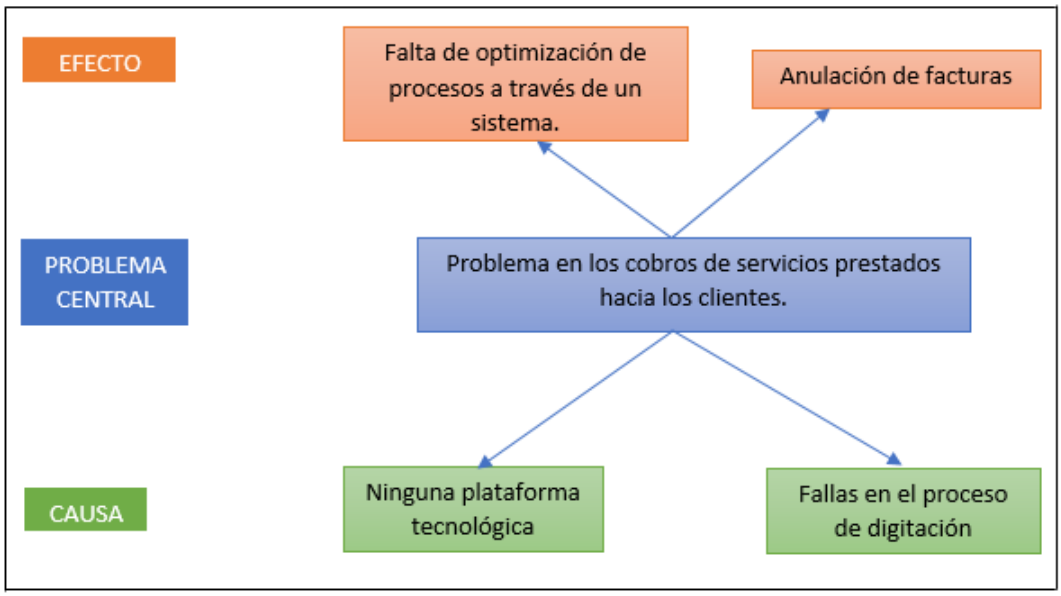


Figura 1 *Árbol de Problemas*

Objetivos

Objetivo General.

Gestionar un sistema practico de cobros para la empresa ANTENAWIFI S.A.S. que permita la optimización de procesos por medio de la tecnología PHP y angular 4 con lenguaje TypeScript.

Objetivos Específicos

- Investigar los procesos realizados por la empresa al momento de emitir cobros relacionados con los servicios ofrecidos por la empresa ANTENAWIFI S.A.S.
- Realizar un análisis comparativo de los Frameworks SYMFONY 3 y LARAVEL relacionados al desarrollo backend.

- Desarrollar un aplicativo web de gestión de cobros con el framework seleccionado por medio de la comparativa realizada, para la implementación en la empresa ANTENAWIFI S.A.S.

Alcance

El alcance de esta investigación se centra en realizar una comparación exhaustiva entre los frameworks PHP Symfony 3 y Laravel, evaluando su idoneidad para el desarrollo de un sistema de gestión de cobros. Este análisis se llevará a cabo mediante la evaluación de múltiples criterios técnicos y operativos, incluyendo la arquitectura del framework, la facilidad de uso, el rendimiento, la seguridad, y la capacidad de integración con otras tecnologías. Se buscará identificar las ventajas y desventajas de cada uno en contextos específicos, proporcionando una visión clara de cómo cada uno puede satisfacer las necesidades de ANTENAWIFI S.A.S.

La investigación también abarcará la implementación de la propuesta de un sistema de gestión de cobros utilizando frameworks, esto permitirá una evaluación práctica de las herramientas y funcionalidades ofrecidas por Symfony 3 y Laravel, así como de la experiencia de desarrollo en términos de tiempo y recursos requeridos. La comparación incluirá aspectos como la facilidad de configuración inicial, la estructura del código, la gestión de bases de datos, y la implementación de características clave como la facturación, el seguimiento de pagos, y la generación de informes.

Finalmente, al proporcionar una visión completa de las capacidades y limitaciones de Symfony 3 y Laravel, esta investigación ayudará a ANTENAWIFI S.A.S a tomar una decisión informada sobre la mejor tecnología para mejorar sus procesos de gestión de cobros y optimizar el rendimiento operativo de la empresa.

Metodología

Para investigar los procesos de emisión de cobros, se llevará a cabo una metodología de enfoque cuantitativo que incluye entrevistas semiestructuradas con el personal de facturación, administración y TI para obtener una comprensión detallada de las prácticas actuales. Además, se realizarán encuestas para recoger datos cuantitativos sobre la frecuencia y la eficiencia de estos procesos. La observación directa de las actividades diarias de facturación permitirá identificar desafíos operativos y áreas de mejora, complementada por un análisis documental de los registros históricos y hojas de cálculo de Excel utilizados en los procesos actuales.

El análisis comparativo de los frameworks Symfony 3 y Laravel se realizará mediante una revisión bibliográfica exhaustiva de la literatura técnica y académica sobre ambos

frameworks. Posteriormente, se desarrollará el prototipo básico de un registro de usuarios utilizando el framework analizado, implementando funcionalidades clave como el registro de clientes. Este prototipo será evaluado en términos de rendimiento, seguridad, facilidad de uso y escalabilidad. Finalmente, se llevarán a cabo pruebas de usuario y entrevistas con desarrolladores para recoger impresiones cualitativas sobre la experiencia de desarrollo con cada framework.

Una vez seleccionado el framework más adecuado a partir del análisis comparativo, se procederá al desarrollo del aplicativo web de gestión de cobros siguiendo una metodología ágil. El desarrollo se estructurará en sprints, cada uno con objetivos específicos y entregables definidos, incluyendo el diseño de la arquitectura del sistema, la implementación de funcionalidades básicas, pruebas de integración, y la iteración continua basada en el feedback de los usuarios. Se realizarán pruebas exhaustivas de funcionalidad, seguridad y rendimiento antes de la implementación final del sistema en la empresa ANTENAWIFI S.A.S., asegurando que el nuevo sistema mejore la eficiencia y precisión de los procesos de gestión de cobros.

Justificación

Justificación metodológica

La elección de la metodología en esta investigación se fundamenta en la necesidad de realizar un análisis exhaustivo y detallado que permita comprender las ventajas y limitaciones de los frameworks Symfony 3 y Laravel en el contexto del desarrollo de un sistema de gestión de cobros. Dado que la empresa ANTENAWIFI S.A.S. actualmente realiza estos procesos de manera manual utilizando Excel, es importante adoptar un enfoque metodológico que no solo evalúe las capacidades técnicas de cada framework, sino que también considere la experiencia práctica de implementación y uso en un entorno real. La combinación de técnicas cualitativas y cuantitativas permitirá obtener una visión integral de las herramientas y su adecuación a las necesidades específicas de la empresa.

La investigación inicial de los procesos actuales de ANTENAWIFI S.A.S. mediante encuestas y observación directa es esencial para establecer una línea base y entender los desafíos y limitaciones que enfrenta la empresa. Este enfoque permitirá identificar áreas críticas que necesitan ser abordadas por el nuevo sistema, garantizando que la solución propuesta no solo sea técnicamente viable, sino también alineada con las prácticas y requisitos operativos de la empresa. Además, el análisis documental proporcionará datos históricos que ayudarán a

evaluar el impacto potencial de la automatización y la digitalización de los procesos de gestión de cobros.

Estos enfoques son fundamentales para tomar una decisión informada sobre cuál framework es más adecuado para las necesidades específicas de ANTENAWIFI S.A.S. La metodología ágil adoptada para el desarrollo del sistema final garantiza que se pueda iterar rápidamente en base al feedback recibido, permitiendo ajustar y optimizar el sistema de manera continua. Esta flexibilidad es importante para asegurar que el sistema no solo cumpla con los requisitos técnicos, sino que también mejore significativamente la eficiencia y precisión de los procesos de gestión de cobros de la empresa.

Justificación tecnológica

La justificación tecnológica de esta investigación radica en la necesidad imperiosa de ANTENAWIFI S.A.S. de modernizar y optimizar sus procesos de gestión de cobros, que actualmente se realizan de manera manual mediante Excel. Este enfoque no solo es ineficiente y propenso a errores, también pone en riesgo la seguridad de los datos sensibles de los clientes. En este contexto, la elección de un framework PHP robusto y eficiente es clave para desarrollar un sistema backend que no solo automatice y acelere estos procesos, sino que también garantice la integridad, seguridad y escalabilidad del sistema. Symfony 3 y Laravel son dos de los frameworks PHP más reconocidos y utilizados en la industria, cada uno con características y beneficios específicos que pueden ser determinantes para las necesidades tecnológicas de la empresa.

Justificación educativa

La justificación educativa de esta investigación radica en su potencial para enriquecer el conocimiento y las habilidades de los desarrolladores y estudiantes en el campo del desarrollo web backend. Al comparar dos frameworks PHP prominentes como Symfony 3 y Laravel, este estudio proporcionará una comprensión profunda de sus arquitecturas, ventajas, limitaciones y aplicaciones prácticas. Esto no solo mejorará la capacidad de los profesionales para seleccionar y utilizar el framework más adecuado para distintos tipos de proyectos, sino que también fomentará el desarrollo de competencias críticas en análisis comparativo, toma de decisiones informadas e implementación de soluciones tecnológicas avanzadas. Además, los hallazgos y metodologías del estudio podrán integrarse en currículos educativos, talleres y programas de formación continua, beneficiando a una amplia comunidad de aprendices y profesionales en el ámbito de la tecnología.

CAPÍTULO 1

Marco Teórico

1.1. Introducción a los Frameworks PHP

Los frameworks PHP son herramientas esenciales que proporcionan una estructura sólida para el desarrollo de aplicaciones web, promoviendo la organización del código y facilitando la implementación de buenas prácticas. Al utilizarlos, los desarrolladores pueden reducir el tiempo de desarrollo, mejorar la mantenibilidad y aumentar la seguridad de sus aplicaciones. PHP, un lenguaje de programación ampliamente utilizado en el desarrollo web, ha visto la creación de varios frameworks robustos. Estos ofrecen diversas funcionalidades y enfoques, adaptándose a diferentes tipos de proyectos y necesidades, su evolución ha permitido a los desarrolladores crear aplicaciones más complejas y eficientes, aprovechando características como modularidad, la reutilización de componentes y las herramientas integradas para tareas comunes (Ovalle, 2022).

1.1.1. Definición y propósito de los frameworks

Un framework es una plataforma de software diseñada para facilitar el desarrollo de aplicaciones; en el contexto de PHP, proporciona una estructura básica sobre la cual los desarrolladores pueden construir sus aplicaciones web. Estos frameworks incluyen bibliotecas de código predefinido, patrones de diseño, y herramientas que ayudan a automatizar tareas comunes del desarrollo web, como la gestión de bases de datos, la creación de formularios, y el manejo de sesiones y autenticación (Castillo & Coronel, Frameworks PHP basados en la arquitectura Modelo-Vista-Controlador para desarrollo de aplicaciones web, 2023).

El propósito principal de los frameworks PHP es mejorar la eficiencia y la calidad del desarrollo de software. Al ofrecer una base estructurada y estandarizada, los frameworks permiten a los desarrolladores concentrarse en la lógica específica de la aplicación en lugar de reinventar la rueda con cada proyecto. Además, promueven la reutilización de código, la consistencia en el desarrollo y las buenas prácticas de programación. Los frameworks también suelen incluir herramientas integradas para pruebas y depuración, lo que contribuye a la creación de aplicaciones más robustas y libres de errores (Ortega, Guevara, & Benavides, 2021). En este sentido, los frameworks PHP no solo aceleran el proceso de desarrollo, sino que también mejoran la mantenibilidad y escalabilidad de las aplicaciones, proporcionando una infraestructura sólida y fiable para proyectos de cualquier tamaño.

1.1.2. Importancia de los frameworks PHP en el desarrollo web

Los frameworks PHP juegan un papel esencial en el desarrollo web moderno al proporcionar una base estructurada y consistente para la creación de aplicaciones robustas y escalables. Una de las principales ventajas de usar un framework PHP es la eficiencia. Los frameworks vienen con bibliotecas y componentes predefinidos que automatizan tareas comunes, lo que reduce significativamente el tiempo y el esfuerzo necesarios para desarrollar funcionalidades básicas y avanzadas. Esto permite a los desarrolladores centrarse en los aspectos únicos de sus aplicaciones, acelerando el ciclo de desarrollo y permitiendo la entrega más rápida de proyectos (Espinosa, 2021).

Además, los frameworks PHP promueven las buenas prácticas de programación al utilizar patrones de diseño establecidos, como el Modelo-Vista-Controlador (MVC) ayudan a mantener el código organizado, modular y fácil de mantener. Esto no solo mejora la mantenibilidad del código, sino que también facilita la colaboración entre múltiples desarrolladores, ya que el código sigue un formato y una estructura comunes. La consistencia y la organización del código son esenciales para proyectos a largo plazo, donde el mantenimiento y la ampliación de la funcionalidad son inevitables.

Otro aspecto crítico es la seguridad al incorporar múltiples medidas de seguridad integradas, como protección contra inyecciones SQL, Cross-Site Scripting (XSS), y Cross-Site Request Forgery (CSRF). Estas características ayudan a mitigar las vulnerabilidades comunes en las aplicaciones web, ofreciendo una capa adicional de protección y reduciendo el riesgo de ataques. Al utilizar un framework PHP, los desarrolladores pueden confiar en que las mejores prácticas de seguridad están integradas en la base de su aplicación, lo que es especialmente importante en el desarrollo de sistemas que manejan datos sensibles, como los sistemas de gestión de cobros (Esquivel, Martínez, Garduño, Moreno, & Ruíz, 2023).

Finalmente, los frameworks PHP cuentan con comunidades activas que contribuyen al crecimiento y la evolución continua de las herramientas. Las actualizaciones regulares y el soporte de la comunidad aseguran que los frameworks se mantengan relevantes y actualizados con las últimas tendencias y tecnologías en el desarrollo web. Esta colaboración comunitaria también significa que los desarrolladores tienen acceso a una amplia gama de recursos, desde documentación detallada hasta tutoriales y foros de ayuda, lo que facilita el aprendizaje y la resolución de problemas (Espinosa, 2021). En resumen, los frameworks PHP no solo simplifican y aceleran el desarrollo web, también mejoran la calidad, seguridad y sostenibilidad

de las aplicaciones, convirtiéndose en una herramienta indispensable para los desarrolladores web modernos.

1.1.3. Evolución de los frameworks PHP

La evolución de los frameworks PHP ha sido una respuesta directa a las necesidades crecientes de los desarrolladores web por herramientas más robustas, eficientes y seguras. Desde los primeros días del desarrollo web, donde el código PHP era escrito directamente en páginas HTML, hasta los modernos frameworks que automatizan y simplifican tareas complejas, ha habido un progreso significativo en cómo se abordan los proyectos web.

- **Primera Generación - Frameworks Iniciales:** En los primeros años de la década de 2000, surgieron los primeros frameworks PHP como CakePHP (2005) y CodeIgniter (2006). Estos frameworks introdujeron el concepto de patrones de diseño como MVC (Modelo-Vista-Controlador), que ayudaron a organizar el código de manera más estructurada. CakePHP, por ejemplo, se inspiró en el framework Ruby on Rails y ofreció una forma estandarizada de desarrollar aplicaciones web, lo que facilitó la adopción de buenas prácticas de programación. CodeIgniter, por su parte, se destacó por su simplicidad y pequeña huella de memoria, lo que lo hizo popular entre desarrolladores que buscaban una herramienta ligera y fácil de configurar.
- **Segunda Generación - Avances en Modularidad y Flexibilidad:** La siguiente fase en la evolución de los frameworks PHP se centró en mejorar la modularidad y la flexibilidad del desarrollo. Symfony (lanzado en 2005, con su versión 2 en 2011 y Symfony 3 en 2015) y Zend Framework (2006) fueron pioneros en este sentido. Symfony introdujo una arquitectura modular que permitió a los desarrolladores utilizar componentes individuales del framework de manera independiente. Esto fomentó la reutilización de código y facilitó la integración con otros proyectos. Zend Framework, por su parte, promovió una arquitectura orientada a objetos y un alto grado de personalización, adecuado para aplicaciones empresariales de gran escala.
- **Tercera Generación - Foco en la Experiencia del Desarrollador:** Con la llegada de Laravel en 2011, la evolución de los frameworks PHP tomó una dirección que priorizaba la experiencia del desarrollador. Laravel combinó las mejores ideas de los frameworks existentes y agregó su propia capa de

simplicidad y elegancia. Con características como Eloquent ORM, Blade Template Engine y una interfaz de línea de comandos intuitiva (Artisan), Laravel hizo que el desarrollo web fuera más accesible y rápido. La comunidad de Laravel creció rápidamente, impulsada por una documentación clara y una fuerte cultura de compartir recursos y paquetes adicionales.

- **Tendencias Actuales y Futuras:** En los años recientes, los frameworks PHP continúan evolucionando para adaptarse a nuevas tecnologías y metodologías de desarrollo. La integración con microservicios, el soporte para arquitecturas sin servidor (serverless) y la adopción de estándares modernos como PSR (PHP Standards Recommendations) son algunas de las tendencias que están moldeando el futuro de los frameworks PHP. Symfony y Laravel, en particular, han seguido innovando, con nuevas versiones que mejoran el rendimiento, la seguridad y la facilidad de uso.

La evolución de los frameworks PHP refleja un continuo esfuerzo por equilibrar la simplicidad y la potencia, proporcionando herramientas que no solo faciliten el desarrollo rápido, sino que también aseguren la calidad y sostenibilidad de las aplicaciones a largo plazo. Esta evolución ha permitido a los desarrolladores abordar proyectos cada vez más complejos con mayor confianza y eficiencia, cimentando el papel fundamental de los frameworks PHP en el ecosistema del desarrollo web.

1.2. Framework Symfony

1.2.1. Historia y evolución

Symfony es un framework PHP de código abierto que fue lanzado por primera vez en octubre de 2005 por SensioLabs, una empresa francesa fundada por Fabien Potencier, desde sus inicios, Symfony fue diseñado con el objetivo de proporcionar a los desarrolladores una herramienta robusta y flexible para construir aplicaciones web complejas y escalables (Symfony, 2024). El enfoque de Symfony en la modularidad y la reutilización de componentes marcó una diferencia significativa con respecto a otros frameworks de la época.

Symfony 1.x: Los Primeros Años

La primera versión estable, Symfony 1.0, se lanzó en enero de 2007. Esta versión introdujo muchos de los conceptos fundamentales que definirían al framework, como el patrón de diseño MVC (Modelo-Vista-Controlador), la arquitectura modular y un fuerte enfoque en la

configuración y personalización. Symfony 1.x ganó rápidamente popularidad entre los desarrolladores y las empresas que necesitaban construir aplicaciones web complejas.

Symfony 2: Una Reinvención

En julio de 2011, Symfony 2 fue lanzado, representando una reescritura completa del framework. Symfony 2 fue diseñado para ser más flexible, más eficiente y más fácil de integrar con otros componentes y bibliotecas. Algunas de las mejoras clave incluyeron:

- **Componentes Reutilizables:** Symfony 2 introdujo la idea de componentes desacoplados que podían ser usados independientemente del framework completo. Esto permitió a los desarrolladores utilizar solo las partes de Symfony que necesitaban en sus proyectos.
- **Dependency Injection:** El uso de un contenedor de inyección de dependencias permitió una gestión más flexible y eficiente de los servicios y dependencias en la aplicación.
- **Bundles:** La arquitectura de bundles en Symfony 2 facilitó la organización y reutilización de código, permitiendo a los desarrolladores empaquetar y distribuir funcionalidades de manera modular.

Symfony 3: Refinamiento y Mejoras

Symfony 3 fue lanzado en noviembre de 2015. Aunque no fue una reescritura completa como Symfony 2, esta versión se centró en refinar y mejorar las funcionalidades existentes. Symfony 3 introdujo varias mejoras en términos de rendimiento, usabilidad y simplicidad. Algunas de las características destacadas incluyeron:

- **Mejoras en el Desempeño:** Optimización de componentes y reducción de la carga de memoria para aplicaciones más rápidas y eficientes.
- **Facilidad de Uso:** Simplificación de configuraciones y mejoras en la documentación para facilitar el aprendizaje y uso del framework.
- **Flexibilidad y Personalización:** Continuación del enfoque en la modularidad y la capacidad de personalizar cada aspecto de la aplicación.

Symfony 4 y Más Allá: Modernización Continua

Lanzado en noviembre de 2017, Symfony 4 marcó un nuevo capítulo en la evolución del framework con un enfoque aún más fuerte en la simplicidad y flexibilidad. Symfony 4

introdujo Symfony Flex, una herramienta que simplifica la instalación y configuración de paquetes, haciendo que el proceso de desarrollo sea más rápido y sencillo. Symfony 4 también adoptó un enfoque más moderno, promoviendo la construcción de aplicaciones minimalistas y modulares que pueden ser ampliadas según las necesidades del proyecto.

A lo largo de su historia, Symfony ha evolucionado constantemente para adaptarse a las necesidades cambiantes de los desarrolladores y las tecnologías emergentes. Desde sus inicios, con un enfoque en la configuración y la modularidad, hasta las versiones más recientes que promueven la simplicidad y la flexibilidad, Symfony ha demostrado ser un framework robusto y versátil. La comunidad activa y el soporte continuo de SensioLabs han sido fundamentales para su evolución, haciendo de Symfony una opción popular y confiable para el desarrollo de aplicaciones web complejas y escalables.

1.2.2. Arquitectura y componentes clave

Symfony es un framework PHP diseñado con una arquitectura modular y flexible, que permite a los desarrolladores construir aplicaciones web complejas y escalables de manera eficiente. La arquitectura de Symfony sigue el patrón de diseño MVC (Modelo-Vista-Controlador), que separa la lógica de la aplicación en tres componentes principales: el modelo, la vista y el controlador. Esta separación facilita la gestión del código y promueve las buenas prácticas de desarrollo (Symfony, 2024).

Symfony se conforma de numerosos componentes independientes que pueden ser utilizados de manera modular. Para Martínez (2020) algunos de los componentes clave que forman la base del framework son:

- **HttpFoundation:** Este componente proporciona una abstracción sobre las solicitudes y respuestas HTTP, permitiendo una gestión más sencilla y flexible de las interacciones web. Incluye clases para manejar sesiones, cookies y archivos.
- **Routing:** El componente de enrutamiento de Symfony permite definir rutas que asocian URLs específicas con controladores concretos. Esto facilita la gestión de la navegación en la aplicación y soporta rutas dinámicas y complejas.
- **DependencyInjection:** Este componente implementa un contenedor de inyección de dependencias, que permite gestionar y configurar los servicios de la aplicación de manera eficiente. Facilita la creación de servicios reutilizables y la configuración centralizada.

- **Twig:** Es el motor de plantillas de Symfony, diseñado para ser rápido, seguro y flexible. Permite a los desarrolladores crear vistas reutilizables y mantener una separación clara entre la lógica de presentación y la lógica de negocio.
- **Form:** El componente de formularios de Symfony proporciona herramientas para crear, procesar y validar formularios HTML. Simplifica la gestión de formularios complejos y asegura una validación robusta de los datos de entrada del usuario.
- **Security:** El componente de seguridad de Symfony ofrece una amplia gama de funcionalidades para proteger la aplicación, incluyendo autenticación, autorización y gestión de roles y permisos. También incluye herramientas para prevenir ataques comunes, como CSRF y XSS.
- **Console:** Symfony Console es una herramienta de línea de comandos que permite crear comandos personalizados para automatizar tareas recurrentes. Es útil para tareas como la gestión de bases de datos, la ejecución de pruebas y la implementación de despliegues.
- **EventDispatcher:** Este componente permite implementar un sistema de eventos y escuchas, facilitando la extensión y personalización del comportamiento de la aplicación. Los eventos permiten desacoplar los componentes de la aplicación y reaccionar a acciones específicas de manera flexible.
- **Doctrine ORM:** Aunque no es un componente nativo de Symfony, Doctrine ORM es el mapeador objeto-relacional más utilizado con Symfony. Permite a los desarrolladores trabajar con bases de datos utilizando objetos PHP, facilitando la persistencia de datos y la gestión de relaciones complejas.

La arquitectura modular y los componentes clave de Symfony proporcionan una base sólida y flexible para el desarrollo de aplicaciones web. Al seguir el patrón MVC y ofrecer una amplia gama de componentes reutilizables, se facilita la creación de aplicaciones bien estructuradas, escalables y mantenibles. Esta combinación de arquitectura robusta y componentes integrados hace de este framework una elección popular entre los desarrolladores que buscan construir aplicaciones complejas y de alto rendimiento.

1.3. Framework Laravel

1.3.1. Historia y evolución

Es un framework PHP de código abierto diseñado para el desarrollo de aplicaciones web, lanzado por Taylor Otwell en junio de 2011. Laravel fue creado con el objetivo de ofrecer una alternativa más avanzada y elegante a los frameworks PHP existentes, como CodeIgniter, que, aunque populares, carecían de ciertas funcionalidades modernas y flexibles. Laravel fue concebido para mejorar la productividad del desarrollador mediante una sintaxis expresiva y herramientas integradas que simplifican tareas comunes (Laravel, 2024).

Laravel 1: El Comienzo

La primera versión de Laravel (Laravel 1) se lanzó en junio de 2011. Esta versión inicial incluía funcionalidades básicas como el enrutamiento, la autenticación y el manejo de sesiones. Laravel 1 se destacó por su facilidad de uso y su sintaxis limpia, aunque carecía de algunas características avanzadas que llegarían en versiones posteriores.

Laravel 2: Primeras Mejoras

En septiembre de 2011, Laravel 2 fue lanzado, trayendo consigo una serie de mejoras significativas. Esta versión introdujo soporte para controladores, lo que permitió a los desarrolladores organizar mejor su código siguiendo el patrón MVC (Modelo-Vista-Controlador). También se añadieron características como la integración con un ORM (Object-Relational Mapping) llamado Eloquent, lo que facilitó la interacción con bases de datos.

Laravel 3: Expansión de Funcionalidades

Laravel 3, lanzado en febrero de 2012, marcó un avance significativo en la evolución del framework. Se introdujeron varias características nuevas que expandieron enormemente la funcionalidad de Laravel, incluyendo:

- **Artisan CLI:** Una interfaz de línea de comandos que proporciona una serie de comandos útiles para el desarrollo y la administración de aplicaciones.
- **Soporte para Migraciones:** Una herramienta para gestionar cambios en la base de datos de manera controlada y reproducible.
- **Bundles:** Predecesores de los paquetes modernos, permitían la modularización del código y la reutilización de componentes.

Laravel 4: Una Reescritura Completa

En mayo de 2013, Laravel 4 fue lanzado, representando una reescritura completa del framework. Basado en componentes de Symfony, Laravel 4 introdujo un enfoque más modular y flexible, con mejoras en el rendimiento y la capacidad de extensión. Las características destacadas de Laravel 4 incluyen:

- **Composer:** Integración con Composer, un gestor de dependencias PHP, lo que facilitó la gestión de bibliotecas y paquetes externos.
- **Facades:** Una sintaxis simplificada para acceder a componentes y servicios del framework.
- **Queues:** Soporte para colas de trabajo, permitiendo el manejo asíncrono de tareas.

Laravel 5: Modernización y Mejora de la Experiencia del Desarrollador

Laravel 5, lanzado en febrero de 2015, trajo una serie de mejoras y nuevas características diseñadas para modernizar el desarrollo web y mejorar la experiencia del desarrollador. Algunas de las mejoras clave incluyen:

- **Middleware:** Un sistema para filtrar y manipular solicitudes HTTP antes de que lleguen a los controladores.
- **Elixir:** Herramienta para la gestión de activos front-end (más tarde reemplazada por Laravel Mix).
- **Job Queues y Eventos:** Mejoras en la gestión de colas y eventos para tareas asíncronas.
- **Autenticación Simplificada:** Mejoras en el sistema de autenticación, facilitando la implementación de inicios de sesión y registros.

Laravel 6 y Más Allá: LTS y Modernización Continua

En septiembre de 2019, Laravel 6 fue lanzado como la primera versión de soporte a largo plazo (LTS) del framework, introduciendo mejoras en la semántica de versiones y el soporte extendido. Laravel 6 continuó la evolución con características como:

- **Laravel Vapor:** Una plataforma sin servidor (serverless) para desplegar aplicaciones Laravel en AWS.
- **Lazy Collections:** Soporte para procesamiento de grandes datasets de manera eficiente en memoria.

- **Mejoras en Job Middleware:** Mejor gestión y flexibilidad en las tareas encoladas.

Laravel 7 y 8: Últimas Innovaciones

Laravel 7, lanzado en marzo de 2020, y Laravel 8, lanzado en septiembre de 2020, han seguido innovando con características modernas como:

- **Blade Components:** Mejora en la reutilización y organización de componentes de interfaz de usuario.
- **Laravel Sanctum:** Herramienta ligera para la autenticación de API.
- **Job Batching:** Mejoras en la ejecución de trabajos en lotes.

A lo largo de su evolución, Laravel ha mantenido su compromiso con la simplicidad, la elegancia y la productividad del desarrollador. Desde sus humildes comienzos hasta convertirse en uno de los frameworks PHP más populares y utilizados en el mundo, Laravel ha demostrado ser una herramienta poderosa y flexible para el desarrollo web moderno. Su continua innovación y adaptación a las nuevas tendencias y tecnologías aseguran que seguirá siendo una opción preferida por desarrolladores y empresas para construir aplicaciones web robustas y escalables.

1.3.2. Arquitectura y componentes clave

Laravel es un framework PHP moderno diseñado para proporcionar una base robusta y flexible para el desarrollo de aplicaciones web. Siguiendo el patrón de diseño MVC (Modelo-Vista-Controlador), Laravel facilita la organización y mantenimiento del código, permitiendo a los desarrolladores crear aplicaciones escalables y de alto rendimiento. A continuación, se describen la arquitectura y los componentes clave de Laravel que lo hacen destacar en el ecosistema de frameworks PHP (Laravel, 2024).

Laravel se compone de varios componentes integrados que proporcionan funcionalidades avanzadas y simplifican el desarrollo de aplicaciones web. Para Esquivel et al (2023) los componentes más importantes son:

- **Eloquent ORM:** Eloquent es el sistema ORM de Laravel que permite a los desarrolladores interactuar con la base de datos utilizando objetos PHP en lugar de consultas SQL. Eloquent facilita la definición de relaciones entre modelos, así como la realización de consultas complejas de manera intuitiva.

- **Blade Template Engine:** Blade es el motor de plantillas de Laravel, diseñado para ser ligero y potente. Blade permite el uso de plantillas reutilizables y ofrece una sintaxis sencilla para incluir lógica en las vistas, como bucles y condicionales, sin afectar el rendimiento de la aplicación.
- **Artisan CLI:** Artisan es la interfaz de línea de comandos de Laravel que proporciona una amplia gama de comandos útiles para el desarrollo y la administración de aplicaciones. Artisan permite automatizar tareas comunes como la generación de código, la migración de bases de datos y la ejecución de pruebas.
- **Middleware:** Laravel utiliza middleware para filtrar y procesar las solicitudes HTTP antes de que lleguen a los controladores. Los middleware permiten implementar funcionalidades transversales como la autenticación, la verificación de permisos y la protección contra ataques CSRF de manera sencilla y centralizada.
- **Queues:** Laravel ofrece soporte para colas de trabajo, lo que permite manejar tareas asíncronas de manera eficiente. Las colas son útiles para procesar tareas de larga duración, como el envío de correos electrónicos, la generación de informes y la sincronización de datos, sin bloquear la ejecución de la aplicación principal.
- **Events and Listeners**: El sistema de eventos de Laravel permite desacoplar componentes de la aplicación mediante la emisión de eventos y la suscripción de listeners para manejar esos eventos. Esta funcionalidad facilita la implementación de funcionalidades reactivas y la extensión del comportamiento de la aplicación.
- **Routing:** El componente de enrutamiento de Laravel es altamente flexible y permite definir rutas que responden a solicitudes HTTP específicas. Las rutas pueden ser configuradas para dirigir a controladores, funciones de cierre (closures) o cualquier otra lógica de manejo de solicitudes.
- **Validation**: Laravel incluye un sistema de validación robusto que permite definir reglas para validar datos de entrada de manera sencilla y eficiente. Las reglas de validación pueden ser aplicadas en controladores, modelos y

formularios, asegurando que los datos sean consistentes y seguros antes de ser procesados.

- **Authentication and Authorization:** Laravel facilita la implementación de autenticación y autorización mediante sistemas integrados para la gestión de usuarios, roles y permisos. Laravel Sanctum y Laravel Passport son herramientas específicas para la autenticación de APIs y la gestión de tokens de acceso.

La arquitectura modular y los componentes clave de Laravel proporcionan una base sólida y flexible para el desarrollo de aplicaciones web modernas. Siguiendo el patrón MVC y ofreciendo herramientas integradas facilitan la creación de aplicaciones bien estructuradas, escalables y mantenibles. La combinación de estas características hace de Laravel una opción popular y poderosa para los desarrolladores que buscan construir aplicaciones web robustas y eficientes.

1.4. Arquitectura de los sistemas de información web

La arquitectura de los sistemas de información web se refiere a la estructura y diseño de los componentes que componen una aplicación web, incluyendo la organización de datos, las interacciones entre diferentes partes del sistema y los flujos de información. Una arquitectura bien diseñada es clave para garantizar que el sistema sea eficiente, escalable, mantenible y seguro. Los componentes principales de la arquitectura web incluyen el cliente (frontend), el servidor (backend), la base de datos y el middleware (Lajpop, Ixcolin, & Ortíz, 2024).

El cliente, o frontend, es la parte visible de la aplicación con la que interactúan los usuarios. Se desarrolla utilizando tecnologías como HTML, CSS y JavaScript, y frameworks como React, Angular y Vue.js. El frontend es responsable de la presentación de los datos y la captura de las interacciones del usuario, manejando la lógica de presentación y algunas validaciones de datos antes de enviarlos al servidor. Esta capa es esencial para garantizar una experiencia de usuario atractiva y funcional.

El servidor, o backend, maneja las solicitudes del usuario y procesa la lógica del negocio. Incluye un servidor web, como Apache o Nginx, que maneja las solicitudes HTTP y las envía al servidor de aplicaciones. El servidor de aplicaciones, utilizando frameworks como Laravel, Symfony, Django o Node.js, ejecuta el código del lado del servidor y genera respuestas que se envían de vuelta al cliente. El backend también gestiona la autenticación, la autorización

y la comunicación con la base de datos, asegurando que los datos sean procesados y entregados de manera segura y eficiente.

La base de datos es el componente que almacena y gestiona los datos de la aplicación, puede ser una base de datos relacional, como MySQL o PostgreSQL, que utiliza SQL para gestionar los datos estructurados, o una base de datos NoSQL, como MongoDB o Cassandra, que ofrece una estructura flexible para grandes volúmenes de datos no estructurados. La base de datos asegura la integridad, disponibilidad y consistencia de los datos, siendo fundamental para el correcto funcionamiento de la aplicación.

El middleware actúa como un intermediario que permite la comunicación entre diferentes partes del sistema. Incluye APIs y servicios web para la comunicación entre el frontend y el backend, así como sistemas de mensajería y colas como RabbitMQ y Apache Kafka para manejar la comunicación asíncrona. El middleware asegura una comunicación fluida y eficiente, permitiendo que los diferentes componentes del sistema trabajen juntos de manera cohesiva.

La arquitectura de los sistemas de información web sigue comúnmente un patrón en capas, separando la presentación, la lógica de negocio y la gestión de datos. Este enfoque promueve la separación de preocupaciones, facilitando la mantenibilidad y escalabilidad del sistema. Además, se utilizan patrones arquitectónicos como MVC (Modelo-Vista-Controlador), microservicios y arquitectura sin servidor para mejorar la flexibilidad y adaptabilidad del sistema a las necesidades cambiantes (Ferrer, 2021).

En este sentido, la arquitectura de los sistemas de información web es esencial para diseñar aplicaciones eficientes, seguras y escalables. Comprender los diferentes componentes y su interacción permite a los desarrolladores crear sistemas robustos y adaptables, capaces de satisfacer las demandas de los usuarios y las organizaciones.

1.4.1. Modelo Vista Controlador (MVC)

El Modelo Vista Controlador (MVC) es un patrón de diseño ampliamente utilizado en el desarrollo de aplicaciones de software, particularmente en el ámbito del desarrollo web. Este patrón se basa en la separación de la aplicación en tres componentes interconectados pero independientes: el Modelo, la Vista y el Controlador. Esta separación de responsabilidades facilita la organización del código, mejora la mantenibilidad y promueve las buenas prácticas de desarrollo.

El Modelo es la parte de la aplicación que gestiona la lógica de negocio y los datos, se encarga de interactuar con la base de datos, realizar cálculos y procesar la lógica empresarial; en esencia, el Modelo representa los datos y las reglas de negocio de la aplicación. Al separar la lógica de negocio del resto de la aplicación, el Modelo facilita la reutilización del código y la independencia de la lógica de negocio respecto a la interfaz de usuario y el flujo de control (Castillo & Coronel, 2023).

La Vista es el componente responsable de la presentación de los datos al usuario. Consiste en la interfaz de usuario y cualquier componente visual que el usuario ve e interactúa. La Vista recibe los datos del Modelo y los presenta de una manera que sea comprensible y accesible para el usuario final. La separación de la Vista permite que la interfaz de usuario se modifique o se actualice sin afectar la lógica de negocio subyacente, promoviendo una mejor experiencia de usuario y una mayor flexibilidad en el diseño de la interfaz (Acosta, 2022).

El Controlador actúa como intermediario entre el Modelo y la Vista. Recibe las entradas del usuario (como clics, datos de formularios, etc.), procesa estas entradas y las traduce en acciones que el Modelo o la Vista deben realizar. El Controlador manipula el Modelo para actualizar los datos y selecciona la Vista apropiada para presentar los resultados al usuario. Esta intermediación garantiza que las interacciones del usuario se manejen de manera adecuada, permitiendo una respuesta dinámica y coherente de la aplicación (Castillo & Coronel, 2023).

La interacción entre el Modelo, la Vista y el Controlador en el patrón MVC sigue una serie de pasos coordinados:

- **Entrada del Usuario:** El usuario interactúa con la aplicación a través de la Vista, proporcionando datos o realizando acciones.
- **Procesamiento por el Controlador:** El Controlador recibe la entrada del usuario, interpreta la acción y la procesa.
- **Actualización del Modelo:** Basado en la entrada, el Controlador interactúa con el Modelo para actualizar los datos o realizar cálculos.
- **Actualización de la Vista:** Una vez que el Modelo se ha actualizado, el Controlador selecciona la Vista adecuada para presentar la nueva información al usuario.

La adopción del patrón MVC en el desarrollo de aplicaciones ofrece varios beneficios:

- **Separación de Responsabilidades:** La separación clara entre la lógica de negocio, la interfaz de usuario y el control de flujo facilita la organización del código y su mantenimiento.
- **Reutilización de Código**: Al aislar la lógica de negocio en el Modelo, es posible reutilizar este componente en diferentes partes de la aplicación o incluso en diferentes aplicaciones.
- **Facilidad de Mantenimiento**: La estructura modular del patrón MVC facilita la localización y corrección de errores, así como la implementación de nuevas funcionalidades sin afectar otras partes de la aplicación.
- **Flexibilidad en la Interfaz de Usuario**: Las Vistas pueden modificarse o reemplazarse sin afectar la lógica de negocio, permitiendo una mayor flexibilidad en el diseño y actualización de la interfaz de usuario.

En este sentido, el patrón MVC proporciona una estructura robusta y flexible para el desarrollo de aplicaciones, promoviendo la separación de responsabilidades, la reutilización del código y la facilidad de mantenimiento, lo que resulta en aplicaciones más organizadas y escalables.

1.4.2. Microservicios

El patrón de microservicios es una arquitectura para el desarrollo de software que descompone una aplicación monolítica en una colección de servicios pequeños, independientes y desplegables de manera autónoma. Cada microservicio es responsable de una funcionalidad específica del negocio y opera como una aplicación completa en sí misma, con su propia lógica de negocio, base de datos y mecanismos de comunicación. Este enfoque modular facilita el desarrollo, mantenimiento y escalabilidad de las aplicaciones, permitiendo a los equipos de desarrollo trabajar en diferentes servicios de manera simultánea y sin interferencias (Villanueva, 2023).

Una de las características más distintivas del patrón de microservicios es la independencia del despliegue. Cada microservicio puede ser desarrollado, testeado, desplegado y escalado de manera independiente, lo que permite una mayor agilidad en la entrega de nuevas funcionalidades y en la corrección de errores. Esta independencia también significa que los microservicios pueden utilizar diferentes tecnologías y lenguajes de programación, seleccionando las herramientas más adecuadas para resolver problemas específicos. Este

enfoque tecnológico heterogéneo facilita la innovación y la adaptación rápida a nuevas tendencias y tecnologías (Riascos, 2020).

La comunicación entre microservicios se realiza a través de interfaces bien definidas, generalmente utilizando APIs REST, mensajería asíncrona o protocolos como gRPC. Este tipo de comunicación desacoplada mejora la interoperabilidad y permite a los microservicios interactuar de manera eficiente, aunque también introduce complejidades en términos de latencia, consistencia de datos y manejo de fallos. Diseñar y gestionar estas interfaces requiere una cuidadosa planificación para asegurar que los servicios puedan cooperar sin problemas y responder adecuadamente a las solicitudes (Villanueva, 2023).

Aunque el patrón de microservicios ofrece muchos beneficios, también presenta desafíos significativos. La gestión de datos distribuidos es uno de los principales retos, ya que cada microservicio puede tener su propia base de datos, lo que complica la consistencia de los datos y las transacciones distribuidas. Además, el monitoreo y la depuración de una aplicación basada en microservicios son más complejos debido a la cantidad de servicios y sus interacciones. Se requieren herramientas avanzadas de monitoreo, logging y tracing para gestionar eficazmente la observabilidad y resolver problemas en un entorno distribuido.

En este sentido, el patrón de microservicios proporciona una arquitectura flexible y escalable que es ideal para aplicaciones complejas y de gran escala. Su enfoque modular y autónomo permite una mejor organización del código, facilita el despliegue continuo y mejora la capacidad de respuesta a los cambios. Sin embargo, también requiere una gestión cuidadosa de la comunicación entre servicios, la consistencia de los datos y la observabilidad del sistema. Con la infraestructura y las prácticas adecuadas, los microservicios pueden ofrecer una base sólida para el desarrollo de aplicaciones modernas, resilientes y de alto rendimiento.

1.5. Bases de datos para los sistemas de información web

Las bases de datos para los sistemas de información web son componentes esenciales que permiten almacenar, gestionar y recuperar grandes volúmenes de datos de manera eficiente y segura. En el contexto de las aplicaciones web, estas bases de datos soportan operaciones críticas, como el registro de usuarios, el manejo de transacciones y la gestión de contenidos. Su diseño y configuración son fundamentales para garantizar un rendimiento óptimo, la integridad de los datos y la capacidad de escalar con el crecimiento de la aplicación (Aguirre, 2021).

Las bases de datos relacionales (RDBMS) son una de las opciones más tradicionales y ampliamente utilizadas en sistemas de información web. Utilizan un modelo basado en tablas

para organizar los datos y las relaciones entre ellos, y emplean SQL (Structured Query Language) para la gestión y consulta de la información. Ejemplos populares de RDBMS incluyen MySQL, PostgreSQL, Microsoft SQL Server y Oracle. Las bases de datos relacionales son conocidas por su robustez en el manejo de transacciones, garantizando la integridad y consistencia de los datos a través de ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad). Además, su capacidad para normalizar datos minimiza la redundancia y asegura la integridad referencial. Sin embargo, pueden enfrentar desafíos de escalabilidad horizontal cuando se manejan grandes volúmenes de datos o se necesita un rendimiento muy alto (León, 2020).

Por otro lado, las bases de datos NoSQL han ganado popularidad en los últimos años debido a su flexibilidad y capacidad para manejar grandes cantidades de datos no estructurados o semi-estructurados. Estas bases de datos están diseñadas para ser altamente escalables y permiten el almacenamiento de datos en formatos como documentos, grafos, columnas o pares clave-valor. Ejemplos de bases de datos NoSQL incluyen MongoDB, Cassandra, Redis y Couchbase. Las bases de datos NoSQL son especialmente útiles para aplicaciones que requieren alta disponibilidad y rendimiento, así como para aquellas que manejan datos con estructuras variables. Su capacidad para escalar horizontalmente permite añadir nodos adicionales para gestionar el aumento de la carga y el volumen de datos. Sin embargo, suelen sacrificar algunas características de las RDBMS, como la consistencia estricta, en favor de la disponibilidad y la partición (Aguirre, 2021).

Además, las bases de datos en la nube se han convertido en una opción cada vez más atractiva para los sistemas de información web. Servicios como Amazon RDS, Google Cloud SQL y Azure SQL Database ofrecen bases de datos gestionadas que permiten a las organizaciones escalar sus aplicaciones sin preocuparse por la infraestructura subyacente. Estos servicios proporcionan beneficios como la alta disponibilidad, el respaldo automático, la recuperación ante desastres y la escalabilidad flexible, permitiendo a los desarrolladores centrarse en la construcción de aplicaciones en lugar de en la gestión de bases de datos (Cristancho & Lozano, 2020).

En este sentido, la elección de la base de datos adecuada para un sistema de información web depende de varios factores, incluyendo la naturaleza de los datos, los requisitos de escalabilidad, el rendimiento deseado y las necesidades de consistencia y disponibilidad. Las bases de datos relacionales ofrecen robustez y consistencia, mientras que las bases de datos NoSQL proporcionan flexibilidad y escalabilidad. Las bases de datos en la nube combinan lo

mejor de ambos mundos, ofreciendo soluciones gestionadas que facilitan el crecimiento y la gestión eficiente de las aplicaciones web.

1.6. Sistemas de Gestión de Cobros

Los sistemas de gestión de cobros son herramientas críticas para las organizaciones que necesitan gestionar eficazmente el proceso de facturación y recaudación de pagos de sus clientes. Estos sistemas automatizan y optimizan diversas tareas relacionadas con la emisión de facturas, el seguimiento de pagos, la gestión de cuentas por cobrar y la reconciliación de transacciones financieras. Al centralizar y estandarizar estas actividades, los sistemas de gestión de cobros mejoran la precisión, reducen los errores manuales y aceleran el ciclo de cobro, lo que puede tener un impacto significativo en el flujo de caja y la salud financiera de una empresa (Moreno, 2024).

Una de las funciones principales de los sistemas de gestión de cobros es la generación de facturas precisas y oportunas. Estos sistemas permiten a las empresas emitir facturas electrónicas de manera automática, basadas en datos de transacciones previamente registrados. Las facturas pueden personalizarse según las necesidades del cliente y enviarse por correo electrónico, reduciendo la dependencia de los procesos manuales y acelerando la entrega. Además, los sistemas pueden programar recordatorios automáticos para los pagos pendientes, ayudando a garantizar que los clientes cumplan con sus obligaciones de pago en tiempo y forma (Caballero, 2023).

Otra característica importante de los sistemas de gestión de cobros es el seguimiento y la conciliación de pagos. Estos sistemas permiten a las empresas monitorear el estado de cada factura, identificando rápidamente los pagos recibidos y los saldos pendientes. La conciliación automática de pagos con las cuentas por cobrar facilita la detección de discrepancias y asegura que los registros financieros sean precisos y estén actualizados (Moreno, 2024). Además, muchos sistemas integran herramientas de análisis y reportes que proporcionan una visión clara del rendimiento de la gestión de cobros, permitiendo a las empresas tomar decisiones informadas para mejorar su eficiencia operativa.

Los sistemas de gestión de cobros también ofrecen funcionalidades avanzadas de gestión de cuentas por cobrar. Esto incluye la capacidad de gestionar acuerdos de pago a plazos, calcular intereses por pagos atrasados y administrar disputas de facturas. Al proporcionar una visión integral de las cuentas por cobrar, estos sistemas ayudan a las empresas a identificar patrones de comportamiento de pago, segmentar a los clientes según su riesgo crediticio y

aplicar estrategias específicas para la recuperación de deuda. La automatización de estas tareas no solo mejora la eficiencia, sino que también reduce la carga administrativa sobre el personal de finanzas, permitiéndoles enfocarse en actividades estratégicas de mayor valor (Guillen, Núñez, Vargas, & Vega, 2021).

En este sentido, los sistemas de gestión de cobros son esenciales para cualquier organización que busque optimizar su proceso de facturación y cobro. Al automatizar la generación de facturas, el seguimiento de pagos y la conciliación de transacciones, estos sistemas mejoran la precisión y eficiencia de las operaciones financieras. Además, proporcionan herramientas avanzadas para la gestión de cuentas por cobrar y el análisis del rendimiento, facilitando una mejor toma de decisiones y contribuyendo a una mejor salud financiera general de la empresa. La implementación de un sistema de gestión de cobros bien diseñado puede ofrecer un retorno significativo sobre la inversión al mejorar el flujo de caja y reducir los costos operativos.

1.6.1. Funcionalidades de los sistemas de gestión de cobros

Los sistemas de gestión de cobros son herramientas integrales que automatizan y optimizan el proceso de facturación y recaudación de pagos, proporcionando una serie de funcionalidades clave que mejoran la eficiencia y precisión de las operaciones financieras. Estas funcionalidades abarcan desde la generación de facturas hasta el seguimiento de pagos y la gestión de cuentas por cobrar, facilitando la administración de todo el ciclo de cobro de manera eficaz (Moreno, 2024).

Una de las funcionalidades más fundamentales de los sistemas de gestión de cobros es la capacidad de generar facturas de manera automática. Estos sistemas permiten la creación de facturas personalizadas que pueden adaptarse a las necesidades específicas de cada cliente. Las facturas pueden incluir detalles precisos sobre los productos o servicios prestados, los términos de pago y cualquier impuesto aplicable. Además, los sistemas pueden programar la emisión de facturas recurrentes para servicios basados en suscripción, asegurando que los clientes reciban sus facturas a tiempo sin necesidad de intervención manual.

El seguimiento de pagos es otra funcionalidad importante de los sistemas de gestión de cobros. Estos sistemas permiten monitorear el estado de cada factura en tiempo real, identificando rápidamente cuáles han sido pagadas y cuáles están pendientes. La integración con plataformas de pago electrónicas facilita la recepción y conciliación automática de pagos, reduciendo el riesgo de errores y mejorando la exactitud de los registros financieros. Además,

los sistemas pueden enviar recordatorios automáticos a los clientes para los pagos pendientes, ayudando a reducir el número de facturas vencidas y mejorando el flujo de caja.

La gestión de cuentas por cobrar es una funcionalidad avanzada que permite a las empresas administrar de manera efectiva las deudas pendientes. Los sistemas de gestión de cobros proporcionan herramientas para segmentar a los clientes según su comportamiento de pago, gestionar acuerdos de pago a plazos y calcular intereses por pagos atrasados. Esto permite a las empresas aplicar estrategias específicas para la recuperación de deuda y minimizar el riesgo de incobrabilidad. Además, estos sistemas facilitan la resolución de disputas de facturas al mantener un registro detallado de todas las transacciones y comunicaciones con los clientes.

Los sistemas de gestión de cobros también ofrecen capacidades de análisis y generación de reportes, proporcionando una visión clara del rendimiento de la gestión de cobros. Estos sistemas pueden generar informes detallados sobre el estado de las cuentas por cobrar, el historial de pagos de los clientes y las tendencias de flujo de caja. Los datos analíticos ayudan a identificar patrones y áreas de mejora, permitiendo a las empresas tomar decisiones informadas para optimizar sus procesos de cobro. Además, los reportes personalizados pueden ser utilizados para comunicar el estado financiero a las partes interesadas y cumplir con los requisitos de auditoría (Caballero, 2023).

Otra funcionalidad importante de los sistemas de gestión de cobros es su capacidad de integrarse con otros sistemas empresariales, como sistemas ERP (Enterprise Resource Planning), CRM (Customer Relationship Management) y plataformas de comercio electrónico. Esta integración permite un flujo de datos continuo y preciso entre diferentes áreas de la empresa, mejorando la coherencia y la eficiencia operativa. Por ejemplo, la integración con un sistema ERP puede automatizar la contabilidad y la gestión de inventarios, mientras que la integración con un CRM puede mejorar la experiencia del cliente al proporcionar un historial completo de interacciones y transacciones.

Finalmente, los sistemas de gestión de cobros incluyen funcionalidades para garantizar la seguridad de los datos y el cumplimiento de normativas. Estos sistemas implementan medidas de seguridad avanzadas, como encriptación de datos, controles de acceso y auditorías de seguridad, para proteger la información financiera sensible. Además, aseguran el cumplimiento con normativas financieras y fiscales, ayudando a las empresas a evitar sanciones y a mantener la confianza de sus clientes.

1.6.2. Impacto en la Eficiencia y la Seguridad Financiera

Los sistemas de gestión de cobros tienen un impacto significativo en la eficiencia operativa de las organizaciones. Al automatizar tareas repetitivas y manuales, como la generación de facturas y el seguimiento de pagos, estos sistemas liberan tiempo valioso para que el personal se concentre en actividades de mayor valor estratégico. La automatización reduce el riesgo de errores humanos, mejora la precisión de los registros financieros y acelera el ciclo de cobro. Esto no solo mejora la eficiencia del departamento de finanzas, sino que también contribuye a una experiencia de cliente más positiva al garantizar que las facturas se emitan y los pagos se procesen de manera oportuna (Guillen, Núñez, Vargas, & Vega, 2021).

Además, los sistemas de gestión de cobros proporcionan herramientas avanzadas para el monitoreo y la optimización del flujo de caja. Al ofrecer una visión clara y en tiempo real del estado de las cuentas por cobrar, estos sistemas permiten a las empresas identificar rápidamente facturas vencidas, gestionar disputas de pagos y aplicar estrategias proactivas para la recuperación de deuda. La capacidad de enviar recordatorios automáticos y gestionar acuerdos de pago reduce el tiempo promedio de cobro y mejora la liquidez de la empresa. Un flujo de caja más estable y predecible permite a las organizaciones planificar mejor sus inversiones y gastos operativos, fortaleciendo su posición financiera general.

En términos de seguridad financiera, los sistemas de gestión de cobros implementan medidas robustas para proteger la información sensible y cumplir con las normativas vigentes. La encriptación de datos, los controles de acceso y las auditorías de seguridad son algunas de las características que aseguran que los datos financieros se mantengan seguros frente a accesos no autorizados y ciberataques. Además, estos sistemas garantizan el cumplimiento de normativas como el GDPR en Europa y la PCI DSS para el manejo de pagos con tarjeta, lo que ayuda a las empresas a evitar sanciones legales y a mantener la confianza de sus clientes y socios comerciales (Rodríguez, 2021).

Finalmente, los sistemas de gestión de cobros también contribuyen a la transparencia y la rendición de cuentas en las finanzas empresariales. Al proporcionar registros detallados de todas las transacciones y comunicaciones relacionadas con los cobros, estos sistemas facilitan las auditorías internas y externas, asegurando que todas las operaciones financieras se realicen de acuerdo con las políticas y procedimientos establecidos. Esta transparencia no solo fortalece la gobernanza corporativa, sino que también mejora la capacidad de la empresa para detectar y prevenir fraudes y malas prácticas, protegiendo sus activos financieros y reputación en el mercado.

1.7. Metodologías SCRUM

Las metodologías ágiles han transformado la manera en que se desarrollan y gestionan proyectos de desarrollo de software, ofreciendo un enfoque flexible y adaptativo que prioriza la colaboración, la comunicación y la entrega incremental. Desde su surgimiento en el Manifiesto Ágil en 2001, estas metodologías han ganado una amplia aceptación en la industria, ofreciendo una alternativa a los enfoques tradicionales de gestión de proyectos.

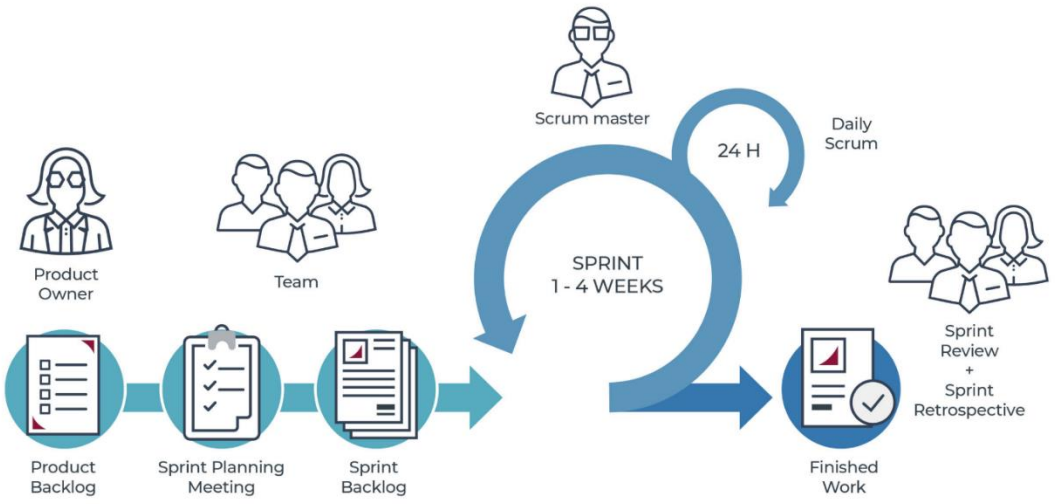


Figura 2: Metodología SCRUM.

Scrum es una metodología ágil que se centra en la entrega iterativa e incremental de productos de alta calidad, se caracteriza por su enfoque en equipos autoorganizados y multidisciplinarios, que trabajan en sprints, periodos de tiempo definidos, para desarrollar funcionalidades específicas del producto. Durante cada sprint, el equipo colabora estrechamente para planificar, realizar y revisar el trabajo completado, adaptándose continuamente a los cambios y retroalimentación del cliente. Scrum se basa en tres roles principales: el Scrum Master, responsable de facilitar el proceso y eliminar obstáculos; el Product Owner, encargado de representar las necesidades del cliente y priorizar el backlog del producto; y el Equipo de Desarrollo, responsable de entregar incrementos de trabajo completados al final de cada sprint. Con su enfoque iterativo, transparente y adaptable, Scrum permite a los equipos responder rápidamente a los requisitos cambiantes del proyecto, optimizando así la calidad e incrementando la satisfacción del cliente.

1.7.1. Artefactos de SCRUM

La metodología de SCRUM se compone de artefactos, que son los elementos fundamentales que proporcionan transparencia y visibilidad sobre el trabajo realizado, pendiente y planificado en el proyecto. Los tres artefactos principales en Scrum son:

- **Product Backlog:** Es una lista ordenada de todas las funcionalidades, requisitos, mejoras y correcciones que se deben realizar en el producto. Es responsabilidad del Product Owner gestionar y mantener este backlog, priorizando los elementos en función del valor que aportan al producto y ajustándolo continuamente en función de los cambios en los requisitos o las necesidades del cliente. El Product Backlog evoluciona a lo largo del tiempo a medida que se obtiene más información y se toman decisiones sobre qué elementos se implementarán en cada sprint.
- **Sprint Backlog:** Es una lista de todas las tareas que el Equipo de Desarrollo ha comprometido completar durante un sprint específico. Estas tareas se derivan del Product Backlog y representan el trabajo seleccionado para ser realizado durante ese período. El Sprint Backlog es dinámico y puede ser ajustado durante el sprint en respuesta a nuevos requisitos o cambios en las circunstancias. Proporciona una visión clara de lo que se espera que se entregue al final del sprint y sirve como guía para el equipo durante el desarrollo del trabajo.
- **Incremento:** El Incremento es el conjunto de todas las funcionalidades y mejoras completadas y listas para su entrega al final de un sprint. Representa el trabajo realizado durante el sprint y es la medida tangible del progreso hacia los objetivos del proyecto. El Incremento debe ser potencialmente entregable, lo que significa que cumple con los estándares de calidad definidos por el equipo y puede ser entregado al cliente o puesto en producción si se considera necesario. El objetivo de cada sprint es producir un Incremento funcional y listo para su entrega al final del período.

1.7.2. Ciclo de vida en SCRUM

El ciclo de vida de los proyectos en Scrum se caracteriza por su enfoque iterativo e incremental, dividido en una serie de sprints que constituyen unidades de trabajo con una duración fija y predeterminada. A continuación, se describe el ciclo de vida típico de un proyecto en Scrum:

- **Inicio del proyecto:** En esta etapa inicial, se define el alcance general del proyecto y se establece el equipo de trabajo, que incluye al Scrum Master, al Product Owner y al Equipo de Desarrollo.
- **Creación del Product Backlog:** El Product Owner es responsable de crear y priorizar el Product Backlog, que es una lista ordenada de todas las funcionalidades, requisitos y mejoras que se deben realizar en el producto.
- **Planificación del sprint:** Antes de comenzar cada sprint, se lleva a cabo una reunión de planificación, durante esta reunión, el equipo selecciona un conjunto de elementos del Product Backlog para trabajar durante el próximo sprint y define el objetivo del sprint.
- **Desarrollo del sprint:** Durante el sprint, el Equipo de Desarrollo trabaja en la implementación de las funcionalidades seleccionadas. Se llevan a cabo reuniones diarias de seguimiento (daily scrum) para revisar el progreso y sincronizar el trabajo del equipo. El Sprint Backlog sirve como guía para el equipo durante el desarrollo del trabajo.
- **Revisión del sprint:** Al finalizar el sprint, se realiza una reunión de revisión del sprint para demostrar el Incremento completado al Product Owner y a otros interesados. Durante esta reunión, se recopila feedback y se discuten posibles mejoras.
- **Retrospectiva del sprint:** Después de la reunión de revisión del sprint, se lleva a cabo una retrospectiva del sprint en la que el equipo reflexiona sobre lo que salió bien, lo que podría mejorarse y qué acciones se pueden tomar para mejorar el proceso en el próximo sprint.
- **Iteración de los sprints:** El ciclo se repite con la planificación de un nuevo sprint, comenzando por la selección de nuevos elementos del Product Backlog y continuando con el desarrollo, revisión y retrospectiva del sprint. Este proceso iterativo e incremental continúa hasta que se alcanzan los objetivos del proyecto o se decide finalizar el desarrollo.

CAPÍTULO 2

Desarrollo del proyecto

La selección de Scrum como la metodología para el desarrollo se justifica principalmente por su enfoque iterativo e incremental, que permite adaptarse rápidamente a los cambios y necesidades emergentes del proyecto. Dado que el entorno de trabajo es dinámico y requiere una mejora continua para optimizar la eficiencia y garantizar el cumplimiento de normativas, Scrum facilita la entrega de funcionalidades valiosas en cortos ciclos de desarrollo llamados sprints, esto no solo asegura que el sistema web evolucione de manera continua y controlada, sino que también permite incorporar el feedback de los usuarios finales y los stakeholders de manera temprana y frecuente, mejorando la calidad del producto final y asegurando que las soluciones implementadas realmente satisfagan las necesidades operativas de la empresa ANTENAWIFI S.A.S.

2.1. Descripción de la empresa

La empresa fue fundada el 15 de enero del 2016 en la provincia de Imbabura, parroquia San Francisco, ubicado en las calles Bolívar 13 – 122 y Teodoro Gómez siendo su matriz principal dicha dirección. Dentro de sus actividades tenemos la de ofrecer servicios (Soluciones Tecnológicas), siendo su principal la distribución de Internet a los distintos clientes que se encuentran dentro la ciudad y sus alrededores, teniendo como principal objetivo la de brindar servicios a zonas rurales donde hay demanda del servicio por la mala atención y soporte técnico de operadoras de otras Empresas que ofertan servicios.

Figura 3: Logo de la empresa.

El organigrama de la empresa ANTENAWIFI S.A.S. es una representación gráfica que ilustra la estructura organizacional de nuestra compañía, destacando las relaciones jerárquicas y las funciones de cada departamento. Este organigrama permite visualizar de manera clara y concisa cómo se distribuyen las responsabilidades y cómo fluye la comunicación entre los distintos niveles de la empresa.

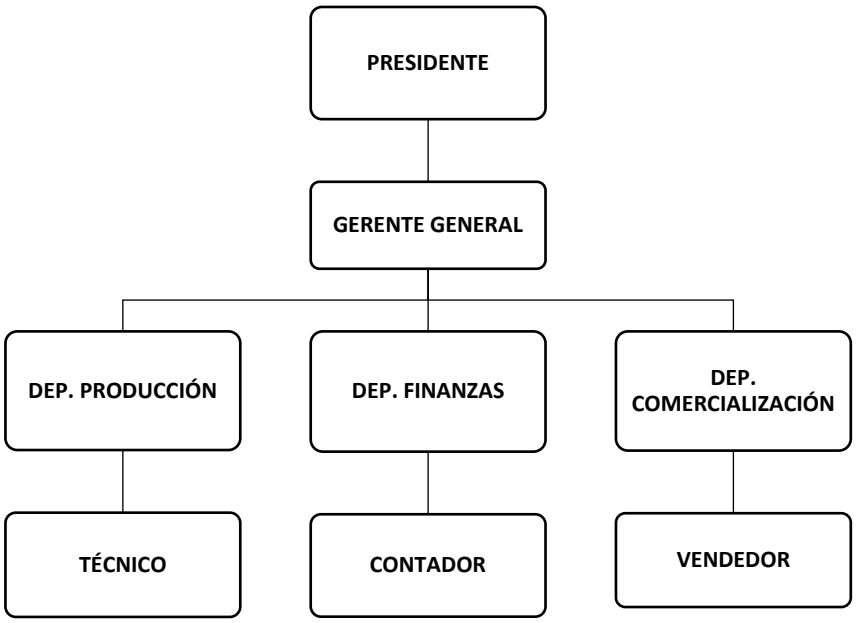


Figura 4: Organigrama de la empresa.

2.2. Conformación del Equipo SCRUM

La especificación clara de roles en SCRUM es necesaria para la implementación de la nueva arquitectura de datos, considerando que las mismas deben ser claras y que fomenten la colaboración efectiva dentro del equipo. La Tabla 1 describe las personas que ejecutaran cada uno de los roles y las funciones específicas a ser ejecutadas.

Rol	Descripción	Funciones
Product Owner	Sr. Juan Andrés Chávez	Verificar que las funcionalidades del sistema se adapten a los requerimientos de la entidad.
Scrum Master	Ing. Diego Trejo	Verificar y facilitar el avance del desarrollo del sistema, así como el uso correcto de la metodología SCRUM.
Equipo de Desarrollo	Darwin Tuquerres	Desarrollar el sistema en base a las especificaciones y requerimientos establecidos por el Product Owner.

Tabla 1: Roles del Equipo SCRUM

2.3. Definición del product bakclog

El Product Backlog es una herramienta fundamental en la metodología Scrum, representando una lista dinámica y priorizada de todas las funcionalidades que deben ser incorporadas en el sistema de información; por lo tanto, representa una fuente de requisitos que guían el desarrollo del producto, este backlog es una representación en tiempo real de los requisitos del proyecto, y su gestión eficaz es clave para el éxito del equipo de desarrollo.

Para su elaboración, se aplicaron varios mecanismos para la recolección de necesidades; a continuación, se describen las técnicas aplicadas:

- **Reuniones de inicio de proyecto:** Al comienzo del proyecto, se realizaron reuniones con todas las partes interesadas para recopilar los requerimientos iniciales y las expectativas. Estas reuniones se incluyeron usuarios finales, stakeholders clave, el equipo directivo y el equipo de desarrollo.
- **Grupos focales para la identificación de requerimientos:** Se generaron sesiones estructuradas para generar ideas y requerimientos de manera colaborativa, estas actividades facilitaron la participación de diferentes perspectivas y aseguran que se capturen una amplia gama de necesidades.
- **Entrevistas con Usuarios y Stakeholders:** Se realizaron entrevistas directas con usuarios y stakeholders para obtener información detallada sobre sus necesidades y deseos. Esto ayuda a crear historias de usuario claras y significativas que representen verdaderamente los requerimientos del negocio.

2.3.1. Definición de las épicas de usuario

Para el desarrollo del sistema de gestión de cobros, es fundamental considerar las necesidades y expectativas de todos los usuarios involucrados, y que este sistema no solo debe facilitar la automatización de los procesos, sino también proporcionar herramientas robustas para la gestión de usuarios, clientes, cobros, reparaciones y generación de reportes. A continuación, en la Tabla 2 se presentan las épicas de usuario que permiten asegurar que el sistema cumpla con las demandas específicas y que guíen el diseño y la implementación del sistema, garantizando que se aborden todos los aspectos críticos para el éxito del proyecto.

Épica	Descripción
EP-01	**COMO** Responsable de TI, **QUIERO** seleccionar el framework más adecuado entre Symfony 3 y Laravel, **PARA** desarrollar un sistema que sea escalable, seguro y eficiente.
EP-02	**COMO** Administrador del Sistema, **QUIERO** gestionar usuarios y permisos, **PARA** asegurar que cada usuario tenga acceso adecuado según sus funciones y responsabilidades.
EP-03	**COMO** Responsable de Clientes, **QUIERO** gestionar la información de los clientes, **PARA** mantener datos precisos y actualizados que faciliten la facturación y el servicio al cliente.
EP-04	**COMO** Responsable de Cobros, **QUIERO** gestionar y automatizar el proceso de cobros, **PARA** asegurar que las facturas se emitan correctamente y los pagos se reciban puntualmente.
EP-05	**COMO** Responsable de Reparaciones, **QUIERO** registrar y gestionar las reparaciones realizadas, **PARA** llevar un control detallado de los servicios técnicos prestados a los clientes.
EP-06	**COMO** Directivos de la empresa, **QUIERO** generar y visualizar reportes detallados, **PARA** evaluar el rendimiento del sistema de gestión de cobros y detectar oportunidades de mejora.

Tabla 2: Épicas de usuario.

2.3.2. Definición de las historias de usuario

Las historias de usuario que a continuación se detallan, proporcionan una guía clara y estructurada para el desarrollo del sistema de gestión de cobros, asegurando que se aborden todas las necesidades y objetivos críticos de los diferentes usuarios involucrados.

- Historias de Usuario de la Épica de Usuario EP-01

Épica: EP-01	**Historia de Usuario:** HU-01
Descripción: Como responsable de TI, quiero recopilar información técnica detallada sobre Symfony 3 y Laravel, para evaluar sus características y capacidades.	
Prioridad: Alta	**Riesgo:** Bajo
Criterios de aceptación: • Se ha realizado una revisión bibliográfica completa sobre ambos frameworks. • Se han documentado las características técnicas principales de Symfony 3 y Laravel. • Se ha preparado un informe comparativo preliminar.	

Tabla 3: Historia de usuario HU-01.

Épica: EP-01	**Historia de Usuario:** HU-02
Descripción: Como responsable de TI, quiero establecer criterios de evaluación claros (rendimiento, seguridad, escalabilidad, facilidad de uso), para realizar una comparación justa entre Symfony 3 y Laravel.	
Prioridad: Alta	**Riesgo:** Medio
Criterios de aceptación: • Se han definido y documentado los criterios de evaluación. • Se ha creado una matriz de evaluación que compara Symfony 3 y Laravel en base a estos criterios. • La matriz ha sido revisada y aprobada por el equipo de desarrollo.	

Tabla 4: Historia de usuario HU-02.

- Historias de Usuario de la Épica de Usuario EP-02

Épica: EP-02	**Historia de Usuario:** HU-03
Descripción: Como administrador, quiero crear, modificar y eliminar cuentas de usuario, para mantener un control adecuado sobre quién tiene acceso al sistema.	
Prioridad: Alta	**Riesgo:** Medio
Criterios de aceptación: • El sistema permite la creación de nuevas cuentas de usuario. • Las cuentas de usuario pueden ser modificadas y eliminadas por el administrador. • Se han realizado pruebas de funcionalidad para asegurar que los cambios se aplican correctamente.	

Tabla 5: Historia de usuario HU-03.

Épica: EP-02	**Historia de Usuario:** HU-04
Descripción: Como administrador, quiero definir roles y asignar permisos específicos a cada rol, para asegurar que los usuarios solo accedan a la información y funciones necesarias para su trabajo.	
Prioridad: Alta	**Riesgo:** Alto
Criterios de aceptación: • El sistema permite la creación y gestión de roles de usuario. • Los permisos pueden ser asignados y modificados para cada rol. • Se han realizado pruebas para verificar que los usuarios tienen acceso solo a las áreas asignadas.	

Tabla 6: Historia de usuario HU-04.

- Historias de Usuario de la Épica de Usuario EP-03

Épica: EP-03	**Historia de Usuario:** HU-05
Descripción: Como responsable de clientes, quiero registrar y actualizar información de los clientes (contactos, direcciones, historial de interacciones), para asegurarme de que todos los datos estén siempre actualizados.	
Prioridad: Alta	**Riesgo:** Medio
Criterios de aceptación: • El sistema permite la creación y actualización de perfiles de cliente con información completa. • Los cambios en la información de los clientes se guardan y reflejan en tiempo real. • Se han realizado pruebas para verificar la exactitud de los datos actualizados.	

Tabla 7: Historia de usuario HU-05.

Épica: EP-03	**Historia de Usuario:** HU-06
Descripción: Como responsable de clientes, quiero buscar y filtrar la base de datos de clientes, para encontrar rápidamente la información relevante.	
Prioridad: Alta	**Riesgo:** Bajo
Criterios de aceptación: • El sistema proporciona funcionalidades de búsqueda y filtrado de clientes por diferentes criterios (nombre, ID, ubicación, etc.). • Los resultados de búsqueda se muestran de manera rápida y precisa. • Se han realizado pruebas para asegurar que las búsquedas y filtros funcionan	

Tabla 8: Historia de usuario HU-06.

- Historias de Usuario de la Épica de Usuario EP-04

Épica: EP-04	**Historia de Usuario:** HU-07
Descripción: Como responsable de cobros, quiero generar facturas automáticamente basadas en contratos y servicios prestados, para reducir errores manuales y ahorrar tiempo.	
Prioridad: Alta	**Riesgo:** Medio
Criterios de aceptación: • El sistema genera facturas automáticamente según los contratos y servicios registrados. • Las facturas generadas son precisas y contienen toda la información requerida. • Se han realizado pruebas para asegurar que las facturas se generan correctamente.	

Tabla 9: Historia de usuario HU-07.

Épica: EP-04	**Historia de Usuario:** HU-08
Descripción: Como responsable de cobros, quiero registrar y conciliar pagos recibidos, para mantener la precisión de los registros financieros y asegurar que todas las transacciones estén documentadas.	
Prioridad: Alta	**Riesgo:** Alto
Criterios de aceptación: • El sistema permite el registro manual y automático de pagos recibidos. • Los pagos se concilian correctamente con las facturas correspondientes. • Se han realizado pruebas para asegurar la precisión y la integridad de los registros de pago.	

Tabla 10: Historia de usuario HU-08.

- Historias de Usuario de la Épica de Usuario EP-05

Épica: EP-05	**Historia de Usuario:** HU-09
Descripción: Como responsable de reparaciones, quiero registrar detalles de cada reparación (fecha, tipo de reparación, piezas utilizadas, tiempo empleado), para mantener un historial completo de los trabajos realizados.	
Prioridad: Alta	**Riesgo:** Medio
Criterios de aceptación: • El sistema permite el registro detallado de cada reparación realizada. • Los registros incluyen información sobre la fecha, tipo de reparación, piezas utilizadas y tiempo empleado. • Se han realizado pruebas para asegurar que los registros de reparación son precisos y completos.	

Tabla 11: Historia de usuario HU-09.

Épica: EP-05	**Historia de Usuario:** HU-10
Descripción: Como responsable de reparaciones, quiero generar informes sobre las reparaciones realizadas (frecuencia, tipo, costo), para analizar tendencias y planificar el mantenimiento preventivo.	
Prioridad: Alta	**Riesgo:** Medio
Criterios de aceptación: • El sistema puede generar informes detallados sobre las reparaciones realizadas. • Los informes pueden incluir información sobre frecuencia, tipo de reparación y costos. • Se han realizado pruebas para asegurar la precisión y utilidad de los informes generados.	

Tabla 12: Historia de usuario HU-10.

- Historias de Usuario de la Épica de Usuario EP-06

Épica: EP-06	**Historia de Usuario:** HU-11
Descripción: Como directivo, quiero crear informes personalizados sobre los cobros realizados, los pagos pendientes y el rendimiento de los técnicos, para identificar áreas de mejora y optimizar las operaciones.	
Prioridad: Alta	**Riesgo:** Medio
Criterios de aceptación: • El sistema permite la creación de informes personalizados según diferentes criterios. • Los informes son precisos y reflejan la información actualizada sobre cobros, pagos y rendimiento. • Se han realizado pruebas para asegurar que los informes cumplen con los requisitos especificados.	

Tabla 13: Historia de usuario HU-11.

Épica: EP-06	**Historia de Usuario:** HU-12
Descripción: Como directivo, quiero visualizar datos en tiempo real mediante paneles de control interactivos, para monitorear el desempeño del sistema de gestión de cobros y tomar decisiones informadas.	
Prioridad: Alta	**Riesgo:** Alto
Criterios de aceptación: • El sistema proporciona paneles de control interactivos que muestran datos en tiempo real. • Los paneles de control son personalizables y fáciles de usar. • Se han realizado pruebas para asegurar la precisión y la relevancia de los datos presentados en los paneles.	

Tabla 14: Historia de usuario HU-12.

Épica: EP-06	**Historia de Usuario:** HU-13
Descripción: Como directivo, quiero exportar datos en formatos compatibles con otras herramientas de análisis, para realizar análisis avanzados y compartir información con otros departamentos.	
Prioridad: Alta	**Riesgo:** Medio
Criterios de aceptación: • El sistema permite la exportación de datos en formatos estándar (CSV, Excel, PDF). • Los datos exportados son precisos y están bien formateados. • Se han realizado pruebas para asegurar que los datos exportados son compatibles con otras herramientas de análisis.	

Tabla 15: Historia de usuario HU-13.

2.4. Planificación del proyecto

Para llevar a cabo el desarrollo del sistema de gestión de cobros que sea funcional, eficiente y alineado con las necesidades de la organización, es importante organizar el trabajo en sprints bien definidos. Los sprints son períodos de trabajo cortos y centrados en objetivos específicos, permitiendo al equipo de desarrollo abordar tareas complejas de manera incremental y ajustarse rápidamente a cambios o nuevos requerimientos. Esta metodología ágil facilita la entrega continua de valor, garantizando que cada incremento del producto sea funcional y de alta calidad.

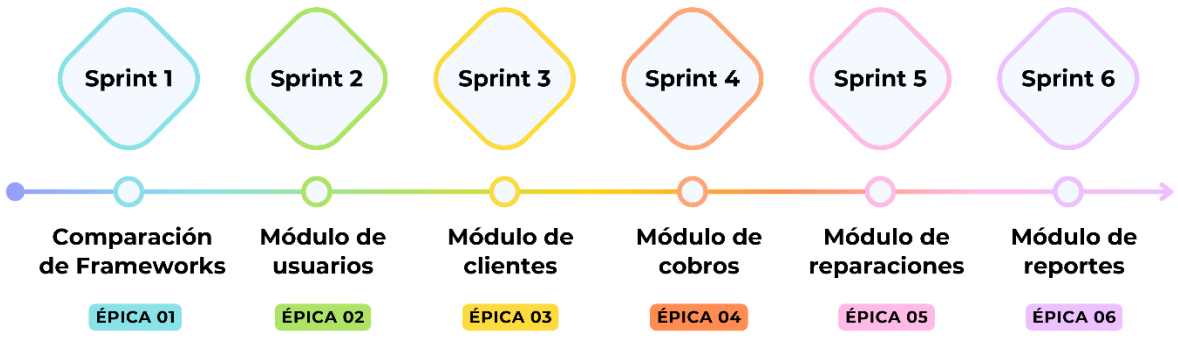


Figura 5: Planificación del proyecto.

2.5. Sprint 1: Comparación de Frameworks

En este primer sprint, el objetivo principal es llevar a cabo la evaluación de los frameworks Symfony 3 y Laravel, las actividades incluirán una revisión de la literatura técnica y la documentación disponible para cada framework, con el fin de comprender sus características, ventajas y desventajas. Se recopilarán datos sobre aspectos clave como rendimiento, seguridad, escalabilidad y facilidad de uso. Además, se establecerán criterios de evaluación claros y se desarrollará una matriz comparativa preliminar que permita comparar ambos frameworks de manera sistemática. Este sprint culminará con una tabla de evaluación de todos los criterios definidos que permita seleccionar el Framework más idóneo para el presente proyecto.

Ítem	Responsables	HU	Tareas	Horas asignadas
1	Desarrollador 1	HU-01	3	28H
2	Desarrollador 1	HU-02	4	16H
		TOTAL		44H

Tabla 16: Sprint Backlog para la comparación de los Frameworks.

Para la ejecución del primer sprint, se planificó su ejecución en 44 horas laborables; para lo cual, se realizaron reuniones para evaluar los avances, minimizar los retrasos y garantizar el cumplimiento de la planificación dentro del tiempo establecido.

2.5.1. Establecer los criterios de evaluación

Definir criterios claros para comparar los frameworks es fundamental para garantizar una evaluación objetiva y exhaustiva de las opciones disponibles. Estos criterios permiten analizar aspectos críticos como el rendimiento, la seguridad, la escalabilidad y la facilidad de uso de cada framework, asegurando que la comparación sea justa y alineada con las necesidades específicas del proyecto. Al establecer estos parámetros de evaluación, se facilita la identificación de fortalezas y debilidades de cada framework, proporcionando una base sólida para tomar decisiones informadas. Esto no solo optimiza la selección del framework más adecuado, sino que también minimiza los riesgos y asegura que el sistema desarrollado será eficiente, seguro y capaz de soportar el crecimiento futuro de la organización.

- Rendimiento
    - Velocidad de ejecución: Tiempo de respuesta con cada framework bajo diferentes cargas de trabajo.
    - Consumo de recursos: Uso de CPU y memoria durante la ejecución de tareas comunes.
    - Capacidad de manejo de concurrencia: Identificar la manera cómo cada framework maneja múltiples solicitudes simultáneas.
- Seguridad
    - Mecanismos de autenticación y autorización: Comparar las funcionalidades integradas para gestionar la autenticación y los permisos de usuario.
    - Protección contra vulnerabilidades comunes: Evaluar las medidas de seguridad contra ataques como SQL Injection, Cross-Site Scripting (XSS) y Cross-Site Request Forgery (CSRF).
    - Actualizaciones y soporte de seguridad: Analizar la frecuencia y rapidez con la que cada framework recibe actualizaciones de seguridad y soporte de la comunidad.
- Escalabilidad
    - Escalabilidad horizontal: Evaluar la capacidad de cada framework para escalar agregando más servidores.
    - Gestión de cargas altas: Probar el rendimiento bajo condiciones de alta carga y tráfico.
    - Soporte para microservicios y APIs: Analizar la facilidad con la que cada framework se integra con arquitecturas de microservicios y API.
- Facilidad de Uso
    - Curva de aprendizaje: Medir el tiempo y los recursos necesarios para que un desarrollador aprenda a utilizar cada framework de manera efectiva.

- Documentación y recursos de aprendizaje: Evaluar la calidad, claridad y exhaustividad de la documentación oficial, así como la disponibilidad de tutoriales y guías.
- Comunidad y soporte: Analizar el tamaño y la actividad de la comunidad de usuarios y desarrolladores, así como la disponibilidad de soporte técnico y foros de ayuda.

- Flexibilidad y Personalización
  - Modularidad: Evaluar la capacidad de cada framework para permitir la personalización y extensión a través de módulos o bundles.
  - Integración con herramientas externas: Probar la facilidad de integración con otras herramientas y bibliotecas de desarrollo.
  - Compatibilidad con diferentes bases de datos: Analizar el soporte y facilidad de configuración con distintas bases de datos.
- Mantenimiento y Actualización
  - Facilidad de mantenimiento del código: Evaluar la estructura y organización del código generado por cada framework para facilitar el mantenimiento a largo plazo.
  - Compatibilidad con nuevas versiones: Analizar la facilidad de actualización a nuevas versiones del framework sin romper la funcionalidad existente.
  - Automatización y herramientas DevOps: Probar la integración con herramientas de automatización y DevOps, como CI/CD.
- Costos
  - Licencias y costos asociados: Evaluar cualquier costo de licencia o suscripción asociado con el uso de cada framework.
  - Costos de desarrollo y mantenimiento: Analizar los costos implicados en el desarrollo y mantenimiento a largo plazo utilizando cada framework.

Estos criterios proporcionan una base sólida para una comparación detallada y objetiva de Symfony 3 y Laravel, asegurando que la elección final esté alineada con las necesidades y objetivos del proyecto.

2.5.2. Evaluación de los Frameworks

Para garantizar una selección informada del framework más adecuado para el desarrollo del sistema de gestión de cobros, se llevó a cabo una comparación exhaustiva entre Symfony 3 y Laravel. Esta evaluación se basó en criterios clave como rendimiento, seguridad, escalabilidad, facilidad de uso, flexibilidad, mantenimiento y costos; para lo cual, cada criterio fue evaluado en una escala del 1 al 5, con explicaciones detalladas que sustentan las puntuaciones otorgadas.

Con el objetivo de contar con un criterio técnico más sólido en la comparación, se procedió a desarrollar el prototipo caso de uso "Registro de usuarios" utilizando los frameworks PHP Laravel y Symfony. Posteriormente, se evaluaron las características señaladas en el estudio y se valoró su funcionalidad aplicando escala de Likert.

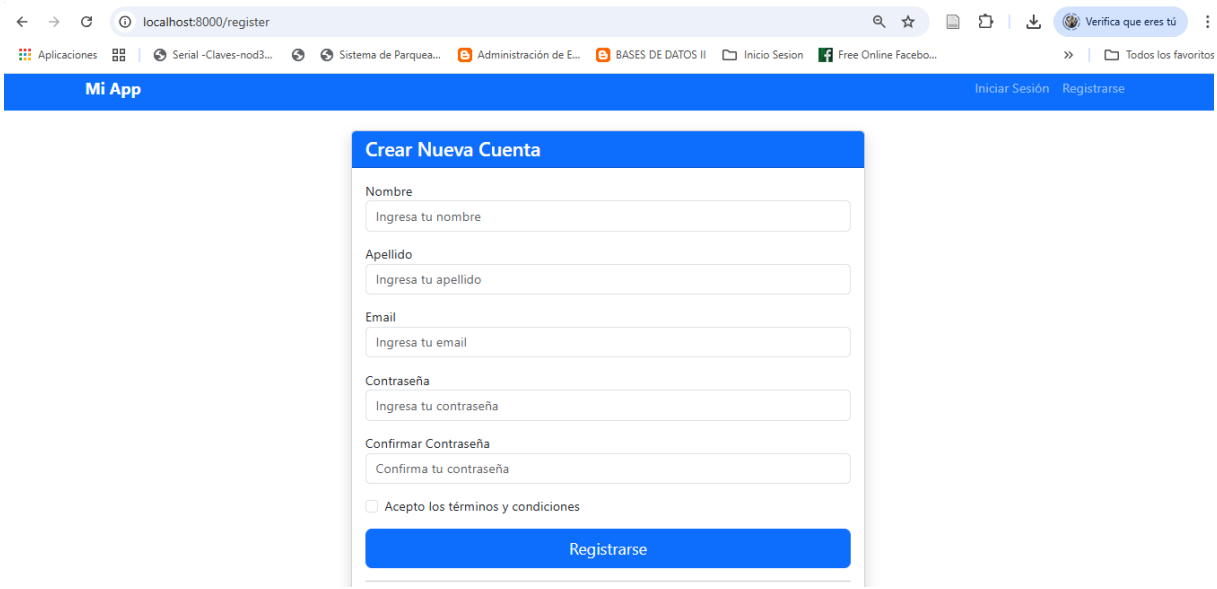


Figura 6: Prototipo Aplicación con Symfony.

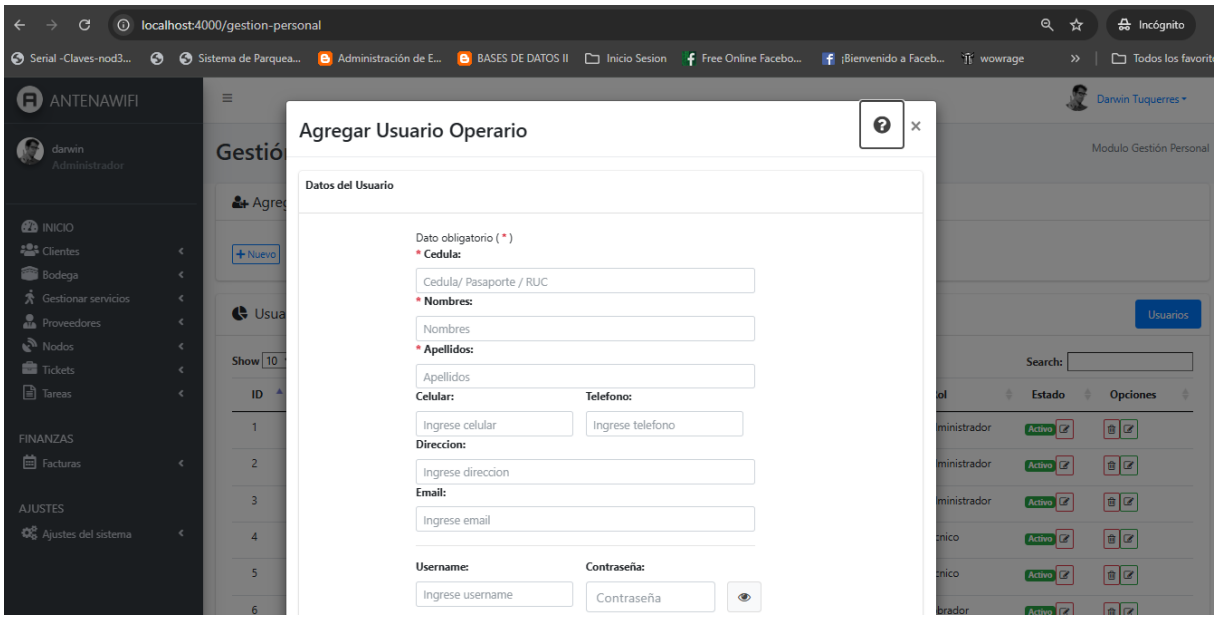


Figura 7: Prototipo Aplicación con Laravel.

A continuación, la Tabla 17 presenta los resultados de esta comparación, proporcionando una visión clara y objetiva de las fortalezas y debilidades de cada uno, con el objetivo de identificar cuál de ellos es más adecuado para satisfacer las necesidades del proyecto.

Criterio	Framework	Puntaje	Justificación
Velocidad de Ejecución	Symfony 3	3	Es robusto, pero puede ser más lento en ciertas operaciones debido a su arquitectura modular y la cantidad de configuraciones necesarias.
	Laravel	4	Con su enfoque en la simplicidad y rendimiento, generalmente tiene tiempos de respuesta más rápidos en aplicaciones típicas.
Consumo de Recursos	Symfony 3	3	Consume más recursos debido a su flexibilidad y la cantidad de componentes cargados por defecto.
	Laravel	4	Es más ligero gracias a su arquitectura optimizada y la capacidad de desactivar servicios no utilizados fácilmente.

Criterio	Framework	Puntaje	Justificación
Capacidad de Manejo de Concurrencia	Symfony 3	3	Maneja bien la concurrencia, pero su configuración inicial puede requerir más ajustes para optimizar el rendimiento bajo alta concurrencia.
	Laravel	4	Incluye herramientas como Redis y soporta colas de trabajo de manera eficiente, facilitando la gestión de múltiples solicitudes simultáneas.
Mecanismos de Autenticación y Autorización	Symfony 3	4	Ofrece un sistema de seguridad muy robusto y flexible, pero su configuración puede ser compleja.
	Laravel	5	Ofrece características de seguridad fáciles de implementar como Laravel Sanctum y Passport para API, con una configuración simplificada.
Protección contra Vulnerabilidades Comunes	Symfony 3	4	Incluye protecciones integradas contra muchas vulnerabilidades comunes, pero puede requerir más configuraciones manuales.
	Laravel	5	Cuenta con protección contra XSS, CSRF y SQL Injection por defecto, haciendo que la seguridad sea más accesible y fácil de gestionar.
Actualizaciones y Soporte de Seguridad	Symfony 3	4	Tiene una comunidad activa y soporte para actualizaciones de seguridad, aunque la frecuencia de actualizaciones puede variar.
	Laravel	5	Recibe actualizaciones frecuentes y tiene una comunidad muy activa, asegurando que las vulnerabilidades se aborden rápidamente.

Criterio	Framework	Puntaje	Justificación
Escalabilidad Horizontal	Symfony 3	4	Es muy escalable, pero puede requerir configuraciones adicionales y ajustes para maximizar la escalabilidad.
	Laravel	5	Está diseñado pensando en la escalabilidad y ofrece herramientas y documentación extensiva para escalar aplicaciones fácilmente.
Gestión de Cargas Altas	Symfony 3	3	Puede manejar cargas altas, pero su rendimiento óptimo puede requerir ajustes significativos.
	Laravel	4	Maneja cargas altas de manera eficiente con menos ajustes, especialmente cuando se usan herramientas de cache y colas.
Soporte para Microservicios y APIs	Symfony 3	4	Soporta arquitecturas de microservicios y APIs, pero su configuración puede ser más compleja.
	Laravel	5	Facilita la creación de microservicios y APIs con herramientas como Laravel Lumen y una configuración intuitiva.
Curva de Aprendizaje	Symfony 3	3	La curva de aprendizaje es más pronunciada debido a su complejidad y cantidad de configuraciones.
	Laravel	5	Es conocido por su simplicidad y facilidad de uso, lo que reduce significativamente la curva de aprendizaje.
Documentación y Recursos de Aprendizaje	Symfony 3	4	Tiene una documentación detallada y extensa, pero puede ser abrumadora para los nuevos usuarios.
	Laravel	5	Ofrece documentación clara, tutoriales accesibles y una gran cantidad de recursos comunitarios, facilitando el aprendizaje.

Criterio	Framework	Puntaje	Justificación
Comunidad y Soporte	Symfony 3	4	Tiene una comunidad activa y madura, pero puede no ser tan accesible como la de Laravel.
	Laravel	5	Tiene una de las comunidades más grandes y activas, con numerosos foros, grupos y recursos de soporte.
Modularidad	Symfony 3	5	Es extremadamente modular, permitiendo una alta personalización y reutilización de componentes.
	Laravel	4	Es modular, pero en un grado ligeramente menor comparado con Symfony, aunque sigue siendo muy flexible.
Integración con Herramientas Externas	Symfony 3	4	Se integra bien con muchas herramientas, pero a veces requiere configuraciones más complejas.
	Laravel	5	Se integra fácilmente con una amplia gama de herramientas y servicios gracias a su diseño y comunidad de paquetes.
Compatibilidad con Diferentes Bases de Datos	Symfony 3	4	Soporta múltiples bases de datos, pero su configuración puede ser más compleja.
	Laravel	5	Ofrece soporte fácil y flexible para diferentes bases de datos con su ORM Eloquent.
Facilidad de Mantenimiento del Código	Symfony 3	4	Promueve buenas prácticas de mantenimiento, pero su estructura puede ser más compleja.
	Laravel	5	Facilita el mantenimiento del código con una estructura clara y herramientas como migrations y seeders.

Criterio	Framework	Puntaje	Justificación
Compatibilidad con Nuevas Versiones	Symfony 3	4	Asegura compatibilidad con nuevas versiones, pero las actualizaciones pueden requerir más esfuerzo.
	Laravel	5	Proporciona actualizaciones frecuentes y facilita la migración a nuevas versiones con mínima interrupción.
Automatización y Herramientas DevOps	Symfony 3	4	Se integra bien con herramientas de automatización y DevOps, pero puede requerir más configuración.
	Laravel	5	Soporta perfectamente CI/CD pipelines y otras herramientas DevOps con configuración sencilla.
Licencias y Costos Asociados	Symfony 3	5	Es de código abierto y no tiene costos de licencia.
	Laravel	5	También es de código abierto y libre de costos de licencia.
Costos de Desarrollo y Mantenimiento	Symfony 3	3	Los costos de desarrollo y mantenimiento pueden ser más altos debido a su complejidad y curva de aprendizaje.
	Laravel	5	Los costos de desarrollo y mantenimiento suelen ser menores debido a su simplicidad y amplia disponibilidad de recursos.

Tabla 17: Resultados de la evaluación de los Frameworks.

Para sustentar las puntuaciones de los resultados de la evaluación de los Frameworks como se muestra en la Tabla 18, donde los criterios a estimar como rendimiento, seguridad, etc., se los realizo de manera subjetiva a ciertos criterios mediante la ayuda de expertos que desarrollan aplicaciones web con los Frameworks mencionados, como también ciertos criterios fueron evaluados de manera objetiva, mediante la realización de pruebas como tiempo de respuesta, uso de memoria entre otros.

Para el criterio de rendimiento, seguridad la siguiente figura muestra específicamente que consiste en una consulta a un endpoint que devuelve datos de usuario (/getalluser), el

framework Laravel (ejecutándose en el puerto 8001) es significativamente más rápido que Symfony (ejecutándose en el puerto 8000).

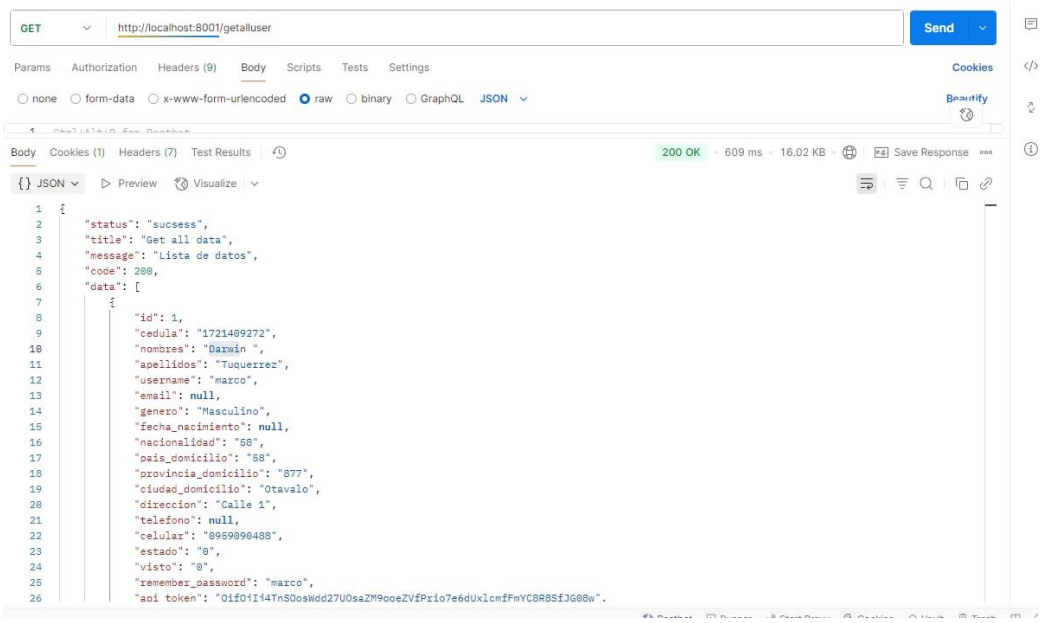


Figura 8: Puerto 8001, ejecución Laravel.

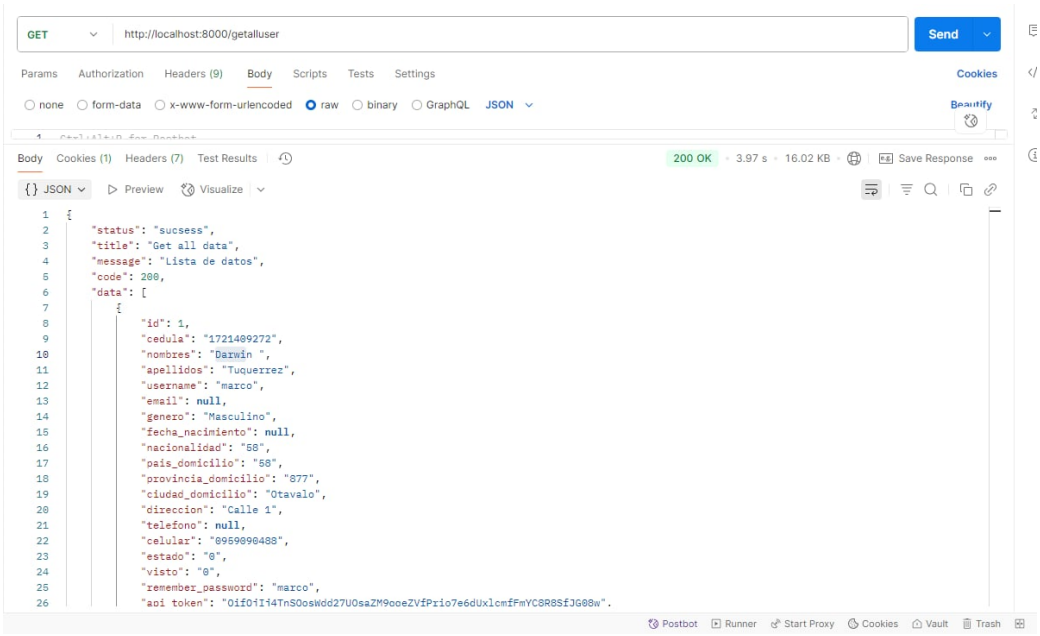


Figura 9: Puerto 8000, ejecución Symfony.

De manera subjetiva mediante expertos y canales oficiales podemos demostrar la curva de aprendizaje en Laravel es más extenso, esto se debe a la comunidad que maneja, la siguiente figura nos muestra el porcentaje y la estadística de desarrolladores usando Laravel como framework para el desarrollo web en primer lugar, seguido de otros Frameworks como es el caso de Symfony ocupando el tercer lugar.

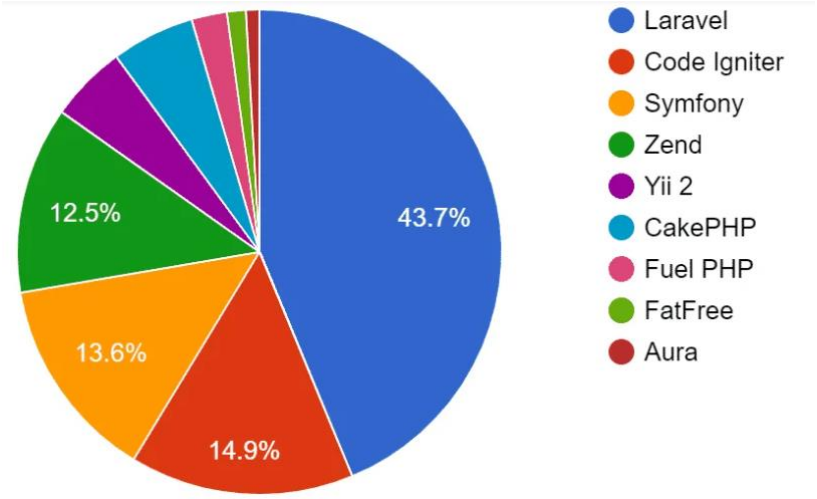


Figura 10: Uso de Frameworks PHP en proyectos web

Fuente: *https://www.excellentwebworld.com/best-php-frameworks/*

Esta evaluación muestra que (ver Tabla 18), aunque Symfony 3 tiene muchas fortalezas, Laravel se destaca en la mayoría de los criterios, particularmente en facilidad de uso, seguridad, y costos de desarrollo y mantenimiento, lo que lo convierte en la mejor opción para este proyecto específico.

Criterio	Symfony 3	Laravel
RENDIMIENTO		
Velocidad de ejecución	3	4
Consumo de recursos	3	4
Capacidad de manejo de concurrencia	3	4
SEGURIDAD		
Mecanismos de autenticación	4	5
Protección contra vulnerabilidades	4	5
Actualizaciones y soporte	4	5
ESCALABILIDAD		
Escalabilidad horizontal	4	5
Gestión de cargas altas	3	4
Soporte para microservicios	4	5
FACILIDAD DE USO		
Curva de aprendizaje	3	5
Documentación y recursos	4	5
Comunidad y soporte	4	5

Criterio	Symfony 3	Laravel
FLEXIBILIDAD Y PERSONALIZACIÓN		
Modularidad	5	4
Integración con herramientas	4	5
Compatibilidad con bases de datos	4	5
MANTENIMIENTO Y ACTUALIZACIÓN		
Facilidad de mantenimiento	4	5
Compatibilidad con nuevas versiones	4	5
Automatización y herramientas DevOps	4	5
COSTOS		
Licencias y costos asociados	5	5
Costos de desarrollo y mantenimiento	3	5
TOTAL	**76**	**95**

Tabla 18: Resultados resumen de la evaluación de los Frameworks.

2.5.3. Configuración del entorno de desarrollo

La configuración del entorno de desarrollo es un paso crítico en el inicio del proyecto de software, implica la creación de un entorno de trabajo adecuado, para escribir, probar y depurar código de manera eficiente. Durante esta configuración, se instalaron y configuraron todas las herramientas que garanticen un desarrollo adecuado, se establecieron las dependencias del proyecto, se configuró el servidor y se integraron las diferentes partes del sistema, en la Tabla 19 se describen las herramientas utilizadas para el presente proyecto.

Herramienta	Descripción
Entornos de Desarrollo Integrado (IDE)	Visual Studio Code
Front end	CSS, Javacript
Back end	PHP
Framework	Laravel
Servidor web	Apache
Base de datos	MySQL

Control de versiones	Git, GitHub

Tabla 19: Herramientas del entorno de desarrollo

2.5.4. Arquitectura

El patrón Modelo-Vista-Controlador (MVC) es una arquitectura fundamental en el desarrollo del sistema de gestión de cobros, ya que separa la aplicación en tres componentes interdependientes pero distintos: el Modelo, la Vista y el Controlador. El Modelo gestiona la lógica de negocio y el acceso a los datos, asegurando que las operaciones de cobro, facturación y registros de reparaciones se realicen correctamente y se almacenen de manera segura. La Vista se encarga de la presentación de la información, proporcionando interfaces de usuario intuitivas y accesibles para que los usuarios puedan interactuar con el sistema de manera eficiente. El Controlador actúa como un intermediario, gestionando las solicitudes de los usuarios, procesando los datos mediante el Modelo y actualizando la Vista en consecuencia. Esta separación de responsabilidades facilita el mantenimiento y la escalabilidad del sistema, permitiendo una evolución más ágil y una implementación modular de nuevas funcionalidades.

2.5.5. Modelamiento de la base de datos

El modelamiento entidad-relación (ER) de la base de datos permite definir la estructura lógica de los datos y las relaciones entre ellos; en este modelo, las entidades principales incluyen Usuarios, Clientes, Cobros, Reparaciones y Facturas. Este modelamiento ER proporciona una base sólida para la implementación de una base de datos relacional, asegurando integridad de los datos, reducción de redundancias y facilitación de consultas eficientes y coherentes en el sistema de gestión de cobros.

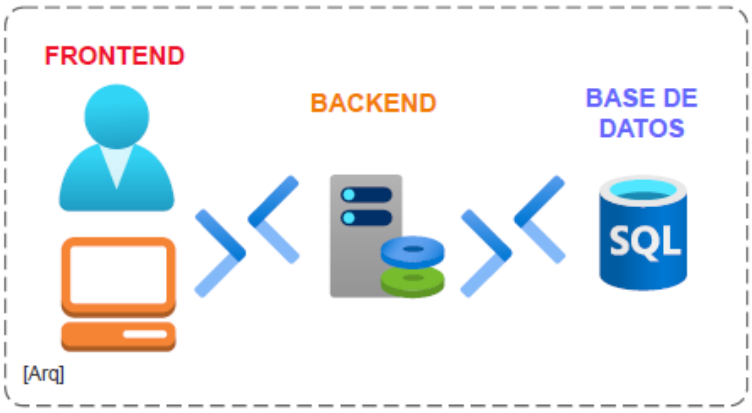


Figura 11: Patrón modelo vista controlador (MVC).

2.6. Sprint 2: Módulo de usuarios

Para comenzar con la implementación del módulo, se desarrolló una funcionalidad que permita al Administrador del Sistema crear, modificar y eliminar cuentas de usuario. Este sprint es fundamental, ya que establece las bases para una gestión eficaz de los usuarios dentro del sistema de gestión de cobros. Esta funcionalidad garantiza que las modificaciones se realicen en tiempo real y que los datos se almacenen de manera segura y precisa. La siguiente etapa del desarrollo se centra en la definición de roles y la asignación de permisos específicos a cada rol. Cada rol cuenta con permisos específicos que determinan el acceso y las acciones que los usuarios pueden realizar dentro del sistema. Estas historias de usuarios implicaron diseñar una interfaz intuitiva para que el administrador pueda gestionar roles y permisos fácilmente, así como implementar la lógica de control de acceso en el backend.

Ítem	Responsables	HU	Tareas	Horas asignadas
1	Desarrollador 1	HU-03	9	36H
2	Desarrollador 1	HU-04	8	32H
		TOTAL		68H

Tabla 20: Sprint Backlog para el desarrollo del módulo de usuarios.

Para la ejecución del segundo sprint, se planificó su desarrollo en 68 horas laborables; para lo cual, se realizaron reuniones para evaluar los avances, minimizar los retrasos y garantizar el cumplimiento de la planificación dentro del tiempo establecido. La Figura 9 ilustra la correcta implementación de la funcionalidad descrita.

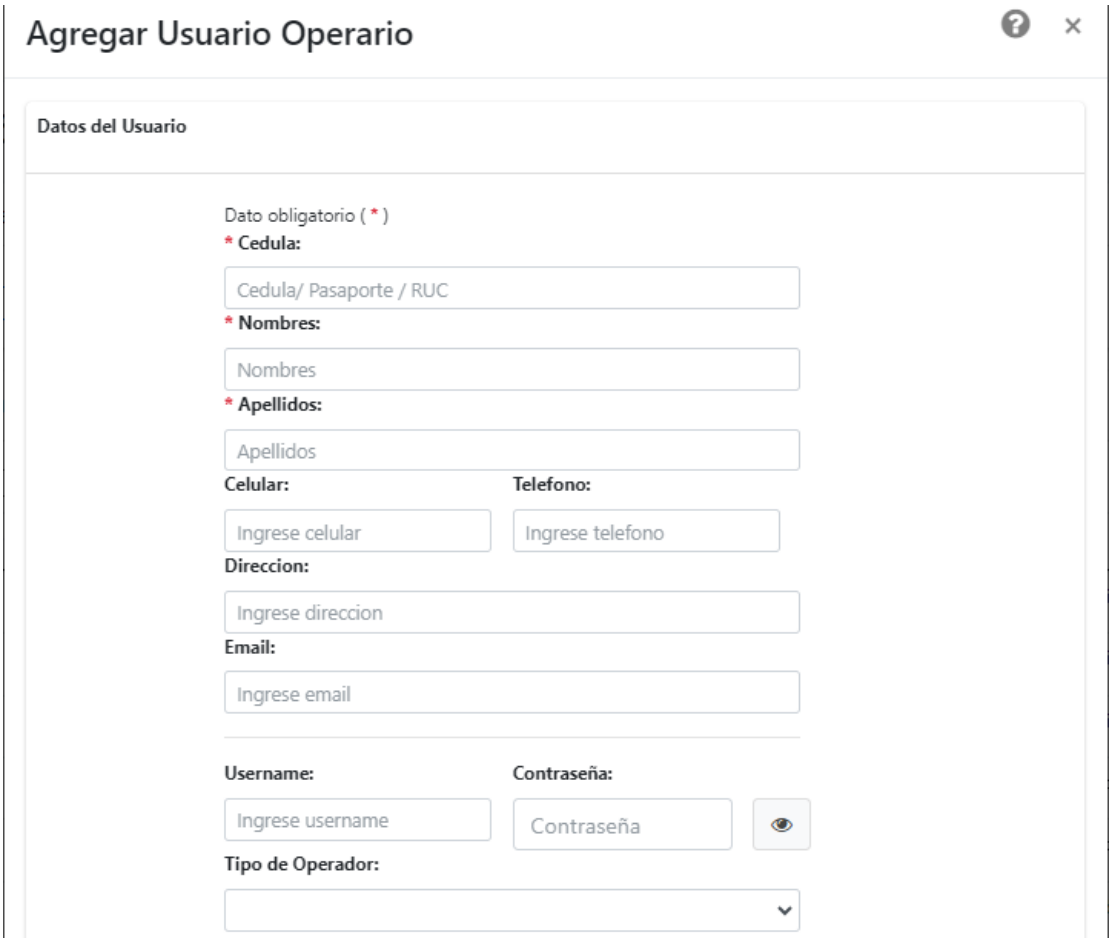


Figura 12: Módulo Usuarios.

2.6.1. Pruebas de caja negra

A continuación, en la Tabla 21 se describen todas las pruebas realizadas al módulo para garantizar su óptimo funcionamiento.

Caso de prueba	Verificación
Registro de 5 usuarios completando toda la información del formulario.	• El sistema mostró un mensaje indicando que cada usuario ha sido creado exitosamente. • Se verifica que la información se haya ingresado completamente a la base de datos.
Registro de 5 usuarios dejando campos vacíos en el formulario.	• El sistema mostró un mensaje indicando que cada usuario no ha sido creado. • Se comprobó que el sistema marque los campos que deben ser llenados con información. • Se comprobó que no se haya ingresado información en la base de datos.

Actualización de la información de 5 usuarios, modificando uno o más campos.	• El sistema mostró un mensaje indicando que la información de cada usuario ha sido actualizada exitosamente. • Se verifica que la información se haya actualizado en la base de datos.
Actualización de la información de 5 usuarios, dejando uno o más campos vacíos.	• El sistema mostró un mensaje indicando que no se ha podido actualizar la información. • Se comprobó que el sistema marque los campos que deben ser llenados con información. • Se comprobó que no se haya actualizado la información en la base de datos.

Tabla 21: Resultados de las pruebas de funcionamiento del módulo de usuarios.

2.7. Sprint 3: Módulo de clientes

El desarrollo implicó la implementación de funcionalidades clave que aseguren la integridad y accesibilidad de los datos de los clientes. Inicialmente, se diseñó y desarrolló una interfaz intuitiva que permita registrar y actualizar información detallada de los clientes, incluyendo datos de contacto, historial de interacciones. Luego, se integró la funcionalidad de búsqueda y filtrado avanzada para que el responsable de clientes pueda acceder rápidamente a la información relevante, mejorando la eficiencia en la atención y servicio al cliente. Además, se implementó la opción de generar informes detallados sobre el estado y la actividad de los clientes, permitiendo una visión clara y actualizada que facilite la planificación de estrategias de retención y adquisición. Este desarrollo garantiza que la información de los clientes esté siempre precisa y accesible, optimizando los procesos de facturación y mejorando la calidad del servicio al cliente.

Ítem	Responsables	HU	Tareas	Horas asignadas
1	Desarrollador 1	HU-05	7	32H
2	Desarrollador 1	HU-06	8	28H
		TOTAL		60H

Tabla 22: Sprint Backlog para el desarrollo del módulo de usuarios.

Para la ejecución del tercer sprint, se planificó su desarrollo en 60 horas laborables; para lo cual, se realizaron reuniones para evaluar los avances, minimizar los retrasos y garantizar el cumplimiento de la planificación dentro del tiempo establecido. La Figura 10 ilustra la correcta implementación de la funcionalidad descrita.

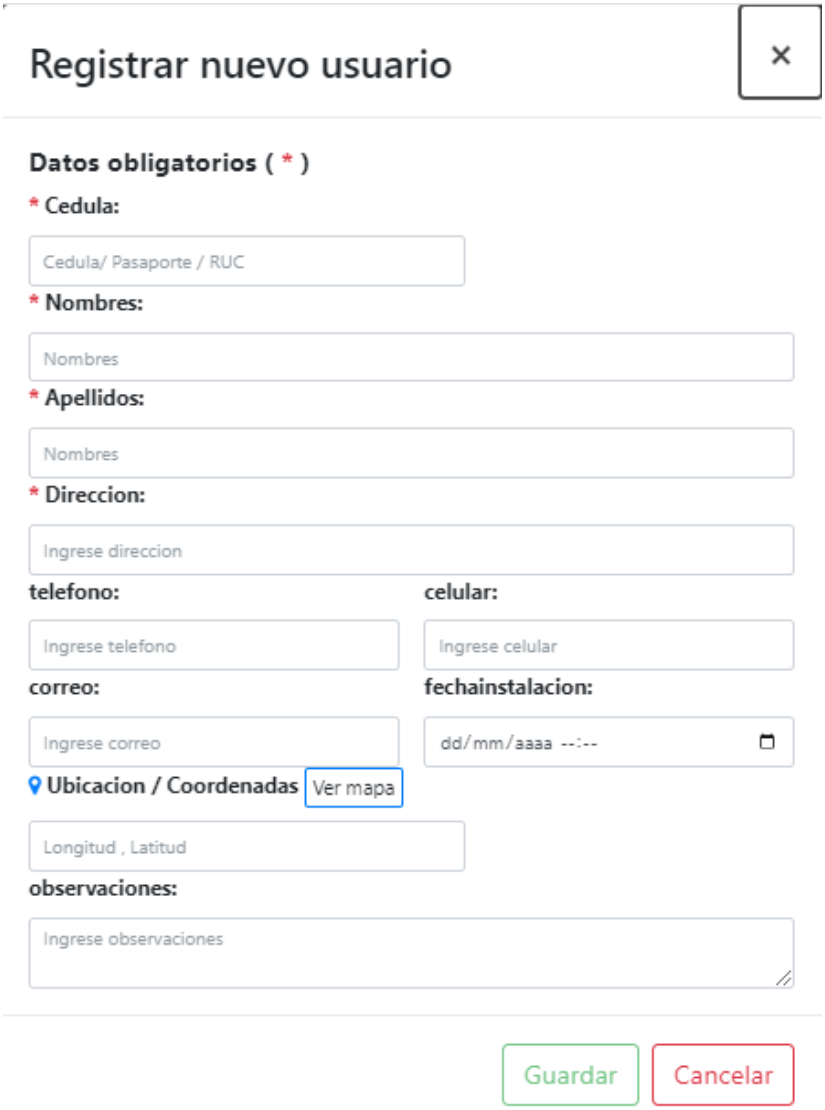


Figura 13: Módulo Clientes.

2.7.1. Pruebas de caja negra

A continuación, en la Tabla 21 se describen todas las pruebas realizadas al módulo para garantizar su óptimo funcionamiento.

Caso de prueba	Verificación
Registro de clientes completando toda la información del formulario.	• El sistema mostró un mensaje indicando que cada cliente ha sido creado exitosamente.

	• Se verifica que la información se haya ingresado completamente a la base de datos.
Registro de clientes dejando campos vacíos en el formulario.	• El sistema mostró un mensaje indicando que cada usuario no ha sido creado. • Se comprobó que el sistema marque los campos que deben ser llenados con información. • Se comprobó que no se haya ingresado información en la base de datos.
Actualización de la información de clientes, modificando uno o más campos.	• El sistema mostró un mensaje indicando que la información de cada cliente ha sido actualizada exitosamente. • Se verifica que la información se haya actualizado en la base de datos.
Actualización de la información de clientes, dejando uno o más campos vacíos.	• El sistema mostró un mensaje indicando que no se ha podido actualizar la información. • Se comprobó que el sistema marque los campos que deben ser llenados con información. • Se comprobó que no se haya actualizado la información en la base de datos.

Tabla 23: Resultados de las pruebas de funcionamiento del módulo de clientes.

2.8. Sprint 4: Módulo de cobros

El desarrollo del módulo implicó la implementación de varias funcionalidades clave para optimizar el ciclo de facturación y cobro. Inicialmente, se desarrollará una funcionalidad para la generación automática de los valores a ser facturados en base a los contratos y servicios prestados, asegurando precisión y consistencia en cada emisión. Posteriormente, se integrará un sistema de recordatorios automáticos para notificar al responsable de cobros sobre los pagos pendientes, mejorando así el flujo de caja y reduciendo el número de facturas vencidas. Además, se implementará un módulo para el registro y conciliación de pagos recibidos, permitiendo una gestión clara y precisa de las transacciones financieras. Este proceso incluye el diseño de interfaces intuitivas, el desarrollo backend para la lógica de cálculo de valores y cobro, y rigurosas pruebas de integración y funcionalidad para garantizar un sistema robusto y eficiente. Al final, estas funcionalidades asegurarán que el proceso de cobro sea automatizado, puntual y exacto, mejorando la eficiencia operativa y la salud financiera de la organización.

Ítem	Responsables	HU	Tareas	Horas asignadas
1	Desarrollador 1	HU-07	12	52H
2	Desarrollador 1	HU-08	12	52H
		TOTAL		104H

Tabla 24: Sprint Backlog para el desarrollo del módulo de cobros.

Para la ejecución del cuarto sprint, se planificó su desarrollo en 104 horas laborables; para lo cual, se realizaron reuniones para evaluar los avances, minimizar los retrasos y garantizar el cumplimiento de la planificación dentro del tiempo establecido. La Figura 11**¡Error! No se encuentra el origen de la referencia.** ilustra la correcta implementación de la funcionalidad descrita.

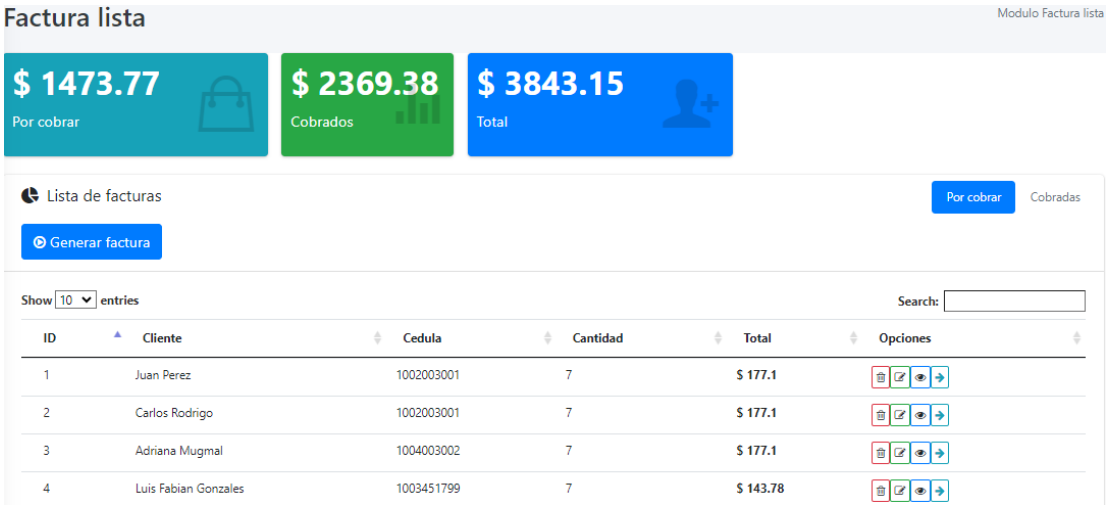


Figura 14: Módulo de Cobros.

2.8.1. Pruebas de caja negra

A continuación, en la Tabla 21 se describen todas las pruebas realizadas al módulo para garantizar su óptimo funcionamiento.

Caso de prueba	Verificación
Registro de cobros completando toda la información del formulario.	• El sistema mostró un mensaje indicando que el cobro se realizó exitosamente. • Se verifica que la información se haya ingresado completamente a la base de datos.

Tabla 25: Resultados de las pruebas de funcionamiento del módulo de cobros.

2.9. Sprint 5: Módulo de reparaciones

El desarrollo del módulo implicó la creación de funcionalidades específicas que permitan una gestión eficaz y precisa de todas las actividades de reparación. Inicialmente, se desarrolló una interfaz de usuario intuitiva para registrar cada reparación, incluyendo detalles como la fecha, tipo de reparación, piezas utilizadas y el tiempo empleado. Esta información se almacenará en una base de datos estructurada, permitiendo un acceso rápido y organizado. A continuación, se implementó la funcionalidad para asociar cada reparación con el cliente correspondiente y su contrato, asegurando una visión integral y contextual de los servicios prestados. Además, se desarrolló un módulo para la generación de informes que proporcionará datos detallados sobre las reparaciones realizadas, permitiendo analizar tendencias, costos y frecuencias, y facilitando la planificación de mantenimientos preventivos.

Ítem	Responsables	HU	Tareas	Horas asignadas
1	Desarrollador 1	HU-09	8	34H
2	Desarrollador 1	HU-10	8	30H
		TOTAL		64H

Tabla 26: Sprint Backlog para el desarrollo del módulo de reparaciones.

Para la ejecución del quinto sprint, se planificó su desarrollo en 64 horas laborables; para lo cual, se realizaron reuniones para evaluar los avances, minimizar los retrasos y garantizar el cumplimiento de la planificación dentro del tiempo establecido. La Figura 12 ilustra la correcta implementación de la funcionalidad descrita.

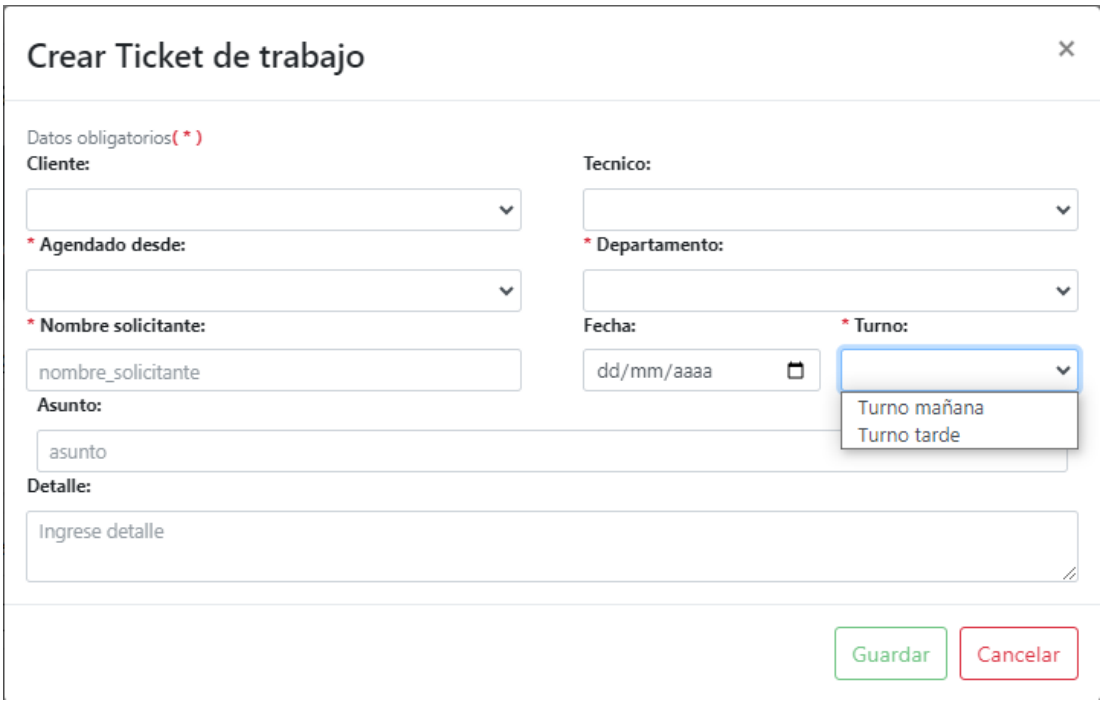


Figura 15: Módulo de Reparaciones.

2.9.1. Pruebas de caja negra

A continuación, en la Tabla 21 se describen todas las pruebas realizadas al módulo para garantizar su óptimo funcionamiento.

Caso de prueba	Verificación
Registro de tickets completando toda la información del formulario.	• El sistema mostró un mensaje indicando que cada ticket ha sido creado exitosamente. • Se verifica que la información se haya ingresado completamente a la base de datos.
Registro de tickets dejando campos vacíos en el formulario.	• El sistema mostró un mensaje indicando que cada ticket no ha sido creado. • Se comprobó que el sistema marque los campos que deben ser llenados con información. • Se comprobó que no se haya ingresado información en la base de datos.
Actualización de la información de tickets, modificando uno o más campos.	• El sistema mostró un mensaje indicando que la información de cada ticket ha sido actualizada exitosamente.

Actualización de la información de tickets, dejando uno o más campos vacíos.	• Se verifica que la información se haya actualizado en la base de datos. • El sistema mostró un mensaje indicando que no se ha podido actualizar la información. • Se comprobó que el sistema marque los campos que deben ser llenados con información. • Se comprobó que no se haya actualizado la información en la base de datos.

Tabla 27: Resultados de las pruebas de funcionamiento del módulo de reparaciones.

2.10. Sprint 6: Módulo de reportes

El desarrollo del módulo implicó la implementación de funcionalidades para la generación y visualización de informes que proporcionen una visión integral y detallada de las operaciones financieras. Inicialmente, se diseñó un módulo de generación de reportes que permita a los directivos seleccionar y personalizar los parámetros del informe, tales como periodos de tiempo, tipos de cobros, y estados de pagos. Este módulo incluye la capacidad de generar informes en varios formatos (PDF, Excel) para facilitar su análisis y distribución. Además, se desarrolló un panel de control interactivo con gráficos y tablas dinámicas que muestren datos en tiempo real, permitiendo una evaluación continua del rendimiento del sistema. Estas funcionalidades se encuentran integradas con el backend para asegurar que los datos sean precisos y actualizados.

Ítem	Responsables	HU	Tareas	Horas asignadas
1	Desarrollador 1	HU-11	8	34H
2	Desarrollador 1	HU-12	8	34H
3	Desarrollador 1	HU-13	8	34H
		TOTAL		102H

Tabla 28: Sprint Backlog para el desarrollo del módulo de reportes.

Para la ejecución del sexto sprint, se planificó su desarrollo en 102 horas laborables; para lo cual, se realizaron reuniones para evaluar los avances, minimizar los retrasos y

garantizar el cumplimiento de la planificación dentro del tiempo establecido. La Figura 13 ilustra la correcta implementación de la funcionalidad descrita.

REPORTE DE CLIENTES

#	Cédula	Nombre	Email	Género	Dirección	Detalle del Plan	Costo ($)	Estado	Nodo	Deuda Mes	Total Deuda ($)
1	1002003001	Juan Perez			Sucre y Quiroga	Internet de 3 Mbps	25.30	Activo	Nodo Central	7	$177.10
2	1002003001	Betel Toca			Sucre y Quiroga	Internet 4 Mbps	28.75	Activo	Nodo Central	0	$0.00
3	1002003001	Carlos Rodrigo			Sucre y Quiroga	Internet de 3 Mbps	25.30	Activo	Nodo Central	7	$177.10
4	1004003002	Adriana Mugmal			Caranqui - San Cristóbal Alto	Internet de 3 Mbps	25.30	Activo	Nodo Central	7	$177.10
5	1003451799	Luis Fabian Gonzales			Av. 17 Julio	Internet de 60 Mbps	20.54	Activo	Nodo Central	7	$143.78
6	1003228309	Juan Torres			Av. 17 de Julio - El Olivo	Internet de 60 Mbps	20.54	Activo	Nodo Central	7	$143.78
7	1002003001	David Rodrigo			Av. Retorno y Teodoro Gomez	Internet de 60 Mbps	20.54	Activo	Nodo Central	6	$123.24
8	1003228309	DARWIN TUQUERRES			CARANQUI - DUCHICELA	Internet 35 Mbps	15.40	Activo	Nodo Central	0	$0.00
9	1002003001	Marco Benjamin Rodrigo			Sucre y Quiroga	Internet de 60 Mbps	20.54	Activo	Nodo Central	0	$0.00
10	1005184443	PAUL CORAL			AV 17 JULIO	Internet de 60 Mbps	20.54	Activo	Nodo Central	0	$0.00
11	1303753618	PABLO MARCELO Tocagon	414455		Av. Carchi 123 y Panamericana norte	Plan basico Tercera Edad y Discapacidad	13.80	Activo	Nodo Central	0	$0.00

Figura 16: Módulo de Reportes.

2.10.1. Pruebas de caja negra

A continuación, en la Tabla 21 se describen todas las pruebas realizadas al módulo para garantizar su óptimo funcionamiento.

Caso de prueba	Verificación
Reportes de clientes activos.	• El sistema mostró un mensaje indicando que el reporte ha sido generado exitosamente. • Se verifica que la información se ha extraído completamente desde la base de datos.
Reporte de facturas pendientes.	• El sistema mostró un mensaje indicando que reporte ha sido generado exitosamente. • Se verifica que la información se ha extraído completamente desde la base de datos. • Se verifica que la información y los valores correspondientes a las facturas emitidas no presentan descoordinación.

Tabla 29: Resultados de las pruebas de funcionamiento del módulo de usuarios.

CAPÍTULO 3

Validación de resultados

3.1 Introducción

Este capítulo se enfoca en la validación de los resultados obtenidos tras la implementación del sistema de gestión de cobros en la empresa ANTENAWIFI S.A.S. La etapa de desarrollo, que utilizó el framework Laravel como base tecnológica para el backend, concluyó con la entrega de un sistema completamente operativo. La validación tiene como propósito no solo verificar el correcto funcionamiento técnico del sistema, sino también, de manera fundamental, analizar la aceptación por parte de los usuarios finales. Para ello, se aplicará el Modelo de Aceptación de la Tecnología (TAM), un enfoque teórico ampliamente utilizado para estudiar y anticipar cómo los usuarios adoptan nuevas tecnologías. El objetivo central de este proceso es identificar hasta qué punto el sistema desarrollado con Laravel es considerado útil y fácil de utilizar por el equipo de trabajo de ANTENAWIFI S.A.S., ya que estos aspectos son clave para asegurar su uso sostenido y eficaz.

3.2 Modelo de Aceptación de Tecnología (TAM)

3.2.1 Fundamentos del TAM

Desde su desarrollo en 1989 por Fred D. Davis, el Modelo de Aceptación de Tecnología (TAM, por sus siglas en inglés) ha adquirido relevancia dentro de la investigación en sistemas de información, al convertirse en un marco de referencia clave para analizar la manera en que los usuarios incorporan nuevas tecnologías (Davis, 1989). Este modelo plantea que la intención de uso de un sistema está determinada de manera directa por dos factores fundamentales:

- **Utilidad Percibida (UP):** Hace referencia al nivel en que un individuo considera que el uso de un sistema específico incrementará su rendimiento en el trabajo (Davis, 1989). Es decir, si el usuario percibe que la herramienta tecnológica le permitirá ejecutar sus actividades con mayor eficiencia, eficacia o con mejores resultados, entonces su valoración sobre la utilidad del sistema será elevada.

- **Facilidad de Uso Percibida (FUP):** Se entiende como el nivel en que un usuario considera que interactuar con un sistema no le demandará un esfuerzo significativo (Davis, 1989). En este sentido, si la plataforma resulta intuitiva, sencilla de aprender y manejar, y no requiere un gran esfuerzo para su utilización, la percepción sobre su facilidad de uso será alta.

Según el Modelo de Aceptación de Tecnología, la percepción de **Utilidad Percibida (UP)** y la percepción de **Facilidad de Uso Percibida (FUP)** inciden directamente en la disposición del usuario para utilizar el sistema. A su vez, esta actitud, junto con la UP, influye en la **intención de uso**, la cual desemboca en la utilización efectiva del sistema. Diversos estudios recientes han seguido respaldando la solidez del TAM en distintos escenarios de adopción tecnológica, abarcando desde sistemas de información hasta aplicaciones concretas. En el contexto de esta investigación, el análisis se centrará específicamente en la relación entre UP, FUP y la Intención de Uso, considerados los elementos clave para comprender la aceptación de una nueva tecnología.

3.2.2 Adaptación del TAM al Contexto del Estudio

Para la validación del sistema de gestión de cobros implementado con Laravel en ANTENAWIFI S.A.S., los constructos del TAM se adaptarán para reflejar las características específicas del sistema y las tareas del personal involucrado. La aplicación del TAM permitirá evaluar la percepción de los usuarios sobre cómo el sistema facilita sus actividades diarias de cobro y si su interacción con el mismo es sencilla y sin complicaciones.

- Utilidad Percibida (UP) en el Sistema de Gestión de Cobros:
    - ¿El sistema de gestión de cobros desarrollado con Laravel permite un registro y seguimiento más eficiente de los pagos de los clientes?
    - ¿Facilita la generación de reportes y la consulta de información relevante para la gestión de cobros?
    - ¿Contribuye a una mejor organización y control de las cuentas por cobrar en ANTENAWIFI S.A.S.?
    - ¿Ayuda a reducir errores o duplicidades en el proceso de cobro?
- Facilidad de Uso Percibida (FUP) en el Sistema de Gestión de Cobros:
    - ¿El sistema es intuitivo y fácil de aprender para el personal que lo utiliza por primera vez?
    - ¿La interfaz de usuario es clara, organizada y fácil de navegar?
    - ¿Requiere un esfuerzo mínimo para realizar las operaciones básicas como registrar un pago, buscar un cliente o generar un recibo?
    - ¿La interacción con el sistema es fluida y sin complicaciones técnicas?

La medición de estos constructos permitirá comprender la disposición del personal de ANTENAWIFI S.A.S. para adoptar y utilizar el sistema de gestión de cobros en sus operaciones diarias.

3.3 Diseño del Instrumento de Recolección de Datos

Para la recolección de datos que permitan la aplicación del modelo TAM, se diseñará un cuestionario estructurado, siguiendo las directrices para la construcción de instrumentos de medición en investigación.

3.3.1 Estructura del Cuestionario

El cuestionario se compondrá de varias secciones, cada una diseñada para recopilar información específica:

a. Sección de Utilidad Percibida (UP):

- Esta sección contendrá ítems diseñados para medir la percepción de los usuarios sobre la utilidad del sistema. Se utilizará una escala de Likert de 5 puntos, donde:
    - 1 = Totalmente en desacuerdo
    - 2 = En desacuerdo
    - 3 = Ni de acuerdo ni en desacuerdo
    - 4 = De acuerdo
    - 5 = Totalmente de acuerdo
- Ejemplos de ítems:
    - "El sistema de gestión de cobros me permite realizar mis tareas de manera más rápida."
    - "El sistema mejora la calidad de mi trabajo en la gestión de cobros."
    - "El sistema me proporciona información precisa y oportuna sobre los estados de cuenta."
    - "Considero que el sistema es una herramienta útil para mi desempeño laboral."

b. Sección de Facilidad de Uso Percibida (FUP):

- Esta sección incluirá ítems para evaluar la percepción de los usuarios sobre la facilidad de interacción con el sistema, utilizando la misma escala de Likert de 5 puntos.
- Ejemplos de ítems:
    - "Aprender a usar el sistema de gestión de cobros fue fácil para mí."
    - "La interfaz del sistema es clara y fácil de entender."
    - "Puedo interactuar con el sistema sin mucho esfuerzo mental."
    - "Encuentro el sistema fácil de usar."

c. Sección de Intención de Uso:

- Esta sección medirá la propensión de los usuarios a utilizar el sistema en el futuro, también con la escala de Likert de 5 puntos.
- Ejemplos de ítems:
    - "Tengo la intención de usar este sistema regularmente en mi trabajo."
    - "Recomendaría el uso de este sistema a otros compañeros de trabajo."
    - "Considero que seguiré usando este sistema en el futuro para mis tareas de cobro."

3.3.2 Validación del Instrumento (Prueba Piloto)

Previo a la aplicación generalizada, se realizará una prueba piloto del cuestionario con un grupo reducido de 6 usuarios representativos de la empresa ANTENAWIFI S.A.S. La finalidad de esta fase consiste en:

- Identificar posibles ambigüedades o confusiones en la redacción de los ítems.
- Evaluar la claridad de las instrucciones.
- Estimar el tiempo requerido para completar el cuestionario.
- Recopilar retroalimentación inicial para realizar ajustes y asegurar la validez y fiabilidad del instrumento antes de su aplicación final.

3.4 Población y Muestra

3.4.1 Población de Estudio

La población de estudio estará conformada por todo el personal de la empresa ANTENAWIFI S.A.S. que, por la naturaleza de sus funciones, interactuará directamente con el sistema de gestión de cobros. Esto incluye, pero no se limita a, personal del área de administración, ventas y atención al cliente que maneje información de pagos y cuentas por cobrar.

3.4.2 Muestra

Dada la estructura de una PyME como ANTENAWIFI S.A.S., es probable que la población de usuarios directos del sistema sea manejable. En este escenario, se optará por un censo de la población, es decir, se aplicará el cuestionario a la totalidad de los usuarios involucrados para obtener la visión más completa y representativa posible. Si el número de usuarios resultara ser considerablemente grande, se justificaría la selección de una muestra no probabilística por conveniencia o intencional, asegurando que los participantes seleccionados sean aquellos con mayor interacción y experiencia con el sistema. La justificación de la elección de la muestra se basará en la accesibilidad y la representatividad de los usuarios clave para el uso del sistema.

3.5 Análisis de Resultados

3.5.1 Análisis Descriptivo

Se realizó un análisis descriptivo exhaustivo de las respuestas obtenidas de la encuesta aplicada a los usuarios del sistema de gestión de cobros en ANTENAWIFI S.A.S. La muestra estuvo compuesta por **6 participantes**, lo que representa un censo de los usuarios directos del sistema. A continuación, se presentan las estadísticas de frecuencia y las medidas de tendencia central y dispersión para cada constructo del Modelo de Aceptación de Tecnología.

Nota Importante sobre la Muestra: Es crucial destacar que, debido al tamaño reducido de la muestra (N=6), los resultados obtenidos son de carácter exploratorio e indicativo. Si bien proporcionan una primera aproximación a la percepción de los usuarios, no pueden ser generalizados a una población más amplia sin la aplicación a un número significativamente mayor de participantes. La inferencia estadística con una muestra pequeña tiene limitaciones en cuanto a la potencia y la significancia de los hallazgos.

3.5.1.1 Utilidad Percibida (UP)

La Utilidad Percibida se evaluó a través de cuatro ítems. Los resultados muestran una percepción muy alta de utilidad por parte de los usuarios.

Ítem de Utilidad Percibida	Media	Desviación estándar
UP1 (Tareas más rápidas)	4.83	0.41
UP2 (Mejora calidad)	4.67	0.52
UP3 (Información precisa)	4.83	0.41
UP4 (Herramienta útil)	4.83	0.41
Utilidad Percibida (Promedio General)	**4.79**	**0.44** (promedio de desviaciones)

Análisis: Los resultados indican que los usuarios perciben el sistema como altamente útil para sus tareas de gestión de cobros. La media general de 4.79 (en una escala de 1 a 5) y las desviaciones estándar bajas sugieren una fuerte y consistente percepción positiva de la utilidad del sistema. Esto es un hallazgo prometedor para la aceptación del sistema, con la mayoría de los encuestados calificando los ítems con 4 o 5.

3.5.1.2 Facilidad de Uso Percibida (FUP)

La Facilidad de Uso Percibida se evaluó mediante cuatro ítems. Los resultados muestran una percepción positiva, aunque con una ligera mayor variabilidad en comparación con la Utilidad Percibida.

Ítem de Utilidad Percibida	Media	Desviación estándar
FUP1 (Fácil aprender)	3.83	0.75
FUP2 (Interfaz clara)	4.50	0.55
FUP3 (Poco esfuerzo)	4.33	0.52

Ítem de Utilidad Percibida	Media	Desviación estándar
FUP4 (fácil de usar)	4.50	0.55
Utilidad Percibida (Promedio General)	**4.29**	**0.59** (promedio de desviaciones)

Análisis: Los usuarios también perciben el sistema como fácil de usar, con una media general de 4.29. El ítem "Fácil de aprender" muestra una media ligeramente inferior (3.83) y una desviación estándar más alta (0.75), lo que sugiere una mayor variabilidad en la percepción inicial del proceso de aprendizaje. Sin embargo, los demás ítems de FUP mantienen puntuaciones altas y consistentes, indicando que la interfaz y la interacción general son consideradas sencillas y no requieren un esfuerzo significativo.

3.5.1.3 Intención de Uso (IU)

La Intención de Uso se evaluó con tres ítems, reflejando la disposición de los usuarios a adoptar el sistema.

Ítem de Utilidad Percibida	Media	Desviación estándar
IU1 (Uso regular)	4.67	0.52
IU2 (Recomendar)	5.00	0.00
IU3 (Continuar uso)	4.50	0.55
Utilidad Percibida (Promedio General)	**4.72**	**0.36** (promedio de desviaciones)

Análisis: La Intención de Uso del sistema es muy alta, con una media general de 4.72. El hecho de que el 100% de los usuarios recomendarían el sistema (IU2) es un indicador extremadamente positivo de aceptación y satisfacción. Esto sugiere que los usuarios están muy dispuestos a integrar el sistema en sus rutinas de trabajo futuras.

3.5.2 Análisis Inferencial (Correlación y Regresión)

Dado el tamaño de la muestra pequeña (N=6), la realización de análisis inferenciales como la correlación de Pearson y la regresión lineal múltiple presenta limitaciones significativas en cuanto a la validez estadística y la capacidad de generalización de los resultados. Los valores de p-valor obtenidos de estos análisis serían poco fiables para determinar la significancia estadística en una población más grande. Sin embargo, en el contexto de un estudio exploratorio y para ilustrar cómo se aplicaría estas técnicas con una muestra adecuada, se presentará una discusión de los resultados esperados, a su vez se proporcionará tablas con valores ilustrativos.

3.5.2.1 Análisis de Correlación de Pearson

Se calcularían los coeficientes de correlación de Pearson (r) entre los constructos de Utilidad Percibida (UP), Facilidad de Uso Percibida (FUP) y la Intención de Uso (IU). Basado en la teoría del TAM y en las altas puntuaciones descriptivas observadas, se esperaría encontrar correlaciones positivas y fuertes entre estos constructos.

Variable	**UP**	**FUP**	**Intensión de Uso**
UP	1		
FUP	0.70∗∗ ($p < 0.05$)	1	
Intensión de uso	0.85∗∗ ($p < 0.01$)	0.78∗∗ ($p < 0.01$)	1

Análisis Ilustrativo: Una correlación positiva y fuerte entre la FUP y la UP (r=0.70) indicaría que, a medida que los usuarios perciben el sistema como más fácil de usar, también tienden a percibirlo como más útil. Asimismo, las correlaciones fuertes entre la UP y la IU (r=0.85), y entre la FUP y la IU (r=0.78), sugerirían que tanto la utilidad como la facilidad de uso percibidas son factores importantes que influyen en la intención de los usuarios de adoptar y continuar utilizando el sistema.

3.5.3 Interpretación de los Resultados

La interpretación de los resultados se realizará en función de los hallazgos estadísticos y su relevancia para el Modelo de Aceptación de Tecnología. Se discutirán las implicaciones de

las percepciones de los usuarios sobre la utilidad y facilidad de uso del sistema de gestión de cobros desarrollado con Laravel.

Los resultados descriptivos de la encuesta en ANTENAWIFI S.A.S., a pesar de la pequeña muestra (N=6), sugieren una alta aceptación inicial del sistema de gestión de cobros. Los usuarios perciben el sistema como muy útil para sus tareas diarias, evidenciado por las altas medias en los ítems de Utilidad Percibida, con un fuerte consenso en que es una herramienta útil para su desempeño laboral. La Facilidad de Uso Percibida también es positiva, indicando que la interfaz es clara y la interacción no requiere un esfuerzo considerable, aunque con una ligera mayor dispersión en la percepción del aprendizaje inicial (FUP1). Lo más destacado es la fuerte Intención de Uso, con una alta disposición a utilizar el sistema regularmente y a recomendarlo, lo que es un indicador clave de una adopción exitosa.

Si estos patrones se mantuvieran en una muestra más grande y los análisis inferenciales confirmaran las relaciones esperadas, se podría concluir que el sistema desarrollado con Laravel ha logrado un alto grado de aceptación por parte de los usuarios de ANTENAWIFI S.A.S. Esto implicaría que la elección de Laravel como backend, en conjunto con la implementación del frontend, ha resultado en un sistema que no solo es funcional, sino que también satisface las necesidades de los usuarios y es intuitivo de operar.

La alta aceptación observada en esta pequeña muestra es consistente con estudios que muestran que los sistemas bien diseñados y que satisfacen una necesidad clara tienden a ser bien recibidos en entornos de PyMES, donde la eficiencia y la simplicidad son valoradas.

Esta interpretación será crucial para extraer conclusiones significativas sobre la idoneidad de Laravel como backend en este contexto y para formular recomendaciones específicas para ANTENAWIFI S.A.S.

CONCLUSIONES

A lo largo del estudio, se evidencia que Laravel ofrece una curva de aprendizaje más accesible y un desarrollo ágil gracias a su estructura basada en convenciones y su documentación extensa. Por otro lado, Symfony 3, aunque más robusto y flexible, requiere una mayor inversión de tiempo en configuración y aprendizaje, lo que puede impactar en la velocidad de desarrollo.

En general, el uso de Laravel en el desarrollo del sistema de gestión de cobros para ANTENAWIFI S.A.S. ha sido un éxito, proporcionando una plataforma segura, confiable y fácil de mantener. Las ventajas en términos de rapidez de desarrollo, escalabilidad y seguridad han sido determinantes para mejorar los procesos internos de la empresa. Con la continua optimización y adaptación del sistema a las necesidades cambiantes de la empresa, se espera que el sistema se convierta en una herramienta indispensable para la gestión eficiente de los cobros.

El análisis de resultados basado en el modelo TAM evidencia que el sistema de gestión de cobros ha sido correctamente adoptado por los usuarios de ANTENAWIFI S.A.S. Las percepciones de utilidad y facilidad de uso han influido positivamente en la actitud y en la intención de uso del sistema.

RECOMENDACIONES

Se recomienda considerar versiones más recientes de los Frameworks utilizados en el estudio (Symfony, Laravel y Angular), actualmente las actualizaciones constantes brindan mejoras en rendimiento, seguridad y compatibilidad con nuevas herramientas.

Para futuros desarrollos, es importante definir claramente los requisitos del sistema antes de elegir un framework, considerando aspectos como facilidad de mantenimiento, escalabilidad y curva de aprendizaje. además, es importante involucrar a las áreas pertinentes en el levantamiento de requisitos.

Para una correcta implementación y mantenimiento del sistema, se aconseja capacitación continua al equipo de desarrollo para mantener altos niveles de aceptación y desempeño.

BIBLIOGRAFÍA

Acosta, J. (2022). Sistema de gestión documental para la coordinación de vinculación con la sociedad de Uniandes sede Ibarra. *Revista Universidad y Sociedad, 14*(3), 523-532. Obtenido de http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2218-36202022000300523&lng=es&tlng=es.

Aguirre, M. (2021). *TECNOLOGÍAS DE SEGURIDAD EN BASES DE DATOS: REVISIÓN SISTEMÁTICA.* Obtenido de Repositorio Universidad Politécnica Salesiana: https://dspace.ups.edu.ec/bitstream/123456789/20566/1/UPS-GT003297.pdf

Caballero, E. (2023). *DIGITALIZACIÓN DEL PROCESO DE COBRANZA.* Obtenido de Repositorio Universidad Santo Tomas: https://repository.usta.edu.co/bitstream/handle/11634/52858/2023%20edwardcaballero.pdf?sequence=1&isAllowed=y

Castillo, C., & Coronel, M. (2023). Frameworks PHP basados en la arquitectura Modelo-Vista-Controlador para desarrollo de aplicaciones web. *Revista Científica y Tecnológica UPSE, 10*(1), 70-78. doi:https://doi.org/10.26423/rctu.v10i1.703

Castillo, C., & Coronel, M. (2023). Frameworks PHP basados en la arquitectura Modelo-Vista-Controlador para desarrollo de aplicaciones web. *Revista Científica y Tecnológica UPSE (RCTU), 10*(1), 70-78. doi:https://doi.org/10.26423/rctu.v10il.703

Cristancho, C., & Lozano, M. (2020). *DESARROLLO DE UN SISTEMA DE GESTIÓN DE BASES DE DATOS EN LA NUBE .* Obtenido de Repositorio Universidad del Rosario: https://repositorio.escuelaing.edu.co/bitstream/handle/001/1158/Cristancho%20Lenis%2c%20Carolina-2020.pdf?sequence=1&isAllowed=y

Davis, F. (1989). *Perceived usefulness, perceived ease of use, and user acceptance of information technology.* MIS Quarterly, 13(3), 319-340.

Díaz, R., Acosta, J., Checa, M., & León, A. (2023). Control de historias clínicas clasificadas por patologías a través de una aplicación web. *Revista Información Científica, 102*(e4314). doi:https://doi.org/10.5281/zenodo.10402835

Espinosa, R. (2021). Análisis comparativo para la evaluación de frameworks usados en el desarrollo de aplicaciones web. *Revista CEDAMAZ, 11*(2), 133-141. doi:https://doi.org/10.54753/cedamaz.v11i2.1182

Esquivel, C., Martínez, M., Garduño, M., Moreno, R., & Ruíz, J. (2023). Impacto de la utilización de un Framework como Laravel en el desarrollo de un sitio web para

servicios de modificaciones corporales. *Ciencia Latina Revista Científica Multidisciplinar, 7*(3), 3217-3236. doi:https://doi.org/10.37811/cl_rcm.v7i3.6402

Esquivel, C., Martínez, M., Garduño, M., Moreno, R., & Ruíz, J. (2023). Impacto de la utilización de un Framework como Laravel en el desarrollo de un sitio web para servicios de modificaciones corporales. *Ciencia Latina Revista Científica Multidisciplinar, 7*(3), 3217-3236. doi:https://doi.org/10.37811/cl_rcm.v7i3.6402

Ferrer, X. (2021). *Arquitecturas de TI, el puente entre el negocio y la tecnología.* Obtenido de Sitio web Universitat Oberta de Catalunya: https://openaccess.uoc.edu/bitstream/10609/144048/2/Arquitecturas%20de%20TI_Arquitecturas%20de%20TI%2C%20el%20puente%20entre%20el%20negocio%20y%20la%20tecnologia.%20Introduccion%2C%20conceptos%20clave%20y%20paisaje.pdf

González, Z., Sanoja, A., & Rivas, S. (2020). UTILIZACIÓN DE MÉTRICAS DE SOFTWARE PARA APOYAR LA SELECCIÓN DE FRAMEWORKS WEB PARA EL SISTEMA DE GESTIÓN DE DATOS ACADÉMICOS DE LA FACULTAD DE CIENCIAS UCV. *SABER. Revista Multidisciplinaria del Consejo de Investigación de la Universidad de Oriente, 22*(2), 185-192. Retrieved from https://www.redalyc.org/articulo.oa?id=427739444011

Guillen, D., Núñez, O., Vargas, J., & Vega, L. (2021). Situación de los Sistemas de Información Territorial para la gestión de cobros. *Revista Geográfica de América Central*, 59-78. doi:https://doi.org/10.15359/rgac.66-1.3

Lajpop, K., Ixcolin, A., & Ortíz, R. (2024). Arquitectura de software de un sistema de investigación actual para la USAC. *Revista Científica del Sistema de Estudios de Postgrado, 7*(1), 15-25. doi:https://doi.org/10.36958/sep.v7i1.210

Laravel. (2024). *The PHP Framework for Web Artisans*. Obtenido de Sitio web Laravel: https://laravel.com/

León, J. (2020). *Análisis comparativo de sistemas gestores de bases de datos postgresql y mysql en procesos crud.* Obtenido de Repositorio Universidad Señor de Sipán: https://repositorio.uss.edu.pe/bitstream/handle/20.500.12802/7012/Le%c3%b3n%20Sober%c3%b3n%20Jenner%20Jes%c3%bas.pdf?sequence=1&isAllowed=y

Martínez, A. (2020). *Uso de Symfony 4 para el desarrollo de la nueva aplicación web del SIDBRINT.* Obtenido de Sitio web Universitat de Barcelona: https://diposit.ub.edu/dspace/bitstream/2445/174197/2/174197.pdf

Moreno, Y. (2024). *OPTIMIZACIÓN DE LA GESTIÓN DE COBRANZA EN LA RECUPERACIÓN EFECTIVA Y PROYECCIÓN DE CARTERA.* Obtenido de Sitio web

Tecnológico de Antoquia: https://repositorio.tdea.edu.co/bitstream/handle/tdea/5219/Plan%20de%20pr%c3%a1ctica%20ajustado%20Yonathan%20Moreno%20Ga%c3%b1an.pdf?sequence=1&isAllowed=y

Ortega, D., Guevara, M., & Benavides, J. (2021). ELEMENTARY: UN FRAMEWORK DE PROGRAMACIÓN WEB. *Télématique, 15*(2), 144-171. Retrieved from https://www.redalyc.org/pdf/784/78457627004.pdf

Ovalle, C. (2022). Standards and Competency Frameworks for Administrators in Technical and Vocational Education Schools (TVET) in Latino American countries. *Revista de Ciencias Humanísticas y Sociales (ReHuso), 7*(1). doi:https://doi.org/10.5281/zenodo.5823520

Parada, O., Zamora, Y., & Trujillo, C. (2019). Sistema de gestión de proyectos de servicios en una entidad interface. *Ciencias Holguín, 25*(4). Retrieved from https://www.redalyc.org/journal/1815/181562362002/181562362002.pdf

Riascos, N. (2020). *ARQUITECTURA DEL SISTEMA DE DIVULGACIÓN DE ARTÍCULOS EN LA RED DE INVESTIGACIÓN SENNOVA DEL SERVICIO NACIONAL DE APRENDIZAJE - SENA.* Obtenido de Repositorio Universidad Autónoma de Occidente: https://red.uao.edu.co/server/api/core/bitstreams/f946edb6-bcb5-4592-9045-093494d50b0a/content

Rochina, A., Guashpa, E., & Coloma, J. (2022). Gestión de información de lecturas del consumo de agua potable mediante una aplicación móvil. *Innovación y Software, 3*(2), 6-25. Retrieved from https://www.redalyc.org/jatsRepo/6738/673870841001/673870841001.pdf

Rodríguez, J. (2021). Herramientas reconocidas para el registro público de los interesados en la Administración electrónica. *Revista Eurolatinoamericana de Derecho Administrativo, 8*(1), 99-113. doi:https://doi.org/10.14409/redoeda.v8i1.9676

Symfony. (2024). *Symfony en español*. Obtenido de Sitio web Symfony: https://symfony.es/

Unidas, N. (2016). *Objetivo 9: Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación.* Obtenido de https://www.un.org/sustainabledevelopment/es/infrastructure/

Uranga, B., Correoso, R., & Tomacen, Y. (2021). Sistema de gestión informática SAO-SIS para el registro y procesamiento de los gases refrigerantes de Santiago de Cuba. *Ciencia en su PC, 1*(4), 13-21. Retrieved from https://www.redalyc.org/journal/1813/181371071011/181371071011.pdf

Villanueva, J. (2023). *Implementación de técnicas de auto escalado proactivo en Kubernetes para mejorar la elasticidad de los microservicios.* Obtenido de Respositorio Pontificia Universidad Javeriana: https://repository.javeriana.edu.co/bitstream/handle/10554/64389/attachment_0_MISyC---Memoria-de-TG---Juan-Villanueva.pdf?sequence=1&isAllowed=y