UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Software

Diseñar un plan de pruebas de integración basado en el estándar ISO/IEC/IEEE 29119 para aplicar en la automatización de procesos de la empresa Reparauto

Trabajo de grado previo a la obtención del título de Ingeniero de Software presentado ante la ilustre Universidad Técnica del Norte.

Autor:

Zamora Moreano Edison Augusto

Director:

MSc. Xavier Mauricio Rea Peñafiel. Ing.

Ibarra – Ecuador

2024

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:	1004786578		
APELLIDOS Y NOMBRES:	ZAMORA MOREANO EDISON AUGUSTO		
DIRECCIÓN:	IBARRA, MATILDE PASQUEL MONJE 1-21 Y ROSA PAREDES		
EMAIL:	eazamoram@utn.edu.ec		
TELÉFONO FIJO:		**TELÉFONO MÓVIL:**	0987396716

DATOS DE LA OBRA	
TÍTULO:	DISEÑAR UN PLAN DE PRUEBAS DE INTEGRACIÓN BASADO EN EL ESTÁNDAR ISO/IEC/IEEE 29119 PARA APLICAR EN LA AUTOMATIZACIÓN DE PROCESOS DE LA EMPRESA REPARAUTO
AUTOR(ES):	EDISON AUGUSTO ZAMORA MOREANO
FECHA:	06/03/2024
PROGRAMA:	PREGRADO
TÍTULO POR EL QUE OPTA:	INGENIERO DE SOFTWARE
DIRECTOR:	MSc. Mauricio Rea
ASESOR 1:	PhD. Cathy Guevara

2. CONSTANCIAS

El autor (es) manifiesta (n) 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 (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de esta y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 06 días del mes de marzo de 2024

EL AUTOR:

ESTUDIANTE

Edison Augusto Zamora Moreano

C.I: 1004786578

CERTIFICACIÓN DIRECTOR

Ibarra, 06 de marzo del 2024

CERTIFICACIÓN DIRECTOR DEL TRABAJO DE TITULACIÓN

Por medio del presente yo MSc. Mauricio Rea, certifico que el Sr. Edison Augusto Zamora Moreano portador de la cedula de ciudadanía número 1004786578, ha trabajado en el desarrollo del proyecto de grado **"Diseñar un plan de pruebas de integración basado en el estándar ISO/IEC/IEEE 29119 para aplicar en la automatización de procesos de la empresa Reparauto",** previo a la obtención del Título de Ingeniero en Software realizado con interés profesional y responsabilidad que certifico con honor de verdad.

Es todo en cuanto puedo certificar a la verdad

Atentamente

MSc. Xavier Mauricio Rea Peñafiel. Ing.

DIRECTOR DE TRABAJO DE GRADO

CERTIFICADO DE IMPLEMENTACIÓN DE SOFTWARE

Ibarra, 01 de marzo del 2024

REPARAUTO

Por medio del presente documento, yo, **Ing. Christian Murillo**, en mi calidad de Gerente Propietario de Reparauto, portador de la cédula de identidad **No. 1002864161**, **CERTIFICO Y DOY FE** que:

El estudiante de la carrera de Software de la Universidad Técnica del Norte, **Sr. Edison Augusto Zamora Moreano**, identificado con cédula de identidad **No. 1004786578**, ha realizado la implementación y capacitación sobre los módulos de "Recepción de Vehículos" y de "Inventarios" dentro de nuestra empresa.

La implementación de estos módulos ha resultado en mejoras significativas en nuestros procesos internos, optimizando la gestión y el manejo de la recepción de vehículos e inventarios, lo que ha contribuido de manera positiva en la eficiencia y eficacia de nuestras operaciones.

Por lo tanto, Reparauto se compromete a continuar utilizando los módulos de software mencionados, expresando nuestra completa satisfacción con el trabajo realizado por el Sr. Edison Zamora. Reconocemos y valoramos el esfuerzo y la dedicación puestos en la realización de este proyecto, el cual ha cumplido nuestras expectativas.

Este certificado se emite a petición del interesado para los usos que considere pertinentes.

Atentamente,

Ing. Christian Murillo
Gerente Propietario Reparauto
CI: 1002864161

DEDICATORIA

A mi madre, *Flor*, cuya dedicación, esfuerzo y trabajo incansable han sido el faro que me ha guiado en los momentos más oscuros y los más brillantes. Su amor incondicional y apoyo constante son la base de cada paso que doy hacia adelante.

A mi padre, *Jaime* (+), quien, aunque no está físicamente con nosotros, su espíritu, enseñanzas y amor continúan viviendo en mí.

A mis hermanas, *Karla* y *Johanna*, pilares fundamentales de mi vida. Su apoyo incondicional, consejos y amor fraternal han sido cruciales en cada etapa de mi vida.

A *Christian*, mi cuñado, a quien considero mi hermano. Su amistad, sabiduría y guía han sido esenciales en mi formación como persona y profesional.

A *Luana*, mi sobrina, su existencia me inspira a ser una mejor persona cada día, con la esperanza de ser un ejemplo para ella.

Edison.

AGRADECIMIENTO

A mi director, MSc. *Mauricio Rea*, un agradecimiento especial por su paciencia y guía excepcionales a lo largo de este proceso. Su apoyo ha sido crucial para mi desarrollo y éxito.

A mi asesora, PhD. *Cathy Guevara*, por su invaluable orientación y amplios conocimientos. Su contribución ha sido clave para elevar la calidad de mi trabajo.

Agradezco a cada uno de mis profesores de la carrera de Software por su dedicación y pasión por enseñar. Su conocimiento y consejos han sido fundamentales para mi crecimiento personal y profesional.

Por último, mi gratitud a la Empresa Reparauto Ibarra por brindarme la oportunidad de desarrollar y aplicar mi trabajo de titulación.

A todos ustedes, mi más sincero agradecimiento.

TABLA DE CONTENIDOS

INDICE DE FIGURAS

INDICE 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: “DISEÑAR UN PLAN DE PRUEBAS DE INTEGRACIÓN BASADO EN EL ESTÁNDAR ISO/IEC/IEEE 29119 PARA APLICAR EN LA AUTOMATIZACIÓN DE PROCESOS DE LA EMPRESA REPARAUTO.”

En la parte de introducción se definen los antecedentes, situación actual, prospectiva, planteamiento del problema, objetivo general y específico, alcance, y justificación.

En el capítulo 1, se encuentra la base teórica para el desarrollo del plan de pruebas y desarrollo del prototipo web propuesto para la empresa Reparauto.

En el capítulo 2, se describe el proceso de construcción del plan de pruebas y del prototipo web, así como también la aplicación del plan de pruebas hacia el prototipo propuesto para la automatización de procesos de la empresa Reparauto.

En el capítulo 3, se detallan la parte del análisis de resultados y la interpretación de resultados.

Finalmente se encuentra las conclusiones, recomendaciones, referencias bibliografías y los anexos.

ABSTRACT

This document is made up of three chapters, in which the entire process to carry out the Degree Project is detailed: "DESIGN AN INTEGRATION TEST PLAN BASED ON THE ISO/IEC/IEEE 29119 STANDARD TO APPLY IN THE AUTOMATION OF THE REPARAUTO COMPANY".

In the introduction part, the background, current and prospective situation, problem statement, general and specific objective, scope, and justification are defined.

In chapter 1, there is the theoretical basis for the development of the test plan and development of the web prototype proposed for the Reparauto company.

In chapter 2, the process of construction of the test plan and the web prototype is described, as well as the application of the test plan towards the proposed prototype for the automation of processes of the Reparauto company.

In chapter 3, the part of the analysis of results and the interpretation of results are detailed.

Finally, there are the conclusions, recommendations, bibliographic references and annexes.

INTRODUCCIÓN

Antecedentes

La empresa Reparauto, ubicada en Ibarra-Ecuador, se dedica a la enderezada y pintura de vehículos colisionados. Desde sus inicios en el año 2015 hasta la actualidad, se ha observado que esta compañía enfrenta desafíos en la gestión de sus procesos de recepción de vehículos y de inventarios, atribuidos principalmente a la falta de automatización y a la presencia de errores humanos. Esta situación ha impactado negativamente la eficiencia operativa, precisión y visibilidad en tiempo real de los inventarios y operaciones de recepción, dificultando la toma de decisiones y limitando el crecimiento del negocio.

Además, Ibarra cuenta con un bajo índice de profesionales especializados en control de calidad de software, lo que agrava aún más el problema. La falta de aplicación de pruebas de software en el proceso de desarrollo puede llevar a la generación de productos de baja calidad, afectando seriamente los procesos industriales automotrices, como la pérdida de datos, errores en el inventario y problemas con la recepción de vehículos. Estos desafíos no solo perjudican la eficiencia y la precisión en el trabajo, sino que también pueden dañar la imagen de la empresa y provocar la pérdida de clientes.

Por consiguiente, resulta fundamental enfrentar estos desafíos y hallar soluciones eficaces que optimicen la gestión de los procesos en Reparauto. La implementación de un sistema integrado con un adecuado control de calidad en el desarrollo del software son medidas que podrían contribuir significativamente al éxito y expansión de la empresa en el sector. Estas mejoras permitirán no solo aumentar la eficiencia y precisión en las operaciones, sino también fortalecer la imagen de la compañía y fomentar la satisfacción y retención de los clientes.

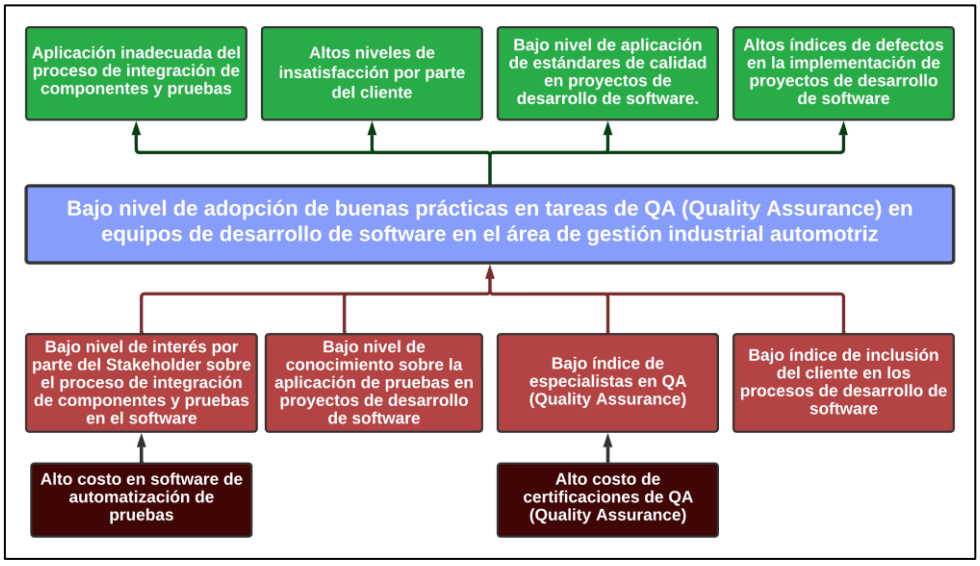
Planteamiento del problema

Actualmente la empresa Reparauto, dedicada a la enderezada y pintura de vehículos colisionados en la ciudad de Ibarra-Ecuador, enfrenta dificultades en la gestión de los procesos de recepción de vehículos y de inventarios debido a la falta de automatización y a la presencia de errores humanos. Esto ha llevado a una reducción en la eficiencia operativa, una disminución en la precisión y una falta de visibilidad en tiempo real de los inventarios y de las operaciones de recepción, lo que dificulta la toma de decisiones y limita el éxito del negocio.

Debido a que la ciudad de Ibarra dónde radica la empresa, cuenta con un bajo índice de profesionales especializados en control de calidad de software y al mismo tiempo la falta de aplicación de pruebas de software en el proceso de desarrollo puede generar software de mala calidad, lo que da a lugar problemas graves en procesos industriales automotrices cómo un software defectuoso que puede causar pérdida de datos, errores en el inventario y problemas con la recepción de vehículos, lo que afecta negativamente la eficiencia operativa y la precisión.

Figura 1

Árbol de Problemas



Nota: Fuente: Elaboración propia.

Objetivos

Objetivo General

Diseñar un plan de pruebas de integración iterativo que se ajuste al modelo ágil diente de tiburón para mejorar el proceso de integración de componentes en el desarrollo del aplicativo web.

Objetivos Específicos

- Diagnosticar las necesidades actuales de la empresa Reparauto en cuanto a los procesos de inventarios y recepción de vehículos.
- Automatizar el módulo de recepción de vehículos y el módulo de inventarios de la empresa Reparauto con una aplicación web usando el modelo diente de tiburón.
- Aplicar el plan de pruebas de integración en el proceso de desarrollo de la aplicación web.

Alcance

El alcance de la investigación consiste en diseñar un plan de pruebas de integración basado en el estándar ISO/IEC/IEEE 29119 para la automatización de los módulos de recepción y de inventarios de la empresa Reparauto con la finalidad de mejorar su gestión de procesos optimizando el uso de materiales, el tiempo de recepción de vehículos y la gestión de documentación de la empresa.

En el desarrollo de la aplicación web se propone aplicar una arquitectura orientada a microservicios ya que cada microservicio tiene su propio ciclo de vida, puede ser testeado y desplegado por separado. Según (Guimarey, 2020), los microservicios pueden ser combinados para realizar una tarea en conjunto, lo que les otorga una ventaja intrínseca al ser independientes de la tecnología y el lenguaje utilizado. la estructura del Backend consistiría en un servidor HTTP integrado en Node.js para manejar las solicitudes y las respuestas de la aplicación, controladores para manejar las solicitudes del usuario, modelos para representar los datos de la aplicación y conectarse a la base de datos PostgreSQL, rutas

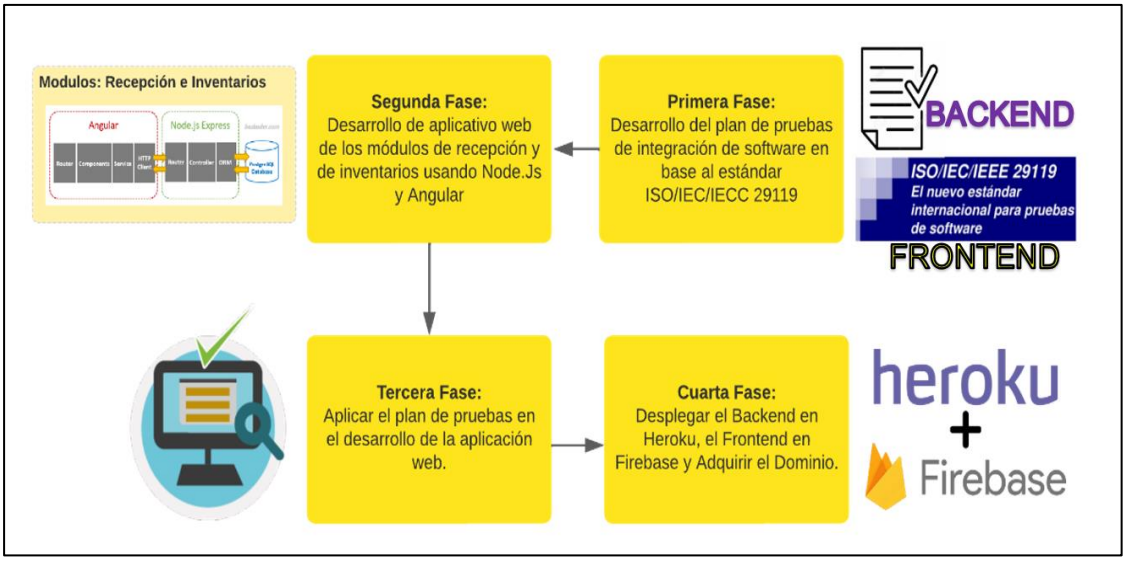
para definir la API, middleware para la autenticación y la validación de datos, una conexión a la base de datos PostgreSQL. El Frontend diseñado con Angular se encargaría de la interfaz de usuario y de la comunicación con el Backend.

Con la finalidad de evaluar la integración de componentes en el aplicativo y él correcto funcionamiento de este, se propone aplicar el plan de pruebas de integración desarrollado en base al estándar ISO/IEC/IEEE 29119.

Como producto final se pretende el despliegue del Backend en la plataforma “Heroku” y el despliegue del Frontend en la plataforma de Google “Firebase”, además de adquirir el nombre del dominio para que la aplicación web sea implementada de manera exitosa en la empresa Reparauto.

Figura 2

Fases del Alcance.



Nota: Fuente: Elaboración propia.

Metodología

La metodología por aplicar para solucionar el objetivo general será desarrollar el plan de pruebas de integración basándose en el estándar ISO/IEC/IEEE- 29119 la cual tiene como finalidad definir un conjunto de estándares acordado internacionalmente para la prueba de software que puede ser utilizado por cualquier organización al realizar cualquier forma de prueba de software (ISO, 2023).

Para dar solución al primer objetivo se va a identificar las necesidades actuales de la empresa en cuanto a los procesos de inventarios y recepción de vehículos, se realizará una revisión exhaustiva de la documentación y registros de la empresa, así como entrevistar al personal encargado de estas áreas para conocer sus necesidades y opiniones. Posterior, se analizarán los procesos actuales de la empresa para determinar si son eficientes y efectivos, identificando los problemas que se presentan y las oportunidades de mejora.

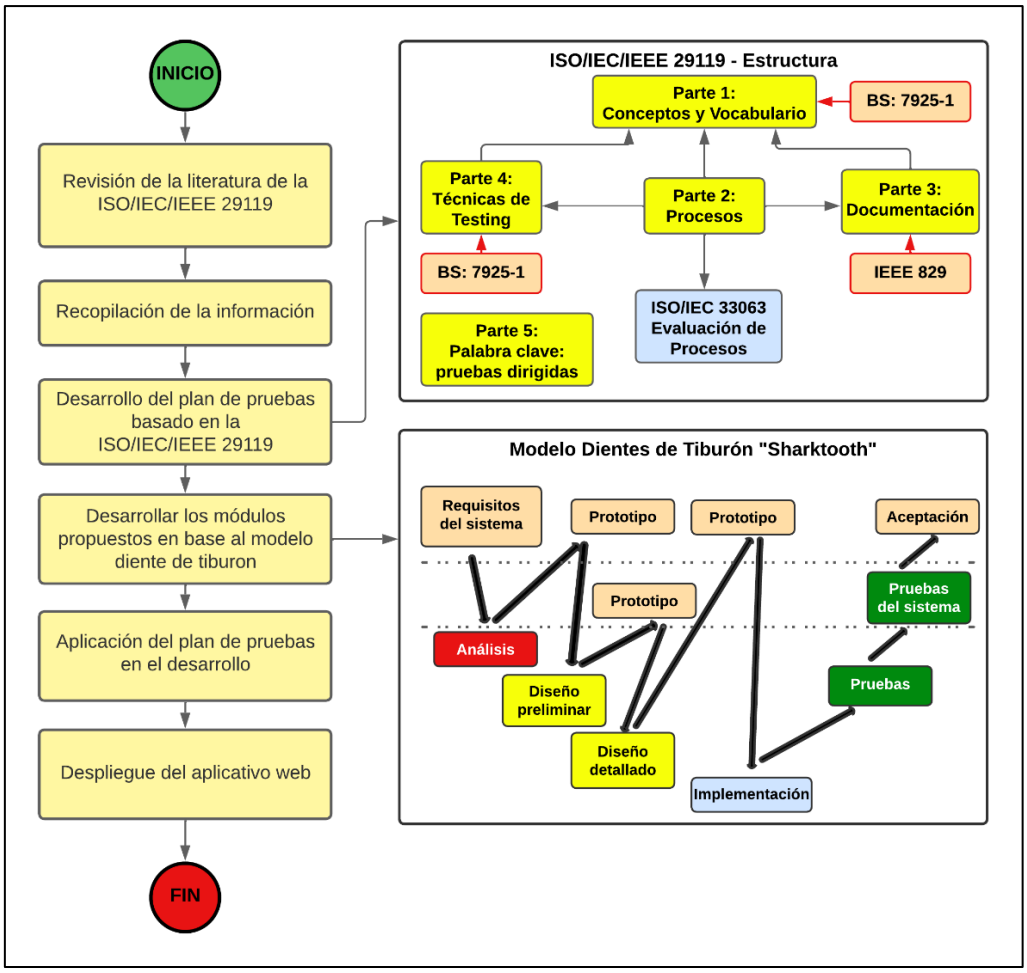
Para solucionar el segundo objetivo y lograr una mayor eficiencia de los procesos de Reparauto, se usará tecnologías de vanguardia como Angular y Node. La automatización de los módulos de recepción de vehículos e inventarios contribuirá a una mayor productividad y reducción de errores en la empresa. Además, el modelo diente de tiburón permitirá mejorar la interacción entre los usuarios y el sistema, asegurando el éxito del proceso de automatización. (Regulwar, 2012).

Además, durante el desarrollo de la aplicación web se implementarán buenas prácticas y se utilizará una arquitectura basada en microservicios. Esta elección se debe a que los microservicios permiten escalar las aplicaciones en cualquier dimensión, así como integrar servicios de alta disponibilidad y mejorar costos en la operación de los servicios (Digital Services, 2020).

Para dar solución al tercer objetivo se propone aplicar el plan de pruebas de integración el cual tiene como base la ISO/IEC/IEEE 29119 al software desarrollado ya que la arquitectura de este usa microservicios los cuales dependen netamente de la integración correcta de componentes.

Figura 3

Diagrama de Metodologías.



Nota: Fuente: Elaboración propia.

Justificación

Según la sección 8 de los Objetivos de Desarrollo Sostenible (ODS), se debe fomentar el trabajo decente y el crecimiento económico a través de la creación de oportunidades laborales de calidad que no perjudiquen al medio ambiente. Además, se busca garantizar condiciones laborales adecuadas para toda la población en edad de trabajar (ONU, 2015). Dónde la empresa Reparauto al implementar un software que automatizara sus procesos mejorara las condiciones de trabajo de sus colaboradores restando actividades que se desarrollan de forma manual y entorpecen el desarrollo de sus actividades.

La sección 9 de la ODS enfatiza la necesidad de modernizar la infraestructura y las industrias para hacerlas sostenibles, lo cual implica el uso eficiente de los recursos y la promoción de tecnologías y procesos industriales limpios y amigables con el medio ambiente. Además, se busca que todos los países tomen medidas para lograr estos objetivos, de acuerdo con sus capacidades respectivas (ONU, 2015). Por lo cual la empresa Reparauto al automatizar sus procesos logrará una reducción significativa de uso de papel y desperdicio de materiales debido al mejor control logrando acatar el objetivo mencionado.

Justificación Tecnológica:

El desarrollo de la aplicación web contribuirá en el mejoramiento de la cadena productiva de la empresa Reparauto ya que reemplazan las tareas repetitivas y mecánicas principalmente que realiza una persona y las decisiones que toma en los procesos de manufacturación (Nexus Integra, 2022), además la implementación de dicho sistema ayudará en la conservación del medio ambiente ya que favorece la reutilización de lo creado para nuevas funcionalidades o nuevos productos, favoreciendo el ahorro y la optimización de los costos y la sostenibilidad (Linux Post Install, 2020).

CAPÍTULO 1

Marco Teórico

1.1. ISO/IEC/IEEE 29119 Software Testing.

Como se menciona en el estándar ISO/IEC/IEEE 29119: Software Testing es una serie de normas internacionales que surgen como respuesta a la creciente necesidad de establecer un marco de referencia común y un lenguaje unificado para las pruebas de software. Con el auge de la tecnología y la informática en todos los ámbitos de la vida cotidiana y empresarial, es fundamental garantizar que los sistemas de software funcionen correctamente, de manera eficiente y segura. La adopción de este estándar en el proceso de desarrollo de software asegura la calidad y la eficacia de las pruebas, minimizando la posibilidad de errores y fallos en los sistemas de software (ISO 29119, 2021).

Las pruebas de software son un componente esencial en el ciclo de vida del desarrollo de software, ya que permiten identificar y corregir errores antes de que los sistemas se implementen y se utilicen. Sin embargo, las metodologías y técnicas de prueba varían ampliamente, lo que dificulta la comparación y el intercambio de información entre equipos y organizaciones.

Este estándar es el resultado de la colaboración entre la Organización Internacional de Normalización (ISO), la Comisión Electrotécnica Internacional (IEC) y el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE). Juntas, estas organizaciones desarrollaron un conjunto de normas que proporcionan un marco integral y coherente para la práctica de las pruebas de software, independientemente del contexto del proyecto, la metodología de desarrollo o el tipo de aplicación (ISO 29119, 2021).

1.1.1. Historia y evolución de la norma ISO/IEC/IEEE 29119.

Según el autor Stuard Reid en su publicación “Los nuevos estándares internacionales de pruebas de software”, las pruebas de software han sido una actividad fundamental desde hace mucho tiempo, incluso antes de que se definieran los modelos de ciclo de vida. En este sentido, se ha referido que "la actividad de prueba de software separada" ya se mencionaba desde el año 1954 como parte importante del proceso de desarrollo de software (Reid Stuard, 2017).

Organismos como IEEE y BSI han publicado una serie de estándares relacionados con diferentes aspectos de las pruebas de software. Entre ellos se encuentra el estándar IEEE 829 Software Test Documentation, que se inició en 1979 y se publicó cuatro años después. También se publicó un estándar de prueba unitaria en 1987, que fue revisado en 2003. Además, BSI publicó dos estándares de prueba en 1998, uno de los cuales es un vocabulario de prueba, mientras que el otro es un estándar de prueba de componentes.

A continuación, se detalla la evolución de la ISO/IEC/IEEE 29119:

- **1947**: Fundación de la Organización Internacional de Normalización (ISO).
- **Mayo de 2007**: La ISO forma un grupo de trabajo para desarrollar nuevos estándares de pruebas de software, con el apoyo del IEEE y el BSI. Esto marca el inicio de la creación de la ISO/IEC/IEEE 29119.
- **2007-2013**: El grupo de trabajo desarrolla y revisa los borradores de las diferentes partes de la ISO/IEC/IEEE 29119, basándose en estándares existentes y retroalimentación de la industria.
- **Septiembre de 2013**: Se publica la primera parte de la ISO/IEC/IEEE 29119, titulada "Conceptos y vocabulario" (ISO/IEC/IEEE 29119-1).
- **2014**: Se publica la segunda parte de la ISO/IEC/IEEE 29119, titulada "Proceso de pruebas" (ISO/IEC/IEEE 29119-2) y también la tercera parte titulada "Documentación de pruebas" (ISO/IEC/IEEE 29119-3).

- **2015**: Se publica la cuarta parte de la ISO/IEC/IEEE 29119, titulada "Técnicas de pruebas" (ISO/IEC/IEEE 29119-4) y también la quinta parte titulada "Pruebas basadas en palabras clave" (ISO/IEC/IEEE 29119-5).
- **Después de 2015**: Se continúa trabajando en la serie de estándares ISO/IEC/IEEE 29119, con la posibilidad de agregar partes adicionales, como la evaluación de la prueba.

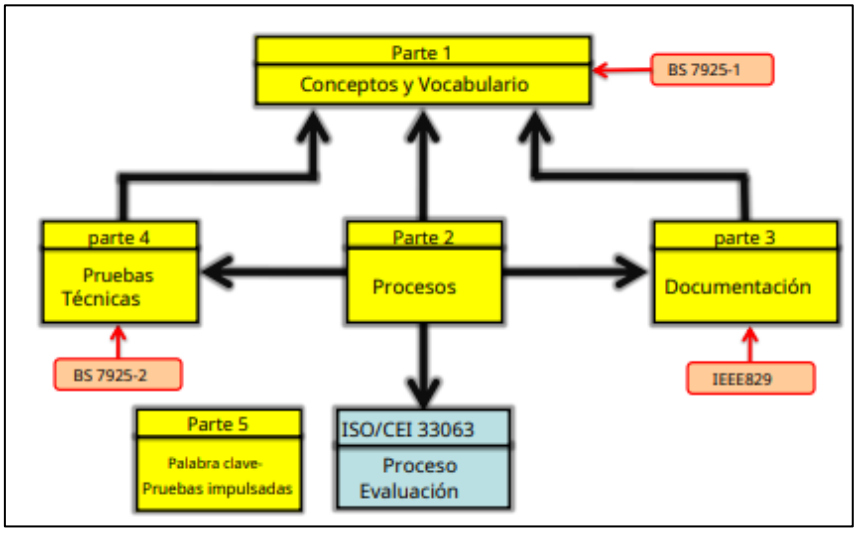
1.1.2. Componentes y estructura de la norma ISO/IEC/IEEE 29119.

En mayo de 2007, ISO aprobó la propuesta de un nuevo conjunto de normas para las pruebas de software, basado en las normas existentes de IEEE y BSI, como IEEE 829, IEEE 1008, BS 7925-1 y BS 7925-2. Como no existía ningún grupo de trabajo con experiencia en pruebas de software dentro del SC7, se creó un nuevo grupo de trabajo llamado "Pruebas de software" (WG26) con representación de más de 20 naciones diferentes para el año 2013 (Reid Stuard, 2017).

Inicialmente, la propuesta era de cuatro partes, pero se añadió una quinta parte sobre la evaluación de procesos. Esta parte se está desarrollando juntamente con ISO WG10 (Evaluación de procesos) y WG26. También se ha comenzado recientemente a trabajar en un estándar separado sobre las pruebas basadas en palabras clave (Reid Stuard, 2020).

Figura 4

Estructura de ISO/IEC/IEEE 29119



Nota: Fuente: ISO-29119-The-New-International-Software-Testing-Standards

La Figura 4 muestra cómo se incorporan los estándares existentes en las partes 1 a 4 de los nuevos estándares ISO/IEC/IEEE 29119, que se basan en cuatro entidades básicas con los procesos de prueba que forman el núcleo central. La documentación de prueba se produce como resultado de la ejecución de los procesos de prueba, mientras que las técnicas para realizar las pruebas se definen como parte de los procesos, y las técnicas reales se definen por separado. La terminología utilizada por las otras partes de este modelo se define en el vocabulario (Reid Stuard, 2020).

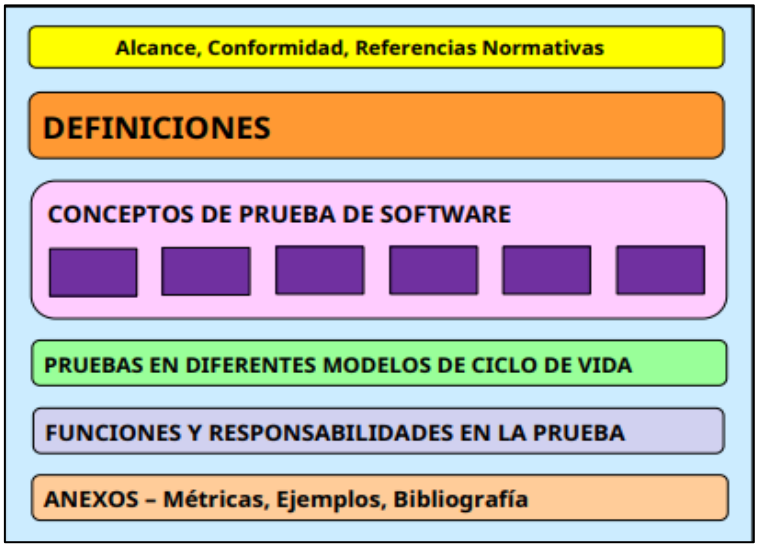
1.1.3. ISO/IEC/IEEE 29119 – 1: Conceptos Y Definiciones

La norma ISO/IEC/IEEE 29119-1 actúa como introducción a los demás estándares, proporcionando una base común de conceptos, vocabulario y ejemplos prácticos para su uso en la implementación de los estándares. Es importante tener en cuenta que ISO/IEC/IEEE 29119-1 es un documento informativo y no prescriptivo, lo que significa que no impone reglas estrictas, sino que proporciona un punto de partida y una guía para la aplicación de los otros estándares de la serie (ISO/IEC/IEEE, 2013b).

A continuación, en la Figura 5 se muestran las diferentes partes del estándar y su aplicación en cuando a proyectos que aplican distintos modelos de ciclo de vida.

Figura 5

ISO/IEC/IEEE 29119 Parte 1: conceptos y vocabulario



Nota: Fuente: ISO-29119-The-New-International-Software-Testing-Standards

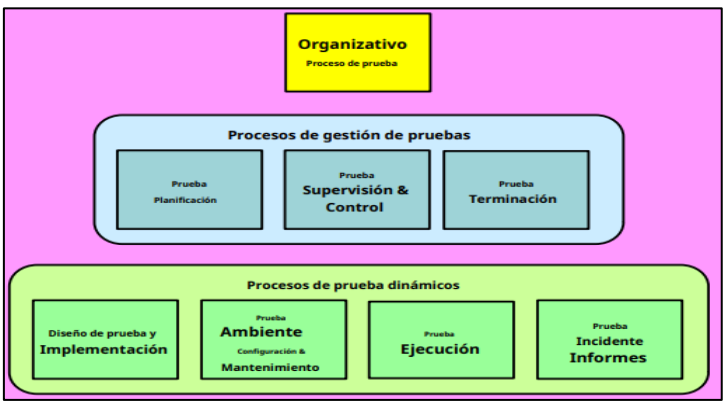
1.1.4. ISO/IEC/IEEE 29119 – 2: Procesos de Prueba

La parte 2 de la norma, se encarga de definir los procesos de prueba de software a diferentes niveles, incluyendo organizacional, gestión de pruebas y ejecución de pruebas dinámicas. Esta parte de la norma respalda una amplia gama de pruebas, incluidas las pruebas dinámicas, tanto funcionales como no funcionales, y abarca tanto las pruebas manuales como las automáticas, además de contemplar las pruebas con y sin guiones predefinidos. Los procesos especificados por la parte 2, son flexibles y pueden aplicarse en combinación con cualquier modelo de ciclo de vida del desarrollo de software. En el contexto del desarrollo de software y la importancia crítica de las pruebas para reducir riesgos, la norma adopta una metodología de pruebas orientada al análisis y manejo de riesgos. Este método prioriza y focaliza las pruebas en las áreas más cruciales y de mayor riesgo del software, lo cual es una práctica estándar en la industria para la planificación y ejecución eficaz de pruebas. (ISO/IEC/IEEE, 2013).

El autor Reid explica que los procesos de gestión de pruebas son necesarios para desarrollar e implementar el plan de prueba del proyecto, así como para cada fase o tipo de prueba que requiera un plan de prueba separado. Aunque se espera que los planes de prueba desarrollados utilizando ISO/IEC/IEEE 29119 incluyan tanto pruebas estáticas como dinámicas.

Figura 6

Los ocho procesos de prueba ISO/IEC/IEEE 29119



Nota: Fuente: ISO-29119-The-New-International-Software-Testing-Standards

La Figura 6 muestra los ocho procesos de prueba definidos en el estándar, cada uno compuesto por un conjunto de actividades y tareas específicas. El Anexo A proporciona descripciones detalladas de cada proceso y sus actividades, incluyendo un ejemplo de una descripción textual típica hasta el nivel de tareas específicas(Reid Stuard, 2017) .

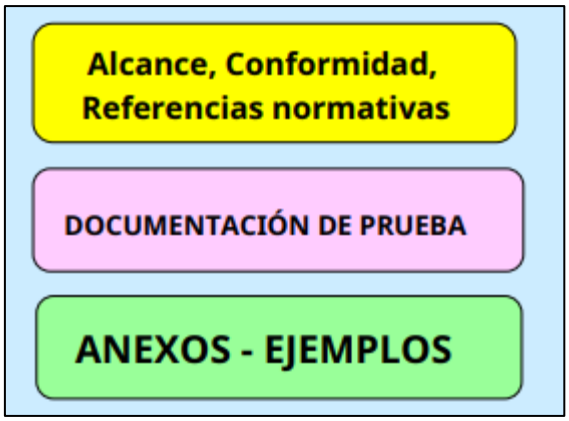
1.1.5. ISO/IEC/IEEE 29119 - 3: Documentación de las Pruebas

La parte 3 de la norma proporciona plantillas y ejemplos para la documentación de pruebas. Estas plantillas están organizadas por proceso de prueba, reflejando la estructura general de descripción de procesos de prueba en la parte 2 de la norma. El Anexo A ofrece un resumen del contenido de cada documento, mientras que el Anexo B contiene asignaciones de la parte 2. Por otro lado, el Anexo C proporciona una descripción general de los ejemplos y los Anexos D a S contienen ejemplos de la aplicación de las plantillas. El Anexo T proporciona asignaciones a estándares existentes y la bibliografía se encuentra al final del documento. La parte 3 permite una amplia gama de pruebas, incluyendo pruebas manuales y automatizadas, dinámicas y funcionales, así como pruebas con y sin guion, y también pruebas no funcionales. Las plantillas de documentación definidas en la tercera parte de la norma se pueden utilizar en conjunto con cualquier modelo de ciclo de vida de desarrollo de software (ISO/IEC/IEEE, 2013).

Según (Reid Stuard, 2017),existe una fuerte relación entre la Parte 2 y la Parte 3 del conjunto de estándares descritos en la norma. Los procesos definidos en la Parte 2 están estrechamente relacionados con la documentación de prueba definida en la Parte 3. Es común que los resultados de los procesos de prueba definidos en la Parte 2 coincidan con la documentación de prueba definida en la Parte 3. Como la Parte 3 es un estándar de documentación, debe cumplir con la estructura y el formato definidos en ISO/IEC 15289 (Documentación del contenido de los productos de información del ciclo de vida). Por lo tanto, la estructura de la Parte 3 podría parecer inusual para algunos.

Figura 7

ISO/IEC/IEEE 29119 Parte 3 – Documentación de prueba



Nota: Fuente: ISO-29119-The-New-International-Software-Testing-Standards

El autor también menciona que la estructura básica de la Parte 3 se muestra en la Figura 7. Los anexos son de mayor utilidad para los profesionales, ya que proporcionan ejemplos de todos los tipos de documentos de prueba definidos para proyectos tanto ágiles como tradicionales.

La tercera parte de la norma ofrece modelos con explicaciones sobre el contenido de los principales tipos de documentos utilizados en pruebas de software.

Documentación de prueba organizacional:

- Política de prueba
- Estrategia de prueba

Documentación de prueba de proyecto:

- Plan de prueba del Proyecto
- Informe de finalización del proyecto de prueba

Documentación de nivel de prueba:

- Plan de prueba
- Especificación de prueba
- Resultados de la Prueba
- Informes de anomalías
- Informe de estado de prueba de nivel

- Informe de entorno de prueba
- Informe de finalización de nivel de prueba

Apéndices:

- Ejemplos de documentos en cada nivel de prueba para proyectos tanto ágiles como tradicionales.

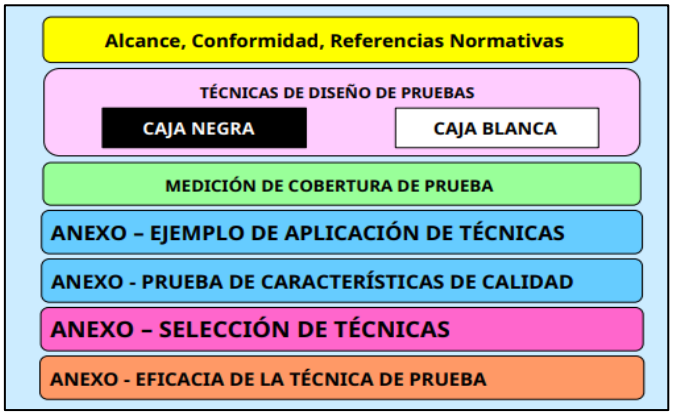
1.1.6. ISO/IEC/IEEE 29119 – 4: Técnicas de Prueba

La cuarta parte establece un conjunto de técnicas para el diseño y la implementación de pruebas en la segunda parte. Estas técnicas están destinadas a una amplia variedad de usuarios, como evaluadores, administradores de pruebas y desarrolladores, y tienen como objetivo proporcionar una guía para la implementación de pruebas de software (ISO/IEC/IEEE, 2013).

El autor Reid, explica que los usuarios que siguen la Parte 2 de la norma deben producir planes de prueba que incluyan la especificación y utilización de técnicas de diseño de casos de prueba y criterios de finalización de la prueba que se alcanzarán. Estas técnicas de diseño de casos de prueba y medidas de cobertura correspondientes se definen en la Parte 4 del estándar. Aunque cada técnica de diseño de caso de prueba se define formalmente en la parte normativa del estándar, se proporcionan ejemplos correspondientes en los anexos que muestran cómo se pueden aplicar las técnicas (Reid Stuard, 2017).

Figura 8

ISO/IEC/IEEE 29119 Parte 4 – Técnicas de prueba



Nota: Fuente: ISO-29119-The-New-International-Software-Testing-Standards

La Figura 8 muestra la estructura de la Parte 4, la cual se basa estrechamente en BS 7925-2, por lo que los usuarios de este estándar notarán una estrecha correspondencia con la Parte 4.

A diferencia de BS 7925-2, también proporciona un anexo sobre las pruebas de las características de calidad o pruebas no funcionales. Este anexo muestra cuáles de las técnicas de diseño de prueba definidas son apropiadas para probar cada una de las características de calidad definidas en la serie de normas ISO/IEC 25000 (SQuaRE) (ISO 2005). Los ejemplos del uso de técnicas de diseño de prueba en la Parte 4 son considerados más prácticos para la mayoría de los usuarios que las definiciones técnicas secas de estas técnicas.

A continuación, se detalla una lista de técnicas de diseño de casos de prueba que se incluyen en la parte 4 del estándar.

Técnicas de prueba basadas en especificaciones:

- Análisis de valor límite
- Gráfica causa – efecto
- Método del árbol de Clasificación
- Técnicas de prueba combinatoria
- Prueba de la tabla de decisiones
- Partición de equivalencia
- Error al adivinar
- Pruebas aleatorias
- Prueba de escenario
- Prueba de transición de estado
- Prueba de sintaxis

Técnicas de prueba basadas en estructura:

- Prueba de rama/decisión
- Prueba de condición de rama

- Prueba de combinación de condiciones de ramificación
- Prueba de flujo de datos
- Prueba de cobertura de decisión de condición modificada (MCDC)
- Prueba de declaraciones

Técnicas de prueba basadas en la experiencia:

- Error al adivinar

1.1.7. ISO/IEC/IEEE 29119 – 5: Pruebas dirigidas por palabras

La parte 5 de la norma define que de pruebas basadas en palabras clave refiere a un enfoque de pruebas que utiliza módulos para elaborar descripciones de casos de prueba. Este método se caracteriza por su enfoque en conceptos clave y características inherentes a este tipo de pruebas. Está diseñado para ser accesible a cualquier persona interesada en desarrollar especificaciones, estructuras o automatizaciones de pruebas que se fundamenten en este enfoque. Además, este estándar detalla los criterios que deben cumplir los sistemas de pruebas basadas en palabras clave, permitiendo así que los profesionales puedan intercambiar diversos componentes de pruebas, incluyendo casos de prueba, datos de prueba, las propias palabras clave, o las especificaciones de manera completa. (Coordinating & Society, 2016).

Según el autor (Reid Stuard, 2017), La quinta sección de la norma no solo establece los criterios para implementar pruebas basadas en palabras clave, sino también los estándares fundamentales para las herramientas requeridas para su empleo adecuado. Este segmento del estándar también define interfaces específicas y un esquema uniforme para el intercambio de datos, asegurando así la compatibilidad entre las herramientas suministradas por diversos fabricantes para el intercambio de información relevante como los casos de prueba, los datos asociados y los resultados obtenidos. Adicionalmente, la norma específica distintos niveles de palabras clave y ofrece orientación sobre el momento apropiado para usar palabras clave organizadas jerárquicamente o en una estructura más plana, e igualmente detalla varios tipos de palabras clave.

1.2. Técnicas de Pruebas de Software:

Según (Luo, 2001),las pruebas de software es una práctica antigua en la historia de las computadoras digitales. Esta técnica es importante para evaluar la calidad del software y dado que suele consumir entre el 40 y el 50% de los esfuerzos de desarrollo. La emergencia de los lenguajes de programación de cuarta generación (4GL) ha impulsado la eficiencia en la fase de desarrollo, lo que ha llevado a un incremento en el tiempo asignado para la evaluación de software. Pese a los progresos alcanzados en métodos formales y estrategias de comprobación, sigue siendo indispensable examinar los sistemas previos a su aplicación práctica. Esta faceta representa uno de los sectores más intrincados y menos entendidos dentro de la ingeniería de sistemas. La ejecución de pruebas permanece como una estrategia fundamental para garantizar la fiabilidad de sistemas informáticos de considerable complejidad y se destaca como un campo relevante de estudio en el ámbito de la ciencia computacional.

Los autores (Sawant et al., 2012), consideran que las pruebas de software se refieren al proceso de evaluación del software con la intención de encontrar errores en él. Es una técnica destinada a evaluar un atributo o capacidad de un programa o producto y determinar si cumple con su calidad. La prueba de software también se utiliza para evaluar otros factores de calidad del software, como confiabilidad, usabilidad, integridad, seguridad, capacidad, eficiencia, portabilidad, mantenibilidad, compatibilidad, entre otros. A lo largo de los años, se han utilizado las mismas técnicas de prueba, algunas de las cuales son métodos creados más por artesanía que por buenos métodos de ingeniería. Si bien las pruebas pueden ser costosas, no probar el software puede ser aún más costoso. Las pruebas de software tienen como objetivo alcanzar ciertos objetivos y principios que deben seguirse.

1.2.1. Clasificación de las técnicas de pruebas de software

Las técnicas de pruebas de software se pueden clasificar de diferentes maneras.

- Basadas en el nivel de pruebas: las técnicas se pueden clasificar según el nivel de pruebas al que se aplican, como pruebas unitarias, de integración, de sistema, de aceptación, entre otras.
- Basadas en el tipo de pruebas: las técnicas se pueden clasificar según el tipo de pruebas que se realizan, como pruebas funcionales, de rendimiento, de seguridad, de usabilidad, entre otras.
- Basadas en la estrategia de pruebas: las técnicas se pueden clasificar según la estrategia de pruebas que se siga, como pruebas de caja negra, de caja blanca, exploratorias, de regresión, entre otras.
- Basadas en la técnica utilizada: las técnicas se pueden clasificar según la técnica específica que se utiliza para realizar las pruebas, como técnicas de particiones de equivalencia, de valores límite, de diagramas de decisión, de análisis de causa-efecto, entre otras.

Las técnicas de pruebas de software se pueden clasificar en varias categorías según la ISO 29119. En primer lugar, están las técnicas basadas en especificaciones, que incluyen pruebas de caja blanca y pruebas de caja negra. Las pruebas de caja blanca son aquellas en las que se examina el código interno del software, mientras que las pruebas de caja negra se centran en las entradas y salidas del software.

En segundo lugar, están las técnicas basadas en experiencia, que utilizan la intuición y la experiencia para diseñar y ejecutar pruebas. Esto incluye técnicas como pruebas exploratorias y pruebas de usabilidad.

En tercer lugar, están las técnicas basadas en defectos, que se centran en identificar y corregir defectos. Esto incluye técnicas como pruebas de regresión y pruebas de rendimiento.

Por último, están las técnicas basadas en modelos, que utilizan modelos matemáticos y estadísticos para diseñar y ejecutar pruebas. Esto incluye técnicas como pruebas de mutación y pruebas de simulación.

1.2.2. Pruebas funcionales y no funcionales

1.2.1.1 Pruebas funcionales

Una prueba funcional, según (Lamancha, 2007), tiene como objetivo validar si el comportamiento observado del software cumple o no con sus especificaciones. En este tipo de prueba, se evalúa el software desde la perspectiva del usuario, probando las funciones a través de la entrada y observando las salidas. No se presta atención a la estructura interna del programa. Se basa en técnicas como partición de equivalencia, análisis del valor límite, grafo causa-efecto y conjetura de errores para derivar casos de prueba. La prueba funcional busca mostrar discrepancias con las especificaciones y no necesariamente demostrar que el programa cumple su especificación.

1.2.2.2 Pruebas no funcionales

Durante el proceso de automatización de los módulos de recepción e inventarios es fundamental llevar a cabo pruebas no funcionales para verificar que el software cumple con los requerimientos no funcionales establecidos por el cliente. Las pruebas no funcionales abarcan diversos aspectos que no se relacionan directamente con las funciones específicas del software, sino más bien con su calidad y desempeño. Estos tipos de pruebas incluyen pruebas de seguridad, pruebas de rendimiento para evaluar la velocidad y la eficiencia, pruebas de usabilidad para medir la facilidad de uso, pruebas de portabilidad para verificar la adaptabilidad a diferentes entornos, entre otras (Ocampo Acosta Alejandro & Correa Tapasco Luisa Marcela, 2011).

1.3. Pruebas de Integración de Software

Según los autores (Ocampo Acosta Alejandro & Correa Tapasco Luisa Marcela, 2011), en su tesis detallan que las pruebas de integración son un componente esencial en el proceso de desarrollo de software. Estas pruebas se aplican especialmente a productos desarrollados a partir de componentes o módulos, con el propósito de garantizar que, al integrar estos módulos, funcionen de acuerdo con las especificaciones establecidas. Las pruebas de integración se realizan con el objetivo de detectar errores asociados al proceso de ensamblaje de los módulos y verificar el correcto funcionamiento de sus interfaces al trabajar en conjunto.

1.3.1. Objetivos de las pruebas de integración

Los objetivos de las pruebas de integración se centran en confirmar que los componentes, después de ser sometidos a pruebas individuales, funcionen de manera correcta al ensamblarse. Esto implica asegurar una interacción eficiente a través de interfaces internas y externas, cumplir con la funcionalidad establecida y ajustarse a los requisitos no funcionales. Además, estas pruebas permiten evaluar las interfaces entre grupos de componentes o subsistemas, garantizando su invocación adecuada y la transmisión correcta de mensajes y datos. En algunos casos, las pruebas de integración se combinan con las pruebas unitarias, utilizando módulos auxiliares para simular las acciones de los componentes involucrados, lo que asegura el cumplimiento de precondiciones necesarias y la evaluación de los resultados obtenidos. (Test de integración)

A continuación, se detallan los principales objetivos de las pruebas de Integración:

- Validar la Interoperabilidad
- Detectar Errores de Integración
- Verificar la Comunicación
- Evaluar el Comportamiento del Sistema
- Comprobar Requisitos No Funcionales
- Prevenir Problemas Futuros

1.3.2. Tipos de pruebas de integración.

Actualmente existen distintos tipos de pruebas de integración, como Big Bang, Top Down, Down Top, entre otros. En el diseño del plan de pruebas basado en la ISO/IEC/IEEE 29119, se espera elegir de estos tipos para aplicar en la automatización e integración de los módulos de recepción e inventarios de la empresa Reparauto. Por lo tanto, es imprescindible describir en el marco teórico el concepto de cada uno de ellos. A continuación, se muestra una tabla con los tipos de pruebas de integración:

Tabla 1

Tipos de Pruebas de Integración.

Nombre	Descripción
Big Bang	Esta prueba concentra todos los módulos del sistema y los ejecuta juntos. Requiere que todas las unidades estén completas antes de su ejecución.
Ad Hoc	Se refiere a pruebas que se pueden realizar en cualquier momento para encontrar errores no previstos.
Top Down	Inicia el análisis de código en los módulos superiores y desciende hacia otros componentes.
Down Top	Comienza desde las interfaces inferiores y asciende. Los problemas son más fáciles de detectar, al igual que las mejoras.
Hybrid	Combina los enfoques Top Down y Down Top, permitiendo la elección de módulos superiores o inferiores para encontrar errores.

Nota: Fuente: Elaboración propia.

1.3.3. Pruebas de integración en arquitecturas basadas en microservicios

En el desarrollo de software, las pruebas de integración en arquitecturas de microservicios se centran en verificar la colaboración sin fisuras entre múltiples microservicios independientes que se comunican a través de interfaces bien definidas. Los microservicios, a diferencia de las aplicaciones monolíticas, ofrecen flexibilidad y escalabilidad al dividir el sistema en componentes modulares. El objetivo principal de las pruebas de integración en este contexto es detectar posibles problemas, como fallos de comunicación y discrepancias en los datos, garantizando la fiabilidad y la solidez del sistema en diversas condiciones de uso. Para abordar los desafíos de la complejidad de la integración, se recurre a estrategias de prueba automatizadas y a la simulación de servicios (*Pruebas de integración de microservicios* | *AppMaster*, s/f).

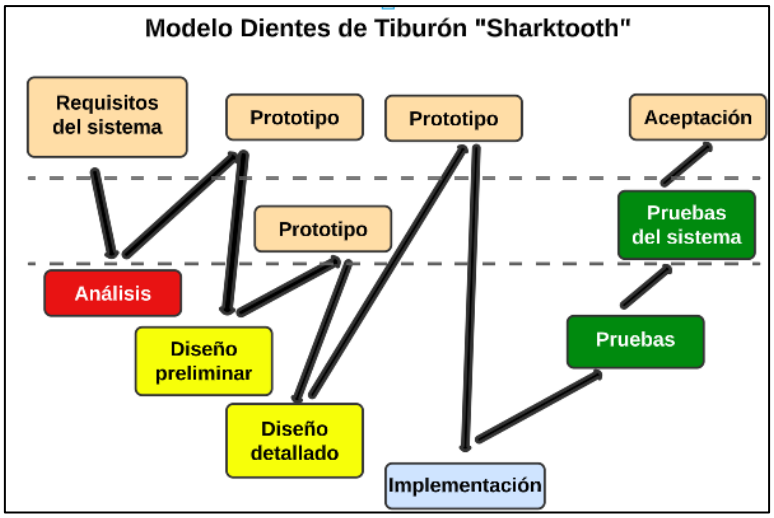
Un aspecto fundamental de las pruebas de integración en arquitecturas basadas en microservicios es manejar la complejidad de múltiples puntos de integración entre los microservicios, lo que requiere un enfoque cuidadoso y sistemático en la planificación y ejecución de los casos de prueba. La automatización de estas pruebas se considera crucial para optimizar el tiempo y el esfuerzo dedicados al proceso de prueba y para reducir la posibilidad de errores humanos. La implementación de herramientas y marcos de automatización, combinada con la práctica de la integración y la entrega continua, permite a los equipos de desarrollo mantener la calidad y la estabilidad del sistema a lo largo de su ciclo de vida.

1.4. Modelo dientes de Tiburón

El modelo Dientes de Tiburón es una metodología de desarrollo de software que se basa en el modelo de dientes de sierra (también conocido como modelo de cascada). Este modelo introduce la Figura del gerente en el proceso de desarrollo, y se enfoca en el control del proceso de desarrollo y la toma de decisiones estratégicas.

Figura 9

Diagrama del Modelo Dientes de Tiburón



Nota: Fuente: Elaboración propia.

Este modelo divide el proceso de desarrollo en varias etapas, que incluyen Análisis de Requerimientos, Revisión de Diseño Preliminar, Integración y Pruebas del Sistema, Desarrollo de Componentes, Diseño Detallado, Pruebas Unitarias y la Implementación. Además, el modelo incluye dos demostraciones de prototipos para asegurar que el software esté siendo desarrollado de manera adecuada y que cumpla con las expectativas del cliente.

También, se enfoca en la comunicación y colaboración entre los miembros del equipo de desarrollo y el gerente, y enfatiza la importancia de la retroalimentación continua para mejorar el proceso de desarrollo. También se destaca la importancia de la documentación adecuada para garantizar la calidad del software desarrollado.

A pesar de que el modelo Dientes de Tiburón comparte muchas características con el modelo de dientes de sierra, también presenta ciertas diferencias significativas. El modelo Dientes de Tiburón pone mayor énfasis en la Figura del gerente y en la toma de decisiones estratégicas, mientras que el modelo de dientes de sierra se enfoca más en la implementación de cada etapa del proceso de desarrollo.

1.4.1. Comparación entre el modelo dientes de Tiburón y otros modelos.

A continuación, se presenta una tabla comparativa entre el Modelo Dientes de tiburón con el modelo en cascada, Modelo en espiral y modelo Ágil.

Tabla 2

Comparación entre el modelo dientes de Tiburón y otros modelos.

Modelo	Enfoque	Etapas	Flexibilidad	Comunicación	Rol del Gerente
Dientes de Tiburón (Sharktooth)	Enfoque en el control del proceso de desarrollo y la toma de decisiones estratégicas.	Análisis de Requerimientos, Diseño Preliminar, Diseño Detallado, Implementación, Pruebas, Pruebas de Sistema.	Menos flexible, ya que enfatiza el control del proceso de desarrollo.	Enfocado en la comunicación y colaboración entre el equipo de desarrollo y el gerente.	El gerente es una Figura clave en el proceso de desarrollo.
Modelo en Cascada (Waterfall)	Enfoque en la implementación de cada etapa del proceso de desarrollo.	Análisis de Requerimientos, Diseño, Implementación, Pruebas y Mantenimiento.	Poca flexibilidad, ya que cada etapa debe ser completada antes de avanzar a la siguiente.	Comunicación limitada, ya que cada etapa se completa antes de avanzar a la siguiente.	El gerente tiene un papel limitado en el proceso de desarrollo.
Modelo en Espiral (Spiral)	Enfoque en la evaluación y mitigación de riesgos en cada etapa del proceso de desarrollo.	Planificación, Evaluación de Riesgos, Desarrollo y Evaluación del Prototipo, Implementación y Pruebas.	Mayor flexibilidad, ya que se enfoca en la mitigación de riesgos en cada etapa del proceso.	Comunicación continua entre el equipo de desarrollo y los stakeholders.	El gerente juega un papel importante en la identificación y mitigación de riesgos en cada etapa del proceso.
Modelo Ágil (Agile)	Enfoque en la iteración continua y la entrega de software funcional.	Planificación, Diseño, Desarrollo, Pruebas y Entrega.	Muy flexible, ya que enfatiza la iteración continua y la adaptabilidad.	Comunicación constante entre los miembros del equipo de desarrollo y los stakeholders.	El gerente juega un papel menos centralizado y más colaborativo en el proceso de desarrollo

Nota: Fuente: Elaboración propia.

1.4.2. Ventajas y desventajas del modelo dientes de Tiburón.

El modelo Dientes de Tiburón es una metodología de desarrollo de software que se enfoca en el control del proceso de desarrollo y la toma de decisiones estratégicas. A continuación, se presentan algunas de las ventajas y desventajas de este modelo.

Ventajas:

- Mejora la planificación y el control del proceso de desarrollo de software: este modelo proporciona un marco estructurado para la planificación y el control del proceso de desarrollo, lo que puede ayudar a garantizar que se cumplan los plazos y objetivos del proyecto.
- Enfatiza la comunicación y la colaboración: se pone gran énfasis en la comunicación y la colaboración entre el equipo de desarrollo y el gerente, lo que puede ayudar a garantizar que todos trabajen juntos de manera efectiva.
- Proporciona una estructura para la toma de decisiones estratégicas: este modelo proporciona una estructura clara para la toma de decisiones estratégicas, lo que puede ayudar a garantizar que las decisiones se tomen de manera informada y efectiva.

Desventajas:

- Menos flexible que otros modelos: Es menos flexible ya que enfatiza el control del proceso de desarrollo en lugar de la adaptabilidad a los cambios.
- Requiere una planificación precisa de los requerimientos: La precisión necesaria en la etapa de levantamiento de requerimientos puede suponer un reto debido a los requisitos cambian con frecuencia.
- Requiere una mayor participación y compromiso del gerente: Esto puede ser un desafío debido a que el compromiso por parte del gerente puede ser demandante y exhaustivo.
- Puede ser costoso y consumir más tiempo que otros modelos: La desventaja del costo se debería a la constante producción de prototipos los cuales también consideran una inferencia temporal en el desarrollo en el lapso que el cliente aprueba el prototipo presentado.

CAPÍTULO 2

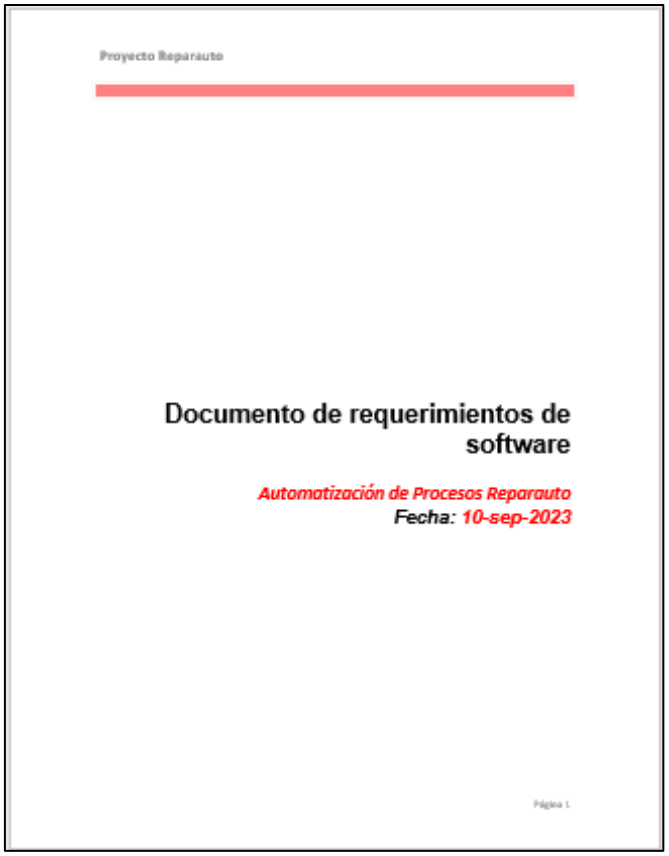
DESARROLLO

2.1. Automatización de los Módulos de Recepción de Vehículos y de Inventarios

Para el desarrollo de los módulos de Recepción de vehículos e Inventarios siguiendo el modelo “Sharktooth” el cual tiene como principio fundamental la interacción constante con el cliente mediante el diseño y muestra de prototipos para su posterior aprobación por parte de este, se optó por empezar el desarrollo siguiendo las directrices el modelo la cual inicia con “Requisitos de Sistema” seguido de” Análisis”. Para cumplir con estas dos fases se creó un documento de análisis de requerimientos en los cuales se detalla el propósito, alcance, funcionalidades, características, entorno operativo, requerimientos funcionales y No funcionales, entre otros.

Figura 10

Portada del documento de Requerimientos de Software



Nota: Fuente: Elaboración propia.

2.1.1. Requisitos Funcionales y No funcionales del Sistema

A continuación, se presentan los cuatro primeros requisitos funcionales y los cuatro primeros requisitos no funcionales descritos en el documento “Requerimientos de Software” con el Proyecto “Automatización de procesos Reparauto”

2.1.1.1. Requisitos Funcionales

Tabla 3

Requisito funcional N° 1.

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF01	Registrar Datos completos del cliente
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El software debe registrar los campos adecuadamente para almacenar información completa del cliente, incluyendo nombre, dirección, número de teléfono, correo electrónico y otros datos relevantes. Esto permitirá una gestión eficiente de la información del cliente en el proceso de recepción de vehículos.	

Nota: Fuente: Elaboración propia.

Tabla 4

Requisito funcional N° 2.

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF02	Capturar información del vehículo
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe permitir la captura de información detallada sobre el vehículo, incluyendo marca, modelo, año, número de placa y todos los datos establecidos por la Agencia Nacional de Tránsito para vehículos que son relevantes para la recepción.	

Nota: Fuente: Elaboración propia.

Tabla 5

Requisito funcional N° 3.

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF03	Almacenar información sobre el estado del vehículo
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
Debe ser posible registrar información sobre el estado del vehículo, incluyendo detalles de daños, kilómetros recorridos y todos los artículos extras que contienen los vehículos (Incluyendo artículos personales)	

Nota: Fuente: Elaboración propia.

Tabla 6

Requisito funcional N° 4.

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF04	Permitir la captura de fotografías del vehículo
Tipo:	**Prioridad:**
Empleados	Alta
Descripción:	
Los técnicos deben poder tomar fotografías del vehículo para documentar su estado y los daños antes de la reparación.	

Nota: Fuente: Elaboración propia.

En el apartado de Anexos se adjuntan los requerimientos completos funcionales y no funcionales, usados para la construcción de los dos módulos.

2.1.1.2. Requisitos No Funcionales

Tabla 7

Requisito No funcional N° 1.

Especificación de Requerimientos No Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQNF01	Tiempo de Respuesta
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe tener un tiempo de respuesta promedio por debajo de 2 segundos para garantizar una experiencia de usuario fluida.	
Manejo de errores:	
El sistema debe mostrar mensajes de error claros y amigables para guiar al usuario en caso de problemas.	
Criterios de aceptación:	
El tiempo de respuesta promedio del sistema debe ser inferior a 2 segundos, medido en condiciones de uso normales.	

Nota: Fuente: Elaboración propia.

Tabla 8

Requisito No funcional N° 2.

Especificación de Requerimientos No Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQNF02	Seguridad de Datos
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe garantizar la seguridad de los datos del cliente, aplicando medidas de cifrado y acceso restringido.	
Manejo de errores:	
Se deben registrar y notificar los intentos de acceso no autorizado a los datos.	
Criterios de aceptación:	
Los datos de los clientes deben estar cifrados y protegidos contra accesos no autorizados.	

Nota: Fuente: Elaboración propia.

Tabla 9

Requisito No funcional N° 3.

Especificación de Requerimientos No Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQNF03	Escalabilidad
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe ser escalable para manejar un aumento futuro en la cantidad de usuarios y datos.	
Manejo de errores:	
Se debe realizar un monitoreo continuo del rendimiento del sistema para detectar cuellos de botella y planificar expansiones.	
Criterios de aceptación:	
El sistema debe poder manejar un aumento del 50% en la cantidad de usuarios sin una degradación significativa del rendimiento.	

Nota: Fuente: Elaboración propia.

Tabla 10

Requisito No funcional N° 4.

Especificación de Requerimientos No Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQNF04	Compatibilidad de Navegadores
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe ser compatible con los navegadores más comunes, como Google Chrome, Mozilla Firefox, Microsoft Edge y Safari.	
Manejo de errores:	
Se debe realizar pruebas de compatibilidad en cada uno de los navegadores compatibles.	
Criterios de aceptación:	
El sistema debe funcionar correctamente en las últimas dos versiones de los navegadores mencionados.	

Nota: Fuente: Elaboración propia.

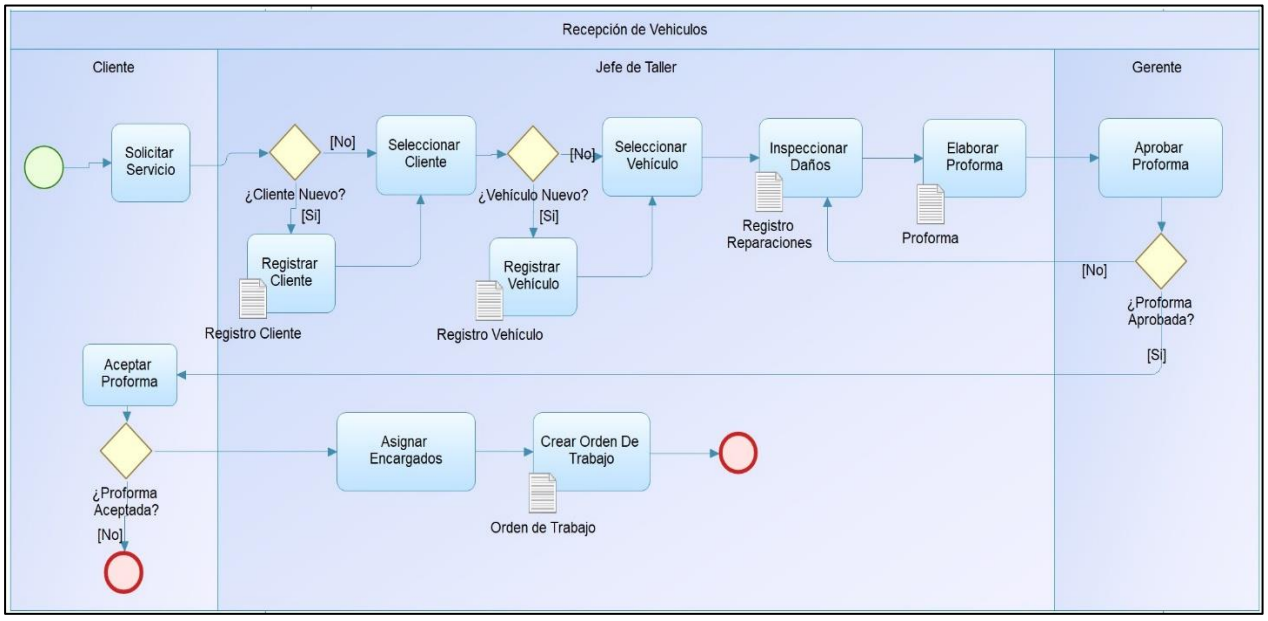
2.1.2. Procesos de Recepción e Inventarios

Para automatizar los dos procesos mencionados se optó por formalizar dichos procesos ya que la empresa actualmente no cuenta con ningún proceso documentado y es fundamental que cada proceso y subproceso este debidamente documentado y diagramado para una correcta automatización.

A continuación, se muestran los diagramas de los procesos de Recepción de vehículos y el de Inventarios. Estos procesos fueron diagramados bajo el esquema BPMN 2.0, el cual es uno de los más usados en el desarrollo de lógicas de negocio, adicional, se usó el programa PowerDesigner V.17 el cual contiene múltiples herramientas de modelado, una de ellas BPMN 2.0

Figura 11

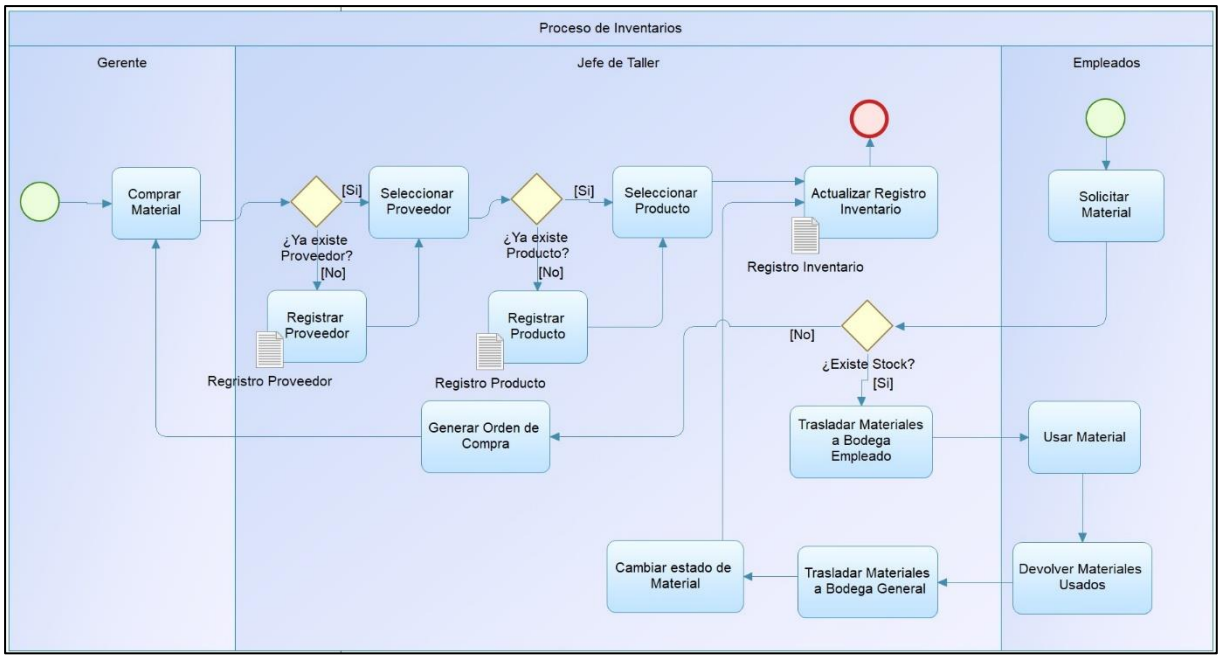
Diagrama de recepción de vehículos en BPMN 2.0



Nota: Fuente: Elaboración propia.

Figura 12

Diagrama de Inventarios en BPMN 2.0



Nota: Fuente: Elaboración propia.

2.1.3. Preparación de Entornos de Desarrollo

Una buena práctica de programación consiste en controlar y documentar las versiones de cada lenguaje, Framework, librería, entre otras. Que usamos en cada proyecto para eventualidades futuras cómo mantenimiento de software o en el caso de iniciar un proyecto con características similares.

Tabla 11

Versiones de entorno de desarrollo.

Entornos de Desarrollo		
Enfoque	**Nombre**	**Versión**
Modelado	SAP Power Designer	16.7
Base de Datos	PostgreSQL	15
	pgAdmin 4	7.2
Backend	Visual Studio Code	1.83.0
	Node	18.17.1
Frontend	Visual Studio Code	1.83.0
	Angular	14.2.12
Pruebas Unitarias	Jasmine	4.0.0

	Karma	6.4.3
	Postman	7.0.9
Pruebas de Integración	Cypress	12.6.0

Nota: Fuente: Elaboración propia.

En el caso del Backend específicamente con Node es necesario documentar el archivo que se crea por defecto cuando desplegamos un nuevo proyecto, dicho archivo lleva el nombre de “package.json” este archivo de configuración guarda toda la información acerca de las librerías y dependencias, así como también las respectivas versiones usadas en el proyecto.

Figura 13

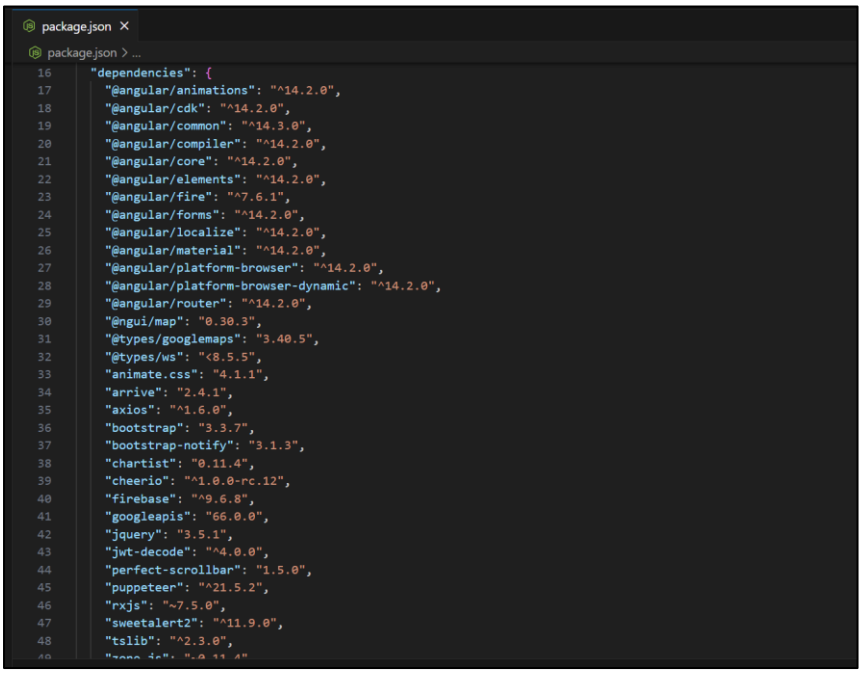
Archivo de Configuración de Node.

Nota: Fuente: Elaboración propia.

En el Frontend sucede algo similar por lo también se debe documentar el archivo “package.json” de configuración, en este caso se usó el Framework Angular basado en TypeScript.

Figura 14

Archivo de configuración de Angular.



Nota: Fuente: Elaboración propia.

2.1.4. Diseño de Base de datos

Luego de una exhaustiva revisión de los requerimientos, procesos actuales, datos históricos de los artefactos usados en la empresa (Documentación), etc. Se procedió a la normalización de la base de datos en conjunto para los dos módulos.

Para el diseño de la base de datos se usó el programa SAP PowerDesigner y el lenguaje SQL conocido específicamente cómo PostgreSQL para bases de datos.

En el artículo "PostgreSQL una alternativa efectiva en las empresas" por Henríquez et al., se resalta el valor de PostgreSQL como un potente sistema de base de datos objeto-relacional de código abierto. Se menciona que PostgreSQL cuenta con más de 15 años de desarrollo activo y una arquitectura que ha ganado reputación por su fiabilidad e integridad de datos (Henríquez N. et al., 2020).

2.1.4.1. Diagrama Físico de Base de Datos

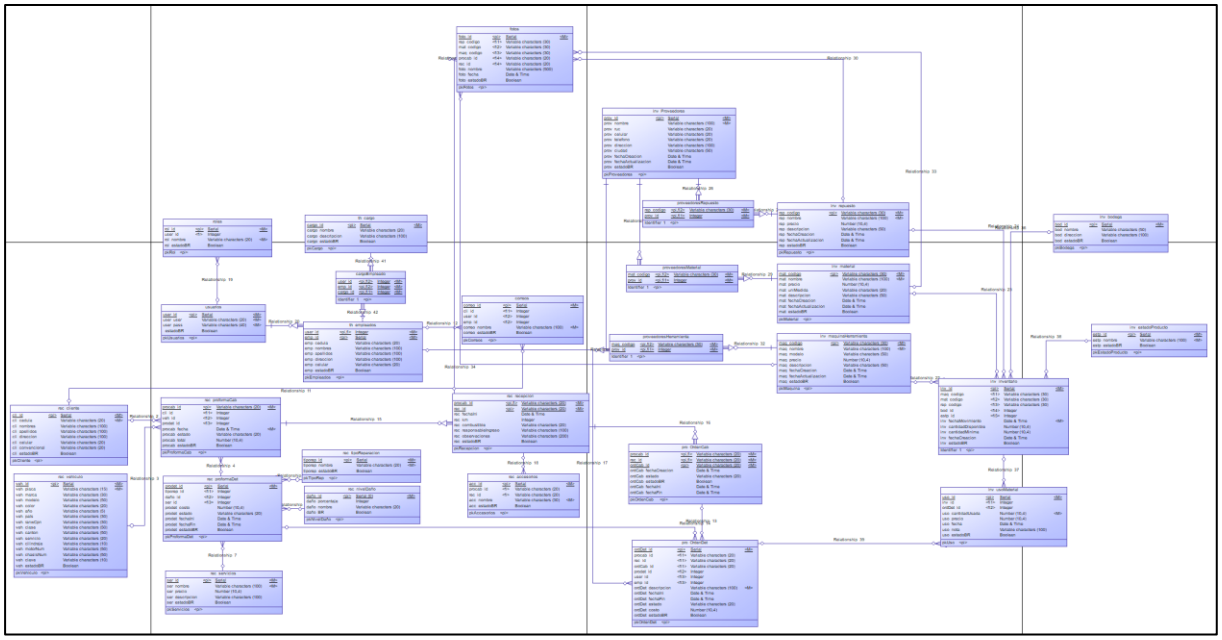
La normalización y modelado de la base de datos se realizó con el programa SAP Power Designer.

En el trabajo de investigación llamado "Enterprise Architecture Management Tool" menciona que SAP Sybase PowerDesigner es una herramienta eficaz para el modelado de procesos de negocio y datos. Esta herramienta es útil en la creación de una diversidad de modelos dentro del ámbito de la Arquitectura Empresarial, abarcando desde mapas de procesos hasta diagramas de arquitectura de aplicaciones y estructuras tecnológicas (Matthes et al., 2021).

La Base de datos cuenta con 29 tablas que gestionan la lógica del negocio incluyendo tablas como: Usuarios, roles, clientes, vehículos, inventarios, subprocesos. Entre otros.

Figura 15

Diagrama de Base de Datos

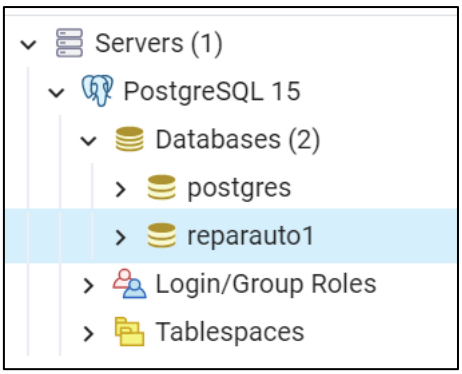


Nota: Fuente: Elaboración propia.

Después de haber diseñado la base de datos se exporta un archivo en formato ".sql" el cual se usa para crear la base de datos en PostgreSQL con la interfaz gráfica PGadmin 4.

Figura 16

Base de datos "Reparauto1" en PostgreSQL



Nota: Fuente: Elaboración propia.

A continuación, se muestran las tablas creadas para su respectivo uso dentro de PostgreSQL.

Figura 17

Tablas en la Base de datos "Reparauto1" en PostgreSQL

Nota: Fuente: Elaboración propia.

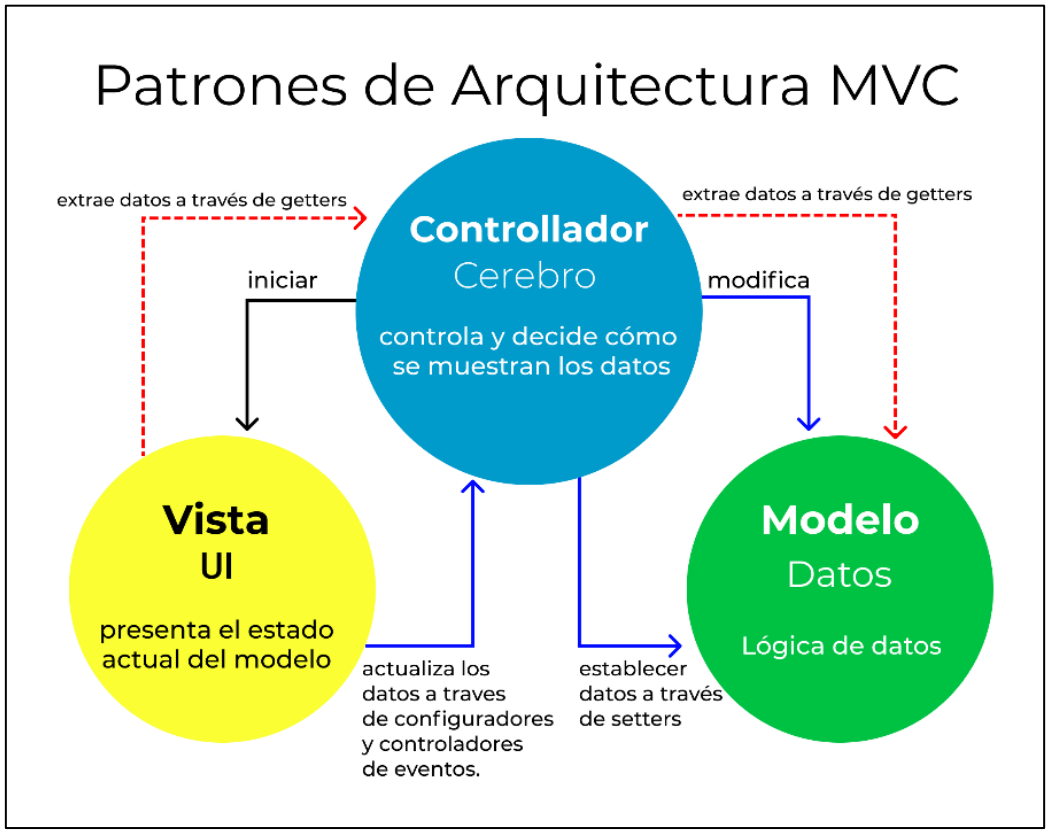
2.1.5. Desarrollo del Backend de la aplicación

2.1.5.1. Patrón de Diseño "MVC"

Para el desarrollo del Backend se optó por usar un patrón de diseño llamado MVC. El autor Gamaliel propone que el patrón preferible para el desarrollo de aplicaciones web es el Modelo Vista Controlador (MVC). Este enfoque implica la división del proyecto en tres componentes clave: la capa de control que se encarga de definir las funciones y acciones del proyecto sin entrar en detalles de implementación, la capa de negocios que aborda cómo se construye la aplicación, y la capa de presentación que se centra en la interacción del usuario con la aplicación (Gamaliel et al., 2012).

Figura 18

Patrón de diseño (MVC)



Nota: Fuente: El patrón modelo-vista-controlador. FreeCodeCamp. Recuperado de: https://www.freecodecamp.org/espanol/news/el-modelo-de-arquitectura-view-controller-pattern/

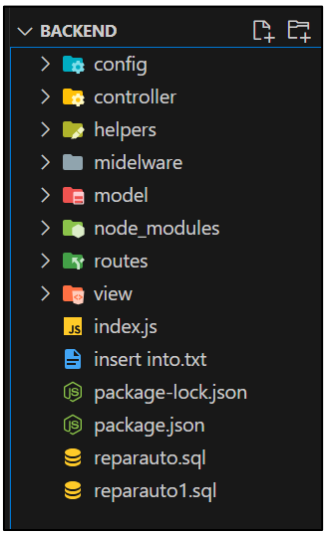
2.1.5.2. Estructura del Backend

Dado que el enfoque de desarrollo se estableció en una arquitectura basada en microservicios existe una variante (Modelo – Ruta – Controlador), en la que la “Vista” se remplaza por la “Ruta” la cual sirve de comunicación directa con el Frontend de la aplicación mediante el consumo de “API’s”.

A continuación, se muestra la estructura en carpetas del Backend:

Figura 19

Estructura del Backend



Nota: Fuente: Elaboración propia.

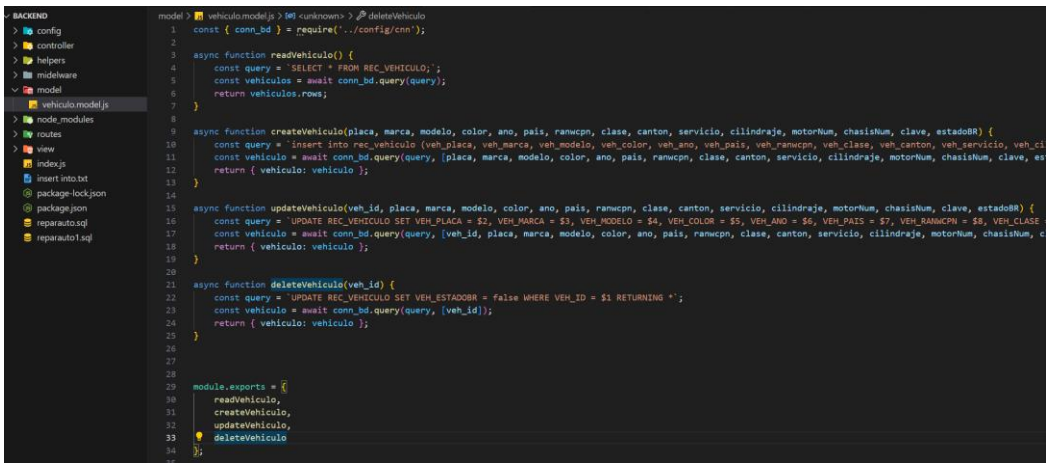
2.1.5.3. Modelo de la tabla “Vehículo”.

El modelo de la tabla "vehículo" proporciona una estructura y definición para almacenar y gestionar datos relacionados con vehículos en la base de datos. Esto facilita la organización, consulta y manipulación de la información de vehículos de la aplicación.

Cabe destacar que las 29 tablas fueron diseñadas con una lógica similar a la descrita en el modelo, controlador y ruta que se muestran a continuación.

Figura 20

Modelo de la tabla Vehículo



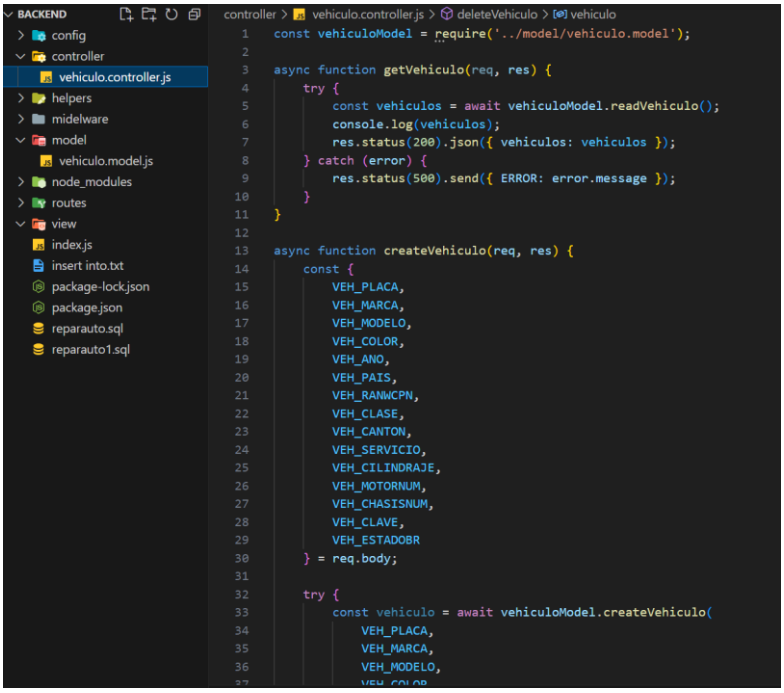
Nota: Fuente: Elaboración propia.

2.1.5.4. Controlador de la tabla “Vehículo”.

El controlador de la tabla "vehículo" actúa como el intermediario que conecta la ruta y el modelo, garantizando que las solicitudes del usuario se procesen correctamente y que los datos se gestionen de acuerdo con las reglas de negocio.

Figura 21

Controlador de la tabla Vehículo



Nota: Fuente: Elaboración propia.

2.1.5.5. Ruta de la Tabla "Vehículo".

A continuación, se presenta las rutas de vehículos la cual es el canal de comunicación directo con el Frontend de la aplicación. Aquí se establecen los nombres de las rutas de los métodos básicos CRUD de cada tabla.

Figura 22

Rutas de la tabla "Vehículo"

```
routes > vehiculos.routes.js > ...
1   const { Router } = require('express')
2
3   const vehiculoController = require('../controller/vehiculo.controller')
4   const scrapingController  = require('../controller/scrapingVehiculos.controller')
5
6   //  Declaramos middelware
7   const middelware = require('../middelware/token.middelware')
8
9   const route = Router();
10
11  //rutas protegidas
12  route.get('/vehiculo', middelware.protegerRuta, vehiculoController.getVehiculo);
13  route.post('/vehiculo', middelware.protegerRuta, vehiculoController.postCreateVehiculo);
14  route.put('/vehiculo', middelware.protegerRuta, vehiculoController.putUpdateVehiculo);
15  route.put('/vehiculoD', middelware.protegerRuta, vehiculoController.deleteVehiculo);
16
17  // Nueva ruta para buscar vehículo por placa
18  route.get('/vehiculo/:veh_placa', middelware.protegerRuta, vehiculoController.getVehiculoByPlaca);
19  route.get('/vehiculoScraping/:veh_placa', middelware.protegerRuta, scrapingController.getConsultaPorPlacas);
20
21
22  module.exports = route
```

Nota: Fuente: Elaboración propia.

2.1.6. Desarrollo del Frontend de la aplicación

El desarrollo del Frontend se llevó a cabo con el Framework Angular una plataforma desarrollada y mantenida por Google que se especializa en la creación de aplicaciones web de una sola página (SPA), optimizando la carga de datos de forma asíncrona para mejorar el rendimiento en dispositivos web y móviles.

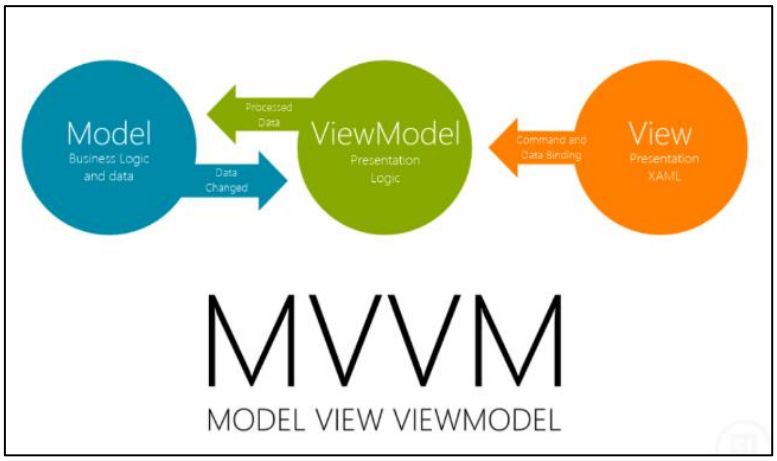
La utilización de TypeScript en Angular, un lenguaje que ofrece tipado estático y objetos basados en clases permite localizar errores de sintaxis antes de su ejecución, contribuyendo a un desarrollo web más eficiente y estructurado (Oriols & Antonio Gómez Gutiérrez, 2021).

2.1.6.1. Patrón de Diseño “MVVM”

Para el diseño del Frontend se aplicó el patrón de diseño conocido como “MVVM” (Modelo – Vista - VistaModelo).

Figura 23

Patrón de diseño (MVVM)



Nota: Fuente: Adaptado de “Patrón de Diseño MVVM. Clicko Blog. Recuperado de: https://blog.clicko.es/patron-diseno-mvvm-usando-wpf-parte-1/

Según la revisión de literatura presentada en el documento "Una revisión de los patrones de diseño de software aplicado a las aplicaciones web" (2020), el Modelo-Vista-VistaModelo (MVVM) se destaca como un patrón de diseño altamente eficiente para el desarrollo de aplicaciones Frontend, especialmente aquellas construidas con Angular.

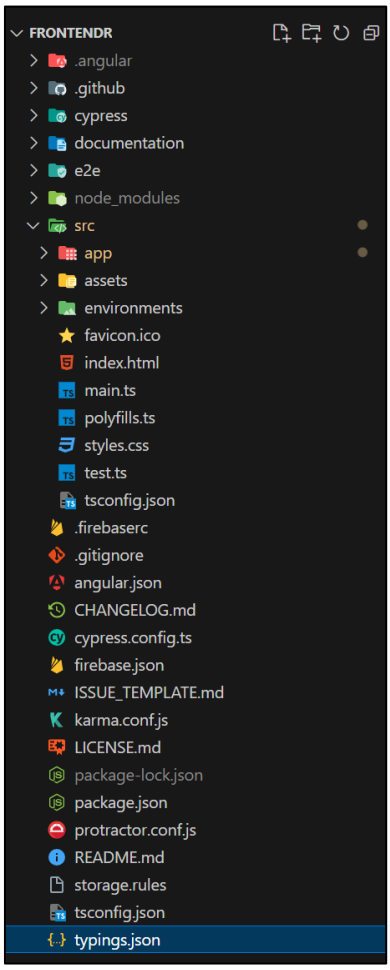
Este patrón facilita la separación clara de la lógica de negocio (Modelo) de la interfaz de usuario (Vista), con la adición de una capa intermedia (VistaModelo) que actúa como un puente entre ambos. Tal estructura no solo asegura una mejor organización y mantenibilidad del código, sino que también potencia la realización de pruebas y la gestión de dependencias de manera más eficiente (Optar et al., 2020).

2.1.6.1 Estructura del Frontend

El Frontend en Angular se estructura con el patrón MVVM, donde los Modelos representan la data, las Vistas son interfaces de usuario compuestas por componentes HTML y TypeScript, y los ViewModels gestionan la lógica. Incluye Servicios para la lógica de negocio, Módulos para agrupar funcionalidades, y archivos de configuración, todo integrado en un marco que facilita el desarrollo eficiente y escalable de aplicaciones web.

Figura 24

Estructura del Frontend



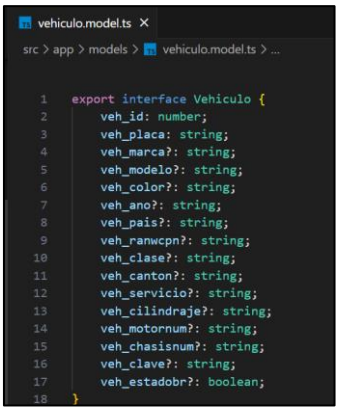
Nota: Fuente: Elaboración propia.

2.1.6.2. Modelo – Interfaz de "Vehículo"

El siguiente "Interfaz" es una forma de definir la estructura que debe tener los atributos del objeto en cuestión, en este caso es el Interfaz de vehículo.

Figura 25

Interfaz de Vehículo



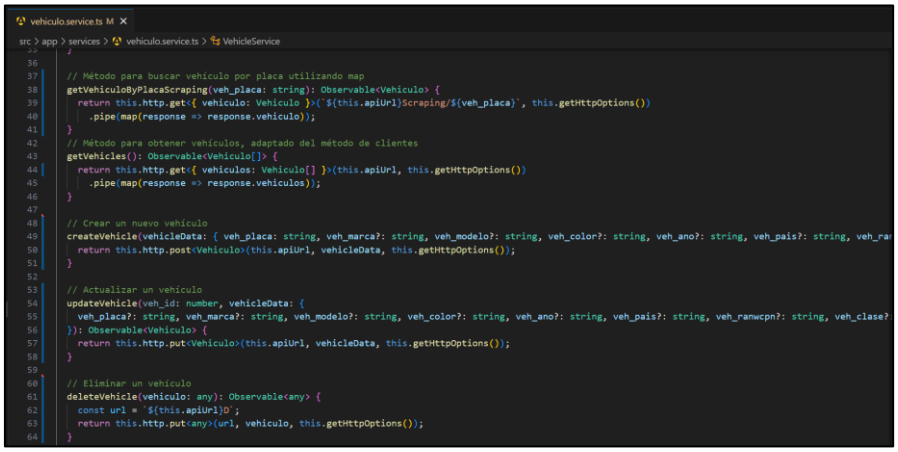
Nota: Fuente: Elaboración propia.

2.1.6.3. Servicio de "Vehículo"

El servicio de vehículo o "VehicleService" como se estructura en el Frontend es una clase de servicio en Angular diseñada para manejar operaciones relacionadas con vehículos, incluyendo la recuperación, creación, actualización y eliminación de datos de vehículos, así como la interacción con el Backend a través de una API RESTful.

Figura 26

Servicio de Vehículo



Nota: Fuente: Elaboración propia.

2.1.6.4. Componentes en Angular

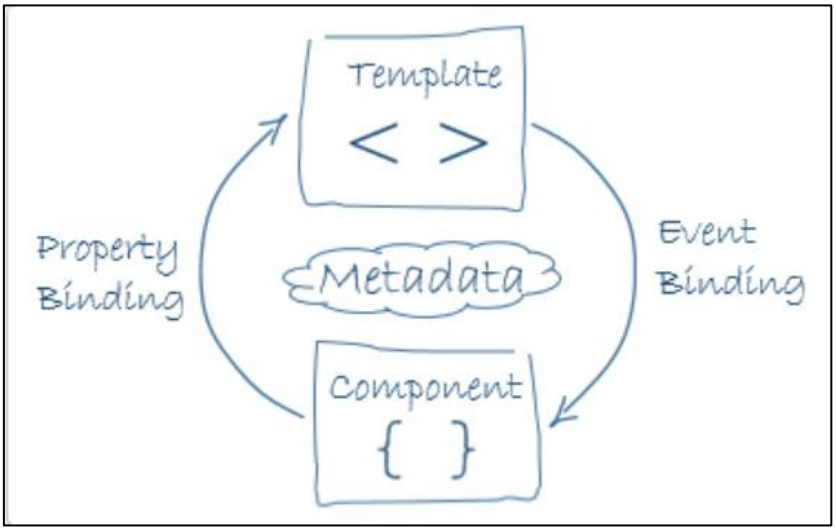
Como se detalla en la documentación oficial de Angular, los componentes son fundamentales en la arquitectura y el desarrollo de aplicaciones web con este Framework. Un componente en Angular es una clase que controla una parte de la pantalla denominada vista.

Cada componente encapsula la lógica específica de la aplicación que respalda la vista, interactuando con ella a través de una interfaz de propiedades y métodos. Esta arquitectura de componentes permite la creación de interfaces de usuario modulares y reutilizables, cada una con responsabilidades bien definidas y separadas. Por ejemplo, en el tutorial "Tour of Heroes", distintos componentes se encargan de la raíz de la aplicación, la lista de héroes y el editor de héroes. La interacción entre la lógica de aplicación y la vista facilita la creación de interfaces dinámicas y reactivas, esenciales para las aplicaciones web modernas (Angular , 2024).

A continuación, se muestra la arquitectura de comunicación bidireccional entre la plantilla (HTML) y la lógica del componente (TS):

Figura 27

Enlace bidireccional en los componentes de Angular



Nota: Fuente: Introducción a componentes y plantillas.

Recuperado de: https://docs.angular.lat/guide/architecture-components

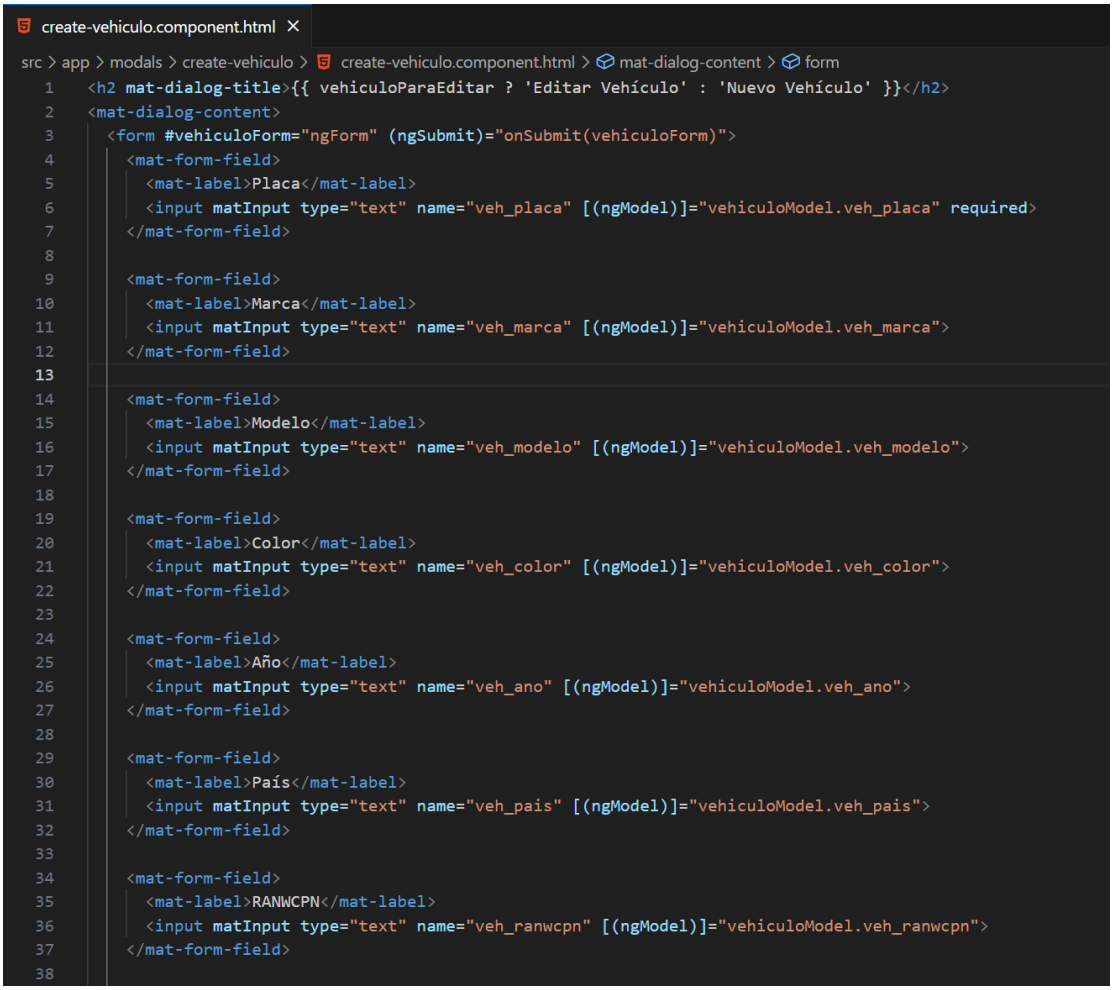
2.1.6.5. Plantilla HTML del componente Vehículos.

El HTML en la plantilla define la estructura de la interfaz de usuario que se renderizará en el navegador. Además de permitir el enlace bidireccional de datos con la lógica del componente (TS), que es muy necesario en la reactividad de formularios en Angular.

A continuación, se muestra un fragmento de la plantilla HTML del componente Vehículos.

Figura 28

HTML del componente Vehículo



Nota: Fuente: Elaboración propia.

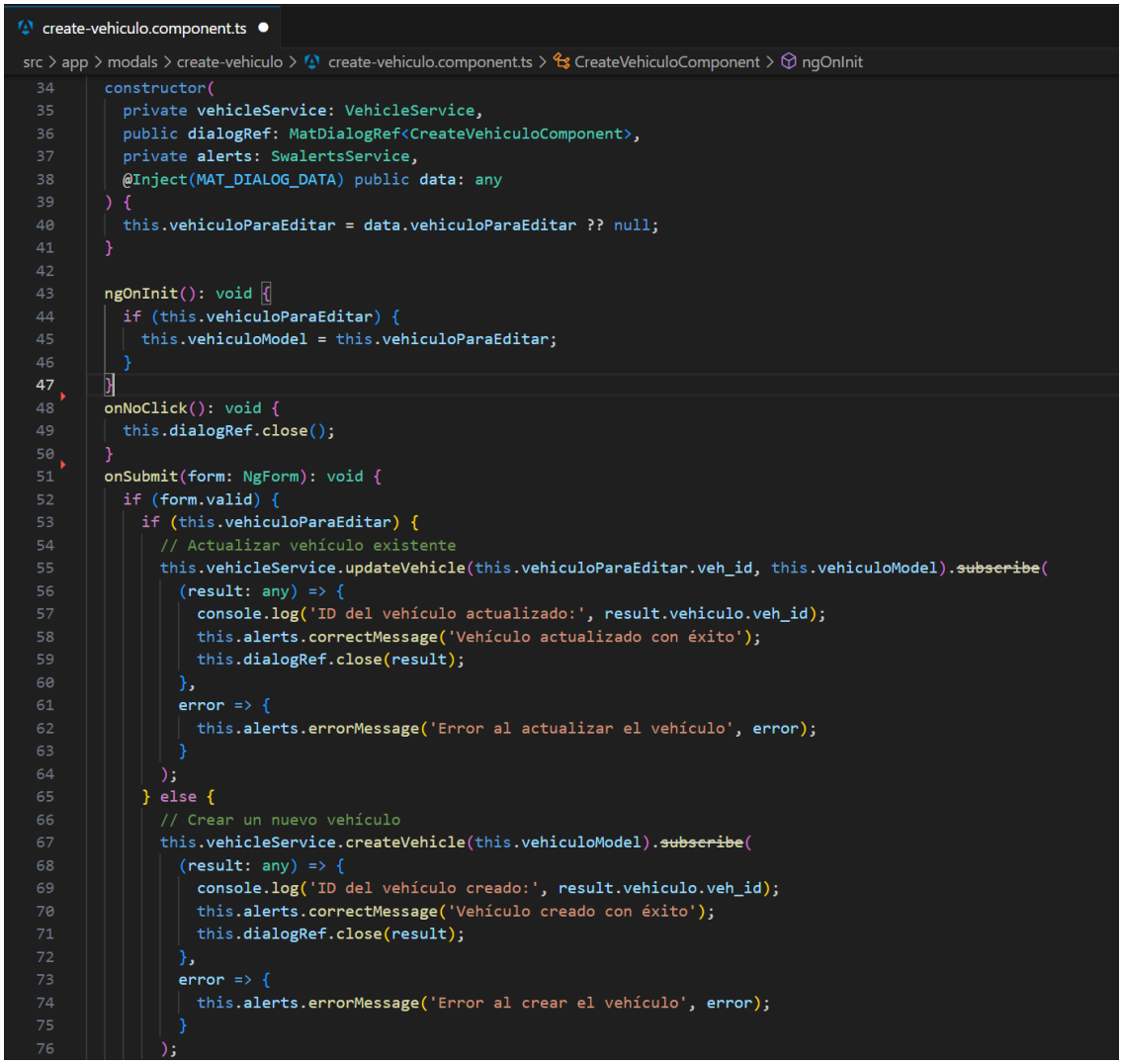
2.1.6.6. Plantilla TS del componente Vehículos.

El TS del componente “Vehículo” como se explicó anterior mente maneja la lógica de este, el cual se encarga de comunicarse con los servicios para crear o editar vehículos en la recepción y la vez mantiene una comunicación bidireccional y reactivo en tiempo real en el formulario el cual visualiza e interactúa el usuario final.

En la siguiente imagen se muestra un fragmento de código del TS del componente vehículo:

Figura 29

TS del componente Vehículo



Nota: Fuente: Elaboración propia.

2.1.6.7. Plantilla SCSS del componente vehículos.

SCSS, conocido como “Sassy CSS”, es una extensión del lenguaje CSS que incorpora características adicionales como variables, reglas anidadas, mixins y funciones, todo ello manteniendo una sintaxis completamente compatible con CSS.

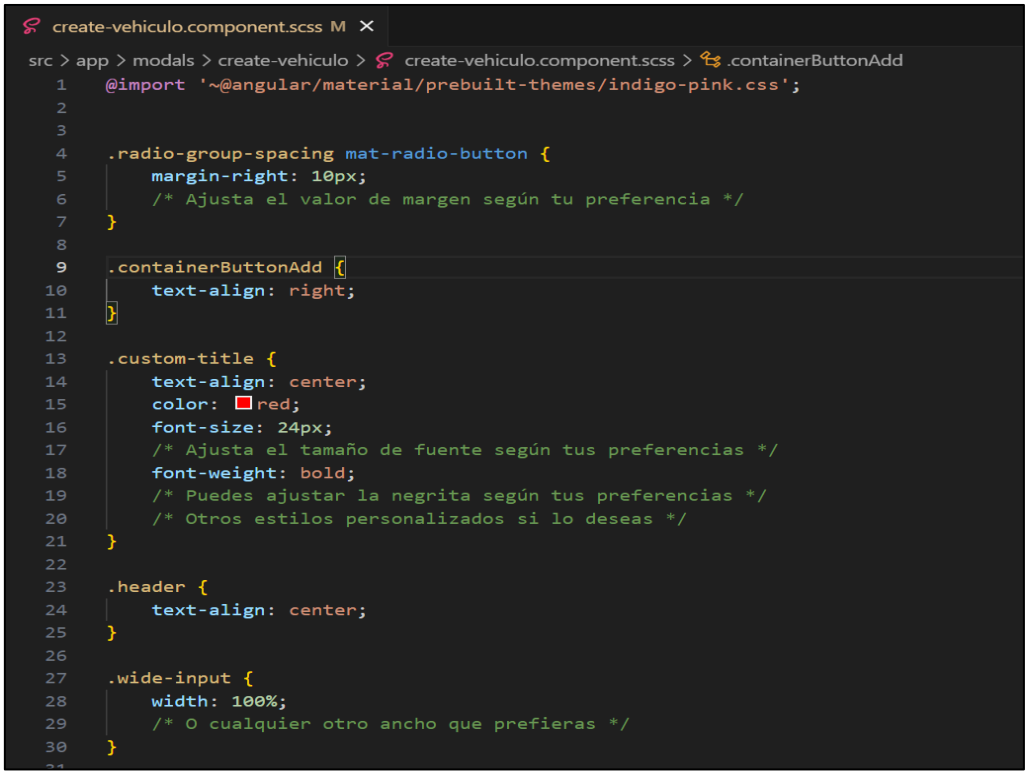
Este preprocesador de CSS ayuda a los desarrolladores a escribir hojas de estilo más eficientes y organizadas, facilitando la gestión de proyectos de diseño a gran escala.

SCSS utiliza la extensión de archivo .scss y es un superconjunto de CSS, lo que significa que todo CSS válido es también SCSS válido. Esta similitud hace que SCSS sea la sintaxis más popular y fácil de usar. Además, SCSS soporta dos sintaxis diferentes: la propia de SCSS y la sintaxis sangrada, original de Sass, que utiliza sangría en lugar de llaves y punto y coma (Sass, 2024).

Dentro del archivo SCSS que se muestra a continuación se definieron los estilos respectivos para el componente de vehículos:

Figura 30

SCSS del componente Vehículo



Nota: Fuente: Elaboración propia.

2.1.7. Presentación de Prototipos

Siguiendo el modelo de desarrollo “Dientes de tiburón” que fue establecido para este proyecto el cual basa sus etapas en entregar prototipos de la aplicación al cliente con cada avance importante y con cambios y/o correcciones establecidas por parte del Stakeholder. Se presentan a continuación los prototipos validados por el cliente en el proceso de desarrollo.

Tabla 12

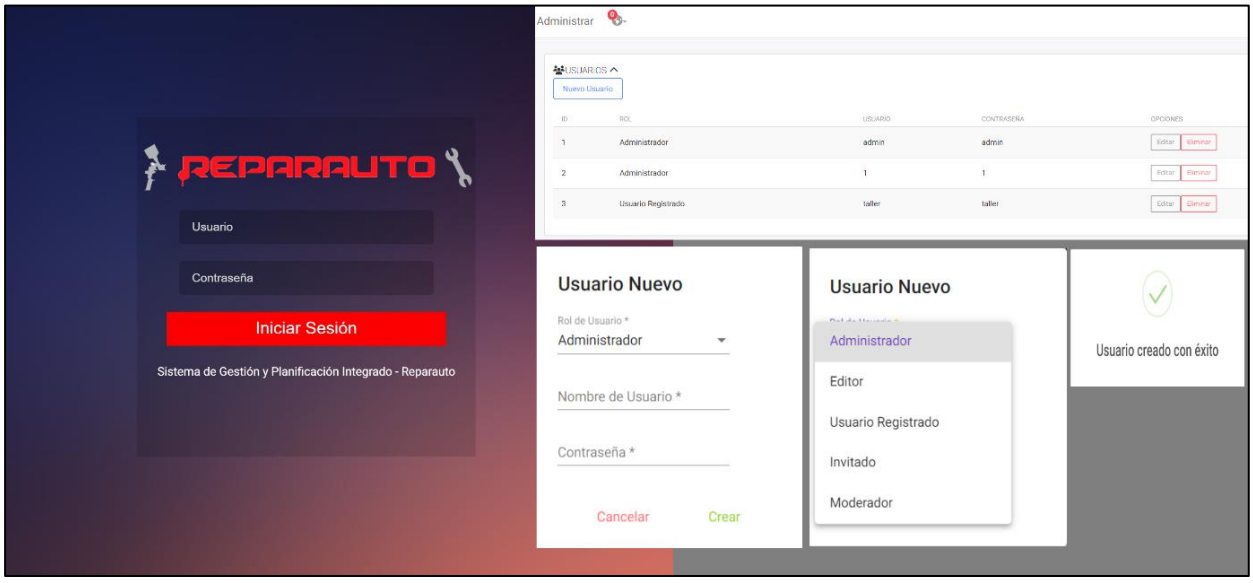
Prototipo N° 01

Presentación de Prototipo N° 01		**Fecha: 05/10/2023**
Proyecto:	**Resumen de Progreso:**	**Hitos Alcanzados**
Automatización de los Módulos: • **Recepción de Vehículos** • **Inventarios**	Creación de Backend completo, Desarrollo de la plantilla a usarse en el Front.	✓ Log in Finalizado. ✓ Interfaz de Vehículos Finalizado.
Próximo Hito:	**Solicitud de Feedback**	**Cronograma**
• **Desarrollo de las interfaces de administración.** • **Desarrollo de la interfaz de Proforma.**	Opinión sobre la usabilidad del prototipo actual.	Entrega del segundo prototipo: **20/10/2023**

Nota: Fuente: Elaboración propia.

Figura 31

Prototipo web 01



Nota: Fuente: Elaboración propia.

Tabla 13

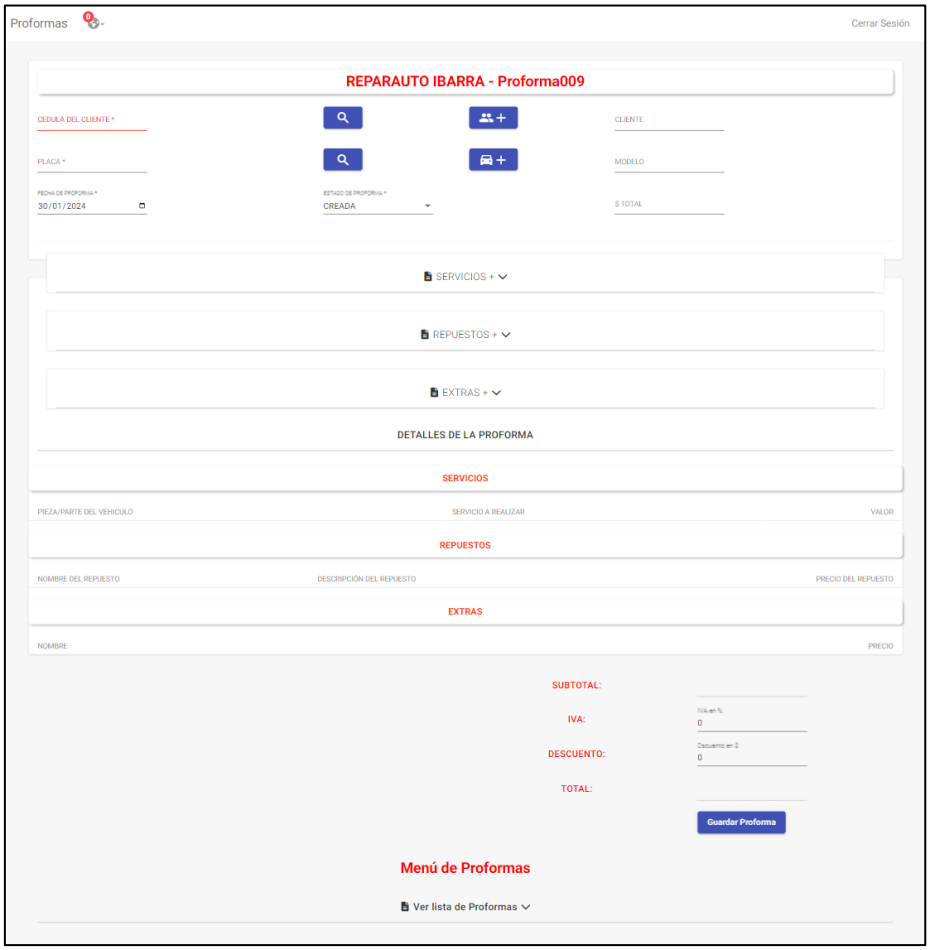
Prototipo N° 02

Presentación de Prototipo N° 02		Fecha: 21/10/2023
Proyecto:	**Resumen de Progreso:**	**Hitos Alcanzados**
Automatización de los Módulos: • **Recepción de Vehículos** • **Inventarios**	Creación de todo el interfaz de administración.	✓ Administración Finalizado. ✓ Proforma Finalizado
Próximo Hito:	**Solicitud de Feedback**	**Cronograma**
• **Desarrollo del interfaz de recepción** • **Desarrollo de la interfaz de Orden de trabajo.**	Opinión sobre la usabilidad y funcionalidad del prototipo actual.	Entrega del tercer prototipo: **06/11/2023**

Nota: Fuente: Elaboración propia.

Figura 32

Prototipo web 02



Nota: Fuente: Elaboración propia.

Tabla 14

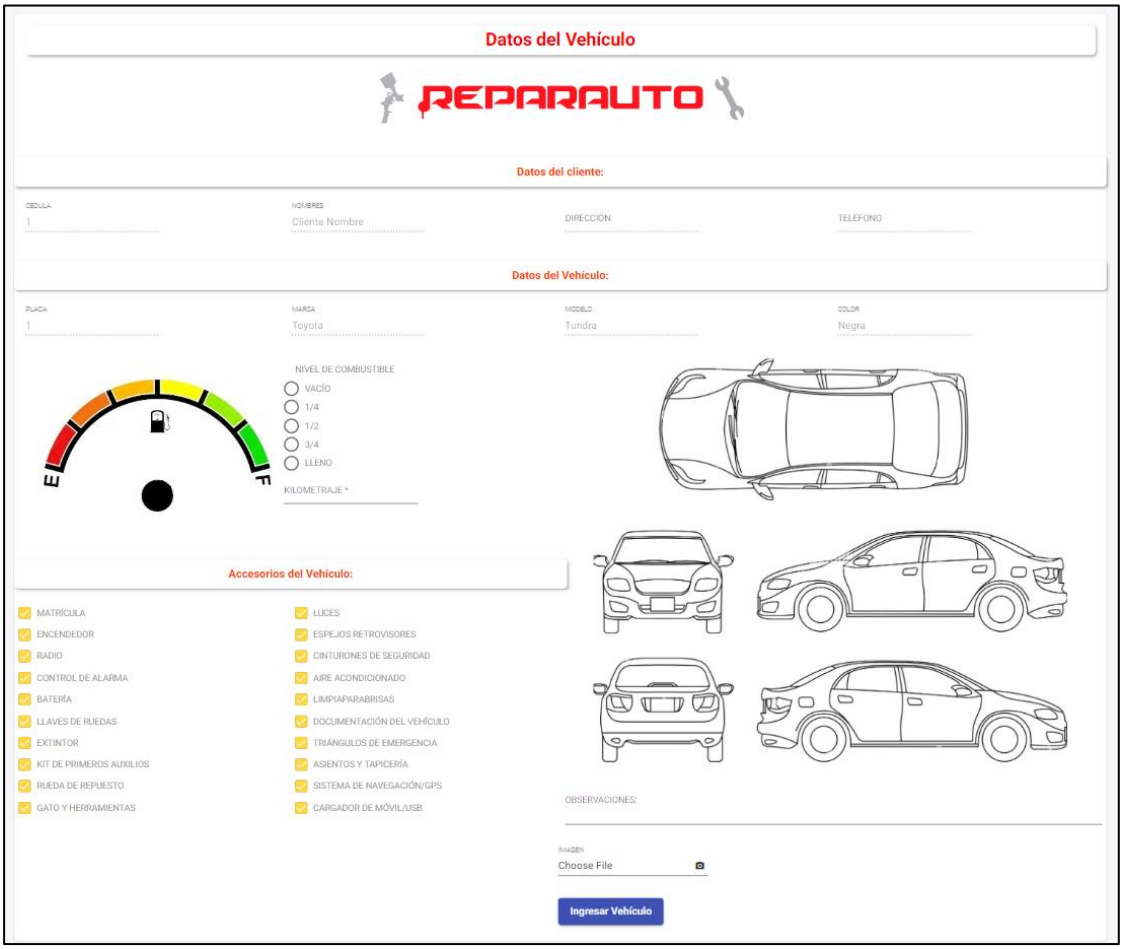
Prototipo N° 03

Presentación de Prototipo N° 03		Fecha: 07/11/2023
Proyecto:	**Resumen de Progreso:**	**Hitos Alcanzados**
Automatización de los Módulos: • **Recepción de Vehículos** • **Inventarios**	Creación del interfaz completo de recepción de vehículos y desarrollo de la interfaz de orden de trabajo	✓ Recepción Finalizado. ✓ Orden de trabajo Finalizado
Próximo Hito:	**Solicitud de Feedback**	**Cronograma**
• **Desarrollo del interfaz crear Inventarios** • **Desarrollo de la interfaz de cargar Stock**	Opinión sobre los cambios realizados en la proforma	Entrega del cuarto prototipo: **20/11/2023**

Nota: Fuente: Elaboración propia.

Figura 33

Prototipo web 03



Nota: Fuente: Elaboración propia.

Tabla 15

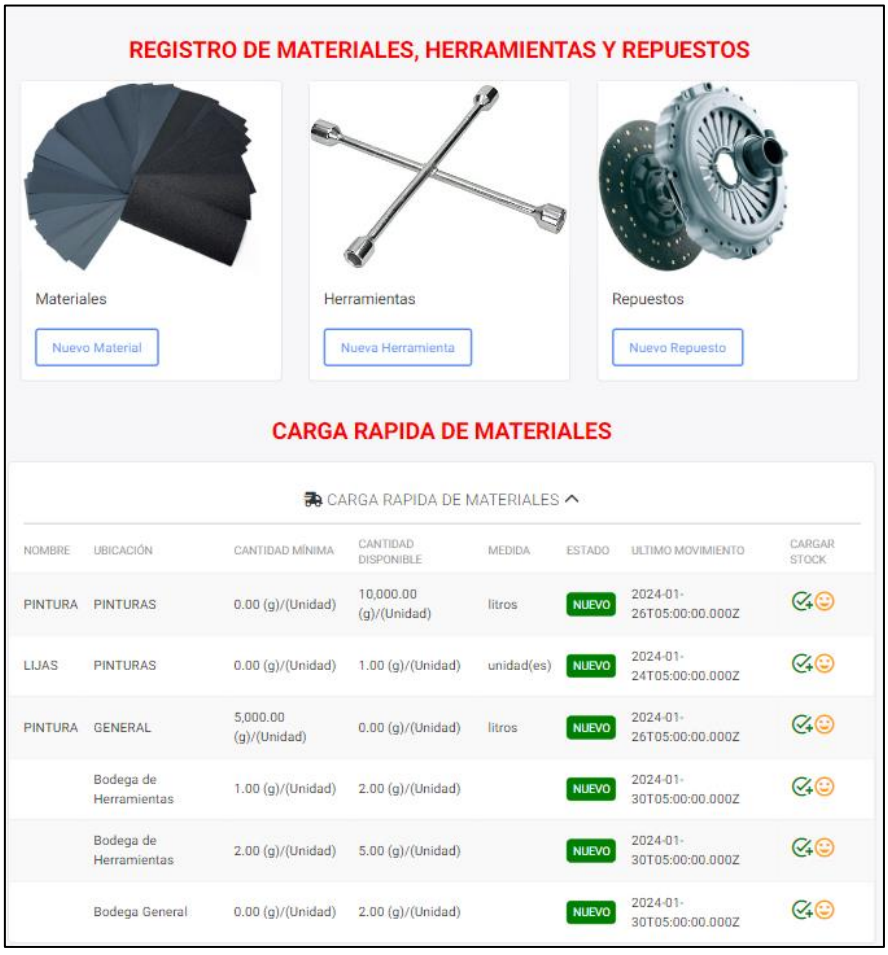
Prototipo N° 04

Presentación de Prototipo N° 04		Fecha: 20/11/2023
Proyecto:	**Resumen de Progreso:**	**Hitos Alcanzados**
Automatización de los Módulos: • **Recepción de Vehículos** • **Inventarios**	Creación del interfaz completo de crear Inventarios y Finalización del interfaz cargar stock	✓ Crear Inventarios Finalizado. ✓ Cargar Stock Finalizado
Próximo Hito:	**Solicitud de Feedback**	**Cronograma**
• **Desarrollo del interfaz de existencias** • **Desarrollo de la interfaz de transacciones**	Opinión la funcionalidad del Prototipo.	Entrega del prototipo Final: **05/12/2023**

Nota: Fuente: Elaboración propia.

Figura 34

Prototipo web 04



Nota: Fuente: Elaboración propia.

Tabla 16

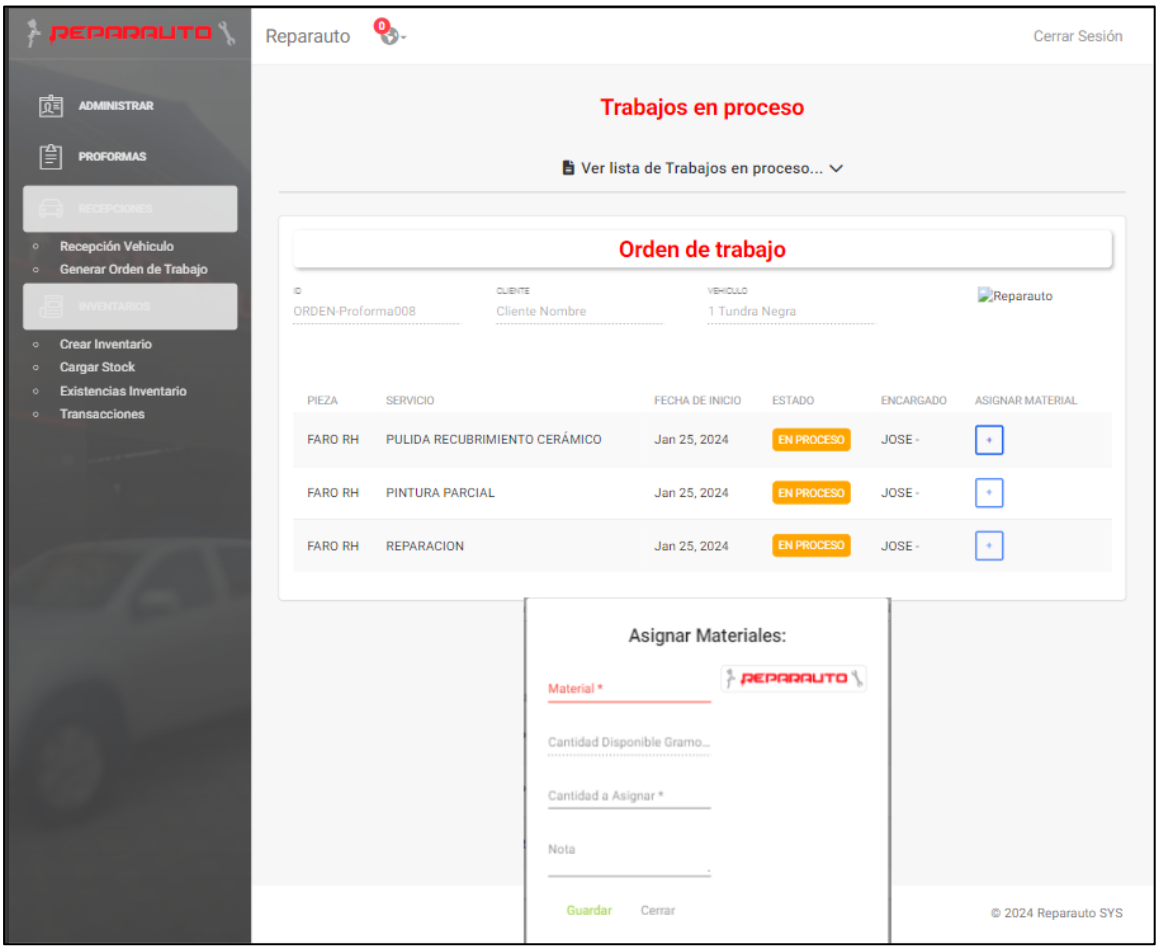
Prototipo Final

Presentación de Prototipo Final		Fecha: 06/12/2023
Proyecto:	**Resumen de Progreso:**	**Hitos Alcanzados**
Automatización de los Módulos: • **Recepción de Vehículos** • **Inventarios**	Finalización de los dos módulos propuestos	✓ Existencias Finalizado ✓ Transacciones Finalizado
Próximo Hito:	**Solicitud de Feedback**	**Cronograma**
• **Correcciones del feedback**	Satisfacción con el software Final.	**No Programado**

Nota: Fuente: Elaboración propia.

Figura 35

Prototipo web final



Nota: Fuente: Elaboración propia.

2.1.8. Pruebas Unitarias

Para la aplicación de pruebas unitarias se usó las librerías “Karma” y “Jasmine”. Karma es una prueba runner desarrollado por el equipo de AngularJS, que facilita la ejecución de pruebas escritas en Jasmine, Mocha, QUnit, etc., en diferentes navegadores de manera automatizada. Karma es ideal para proyectos donde se necesita probar el código en múltiples navegadores o entornos.

Por otro lado, Jasmine es un Framework de pruebas para JavaScript, que permite escribir pruebas de una manera fácil y legible, utilizando una sintaxis que describe los casos de prueba de forma clara.

2.1.8.1. Configuración del entorno de pruebas Unitarias

A continuación, se muestra el archivo de configuración “Karma.conf.js” el cual es esencial para el correcto funcionamiento del proceso de pruebas unitarias en Angular.

Figura 36

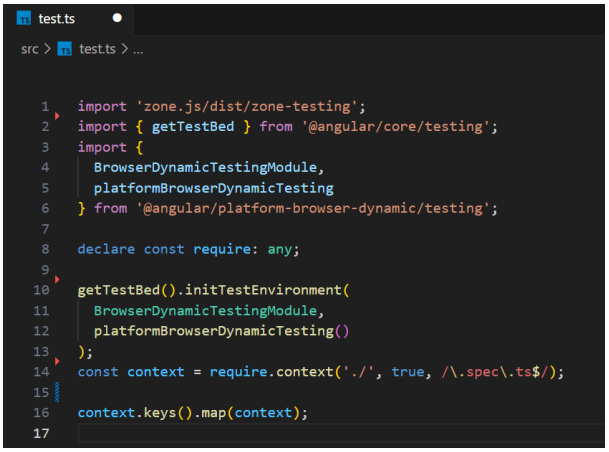
Archivo “Karma.conf.js”

Nota: Fuente: Elaboración propia.

El archivo “test.ts” es un archivo de configuración utilizado en el entorno de pruebas de Angular. Su función principal es inicializar el entorno de prueba de Angular y cargar dinámicamente todos los archivos de especificaciones de prueba (archivos. “spec.ts”) para que puedan ejecutarse durante las pruebas unitarias.

Figura 37

Archivo “Test.ts”



Nota: Fuente: Elaboración propia.

En la configuración del entorno de pruebas unitarias fue crucial configurar el archivo “tsconfig.spec.json” debido a que juega un papel crucial al definir las configuraciones específicas de compilación para el código TypeScript además de proporcionar un entorno de prueba aislado y optimizado.

Figura 38

Archivo “tsconfig.spec.json”

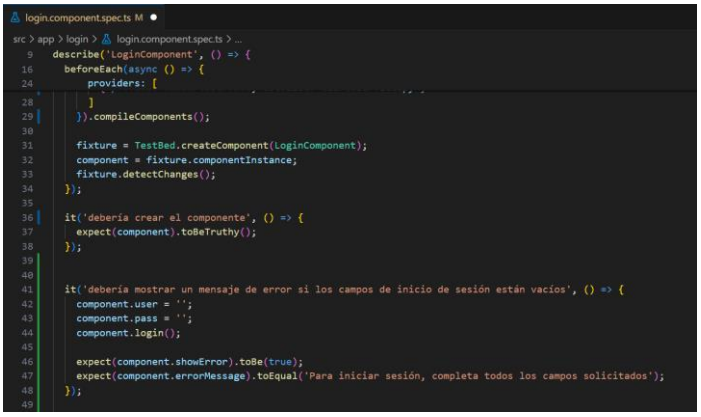
Nota: Fuente: Elaboración propia.

2.8.1.2. Pruebas unitarias al “Login” del aplicativo.

En la siguiente Figura se observan las pruebas unitarias realizadas al componente del “Login” del aplicativo. Estas pruebas comprenden los distintitos casos a los que se expone el componente “Login” como credenciales incorrectas, faltantes entre otras.

Figura 39

Archivo “Login.component.spec.js”



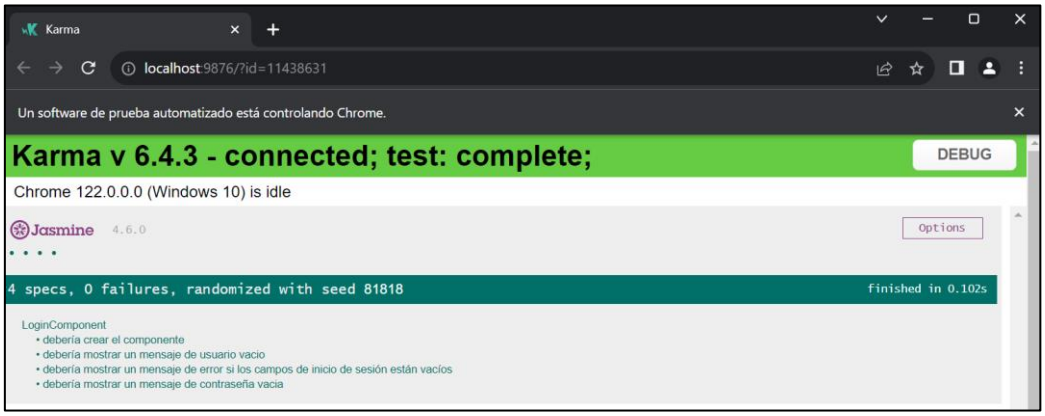
Nota: Fuente: Elaboración propia.

2.8.2. Resultado de las Pruebas Unitarias del componente “Login”.

Como resultado de la ejecución de las distintas pruebas unitarias, en la siguiente imagen observamos específicamente los resultados obtenidos de las pruebas unitarias aplicadas al componente “Login”.

Figura 40

Resultado de las pruebas



Nota: Fuente: Elaboración propia.

2.2. Creación del plan de pruebas de Integración

Para la creación del plan de pruebas de Integración se utilizaron específicamente las partes 2 y 4 de la norma ISO/IEC/IEEE 29119. Obteniendo como resultado un documento de 20 páginas titulado “Plan de pruebas de integración basado en la Norma ISO/IEC/IEEE 29119” donde se detalla el proyecto “Automatización de procesos de Recepción de vehículos y de Inventarios”.

Este documento consta de 9 partes las cuales son: Introducción, Contexto de las pruebas, Comunicación de las pruebas, Registro de las Pruebas, Estrategia de Prueba, Actividades y estimados de Prueba, Personal, Cronograma y Anexos.

Figura 41

Portada del plan de pruebas.

Nota: Fuente: Elaboración propia.

2.3. Aplicación del plan de pruebas de Integración

Para la aplicación del plan de pruebas de Integración se usó los dos casos de pruebas de integración de componentes detallados en el plan de pruebas.

2.3.1. Interfaces de Usuario del Módulo de Recepción de Vehículos.

Tabla 17

Caso de prueba de Integración TC-001

ID Caso de prueba	**TC-001**
Título	Integración de Módulo de Recepción
Descripción	Probar la integración completa del módulo de recepción de vehículos, incluyendo el registro de vehículos y clientes, y la captura y carga de imágenes todo desde el inicio de sesión.
Precondiciones	El usuario se encuentra en la página de inicio de sesión.
Pasos de la Prueba	1. Ingresar las credenciales de inicio de sesión 2. Ingresar a subproceso “Proforma” 3. Ingresar las placas del vehículo 4. Ingresar la cedula del Cliente 5. Registrar daños y valores de reparación 6. Confirmar y guardar información en el sistema 7. Ingresar a subproceso “Recepciones” 8. Registrar información del vehículo (Estado, Accesorios, fecha. etc.) 9. Subir fotografías del vehículo 10. Confirmar y guardar información en el sistema
Resultados Esperados	Los detalles del cliente y del vehículo se registran correctamente. Las imágenes se capturan/suben y se asocian con el registro del vehículo en el sistema. Se registra y se genera una orden de trabajo asociada al vehículo.
Postcondiciones	El registro del vehículo, junto con las imágenes y los detalles del cliente, se guarda y se puede acceder a él en el sistema

Nota: Fuente: Elaboración propia.

2.3.2. Interfaces de Usuario del Módulo de Inventarios.

Tabla 18

Caso de prueba de Integración TC-002

ID Caso de prueba	**TC-002**
Título	Integración de Módulo de Inventarios
Descripción	Verificar la funcionalidad completa de las interfaces de usuario para la visualización y gestión del inventario de partes y suministros, y para actualizar el estado del inventario y registrar el uso de recursos.
Precondiciones	El sistema debe estar operativo y accesible al usuario.
Pasos de la Prueba	1. Ingresar las credenciales de inicio de sesión 2. Navegar al Módulo de Inventarios 3. Revisar la Lista de materiales disponibles 4. Seleccionar un material y actualizar la cantidad y/o estado 5. Revisar la Lista de herramientas disponibles 6. Seleccionar una herramienta y actualizar la cantidad y/o estado 7. Revisar la Lista de maquinaria disponibles 8. Seleccionar una maquina y actualizar la cantidad y/o estado 9. Asignar un material a un proceso específico 10. Confirmar y guardar los cambios realizados en el inventario.
Resultados Esperados	Las pantallas muestran la información actual del inventario. Las actualizaciones y registros de uso se procesan y reflejan correctamente en el sistema.
Postcondiciones	Los cambios en el inventario se guardan de forma permanente y están visibles para futuras consultas y gestiones.

Nota: Fuente: Elaboración propia.

2.3.3. Ejecución de Pruebas de Integración

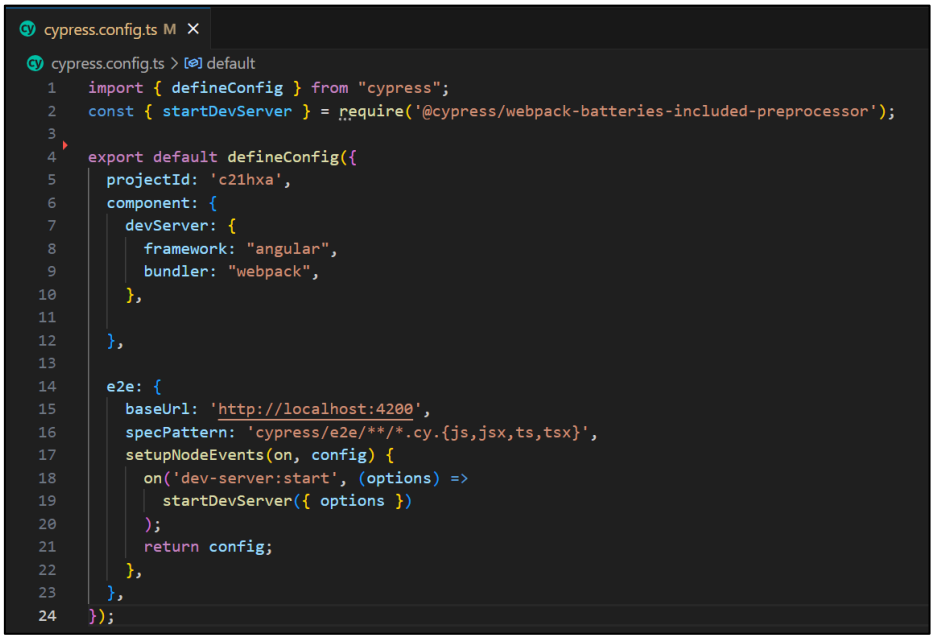
Para la ejecución de las pruebas de integración se utilizó el marco de pruebas llamado Cypress, esta biblioteca automatiza las pruebas de integración en múltiples navegadores. Los dos casos de pruebas de integración los cuales son “TC001” y “TC002” se escribieron paso a paso bajo la sintaxis de Cypress.

2.3.3.1. Configuración de entorno

Para la instalación y configuración de Cypress el cual es el entorno de pruebas de integración únicamente se ejecutó el comando “npm install Cypress” y posterior se configura el archivo “Cypress.config.ts” el cual contiene los parámetros necesarios para el correcto funcionamiento. En la siguiente imagen se puede observar el archivo antes mencionado:

Figura 42

Archivo de configuración de Cypress.



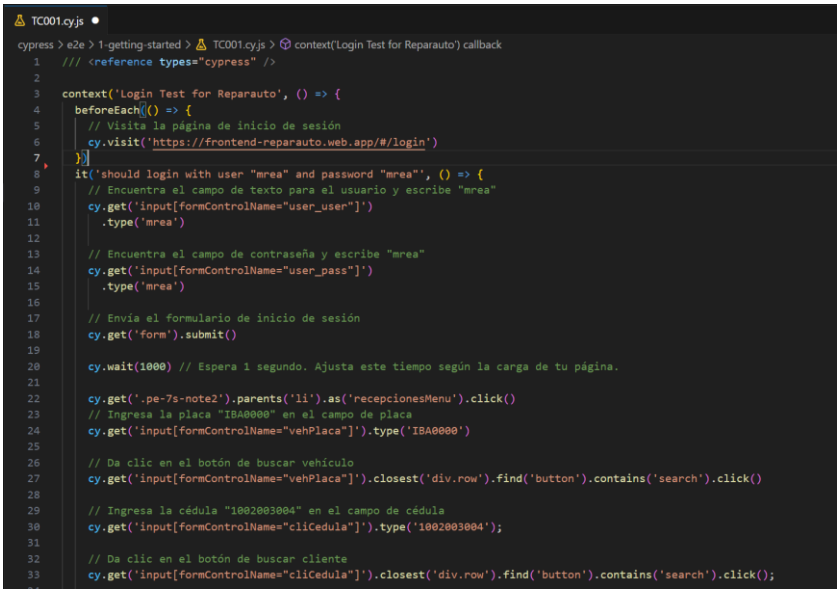
Nota: Fuente: Elaboración propia.

2.3.3.2. Ejecución de pruebas automatizadas.

En las siguientes ilustraciones se muestran fragmentos de los archivos “TC001.cy.js” con 117 líneas de código y “TC002.cy.js” con 66 líneas de código los cuales contienen el código correspondiente a cada caso de prueba automatizado.

Figura 43

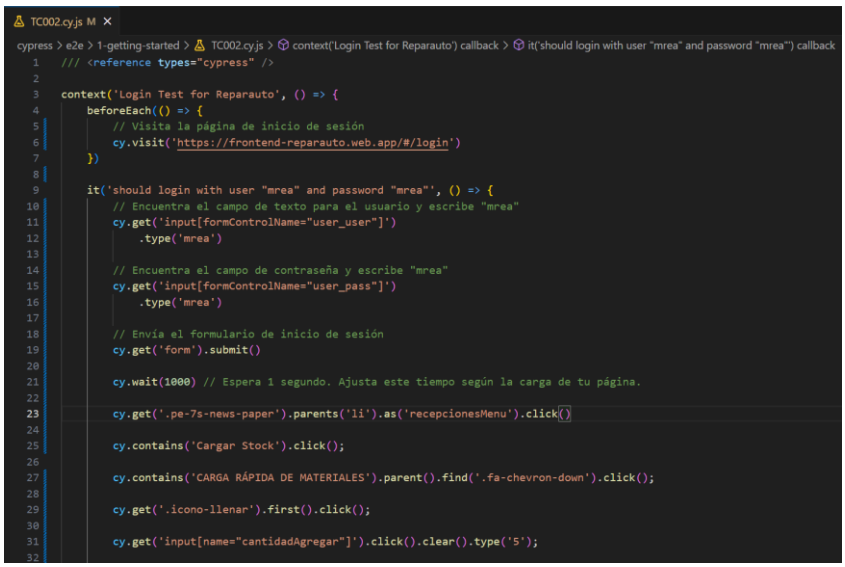
Archivo “TC001.cy.js”



Nota: Fuente: Elaboración propia.

Figura 44

Archivo “TC002.cy.js”



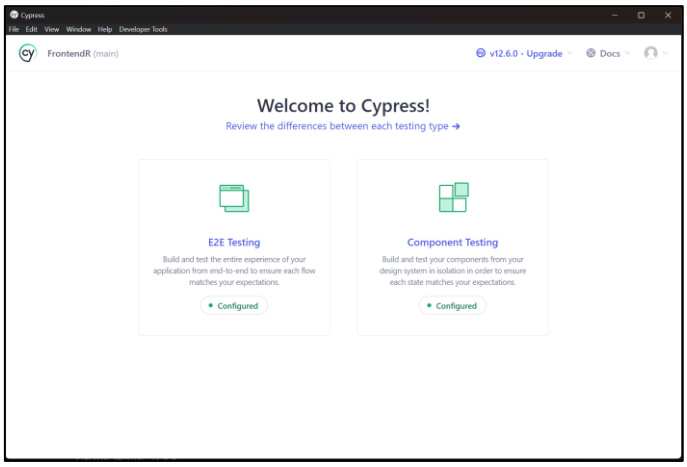
Nota: Fuente: Elaboración propia.

2.3.3.3. Entorno de Cypress

Al ejecutar el comando “npx Cypress open” en la terminal de VScode se abre el entorno en el cual ejecutamos las pruebas que se mostraron anteriormente.

Figura 45

Interfaz de Cypress

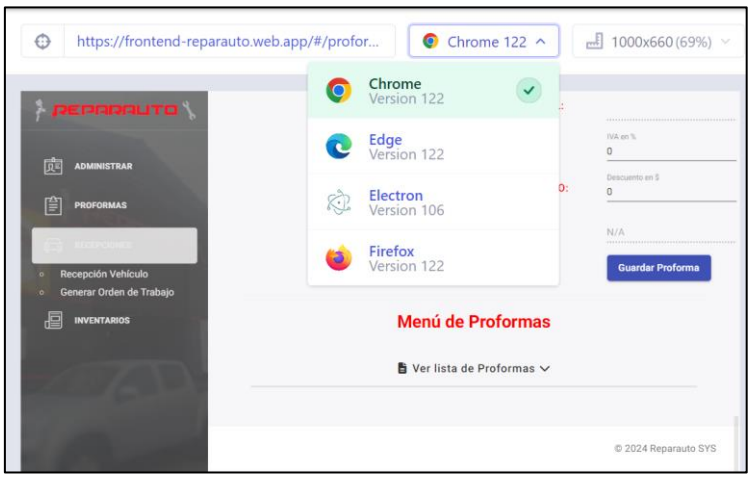


Nota: Fuente: Elaboración propia.

En el entorno Cypress se realizaron las pruebas de integración descritos en el documento “Plan de pruebas de Integración basado en la Norma ISO/IEC/IEEE 29119” así como también las pruebas de regresión a cada caso y en cada Navegador descrito en el documento mencionado.

Figura 46

Testing de componentes

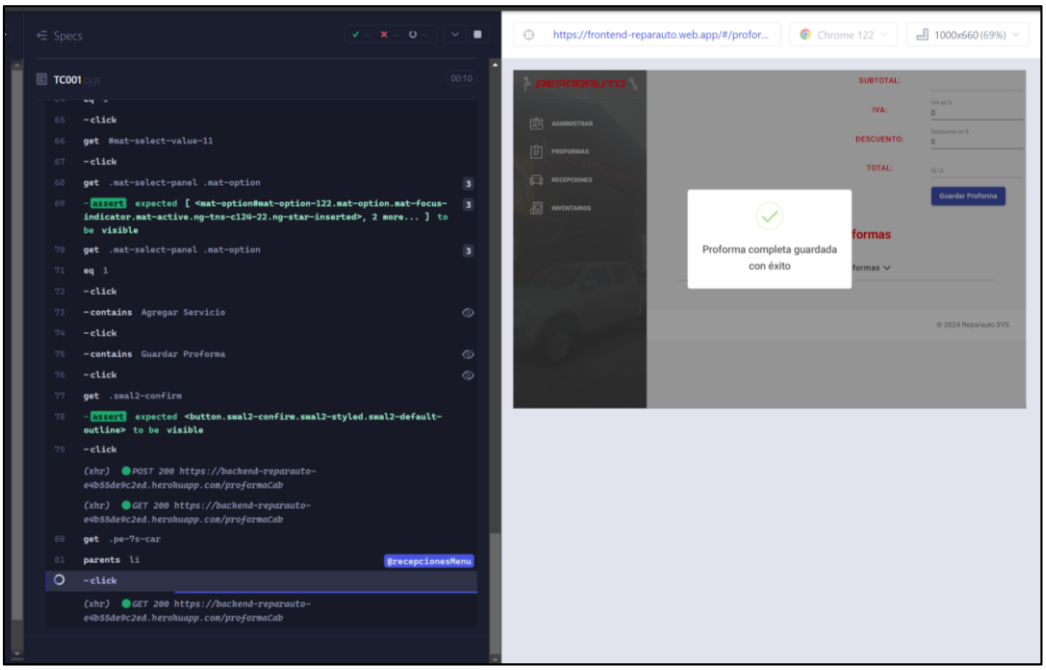


Nota: Fuente: Elaboración propia.

2.3.3.4. Ejecución caso TC001

Figura 47

Ejecución del caso TC001

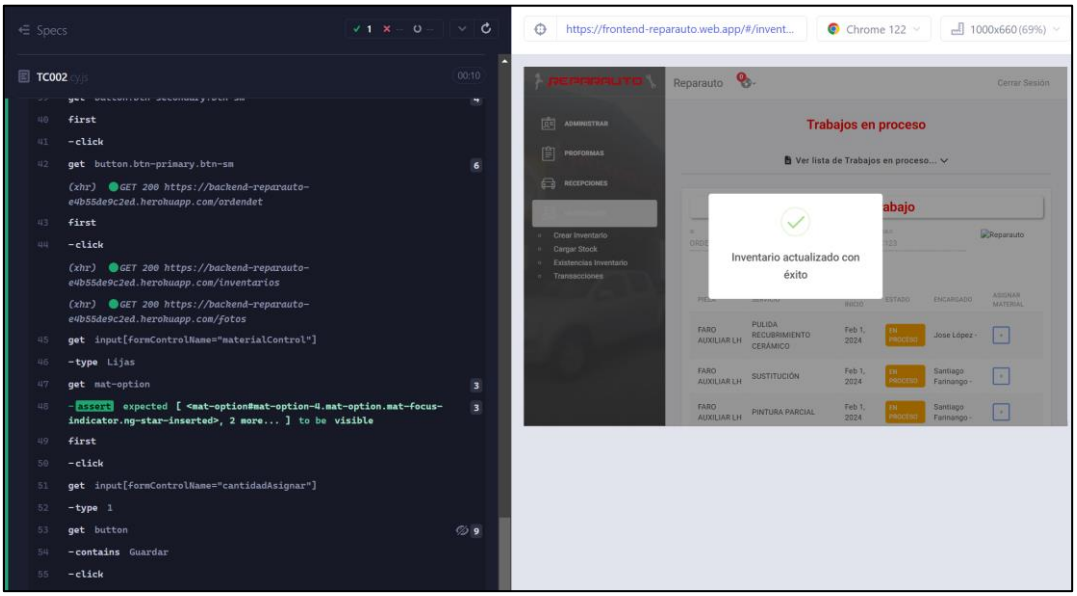


Nota: Fuente: Elaboración propia.

2.3.3.5. Ejecución caso TC002

Figura 48

Ejecución del caso TC002



Nota: Fuente: Elaboración propia.

2.3.4. Despliegue de la aplicación

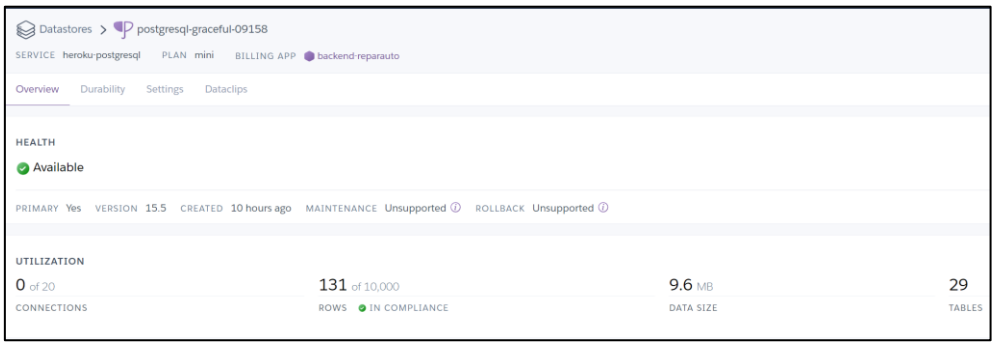
2.3.4.1. Despliegue de la base de datos en Heroku.

Con el fin de desplegar la base de datos al servidor Heroku se creó una aplicación llamada “backend-reparauto” dentro de Heroku la cual aloja el Backend y la base de datos.

Heroku entrega las credenciales necesarias para la base de datos la cual sirve para integrar con “pgAdmin” que es la interfaz gráfica de “PostgreSQL”. En la siguiente imagen podemos ver la base de datos alojada en Heroku:

Figura 49

Base de datos en Heroku.



Nota: Fuente: Elaboración propia.

Después de integrar el respaldo de la base de datos en Heroku desde la base de datos local se configuró el archivo “cnn.js” el cual usa el Backend para generar una conexión directa con la base de datos en cuestión, la siguiente imagen detalla la configuración necesaria para el correcto funcionamiento de esta.

Figura 50

Archivo de configuración “cnn.js”

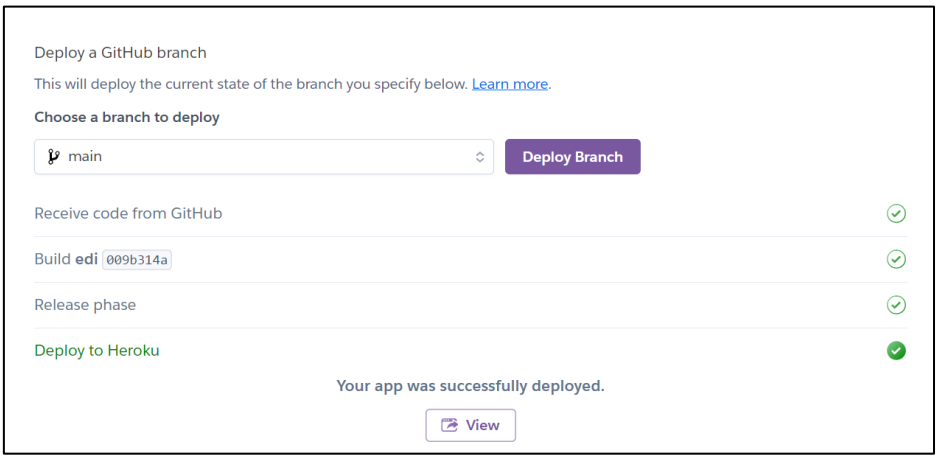
Nota: Fuente: Elaboración propia.

2.3.4.2. Despliegue del Backend en Heroku.

Con los archivos previamente configurados para la conexión de la base de datos de Heroku, se enlazó GitHub con Heroku haciendo el despliegue automático como se observa en la siguiente imagen:

Figura 51

Despliegue desde Heroku con Github.



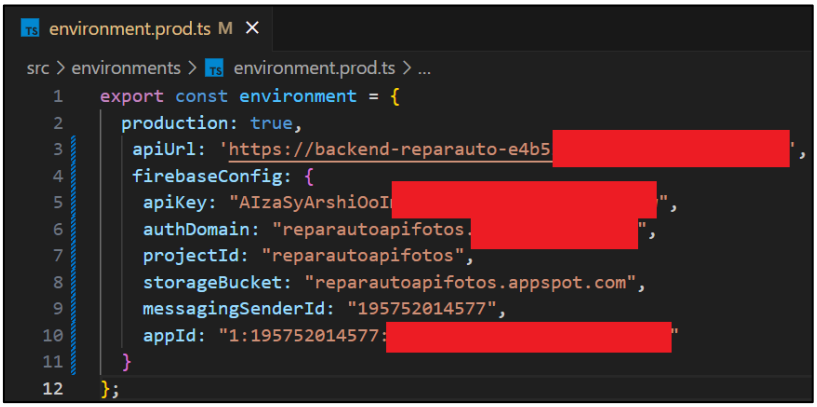
Nota: Fuente: Elaboración propia.

2.3.4.3. Despliegue del Frontend en Firebase.

Con el Backend alojado en Heroku, se remplazó el archivo “environment” de desarrollo por el de producción en todos los servicios, además de cambiar la ruta local por la ruta que nos entrega Heroku al momento del despliegue del Backend,

Figura 52

Archivo de configuración “environment.prod.ts”

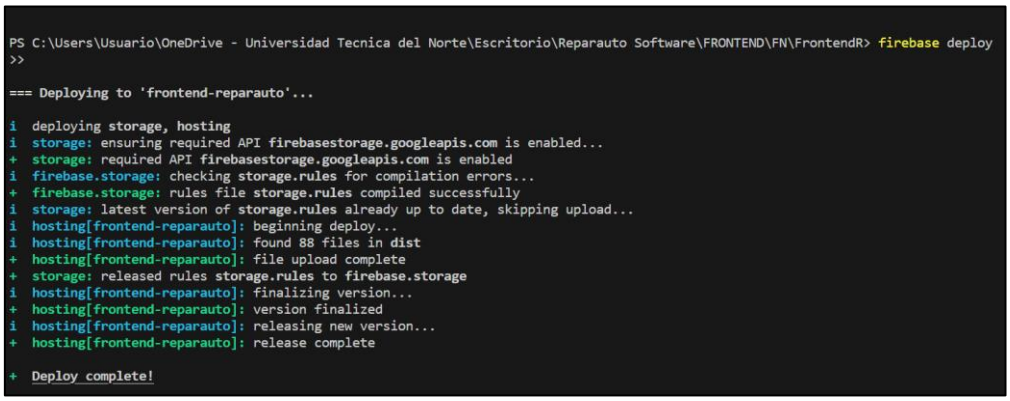


Nota: Fuente: Elaboración propia.

En la consola de Firebase se creó un nuevo proyecto llamado “Frontend-reparauto” el cual sirvió de hosting para el Frontend de la aplicación. Se configuró siguiendo el manual de Firebase y posteriormente se guardaron los cambios en GitHub para desplegar el Frontend.

Figura 53

Despliegue del Frontend en Firebase

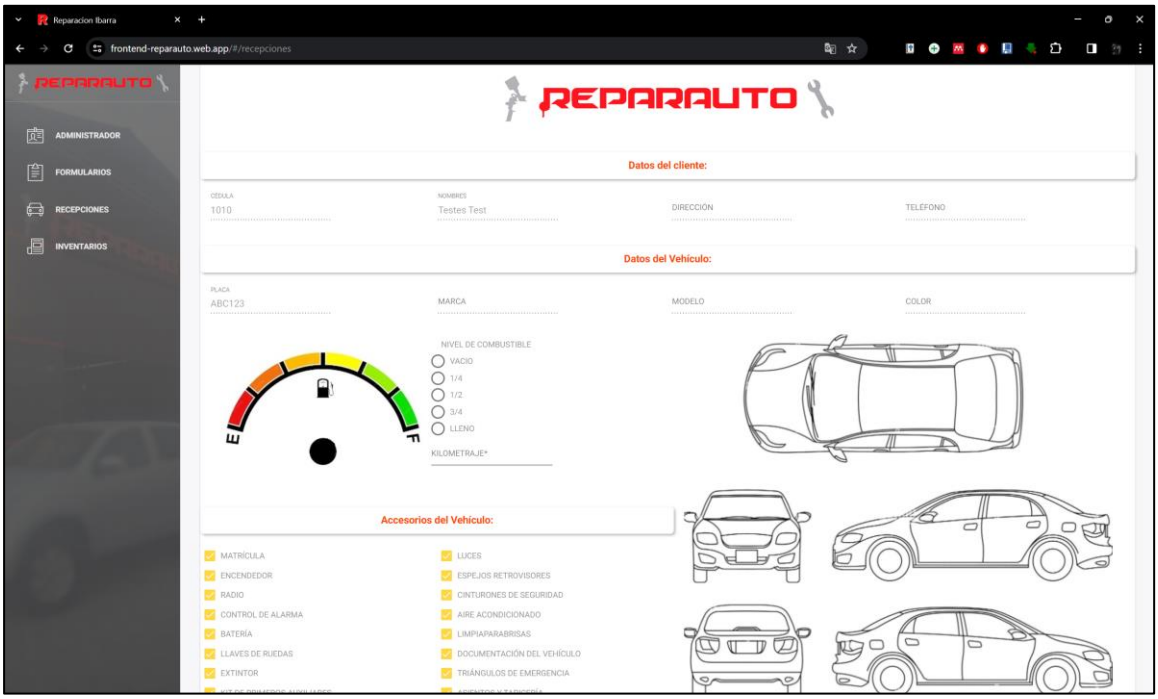


Nota: Fuente: Elaboración propia.

En la siguiente Figura se observa la aplicación funcional y alojada en Firebase y Heroku como se planteó en la arquitectura:

Figura 54

Aplicación Funcional desplegada



Nota: Fuente: Elaboración propia.

CAPÍTULO 3

3. Análisis de Resultados

3.1. Estructuración del Análisis

El presente análisis de resultados adopta un enfoque metodológico mixto, integrando análisis cuantitativos y cualitativos para evaluar exhaustivamente la implementación del plan de pruebas de integración basado en el estándar ISO/IEC/IEEE 29119 en los procesos de automatización de Reparauto. Esta metodología se seleccionó con el fin de obtener una comprensión detallada y multifacética del impacto generado por la incorporación del software, abarcando tanto aspectos objetivos como subjetivos relacionados con la operatividad y la percepción interna de la herramienta.

3.1.1. Enfoque Metodológico Aplicado

El diseño mixto de la investigación se fundamenta en la premisa de que la combinación de datos cuantitativos y cualitativos ofrece una visión más completa y rica de la realidad investigada. Los datos cuantitativos proporcionan mediciones objetivas sobre parámetros de eficiencia, rendimiento y optimización de recursos, mientras que los datos cualitativos ofrecen perspectivas valiosas sobre la experiencia del usuario, la aceptación del sistema y su integración en la cultura organizacional de Reparauto.

3.1.2. Planificación y Ejecución de la Recolección de Datos.

Previo a la implementación del software, se realizó una encuesta cualitativa y cuantitativa para recabar información base sobre los procedimientos operativos estándar en Reparauto. Subsecuentemente, tras la implementación del sistema, se replicaron esta encuesta para capturar cualquier variación atribuible directamente al nuevo software.

Las encuestas fueron dirigidas exclusivamente a los 12 colaboradores de Reparauto involucrados directamente con los procesos de recepción e inventarios. Esta selección precisa aseguró la obtención de perspectivas críticas de aquellos directamente afectados por la implementación del software, permitiendo una evaluación precisa de su impacto operativo y funcional.

3.2. Desarrollo del Instrumento de Medición

3.2.1. Adaptación de cuestionario a partir del Modelo DeLone & McLean

Para el desarrollo del cuestionario, se procedió a una meticulosa adaptación de los parámetros establecidos por el Modelo DeLone & McLean. Los aspectos clave del modelo, como la calidad del sistema, la calidad de la información, la satisfacción del usuario, el uso del sistema y el beneficio neto, fueron cuidadosamente integrados en el diseño del cuestionario. Esta adaptación permitió no solo medir el impacto operativo y funcional del software sino también captar la percepción y satisfacción de los usuarios respecto a su implementación

3.2.2. Construcción del Cuestionario

Los cuestionarios se diseñaron con el objetivo de evaluar de manera integral el impacto de la implementación de un nuevo software de gestión de procesos de recepción e inventarios en Reparauto. Este instrumento se compuso de dos secciones principales: preguntas cuantitativas y preguntas cualitativas con escala Likert, pre y post-implantación del software.

- **Selección de Preguntas Cuantitativas**: Las preguntas cuantitativas se enfocaron en medir aspectos operativos específicos como el tiempo de procesamiento, errores en registros de inventario, volumen de vehículos procesados, tiempo para localizar ítems en inventario y horas dedicadas a la gestión de inventarios. Estas preguntas fueron esenciales para obtener datos objetivos sobre la eficiencia y eficacia de los procesos antes y después de la implementación del software.

- **Escala Likert en Preguntas Cualitativas**: Para las preguntas cualitativas, se utilizó una escala Likert de 5 puntos para evaluar percepciones y actitudes respecto a la facilidad de uso, beneficios percibidos, eficiencia del sistema, impacto en los procesos de trabajo y satisfacción general con el software. La inclusión de estas preguntas permitió capturar la dimensión subjetiva de la experiencia de los usuarios con el nuevo sistema.

3.2.3. Variables de Estudio

Tabla 18

Variables de estudio en las preguntas Post Implantación

Pregunta del Cuestionario Post-Implantación	Variable de Estudio	Impacto
¿Cuánto cree que ha mejorado la eficiencia y rapidez en la recepción de vehículos con el nuevo software?	Uso y Satisfacción del Usuario	Indica un aumento en la adopción y satisfacción del usuario debido a mejoras en la eficiencia operativa, reflejando una respuesta positiva hacia el sistema.
¿Cuánto ha cambiado la gestión del inventario con el uso del nuevo sistema?	Calidad de la Información y Uso	Muestra cómo la mejora en la calidad de la información afecta directamente el uso del sistema, facilitando una gestión de inventario más eficiente y precisa.
¿Cuán beneficiosos han sido los cambios desde la implantación del software?	Impacto Individual y Organizacional	Evalúa el valor añadido percibido del software a nivel individual y organizacional, destacando los beneficios tangibles e intangibles derivados de su implementación.
¿Qué tan desafiantes han sido los problemas encontrados después de la implementación del software?	Satisfacción del Usuario	Mide cómo los desafíos post-implantación afectan la satisfacción general del usuario, indicando áreas críticas para la mejora continua del sistema.

¿Cómo calificaría la facilidad de uso del nuevo software y qué tan dispuesto estaría a recomendar mejoras?	Calidad del Sistema y Satisfacción del Usuario	Evalúa la percepción de la calidad del sistema en términos de usabilidad y la disposición a promover el sistema, lo cual es indicativo de una alta satisfacción del usuario.

Nota: Fuente: Elaboración propia.

3.3. Recopilación de Datos

Para la recolección de datos se digitalizó las dos encuestas “Pre” y “Post” Implantación utilizando la herramienta Google Forms , las dos encuestas se dirigieron a las 12 personas que forman parte de la empresa Reparauto, las mismas que interactúan en los dos procesos que fueron automatizados.

3.4. Análisis de Datos

El análisis de los datos obtenidos en las dos encuetas fue llevado a cabo usando el software “IBM SPSS Stadistics 25” el cual evaluó Media, Mediana, Moda, Rango, Frecuencias, Alfa de Cronbach entre otros.

3.4.1. Tratamiento Estadístico de los Datos

A continuación, se muestra una tabla con un análisis descriptivo de las primeras 5 preguntas cuantitativas Pre-Implantación:

Tabla 19

Estadísticas descripticas cuantitativas pre-implantación

Estadísticos descriptivos							
	N	Rango	Mínimo	Máximo	Media	Desv. estándar	Varianza
¿Cuánto tiempo tarda, en promedio, procesar la recepción de un vehículo?	12	20	10	30	18,33	8,348	69,697
¿Cuántos errores se encuentran, en promedio, en los registros de inventario cada mes?	12	10	5	15	11,25	3,108	9,659
¿Cuántos vehículos son recibidos y registrados en promedio diariamente?	12	1	1	2	1,75	0,452	0,205
¿Cuál es el tiempo promedio necesario para localizar un ítem específico en el inventario?	12	0,5	1	1,5	1,083	0,1946	0,038
¿Cuántas horas al día dedica el personal a tareas relacionadas con el tratamiento de materiales en los inventarios?	12	1.0	.5	1.5	.917	.3589	0,129
N válido (por lista)	12						

Nota: Fuente: Elaboración propia.

La siguiente tabla muestra un análisis descriptivo de las 5 preguntas cuantitativas con enfoque Post-Implantación:

Tabla 20

Estadísticas descripticas cuantitativas post-implantación

Estadísticos descriptivos							
	N	Rango	Mínimo	Máximo	Media	Desv. estándar	Varianza
¿Cuánto tiempo tarda, en promedio, procesar la recepción de un vehículo?	12	0	10	10	10,00	0,000	0,000
¿Cuántos errores se encuentran, en promedio, en los registros de inventario cada mes?	12	5	0	5	0,42	1,443	2,083

¿Cuántos vehículos son recibidos y registrados en promedio diariamente?	12	2	1	3	1,75	0,754	0,568
¿Cuál es el tiempo promedio necesario para localizar un ítem específico en el inventario?	12	0,0	0,5	0,5	0,500	0,0000	0,000
¿Cuántas horas al día dedica el personal a tareas relacionadas con el tratamiento de materiales en los inventarios?	12	0,0	0,5	0,5	0,500	0,0000	0,000
N válido (por lista)	12						

Nota: Fuente: Elaboración propia.

Frecuencias de las preguntas cualitativas en el cuestionario pre-implantación:

Tabla 21

Frecuencias Pregunta 6 Pre-Implantación

En una escala del 1 al 5, ¿cómo calificaría la dificultad actual al recibir y registrar nuevos vehículos?					
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	2	1	8,3	8,3	8,3
	3	11	91,7	91,7	100
	Total	12	100	100	

Nota: Fuente: Elaboración propia.

Tabla 22

Frecuencias Pregunta 7 Pre-Implantación

En una escala del 1 al 5, ¿cuán beneficioso cree que será el nuevo software para su trabajo?					
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	4	2	16,7	16,7	16,7
	5	10	83,3	83,3	100
	Total	12	100	100	

Nota: Fuente: Elaboración propia.

Tabla 23

Frecuencias Pregunta 8 Pre-Implantación

En una escala del 1 al 5, ¿cómo evaluaría la precisión y eficiencia de su actual manejo de inventario?					
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	1	1	8,3	8,3	8,3
	2	6	50	50	58,3
	3	5	41,7	41,7	100
	Total	12	100	100	

Nota: Fuente: Elaboración propia.

Tabla 24

Frecuencias Pregunta 9 Pre-Implantación

En una escala del 1 al 5, ¿cuán significativas son las limitaciones del proceso actual de recepción de vehículos e inventarios?					
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	2	2	16,7	16,7	16,7
	3	10	83,3	83,3	100
	Total	12	100	100	

Nota: Fuente: Elaboración propia.

Tabla 25

Frecuencias Pregunta 10 Pre-Implantación

En una escala del 1 al 5, ¿cuán altas son sus expectativas respecto al nuevo software?					
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	4	1	8,3	8,3	8,3
	5	11	91,7	91,7	100
	Total	12	100	100	

Nota: Fuente: Elaboración propia.

Frecuencias de las preguntas cualitativas en el cuestionario Post-implantación:

Tabla 26

Frecuencias Pregunta 6 Post-Implantación

En una escala del 1 al 5, ¿cuánto cree que ha mejorado la eficiencia y rapidez en la recepción de vehículos con el nuevo software?					
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	1	1	8,3	8,3	8,3
	4	2	16,7	16,7	25
	5	9	75	75	100
	Total	12	100	100	

Nota: Fuente: Elaboración propia.

Tabla 27

Frecuencias Pregunta 7 Post-Implantación

En una escala del 1 al 5, ¿cuánto ha cambiado la gestión del inventario con el uso del nuevo sistema?					
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	3	1	8,3	8,3	8,3
	4	5	41,7	41,7	50
	5	6	50	50	100
	Total	12	100	100	

Nota: Fuente: Elaboración propia.

Tabla 28

Frecuencias Pregunta 8 Post-Implantación

En una escala del 1 al 5, ¿cuán beneficiosos han sido los cambios desde la implantación del software?					
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	4	3	25	25	25
	5	9	75	75	100
	Total	12	100	100	

Nota: Fuente: Elaboración propia.

Tabla 29

Frecuencias Pregunta 9 Post-Implantación

En una escala del 1 al 5, ¿Qué tan críticos han sido los problemas encontrados después de la implementación del software?					
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	4	4	33,3	33,3	33,3
	5	8	66,7	66,7	100
	Total	12	100	100	

Nota: Fuente: Elaboración propia.

Tabla 30

Frecuencias Pregunta 10 Post-Implantación

En una escala del 1 al 5, ¿cómo calificaría la facilidad de uso del nuevo software y qué tan dispuesto estaría a recomendar mejoras?					
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	4	5	41,7	41,7	41,7
	5	7	58,3	58,3	100
	Total	12	100	100	

Nota: Fuente: Elaboración propia.

3.4.2. Prueba T para Muestras Emparejadas

La Prueba T para muestras emparejadas se utilizó para comparar las medias de dos conjuntos de datos relacionados, como las mediciones pre y post-implantación. A partir de esta prueba, obtenemos el valor de **t**, los grados de libertad y el valor **p**, que juntos nos indican si la diferencia en las medias es estadísticamente significativa. Si el valor p es bajo (típicamente menor que 0.05), podemos concluir que es probable que el software haya tenido un efecto significativo en las variables medidas.

Tabla 31

Prueba T de Muestras Emparejadas

Prueba de muestras emparejadas										
		Diferencias emparejadas							Significación	
					95% de intervalo de confianza de la diferencia					
		Media	Desv. estándar	Media de error estándar	Inferior	Superior	t	gl	P de un factor	P de dos factores
P1	**¿Cuánto tiempo tarda, en promedio, procesar la recepción de un vehículo?**	8,333	8,348	2,410	3,029	13,638	3,458	11	0,003	0,005
P2	**¿Cuántos errores se encuentran, en promedio, en los registros de inventario cada mes?**	10,833	2,887	0,833	8,999	12,667	13,000	11	0,000	0,000
P3	**¿Cuántos vehículos son recibidos y registrados en promedio diariamente?**	0,000	0,953	0,275	-0,606	0,606	0,000	11	0,500	1,000
P4	**¿Cuál es el tiempo promedio necesario para localizar un ítem específico en el inventario?**	0,5833	0,1946	0,0562	0,4597	0,7070	10,383	11	0,000	0,000
P5	**¿Cuántas horas al día dedica el personal a tareas relacionadas con el tratamiento de materiales en los inventarios?**	0,4167	0,3589	0,1036	0,1887	0,6447	4,022	11	0,001	0,002

Nota: Fuente: Elaboración propia.

3.4.3. Evaluación de la Fiabilidad del Instrumento.

Para evaluar la fiabilidad del instrumento de medición, se aplicó el coeficiente alfa de Cronbach. Donde se establece que un valor de alfa cercano a 1 indica alta fiabilidad, mostrando que los ítems son consistentes entre sí. Un valor igual o superior a 0.7 se considera aceptable, señalando una consistencia adecuada y valores por debajo de 0.7 sugieren la necesidad de revisar el cuestionario para mejorar su fiabilidad.

3.4.3.1. Alfa de Cronbach en cuestionario Pre-Implantación

Tabla 32

Alfa de Cronbach Pre-Implantación

Estadísticas de total de elemento				
	Media de escala si el elemento se ha suprimido	Varianza de escala si el elemento se ha suprimido	Correlación total de elementos corregida	Alfa de Cronbach si el elemento se ha suprimido
En una escala del 1 al 5, ¿cómo calificaría la dificultad actual al recibir y registrar nuevos vehículos?	14,92	0,629	-0,033	-,386[a]
En una escala del 1 al 5, ¿cuán beneficioso cree que será el nuevo software para su trabajo?	13	0,727	-0,274	-,028[a]
En una escala del 1 al 5, ¿cómo evaluaría la precisión y eficiencia de su actual manejo de inventario?	15,5	0,455	-0,207	-,044[a]
En una escala del 1 al 5, ¿cuán significativas son las limitaciones del proceso actual de recepción de vehículos e inventarios?	15	0,545	0	-,481[a]
En una escala del 1 al 5, ¿cuán altas son sus expectativas respecto al nuevo software?	12,92	0,629	-0,033	-,386[a]

Nota: Fuente: Elaboración propia.

3.4.3.2. Alfa de Cronbach en cuestionario Post-Implantación

Tabla 33

Alfa de Cronbach Post-Implantación

Estadísticas de total de elemento				
	Media de escala si el elemento se ha suprimido	Varianza de escala si el elemento se ha suprimido	Correlación total de elementos corregida	Alfa de Cronbach si el elemento se ha suprimido
En una escala del 1 al 5, ¿cuánto cree que ha mejorado la eficiencia y rapidez en la recepción de vehículos con el nuevo software?	18,42	2,265	0,543	0,651
En una escala del 1 al 5, ¿cuánto ha cambiado la gestión del inventario con el uso del nuevo sistema?	18,5	4,091	0,37	0,657
En una escala del 1 al 5, ¿cuán beneficiosos han sido los cambios desde la implantación del software?	18,17	3,788	0,878	0,517
En una escala del 1 al 5, ¿Qué tan críticos han sido los problemas encontrados después de la implementación del software?	18,25	4,932	0,166	0,717
En una escala del 1 al 5, ¿cómo calificaría la facilidad de uso del nuevo software y qué tan dispuesto estaría a recomendar mejoras?	18,33	4,061	0,584	0,592

Nota: Fuente: Elaboración propia.

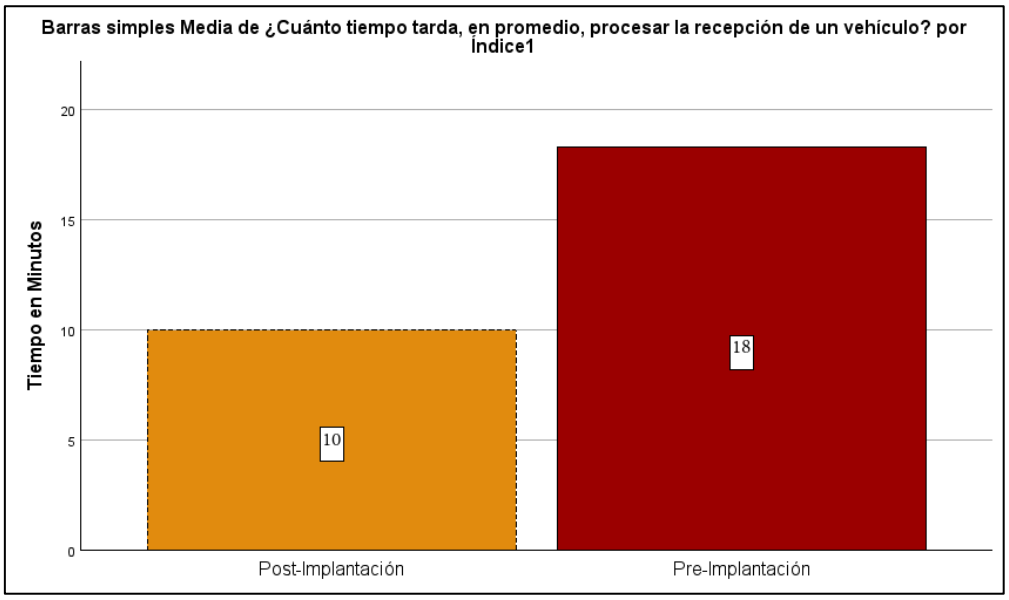
3.5. Presentación y Discusión de los Resultados

3.5.1. Interpretación de los Resultados Cuantitativos

Pregunta 1: ¿Cuánto tiempo tarda, en promedio, procesar la recepción de un vehículo?

Figura 55

Gráfico de Barras – P1



Nota: Fuente: Elaboración propia.

- **Reducción Significativa del Tiempo de Procesamiento**: La media de tiempo para procesar la recepción de un vehículo disminuyó de 18,33 minutos (pre-implantación) a 10 minutos (post-implantación). Esto indica una mejora notable en la eficiencia del proceso de recepción de vehículos gracias a la implementación del software.
- **Eliminación de la Variabilidad:** La desviación estándar en el tiempo de procesamiento se redujo a 0 en la medición post-implantación, lo que sugiere que el proceso se ha estandarizado y la variabilidad en el tiempo de procesamiento se ha eliminado completamente.

Pregunta 2: ¿Cuántos errores se encuentran, en promedio, en los registros de inventario cada mes?

Figura 56

Gráfico de Barras – P2

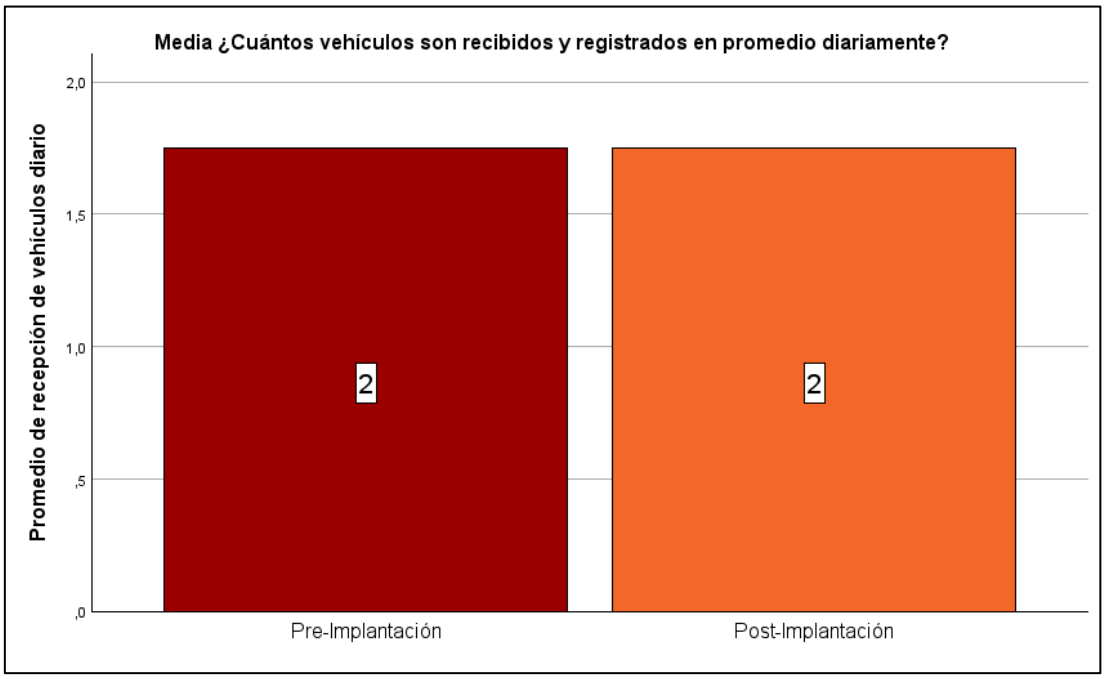
Nota: Fuente: Elaboración propia.

- **Mejora en la calidad de Información**: La cantidad promedio de errores en los registros de inventario se redujo de 11,25 errores por mes (pre-implantación) a 0,42 errores por mes (post-implantación). Esta notable disminución de 10,83 errores representa una mejora del 96,26% en la precisión de los registros de inventario.
- **Incremento en la Eficiencia Operativa:** Con un promedio de 0,42 errores por mes post-implantación, comparado con los 11,25 errores por mes pre-implantación, se evidencia una optimización significativa en la gestión de inventarios

Pregunta 3: ¿Cuántos vehículos son recibidos y registrados en promedio diariamente?

Figura 57

Gráfico de Barras – P3



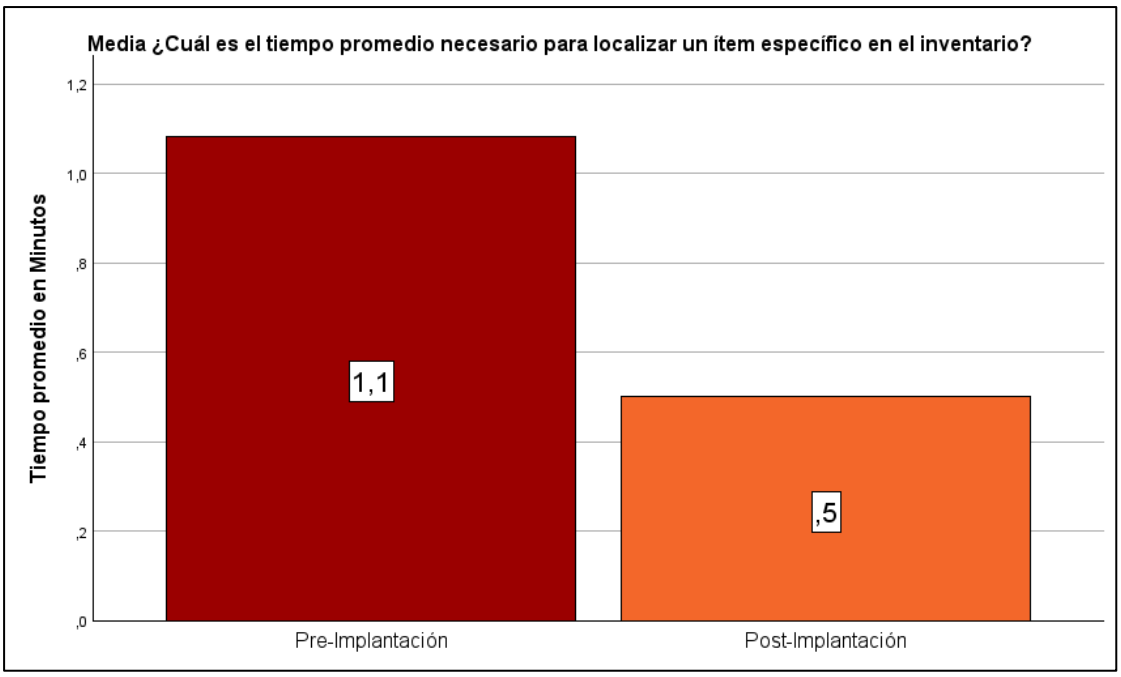
Nota: Fuente: Elaboración propia.

- **Mantenimiento del Volumen de Trabajo**: La cantidad promedio de vehículos recibidos y registrados diariamente se mantuvo constante en 1,75. Esto indica que la implementación del software no ha tenido un impacto directo en alterar la cantidad de vehículos que Reparauto procesa diariamente.
- **Aumento de la Variabilidad Post-Implantación**: A pesar de mantenerse constante en promedio, el aumento en la desviación estándar post-implantación sugiere una mayor variabilidad en el número de vehículos procesados diariamente, lo que podría indicar una adaptabilidad mejorada a las fluctuaciones del mercado o cambios en la demanda del servicio.

Pregunta 4: ¿Cuál es el tiempo promedio necesario para localizar un ítem específico en el inventario?

Figura 58

Gráfico de Barras – P4



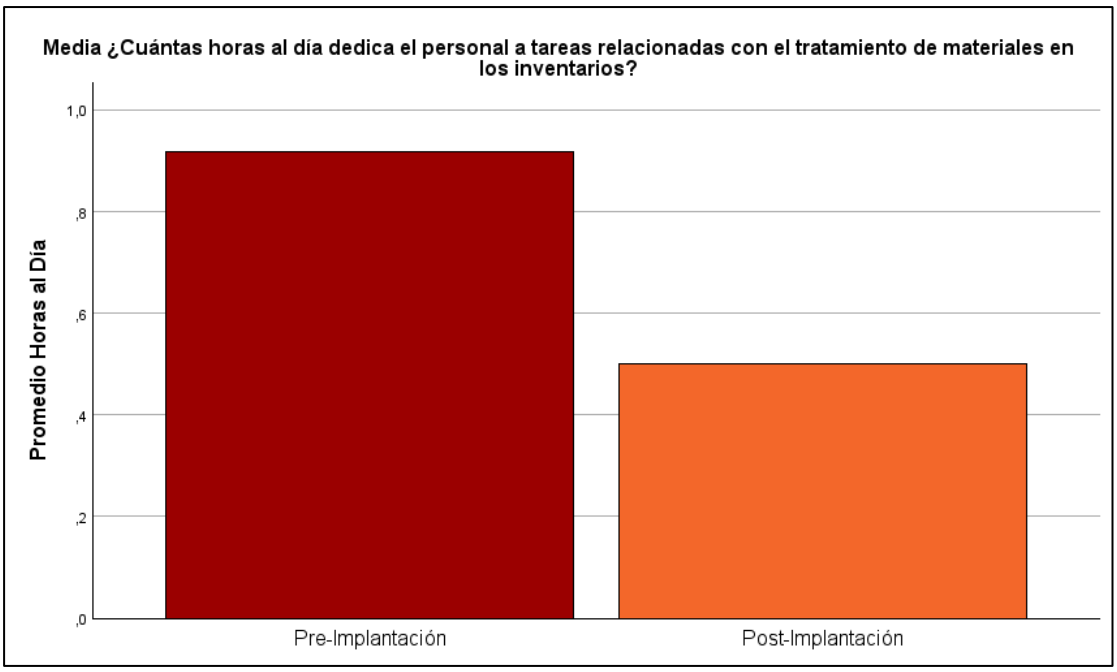
Nota: Fuente: Elaboración propia.

- **Reducción del Tiempo de Localización:** La implementación del software ha reducido a la mitad el tiempo promedio necesario para localizar un ítem específico en el inventario, pasando de 1,08 minutos a 0,5 minutos, lo que refleja una mejora significativa en la eficiencia de la gestión de inventario.
- **Eliminación de la Variabilidad en la Localización de Ítems**: La desviación estándar reducida a cero post-implantación evidencia la estandarización completa del proceso de localización de ítems, indicando que el software ha logrado uniformizar y predecir con precisión este aspecto crítico de la gestión de inventarios.

Pregunta 5: ¿Cuántas horas al día dedica el personal a tareas relacionadas con el tratamiento de materiales en los inventarios?

Figura 59

Gráfico de Barras – P5



Nota: Fuente: Elaboración propia.

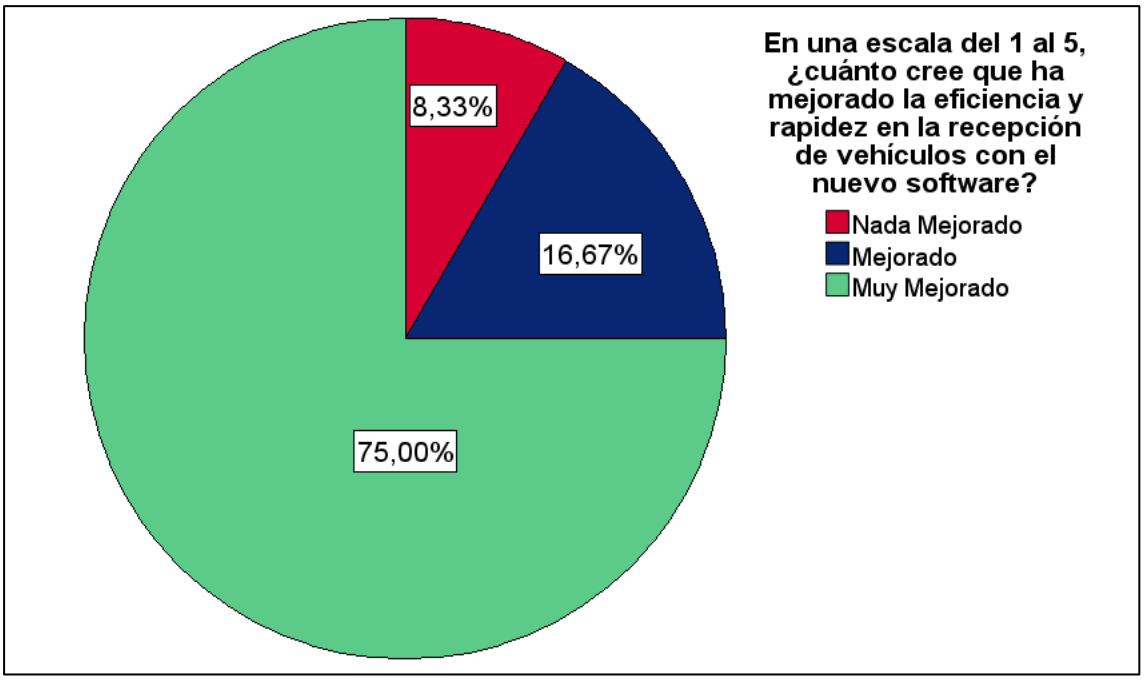
- **Reducción en el Tiempo Dedicado a Inventarios**: El tiempo promedio dedicado por el personal a tareas relacionadas con el tratamiento de materiales en los inventarios se redujo en un 45,47%, pasando de 0,91 horas a 0,5 horas al día. Esta mejora destaca la eficiencia operativa lograda a través de la implementación del software.
- **Estandarización del Manejo de Inventarios**: La desviación estándar reducida a cero post-implantación señala la eliminación completa de la variabilidad en el tiempo dedicado a estas tareas, lo que refleja la capacidad del software para homogeneizar y optimizar las operaciones de tratamiento de materiales.

3.5.2. Interpretación de los Resultados Cualitativos Post-Implantación

Pregunta 6: En una escala del 1 al 5, ¿cuánto cree que ha mejorado la eficiencia y rapidez en la recepción de vehículos con el nuevo software?

Figura 60

Gráfico de Pastel – P6



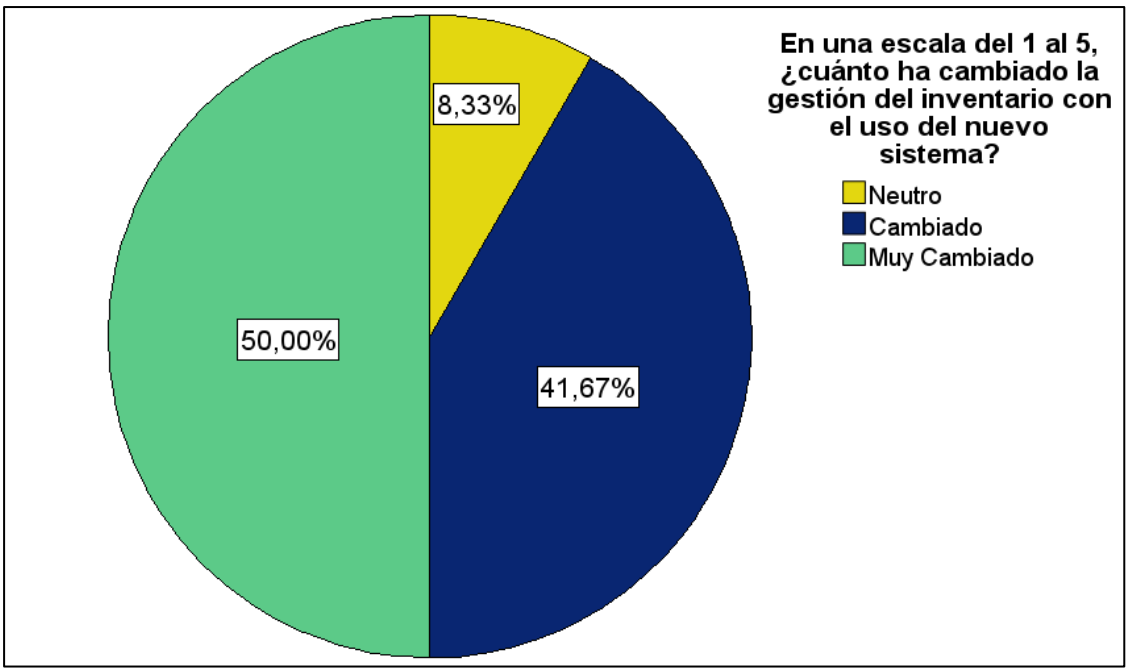
Nota: Fuente: Elaboración propia.

La mayoría de los usuarios (75%) reportaron una mejora muy significativa en la eficiencia y rapidez en la recepción de vehículos tras la implementación del software. Adicionalmente, un 16,7% observó mejoras significativas, sumando un total de 91,7% de respuestas positivas en las categorías de mejora alta. Solo un 8,3% de los usuarios reportaron una mejora mínima.

Pregunta: 7: En una escala del 1 al 5, ¿cuánto ha cambiado la gestión del inventario con el uso del nuevo sistema?

Figura 61

Gráfico de Pastel – P7



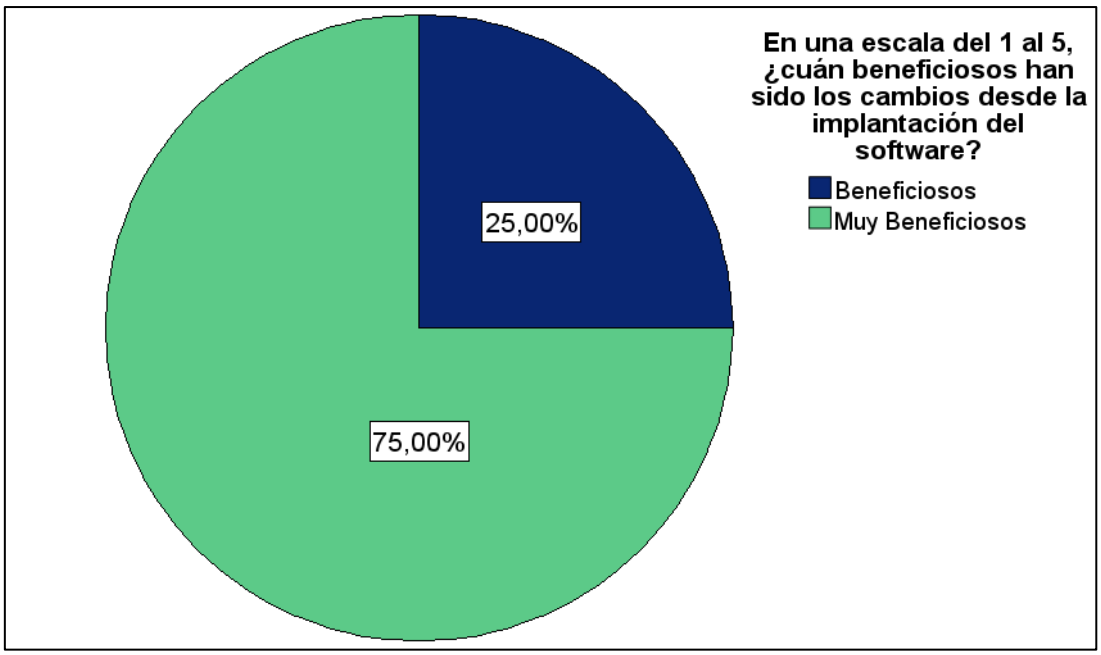
Nota: Fuente: Elaboración propia.

Con relación a la gestión del inventario del nuevo sistema, el 50% de los usuarios calificó la mejora con la máxima puntuación, lo que indica una transformación positiva sustancial. Adicionalmente, un 41,7% otorgó una calificación de 4, sumando entre ambas categorías un total de 91,7% que percibe una mejora significativa en la gestión del inventario. Solo un 8,3% de los usuarios otorgaron una calificación media

Pregunta 8: En una escala del 1 al 5, ¿cuán beneficiosos han sido los cambios desde la implantación del software?

Figura 62

Gráfico de Pastel – P8



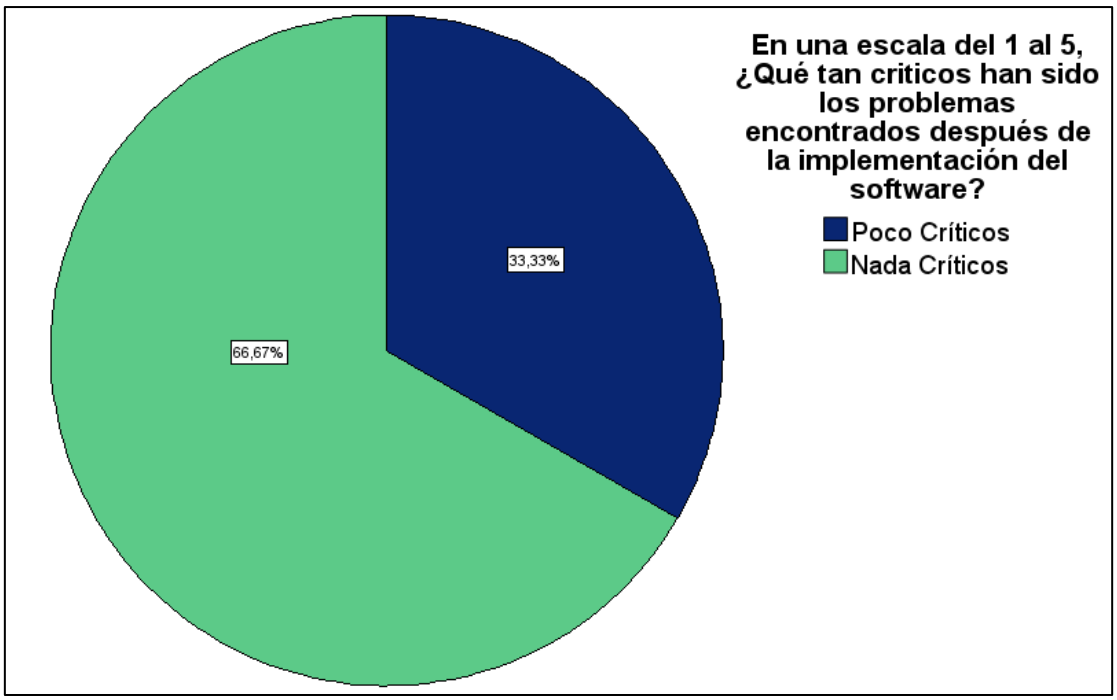
Nota: Fuente: Elaboración propia.

La percepción de los beneficios derivados de la implantación del software es notablemente positiva entre los usuarios. Un 75% otorgó la calificación más alta en términos de beneficios percibidos, mientras que un adicional 25% calificó los cambios como altamente beneficiosos. Esto resulta en un 100% de los encuestados que consideran que el software ha tenido un impacto positivo desde su implementación.

Pregunta 9: En una escala del 1 al 5, ¿Qué tan críticos han sido los problemas encontrados después de la implementación del software?

Figura 63

Gráfico de Pastel – P9



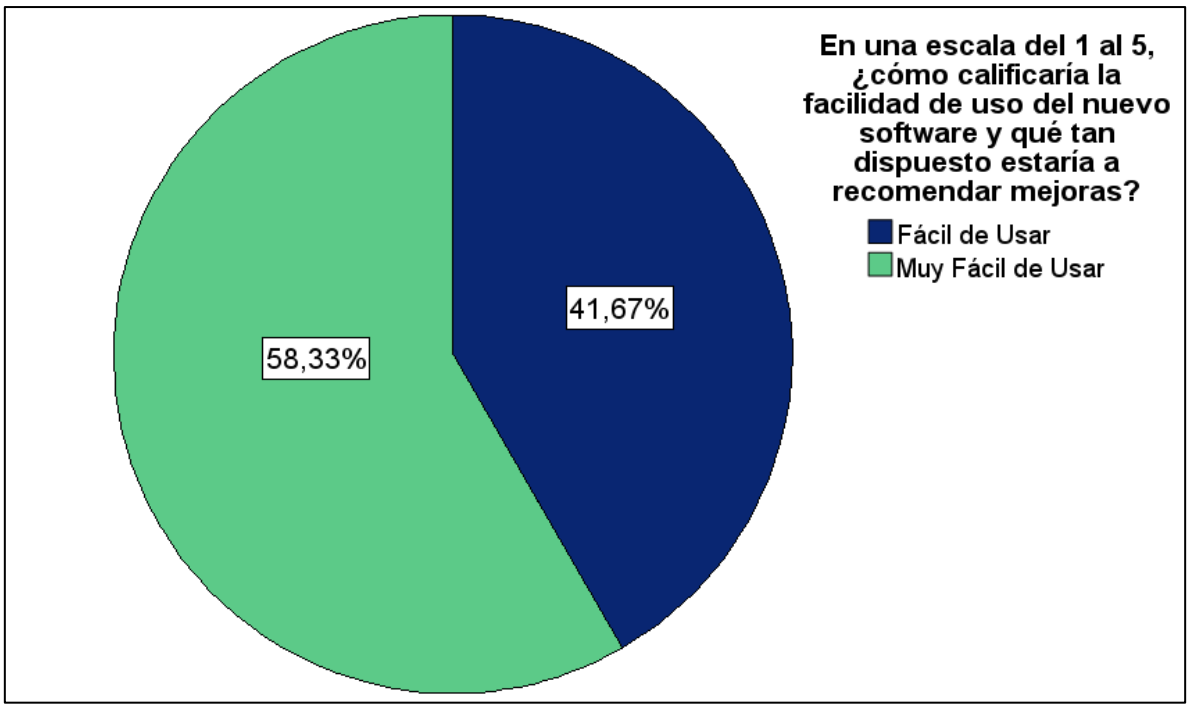
Nota: Fuente: Elaboración propia.

Los datos reflejan una respuesta favorable con respecto a los problemas encontrados tras la implementación del nuevo software. Con un 66,7% de los usuarios indicando que no consideran críticos los problemas surgidos y un 33,3% calificando los problemas como poco críticos

Pregunta 10: En una escala del 1 al 5, ¿cómo calificaría la facilidad de uso del nuevo software y qué tan dispuesto estaría a recomendar mejoras?

Figura 64

Gráfico de Pastel – P10



Nota: Fuente: Elaboración propia.

La mayoría de los encuestados (58,3%) otorgó la calificación más alta, lo que implica que encuentran el software muy fácil de usar y están altamente dispuestos a recomendar mejoras. Un 41,7% adicional calificó estas características con un 4, lo que sugiere que también están satisfechos con la facilidad de uso

CONCLUSIONES

La revisión detallada de las necesidades operativas de Reparauto, así como también el análisis de la literatura de la norma ISO/IEC/IEEE 29119 han sido un pilar esencial en este proyecto, permitiéndome el correcto desarrollo del plan de pruebas de Integración y ajustar mi solución tecnológica con gran precisión a los requerimientos específicos de la empresa

La implementación de una aplicación web dedicada a la recepción de vehículos y la gestión de inventarios de forma automatizada, basada en el modelo diente de tiburón, ha introducido un cambio significativo en el flujo de trabajo de Reparauto. Esta transformación ha reducido notablemente el tiempo de operación y ha minimizado los errores, mejorando así la experiencia general del usuario y demostrando el valor de integrar soluciones tecnológicas avanzadas en procesos empresariales clave.

Abordar la Norma ISO/IEC/IEEE 29119 fue todo un desafío por su tamaño y complejidad. Esta normativa, siendo un referente en el ámbito de las pruebas de software, presenta un marco exhaustivo que requiere una comprensión profunda para su correcta aplicación. La dificultad radicó no solo en la cantidad de información a asimilar sino también en la necesidad de interpretar correctamente sus directrices para adaptarlas a un contexto específico como el desarrollar un plan de pruebas.

El rigor aplicado en el plan de pruebas de integración ha jugado un papel crucial para asegurar la calidad y el rendimiento óptimo de la aplicación web. Esta estrategia proactiva de pruebas ha facilitado la identificación y resolución temprana de problemas, garantizando que la aplicación no solo cumpla con los requisitos establecidos, sino que también se integre sin problemas en las operaciones diarias de Reparauto.

RECOMENDACIONES

Se sugiere continuar con la utilización del modelo diente de tiburón en la automatización de procesos dentro de Reparauto. Este enfoque ha demostrado ser efectivo para mejorar la eficiencia y precisión en los procesos de inventarios y recepción de vehículos, por lo que su aplicación podría explorarse en otras áreas operativas de la empresa para lograr beneficios similares.

Es importante continuar con la práctica de pruebas de integración rigurosas durante las fases de desarrollo y después de cualquier actualización significativa del sistema. El plan de pruebas de integración debe ser revisado y actualizado regularmente para incluir nuevos escenarios y asegurar que el software siga cumpliendo con los estándares de calidad y las necesidades de la empresa.

Se recomienda establecer un mecanismo de retroalimentación continuo con los usuarios de la aplicación, incluyendo tanto a empleados de Reparauto como a clientes finales ya que permitirá identificar oportunidades de mejora en la aplicación y ajustar la funcionalidad para satisfacer mejor las necesidades del usuario.

BIBLIOGRAFÍA

Referencias

Angular . (2024). *Angular Team*. Retrieved from Introducción a componentes y plantillas: https://docs.angular.lat/guide/architecture-components

Digital Services. (2020). *Microservicios Creando Software de alta disponibilidad.* Ciudad de méxico: The New Normal.

Guimarey, A. (2020). *Beneficios y riesgos de migrar una arquitectura monolítica a microservicios.* Palermo: Universidad de Palermo.

ISO. (2023, 02 02). *ISO/CEI/IEEE 29119-1:2013*. Retroceded 12 2022, 04, from https://www.iso.org/standard/45142.html

ISO/IEC/IEEE 29119. (2022, 01 01). *Grupo de Trabajo AEN/CTN71/SC7/GT26 Pruebas de Software*. Retrieved from in2test.

Linux Post Install. (2020, 02 01). *Ecología: ¿Cómo puede el Software Libre ayudar a salvar el planeta?* Retrieved 12 2022, 04, from https://blog.desdelinux.net/ecologia-software-libre-ayudar-salvar-planeta/#Como_puede_el_Software_Libre_ayudar_mejorar_la_Ecologia_del_planeta

Nexus Integra. (2022). *10 beneficios de contar con un sistema de automatización industrial.* Industria 4.0.

ONU. (2015). *La Agenda 2030 y los Objetivos de Desarrollo Sostenible.*

Regulwar, G. B. (2012). Variations in V Model for Software Development. *International Journal of Advanced Research in Computer Science*, 7.

Sass. (2024). *Sass*. Retrieved from Sintaxis : https://sass-lang.com/documentation/syntax/#the-indented-syntax

Coordinating, S., & Society, C. (2016). ISO/IEC/IEEE International Standard - Software and systems engineering -- Software testing -- Part 5: Keyword-Driven Testing. *ISO/IEC/IEEE 29119-5 First edition 2016-11-15*, *2013*, 1–69. ieee.com

Gamaliel, J., Sagredo, C., Espinosa, A. T., Reyes, M., & De Lourdes López García, M. (2012). *Automation of the Codification of the Model-View-Controller Pattern (mvc Pattern) in Projects Oriented to the Web*.

Henríquez N., Iglesias A., Amaris Ramos L., & Ropain Y. (s/f). *Postgresql una alternativa efectiva en las empresas*.

ISO 29119. (2021). *29119 parte 1*.

ISO/IEC/IEEE. (2013a). 29119-3:2013 - ISO/IEC/IEEE International Standard for Software and systems engineering — Software testing — Part 3: Test documentation. *ISO/IEC/IEEE 29119-3:2013(E)*, *2013*, 1–138.

ISO/IEC/IEEE. (2013b). INTERNATIONAL STANDARD ISO / IEC / IEEE Software and systems engineering — Part 1: Concepts and definitions. *Iso/Iec/Ieee 29119-1:2013(E)*, *2013*, 1–64.

ISO/IEC/IEEE. (2013c). INTERNATIONAL STANDARD ISO / IEC / IEEE Software and systems engineering — Part 2: Test processes. *Iso/Iec/Ieee 29119-2*, *2013*, 1–68.

ISO/IEC/IEEE. (2013d). ISO/IEC/IEEE 29119-4:2015 - Software and systems engineering -- Software testing -- Part 4: Test techniques. *ISO - International Organization for Standardization*, 139.

Lamancha, B. P. (2007). *Gestión de las Pruebas Funcionales*. *1*(4).

Luo, L. (2001). *Software Testing Techniques Technology Maturation and Research Strategy Class Report for 17-939A Software Testing Techniques Technology Maturation and Research Strategies*.

Matthes, F., Hauder, M., & Katinszky, N. (2021). *Enterprise Architecture Management Tool Survey 2014 Update*.

Ocampo Acosta Alejandro, & Correa Tapasco Luisa Marcela. (2011). *IMPACTO DE LAS PRUEBAS NO FUNCIONALES EN LA MEDICIÓN DE LA CALIDAD DEL PRODUCTO SOFTWARE DESARROLLADO*.

Optar, P., Gonzales, G., Asesor, C. E., Cachay, M., & Eugenio, M. J. (2020). *FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS TRABAJO DE INVESTIGACIÓN UNA REVISIÓN DE LOS PATRONES DE DISEÑO DE SOFTWARE APLICADO A LAS APLICACIONES WEB*.

Oriols, M. B., & Antonio Gómez Gutiérrez, J. (2021). *El gran libro de Angular*.

Pruebas de integración de microservicios | AppMaster. (s/f). Recuperado el 15 de octubre de 2023, de https://appmaster.io/es/glossary/pruebas-de-integracion-de-microservicios

Reid Stuard. (2017). *ISO-29119-The-New-International-Software-Testing-Standards*. https://www.stureid.info/wp-content/uploads/2015/08/ISO-29119-The-New-International-Software-Testing-Standards.pdf

Sawant, A. A., Bari, P. H., & Chawan, P. M. (2012). *Software Testing Techniques and Strategies*. *2*, 980–986. www.ijera.com

Test de integración: Objetivos, Tipos y Ejemplos. (s/f). Recuperado el 15 de octubre de 2023, de https://ginzo.tech/test-integracion/

REPARAUTO

Plan de Pruebas de Integración basado en la Norma ISO/IEC/IEEE 29119

Proyecto: Automatización de procesos de Recepción de vehículos y de Inventarios

Versión: 1.2

Historial de versiones

Versión	Autor(es)	Descripción	Fecha
1.0	Edison Zamora	Creación del documento	Oct 2023
1.1	Edison Zamora	Identificación de un nuevo riesgo	Oct 2023
1.2	Edison Zamora	Creación de Casos de Prueba	Nov 2023

Índice

I. INTRODUCCIÓN

Este documento define el Plan de Pruebas a aplicarse la automatización de módulos de Reparauto Ibarra, específicamente enfocado en las pruebas de integración de los módulos de recepción y de inventarios del sistema Reparauto.

Siguiendo las directrices de los estándares ISO/IEC/IEEE 29119, este plan tiene como objetivo fundamental asegurar la correcta interacción y funcionamiento conjunto de estos módulos, considerando su importancia crítica para la operatividad del sistema completo.

1.1 Alcance

Objetivo del Plan de Pruebas: El propósito principal de este plan es evaluar y validar la integración efectiva de los módulos de recepción y de inventarios, garantizando así su funcionamiento armónico y eficiente en el contexto del sistema global de Reparauto. Se busca identificar y resolver cualquier problema relacionado con la interacción de estos componentes, asegurando que el flujo de datos y las operaciones entre ellos se ejecuten sin errores ni interrupciones.

1.2 Referencias

- Especificación de Requisitos del Proyecto
- Norma ISO/IEC/IEEE 29119 parte II
- Norma ISO/IEC/IEEE 29119 parte III
- Norma ISO/IEC/IEEE 29119 parte IV

1.3 Glosario

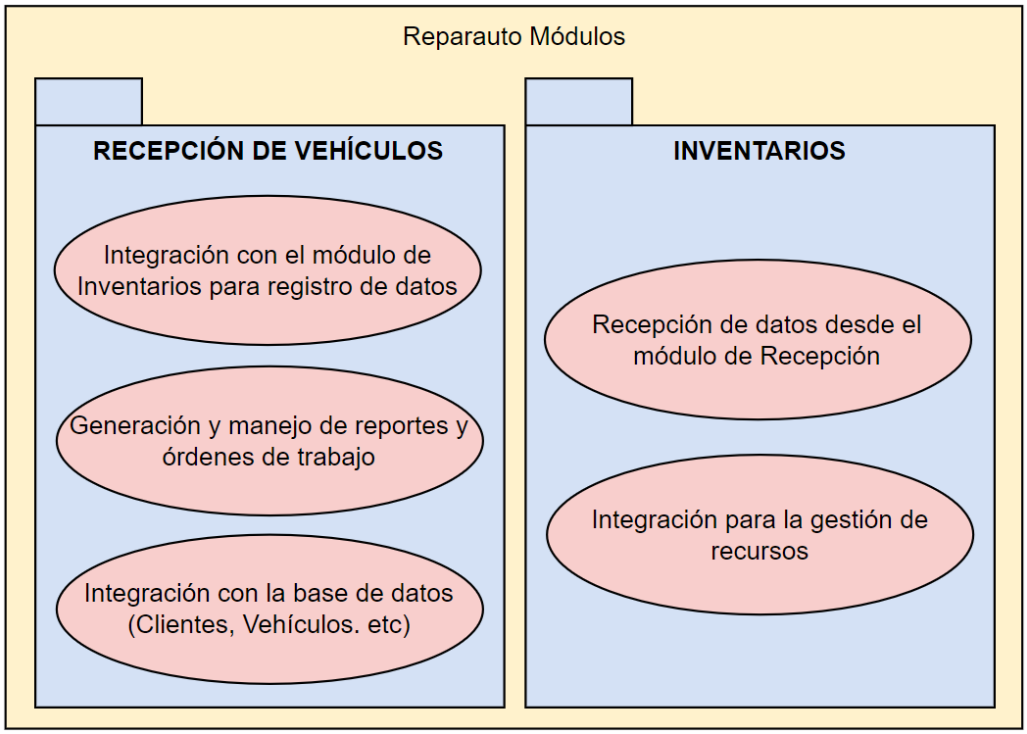
- **Backend**: Parte del software que opera en el servidor, manejando la lógica de negocio y la interacción con la base de datos.
- **Frontend**: Interfaz de usuario a través de la cual interactúan los usuarios finales con la aplicación.
- **Cypress**: Herramienta de automatización para realizar pruebas end-to-end en aplicaciones web.
- **Postman**: Aplicación para probar y desarrollar APIs, permitiendo enviar solicitudes HTTP y analizar respuestas.
- **Pruebas de Integración Basadas en Interfaces**: Evaluación del funcionamiento conjunto de dos o más componentes de software a través de sus interfaces.
- **Pruebas de Contrato**: Método para verificar que los componentes de software cumplan con los requisitos y especificaciones predefinidos.
- **Simulaciones**: Técnica para imitar el funcionamiento de un componente de software en un entorno de prueba.
- **Mocks**: Objetos de prueba que simulan el comportamiento de objetos reales de manera controlada.
- **ISO/IEC/IEEE 29119**: Serie de estándares internacionales para prácticas de prueba de software, incluyendo procesos, documentación, y técnicas.
- **UAT (User Acceptance Test):** Pruebas realizadas por usuarios finales para asegurarse de que el sistema cumple con los requisitos y es apto para el uso.
- **Entorno de Pruebas**: Configuración de software y hardware utilizada para ejecutar pruebas.
- **Rendimiento bajo carga**: Medida de cómo se comporta un sistema bajo un volumen de trabajo significativo.
- **Pruebas de Sistema**: Proceso de verificar que el sistema completo funciona según lo especificado.
- **Métricas de Pruebas**: Datos cuantitativos recopilados durante las pruebas que ayudan a evaluar la calidad y efectividad de estas.
- **Criterios de Suspensión y Reanudación**: Condiciones bajo las cuales las pruebas se detendrán temporalmente y luego continuarán.

II. CONTEXTO DE LAS PRUEBAS

2.1 Proyecto / Subprocesos de Prueba

- **Módulo de Recepción de Vehículos**: Este módulo es esencial en el proceso inicial de recepción de los vehículos, abarcando desde el registro de los datos del cliente y del vehículo hasta la documentación detallada del estado del vehículo. La prueba de integración se centrará en cómo este módulo interactúa con el sistema de inventarios, especialmente en la generación y el manejo de reportes y órdenes de trabajo.
- **Módulo de Inventarios**: Fundamental para la gestión eficiente de los recursos, este módulo controla el inventario de partes y suministros. Las pruebas se centrarán en la integración con el módulo de recepción, verificando cómo la información del inventario se actualiza y se refleja en tiempo real en respuesta a las operaciones de recepción de vehículos.

A continuación, se muestran los enfoques de Integración del proyecto.



2.2 Elementos de Prueba

Los elementos de prueba para la integración de los módulos de Recepción de Vehículos y de Inventarios en el sistema Reparauto incluyen:

1. **Interfaces de Usuario del Módulo de Recepción de Vehículos:**
   - Pantallas de registro de vehículos y clientes.
   - Formularios para la entrada de detalles del estado del vehículo.
   - Funcionalidades para la captura y carga de imágenes del vehículo.
2. **Base de Datos y Lógica de Negocio del Módulo de Recepción:**
   - Procedimientos de almacenamiento y recuperación de datos de clientes y vehículos.
3. **Interfaces de Usuario del Módulo de Inventarios:**
   - Pantallas para la visualización y gestión del inventario de partes y suministros.
   - Interfaces para actualizar el estado del inventario y registrar el uso de recursos.
4. **Base de Datos y Lógica de Negocio del Módulo de Inventarios:**
   - Sistemas para la gestión de datos de inventario.
   - Procesos automatizados para la generación de alertas de bajo nivel de inventario y pedidos de reposición.
5. **Integración entre los Módulos de Recepción y de Inventarios:**
   - Comunicación y transferencia de datos entre los dos módulos.
   - Sincronización de operaciones entre recepción de vehículos y gestión de inventarios.

2.3 Alcance de la Prueba

El alcance se enfoca en la interacción y el funcionamiento coordinado de los módulos de Recepción de Vehículos y de Inventarios. Estas pruebas verificarán cómo los datos se transfieren y gestionan entre estos dos módulos, asegurando que la información fluye correctamente desde el momento en que un vehículo es registrado en el módulo de Recepción hasta su reflejo en el sistema de Inventarios.

Las pruebas se concentrarán en validar la funcionalidad de las interfaces de usuario para ambos módulos, asegurándose de que sean capaces de interactuar efectivamente entre sí. Además, se evaluará la precisión y eficiencia en la

actualización de los registros de inventario en respuesta a las operaciones de recepción de vehículos.

2.4 Suposiciones y Restricciones

Suposiciones:

Ambiente de Pruebas Similar a la Producción:

- El ambiente de pruebas es un clon exacto del ambiente de producción. las pruebas se realizarán en un entorno que replica con precisión las condiciones reales de operación, incluyendo la configuración del hardware, software y la base de datos.

Disponibilidad de Datos de Prueba Representativos:

- Los datos utilizados en las pruebas reflejarán fielmente los datos reales de operación en Reparauto. Esto incluye datos de vehículos, clientes e inventarios que son representativos de los escenarios de uso cotidianos.

Estabilidad de los Requisitos del Sistema:

- Se supone que los requisitos del sistema para los módulos de recepción y de inventarios están bien definidos y no sufrirán cambios significativos durante el período de prueba.

Acceso a Recursos Necesarios:

- Se asume que el equipo de pruebas tendrá acceso completo a todos los recursos necesarios, incluyendo software, hardware y conocimientos técnicos, para llevar a cabo las pruebas de manera efectiva.

Restricciones:

Recursos y Capacidades del Entorno de Pruebas:

- El entorno de pruebas puede tener limitaciones en términos de capacidad y rendimiento en comparación con el entorno de producción, lo que podría afectar la extrapolación de los resultados de las pruebas a un entorno real.

Ventana de Tiempo para Pruebas:

- Las pruebas se llevarán a cabo dentro de un marco de tiempo específico, lo que puede limitar la profundidad y el alcance de las pruebas de regresión y de escenarios no planificados.

Cambios en el Entorno Tecnológico:

- Posibles actualizaciones o cambios en el entorno tecnológico durante el período de pruebas podrían requerir ajustes en el plan de pruebas.

2.5 Partes Interesadas

Parte Interesada	Roles y Responsabilidades
Cliente (Gerente de Reparauto Ibarra)	• Aprobación del Plan de Pruebas, el Cronograma de las Pruebas y los entregables. • Participación en las pruebas de aceptación del usuario (UAT). • Proporcionar feedback y requisitos para las UAT, que comenzarán con la versión Beta del producto.
Edison Zamora (Líder del Equipo de Pruebas)	• Desarrollo y coordinación del Plan de Pruebas. • Gestión del equipo de pruebas. • Comunicación continua con la gerencia y otros stakeholders. • Asegurar la alineación del plan de pruebas con los objetivos del proyecto.
Equipo de Desarrollo	• Implementación de correcciones y mejoras basadas en los resultados de las pruebas. • Colaboración con el equipo de pruebas para resolver problemas técnicos.
Usuarios Finales (Empleados de Reparauto)	• Participación en las UAT. • Proporcionar retroalimentación sobre la funcionalidad y usabilidad del sistema.
Equipo de Soporte Técnico	• Asistencia en la configuración y mantenimiento del entorno de pruebas. • Proporcionar soporte técnico durante la ejecución de las pruebas.

Stakeholders Externos	• En caso de integración con sistemas de terceros, su participación en las pruebas de integración. • Proporcionar información y requisitos necesarios para las pruebas.

III. COMUNICACIÓN DE LAS PRUEBAS

A continuación, se detallan los principales puntos de comunicación, sus propósitos, frecuencia, medios utilizados, responsables y audiencias:

Punto de Comunicación	Propósito	Frecuencia	Medios	Responsable	Audiencia
Reunión de Inicio	Iniciar oficialmente el proyecto de pruebas y establecer expectativas.	Una vez al comienzo del proyecto.	Reunión presencial	Líder de Proyecto / QA Manager	Equipo de pruebas, Desarrolladores, Stakeholders clave.
Reuniones Internas de Seguimiento	Monitorear el progreso y discutir desafíos en el proceso de pruebas.	Semanal.	Reunión presencial	Líder de Proyecto / QA Manager	Equipo de pruebas.
Reportes de Estado	Informar el estado actual y progreso de las pruebas.	Semanal.	Documento	Líder de Proyecto / QA Manager	Equipo de pruebas, Gerencia de Reparauto, Stakeholders clave.
Reporte de Hitos y Hallazgos Clave	Comunicar logros importantes o descubrimientos críticos.	Como sea apropiado (tras alcanzar hitos clave).	Documento herramienta de gestión de proyectos.	Líder de Proyecto / QA Manager	Equipo de pruebas, Gerencia de Reparauto, Desarrolladores.
Reuniones de Coordinación con Desarrollo	Coordinar esfuerzos entre el equipo de pruebas y desarrollo para la resolución de problemas.	Según necesidad.	Reunión presencial	Líder de Proyecto / QA Manager	Equipo de pruebas, Equipo de desarrollo.
Sesiones de Retroalimentación con Usuarios Finales	Obtener feedback directo de los usuarios finales sobre la usabilidad y funcionalidad.	Puntual (post-implementación y durante UAT).	Reunión presencial y/o encuestas	Líder de Proyecto / QA Manager	Usuarios finales (empleados de Reparauto).

IV. REGISTRO DE RIESGOS

En la siguiente tabla se identifican los riesgos del proyecto, así como se determina la severidad de cada uno de los riesgos multiplicando el impacto por la probabilidad de ocurrencia.

El impacto y la probabilidad se determinan teniendo en cuenta una escala de 1 al 5, donde 5 es el más alto.

No	Riesgos	Probabilidad (1-5)	Impacto (1-5)	Severidad (Probabilidad *Impacto)	Plan de Mitigación
1	Retrasos en la implementación de las funcionalidades.	2	5	**10**	Evaluar el avance del desarrollo de las funcionalidades y replanificar acorde al avance de ser necesario.
2	Fallos en la integración entre módulos de recepción e inventarios.	3	4	**12**	Realizar pruebas de integración frecuentes y tempranas para identificar y resolver problemas rápidamente.
3	Incompatibilidad entre el Frontend y Backend debido a actualizaciones.	3	4	**12**	Establecer protocolos de versión y pruebas de compatibilidad antes de implementar actualizaciones.
4	Pérdida de datos críticos o corrupción durante las pruebas.	2	5	**10**	Implementar procedimientos de respaldo y restauración de datos y usar datos de prueba representativos.

5	Rendimiento inadecuado del sistema bajo carga elevada.	3	4	**12**	Realizar pruebas de carga y estrés para evaluar y mejorar el rendimiento del sistema.
6	Problemas de seguridad y vulnerabilidades en los módulos.	3	5	**15**	Aplicar revisiones de seguridad regularmente y realizar pruebas de penetración.
7	Resistencia al cambio por parte de los usuarios finales.	4	3	**12**	Desarrollar y proporcionar formación y materiales de soporte para facilitar la transición a los nuevos sistemas.
8	Limitaciones técnicas o fallos del entorno de pruebas.	2	4	**8**	Asegurar la robustez del entorno de pruebas y tener un plan de contingencia para fallos técnicos.

V. ESTRATEGIA DE PRUEBA

5.1 Subprocesos de prueba

Para el proyecto Reparauto, los subprocesos de prueba serán los siguientes:

- Pruebas de Integración de Componentes: Se enfocarán en verificar la interacción entre el Backend desarrollado en Node.js y el Frontend en Angular, utilizando Cypress.
- Pruebas de Sistema: Evaluarán la funcionalidad completa del sistema para garantizar que todas las partes trabajan juntas correctamente.

5.2 Entregables de Prueba

Se debe generar la siguiente documentación:

- **Plan de Pruebas de Integración de Componentes.**
- **Especificación de Casos de Pruebas.**
- **Informes de Estado de las Pruebas.**
- **Informe de Finalización del Subproceso de Prueba.**

5.3 Técnicas de diseño de Prueba

Se utilizarán técnicas de diseño de pruebas enfocadas en la integración de componentes:

- **Pruebas de Integración Basadas en Interfaces:** Evaluación de la comunicación entre el Backend (Node.js) y el Frontend (Angular), asegurando que los datos se transmitan y procesen correctamente.
- **Simulaciones y Mocks**: Uso de herramientas para simular servicios y componentes del Backend durante las pruebas del Frontend, y viceversa.

5.4 Criterio de Finalización y Prueba

Se espera una cobertura de pruebas de integración del 80%, enfocándose en las interacciones clave entre los componentes.

No deben quedar defectos críticos que afecten la funcionalidad básica del sistema al concluir las pruebas.

5.5 Métricas

- **Casos de Prueba de Integración Ejecutados:** Número total de pruebas de integración realizadas.
- **Defectos Encontrados en la Integración**: Cantidad y gravedad de los defectos encontrados en las pruebas de integración.
- **Tiempo Medio de Solución**: Tiempo promedio requerido para solucionar problemas identificados en las pruebas de integración.

5.6 Requisitos del entorno de Pruebas

5.6.1 Ambiente de pruebas

Componente	Requisitos
Sistema Operativo	Windows 11
Navegadores	Chrome, Mozilla Firefox
Velocidad de Internet	Mínima (10Mbps), Recomendada (30Mbps)

5.6.2 Herramientas de Pruebas

Herramienta	Función	Uso Específico
Cypress	Automatización de pruebas end-to-end	Pruebas de integración de componentes Frontend y Backend, pruebas de interfaz de usuario
Postman	Automatización de pruebas de API	Pruebas de servicios RESTful del Backend, validación de contratos de API

5.7 Re-testing y regresión de las Pruebas

Re-testing: Las pruebas de confirmación (re-testing) se realizarán para verificar que las correcciones de errores anteriores se han implementado correctamente.

Regresión de Pruebas: Se realizarán pruebas de regresión para asegurar que los cambios recientes no hayan afectado negativamente a otras partes del sistema. **Se estima realizar al menos 3 ciclos de pruebas**, donde el último ciclo incluirá una prueba de regresión completa

5.8 Criterios de Suspensión y Reanudación

5.8.1 Criterios de suspensión

Se suspenderá el proceso de pruebas cuando:

1. La solución no cumple con las funcionalidades especificadas en la Especificación de Requisitos del Proyecto.
2. Una característica principal contiene errores que impiden probar áreas críticas del sistema.
3. El entorno de pruebas no es estable o no ofrece resultados confiables.

5.8.2. Criterio de reanudación

Se reanudará el proceso de pruebas cuando:

1. Se reanudarán las pruebas una vez que las partes interesadas acuerden continuar o tras solucionar los defectos/problemas identificados.

5.9 Desviaciones de la Estrategia de Prueba Organizacional

La cobertura de requisitos se reduce al 80% debido a que se espera que las pruebas de componente sean minuciosas y que los riesgos sean relativamente bajos. Esto se alinea con el enfoque de pruebas de integración centrado en componentes específicos del sistema

VI. ACTIVIDADES Y ESTIMADOS DE PRUEBA

Las pruebas se dividirán en las siguientes actividades principales:

1. Diseño y Desarrollo de Casos de Prueba de Integración
2. Configuración y Validación del Entorno de Pruebas
3. Ejecución de Pruebas de Integración - Ciclo Inicial
4. Análisis de Resultados y Ajustes
5. Ejecución de Ciclos de Re-testing y Regresión
6. Monitoreo Continuo y Reporte de Estado
7. Evaluación Final y Reporte de Cierre

VII. PERSONAL

7.1 Roles, Actividades y Responsabilidades

La matriz RACI (Responsible-Accountable-Consulted-Informed) a continuación ilustra qué rol está involucrado en qué actividad (es) y cuál es el nivel de participación. Los números de las actividades se refieren a la lista de actividades anterior.

Actividad N°	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Líder de QA	R	A	A	R	A	R	R
Analista de QA	A	R	C	A	R	A	C
Ingeniero de Automatización	C	C	R	C	R	C	I

- **R**esponsable **A**poyo **C**onsultado **I**nformado

7.2 Necesidades de Contratación

Dado que el proyecto de pruebas de Reparauto se llevará a cabo por un único tester, no se requiere la contratación adicional de Analistas de Calidad.

7.3 Necesidades de Entrenamiento

Para el tester encargado del proyecto Reparauto, se programará una sesión de capacitación intensiva en el uso del sistema y las herramientas de pruebas como Cypress y Postman. Se prevé que esta formación se extienda más allá de 6 horas para cubrir en profundidad los procedimientos de pruebas de integración y uso efectivo de las herramientas, acorde a los estándares de la ISO 29119.

VIII. CRONOGRAMA

El cronograma general de las pruebas dividido en días.

Actividades de Pruebas	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Análisis																														
Diseño																														
Entorno de Pruebas																														
Ejecución (3 ciclos)																														
Finalización																														

IX. ANEXOS

Casos de prueba de Integración

ID Caso de prueba	TC-001
Título	Integración de Módulo de Recepción
Descripción	Probar la integración completa del módulo de recepción de vehículos, incluyendo el registro de vehículos y clientes, y la captura y carga de imágenes todo desde el inicio de sesión.
Precondiciones	El usuario se encuentra en la página de inicio de sesión.
Pasos de la Prueba	1. Ingresar las credenciales de inicio de sesión 2. Ingresar a subproceso "Proforma" 3. Ingresar las placas del vehículo 4. Ingresar la cedula del Cliente 5. Registrar daños y valores de reparación 6. Confirmar y guardar información en el sistema 7. Ingresar a subproceso "Recepciones" 8. Registrar información del vehículo (Estado, Accesorios, fecha. etc.) 9. Subir fotografías del vehículo 10. Confirmar y guardar información en el sistema
Resultados Esperados	Los detalles del cliente y del vehículo se registran correctamente. Las imágenes se capturan/suben y se asocian con el registro del vehículo en el sistema. Se registra y se genera una orden de trabajo asociada al vehículo.
Postcondiciones	El registro del vehículo, junto con las imágenes y los detalles del cliente, se guarda y se puede acceder a él en el sistema

ID Caso de prueba	TC-002
Título	Integración de Módulo de Inventarios
Descripción	Verificar la funcionalidad completa de las interfaces de usuario para la visualización y gestión del inventario de partes y suministros, y para actualizar el estado del inventario y registrar el uso de recursos.
Precondiciones	El sistema debe estar operativo y accesible al usuario.
Pasos de la Prueba	1. Ingresar las credenciales de inicio de sesión 2. Navegar al Módulo de Inventarios 3. Revisar la Lista de materiales disponibles 4. Seleccionar un material y actualizar la cantidad y/o estado 5. Revisar la Lista de herramientas disponibles 6. Seleccionar una herramienta y actualizar la cantidad y/o estado 7. Revisar la Lista de maquinaria disponibles 8. Seleccionar una maquina y actualizar la cantidad y/o estado 9. Asignar un material a un proceso específico 10. Confirmar y guardar los cambios realizados en el inventario.
Resultados Esperados	Las pantallas muestran la información actual del inventario. Las actualizaciones y registros de uso se procesan y reflejan correctamente en el sistema.
Postcondiciones	Los cambios en el inventario se guardan de forma permanente y están visibles para futuras consultas y gestiones.

Documento de requerimientos de software

Automatización de Procesos Reparauto

Fecha: 10-sep-2023

Tabla de contenido

Contenido

Historial de Versiones

Fecha	Versión	Autor	Organización	Descripción
10/09/23	V 1.0	Edison Zamora	Reparauto	Inicio

Información del Proyecto

Empresa / Organización	Reparauto Ibarra
Proyecto	Automatización de Procesos
Fecha de preparación	
Cliente	Ing. Christian Murillo
Patrocinador principal	
Gerente / Líder de Proyecto	Sr. Edison Zamora
Gerente / Líder de Análisis de negocio y requerimientos	Sr. Edison Zamora

1. Propósito

Nombre: Automatización de Procesos de Recepción de Vehículos y de Inventarios.

Versión: v1.0

2. Alcance del producto / Software

El propósito del software es automatizar y mejorar la gestión de los procesos de recepción de vehículos y de inventarios en la empresa Reparauto, ubicada en Ibarra, Ecuador. El software busca aumentar la eficiencia operativa, la precisión y la visibilidad en tiempo real de los inventarios y las operaciones de recepción, lo que facilitará la toma de decisiones y contribuirá al crecimiento del negocio.

Beneficios: El software proporcionará los siguientes beneficios a la empresa y su organización:

- Mayor eficiencia en la gestión de inventarios.
- Reducción de errores humanos en los procesos de recepción.
- Visibilidad en tiempo real de la información clave.
- Mayor capacidad para tomar decisiones informadas.
- Mejora de la imagen de la empresa y retención de clientes.

Objetivos y Metas: Los objetivos del software están alineados con los objetivos corporativos y las estrategias de negocio de Reparauto. Estos objetivos incluyen:

- Optimizar los procesos de recepción de vehículos.
- Automatizar el módulo de inventarios.
- Implementar un plan de pruebas de integración.

3. Funcionalidades del producto

1. Automatización del módulo de recepción de vehículos.
2. Automatización del módulo de inventarios.
3. Integración de componentes de la aplicación web.
4. Pruebas de integración de componentes.

4. Clases y características de usuarios

1. Gerente
2. Jefe de Taller
3. Empleados
4. Clientes de Reparauto

5. Entorno operativo

Plataforma de Hardware: El sistema se ejecutará en PC's, Laptops y Dispositivos móviles como Celulares y Tabletas con conexión a Internet.

Versiones de Navegador: El sistema es compatible con las siguientes versiones de navegador

Navegadores de Escritorio:

- Google Chrome
- Mozilla Firefox
- Microsoft Edge (anteriormente conocido como Internet Explorer)
- Apple Safari (para macOS)
- Opera
- Brave

Navegadores Móviles:

- Google Chrome (para Android y iOS)
- Safari (para dispositivos iOS)
- Mozilla Firefox (para Android y iOS)
- Microsoft Edge (para Android y iOS)

El sistema será desplegado en un sistema web con acceso mediante dominio, todo el hosting será en la Nube, El Backend estará alojado en Heroku y El Frontend en Firebase.

6. Requerimientos funcionales

6.1. Requerimientos de Módulo de Recepción:

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF01	Registrar Datos completos del cliente
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El software debe registrar los campos adecuadamente para almacenar información completa del cliente, incluyendo nombre, dirección, número de teléfono, correo electrónico y otros datos relevantes. Esto permitirá una gestión eficiente de la información del cliente en el proceso de recepción de vehículos.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF02	Capturar información del vehículo
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe permitir la captura de información detallada sobre el vehículo, incluyendo marca, modelo, año, número de placa y cualquier dato relevante para la recepción.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF03	Almacenar información sobre el estado del vehículo
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
Debe ser posible registrar información sobre el estado del vehículo, incluyendo detalles de daños, kilómetros recorridos y condiciones generales.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF04	Permitir la captura de fotografías del vehículo
Tipo:	**Prioridad:**
Empleados	Alta
Descripción:	
Los técnicos deben poder tomar fotografías del vehículo para documentar su estado y los daños antes de la reparación.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF07	Generar un comprobante de recepción para el cliente
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe generar un comprobante de recepción que el cliente pueda conservar como comprobante de entrega del vehículo.	

6.2. Requerimientos de Módulo de Inventarios:

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF08	Registrar y mantener una lista de inventario de piezas y repuestos
Tipo:	**Prioridad:**
Jefe de Taller	Alta
Descripción:	
El sistema debe permitir llevar un registro detallado de todas las piezas y repuestos disponibles en el inventario de la empresa.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF09	Asociar cada elemento del inventario con un código único
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
Cada artículo en el inventario debe estar identificado con un código único para un seguimiento preciso.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF10	Permitir actualizaciones de inventario cuando se utilizan piezas en reparaciones
Tipo:	**Prioridad:**
Jefe de Taller	Alta
Descripción:	
El sistema debe actualizar automáticamente el inventario cuando se utilizan piezas en reparaciones, ajustando las cantidades disponibles.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF11	Notificar al personal cuando el inventario de un artículo es bajo
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe enviar notificaciones cuando las existencias de un artículo en el inventario alcanzan un nivel bajo, permitiendo la reposición oportuna.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF12	Generar informes de inventario periódicos
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
Debe ser posible generar informes periódicos que muestren el estado actual del inventario y las tendencias de uso de piezas.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF13	Realizar seguimiento de la fecha de adquisición y fecha de vencimiento de los materiales
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe llevar un seguimiento de la fecha de adquisición de los materiales y su fecha de vencimiento para garantizar el uso de materiales en buen estado.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF14	Asignar ubicaciones específicas para el almacenamiento de piezas en el inventario
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
Debe ser posible asignar ubicaciones específicas en el inventario para cada tipo de pieza o repuesto, facilitando su localización.	

6.3. Requerimientos de Integración de Módulos:

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF15	Sincronizar la información del vehículo registrado en el módulo de recepción con el módulo de inventarios
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
La información del vehículo registrado, incluyendo las piezas necesarias para la reparación, debe sincronizarse con el módulo de inventarios para garantizar disponibilidad.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF16	Actualizar el estado del vehículo en función de las reparaciones realizadas
Tipo:	**Prioridad:**
Empleados	Alta
Descripción:	
El sistema debe permitir a los técnicos actualizar el estado del vehículo en función de las reparaciones realizadas y su estado actual.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF17	Permitir la búsqueda de piezas del inventario necesarias para las reparaciones programadas
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
Los técnicos deben poder buscar y encontrar rápidamente las piezas necesarias en el inventario para las reparaciones programadas.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF18	Notificar al jefe de taller cuando una reparación está completa y el vehículo está listo para la entrega
Tipo:	**Prioridad:**
Empleados	Alta
Descripción:	
El sistema debe enviar notificaciones al jefe de Taller cuando una reparación ha sido completada y el vehículo está listo para la entrega al cliente.	

6.4. Requerimientos de Pruebas y Validación:

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF19	Implementar un sistema de pruebas de calidad antes de la entrega de vehículos reparados
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
Antes de la entrega de un vehículo reparado, se debe llevar a cabo un proceso de pruebas de calidad para garantizar que todas las reparaciones se han realizado adecuadamente.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF20	Generar informes de pruebas de calidad para su revisión
Tipo:	**Prioridad:**
Jefe de taller	Alta
Descripción:	
El sistema debe generar informes detallados de las pruebas de calidad realizadas para su revisión y archivo.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF21	Validar que los vehículos estén listos para la entrega antes de notificar al cliente
Tipo:	**Prioridad:**
Jefe de taller	Alta
Descripción:	
Antes de notificar al cliente sobre la disponibilidad de su vehículo, se debe validar que esté listo y en condiciones óptimas para la entrega.	

6.5. Requerimientos de Gestión de Usuarios

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF22	Asignar roles de usuario (gerente, técnico, recepcionista) con diferentes permisos
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe permitir la asignación de roles de usuario con diferentes permisos y niveles de acceso para garantizar la seguridad de la información.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF23	Realizar copias de seguridad regulares de los datos del sistema
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe realizar copias de seguridad regulares de los datos para garantizar la integridad de la información en caso de fallos del sistema.	

6.6. Requerimientos de Seguridad

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF24	Ofrecer soporte técnico y asistencia a los usuarios del sistema
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
Debe estar disponible un servicio de soporte técnico y asistencia para resolver problemas y brindar ayuda a los usuarios del sistema.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF25	Permitir el acceso al sistema desde ubicaciones remotas
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe permitir el acceso desde ubicaciones remotas, lo que facilita la gestión y supervisión del negocio.	

6.7. Requerimientos de Notificaciones

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF26	Generar un historial de cambios y actividad en el sistema
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe mantener un historial de cambios y actividad, permitiendo la revisión de acciones realizadas por los usuarios.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF27	Realizar actualizaciones y mejoras del sistema de forma periódica
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe someterse a actualizaciones y mejoras periódicas para mantenerlo actualizado y eficiente.	

1.5. Requerimientos de Informes de Gestión.

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF28	Generar informes de gastos en insumos por vehículo
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema deberá generar un informe detallado de los gastos por cada material, pieza, repuesto, mano de obra, etc. De cada vehículo.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF29	Generar informe detallado del tiempo de reparación de cada vehículo.
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe generar un informe del tiempo empleado en cada proceso de reparación del vehículo.	

Especificación de Requerimientos Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQF30	Generar reporte mensual de reparaciones
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe generar un reporte detallado sobre las reparaciones realizadas en el mes cada día N° 25.	

7. Requerimientos no funcionales

Especificación de Requerimientos No Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQNF01	Tiempo de Respuesta
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe tener un tiempo de respuesta promedio por debajo de 2 segundos para garantizar una experiencia de usuario fluida.	
Manejo de errores:	
El sistema debe mostrar mensajes de error claros y amigables para guiar al usuario en caso de problemas.	
Criterios de aceptación:	
El tiempo de respuesta promedio del sistema debe ser inferior a 2 segundos, medido en condiciones de uso normales.	

Especificación de Requerimientos No Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQNF02	Seguridad de Datos
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe garantizar la seguridad de los datos del cliente, aplicando medidas de cifrado y acceso restringido.	
Manejo de errores:	
Se deben registrar y notificar los intentos de acceso no autorizado a los datos.	
Criterios de aceptación:	
Los datos de los clientes deben estar cifrados y protegidos contra accesos no autorizados.	

Especificación de Requerimientos No Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQNF03	Escalabilidad
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe ser escalable para manejar un aumento futuro en la cantidad de usuarios y datos.	
Manejo de errores:	
Se debe realizar un monitoreo continuo del rendimiento del sistema para detectar cuellos de botella y planificar expansiones.	
Criterios de aceptación:	
El sistema debe poder manejar un aumento del 50% en la cantidad de usuarios sin una degradación significativa del rendimiento.	

Especificación de Requerimientos No Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQNF04	Compatibilidad de Navegadores
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe ser compatible con los navegadores más comunes, como Google Chrome, Mozilla Firefox, Microsoft Edge y Safari.	
Manejo de errores:	
Se debe realizar pruebas de compatibilidad en cada uno de los navegadores compatibles.	
Criterios de aceptación:	
El sistema debe funcionar correctamente en las últimas dos versiones de los navegadores mencionados.	

Especificación de Requerimientos No Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQNF05	Disponibilidad
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe estar disponible las 24 horas del día, los 7 días de la semana, con un tiempo de inactividad programado mínimo.	
Manejo de errores:	
Se deben programar actividades de mantenimiento y actualizaciones en momentos de menor tráfico. (10pm – 7am)	
Criterios de aceptación:	
El sistema debe estar disponible el 99.9% del tiempo, excluyendo el tiempo de inactividad programado.	

Especificación de Requerimientos No Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQNF06	Usabilidad
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe ser intuitivo y fácil de usar, requiriendo un mínimo de capacitación para los usuarios finales.	
Manejo de errores:	
Se deben recopilar y abordar retroalimentaciones de los usuarios para mejorar la usabilidad.	
Criterios de aceptación:	
El sistema debe ser calificado positivamente por al menos el 90% de los usuarios en encuestas de satisfacción.	

Especificación de Requerimientos No Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQNF07	Tolerancia a fallos
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe ser capaz de gestionar y recuperarse de fallos sin afectar la experiencia del usuario.	
Manejo de errores:	
Debe generarse un registro de errores detallado para identificar la causa de las fallas.	
Criterios de aceptación:	
El sistema debe continuar funcionando de manera adecuada incluso después de experimentar fallos menores y debe recuperarse de manera automática en caso de fallos críticos.	

Especificación de Requerimientos No Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQNF08	Consumo de Recursos
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe utilizar eficientemente los recursos de hardware para minimizar la carga en servidores y reducir costos operativos.	
Manejo de errores:	
Se deben realizar análisis de rendimiento para optimizar el uso de recursos.	
Criterios de aceptación:	
El sistema debe mantener el uso de CPU y memoria por debajo del 70% de capacidad en condiciones de carga máxima.	

Especificación de Requerimientos No Funcionales	
Requerimiento Funcional N°:	**Nombre:**
RQNF09	Documentación del Sistema
Tipo:	**Prioridad:**
Gerente	Alta
Descripción:	
El sistema debe contar con una documentación completa que describa su funcionamiento, instalación y solución de problemas.	
Manejo de errores:	
La documentación debe mantenerse actualizada a medida que se realicen cambios en el sistema.	
Criterios de aceptación:	
Debe existir una guía de usuario completa y una documentación técnica detallada que permita a los administradores resolver problemas de manera eficiente.	