

Maestría en ingeniería del software

Trabajo Fin de Maestría

DISEÑO DE UNA ARQUITECTURA DE SOFTWARE PARA LA GESTIÓN, MONITOREO E HISTORIFICACIÓN PARA TALLERES DE MANTENIMIENTO DE VEHÍCULAR

AUTOR: Ing. Triviño Guerra Esteban Rafael

Localidad industrial de la bahía, Cartagena, Colombia.

PROFESOR:Dr. Cadena Martínez Rodrigo

Junio 2022

Resumen

El presente trabajo se presenta con el propósito de describir de forma estructurada cada uno de los conceptos, actividades y procesos tenidos en cuenta para desarrollar el proyecto de investigación cuyo objetivo es diseñar la arquitectura de un *software* que le permita a los talleres de mantenimiento y reparación de vehículos llevar de forma digital los procesos relacionados con la gestión, el monitoreo y el manejo de históricos de los mantenimientos realizados a los vehículos.

Con el propósito de cumplir cada uno de los objetivos de este proyecto, se llevó a cabo una investigación descriptiva y de campo, mediante la que se ejecutó una serie de pasos que llevaron al desarrollo del producto final. Como primer paso se hizo un levantamiento de información, a partir de la que, como segundo paso, se establecieron los requisitos de software, que, a su vez fueron validados mediante una labor de campo y posteriormente refinados para luego, como paso final, diseñar la arquitectura del *software* de acuerdo a los requisitos establecidos.

Como resultado de la investigación se obtuvo la arquitectura de un de software permita a los talleres de mantenimiento de vehículos llevar de forma digital los procesos relacionados con la gestión, el monitoreo y el manejo de históricos de los mantenimientos realizados a los vehículos.

Palabras clave: arquitectura de *software*, mantenimiento de vehículos, historial de mantenimientos, gestión de la información.

Abstract

The present work is presented to describe in a structured way each one of the concepts, activities, and processes taken into account to develop the research project whose objective is to design a software architecture that allows maintenance workshops and repair of vehicles, digitally carry out the processes related to the management, monitoring, and management of historical of maintenance carried out on vehicles.

To fulfill each of the objectives of this project, a descriptive and field investigation was carried out, through which, it was executed a series of steps that led to the development of the final product. As a first step, an information survey was carried out, from which, as a second step, the software requirements were established, which in turn were validated through fieldwork and later refined to then, as a final step, design the architecture. of the software according to the established requirements.

As the main result of the investigation, a software architecture was obtained that allows vehicle maintenance workshops to carry out digitally the processes related to the management, monitoring, and historical management of maintenance carried out on vehicles.

Keywords: software architecture, vehicle maintenance, maintenance history, information management.

Dedicatoria

Este trabajo se lo dedico a mi hermosa madre, que sin su motivación para hacer la maestría en ingeniería de software este trabajo no se hubiese llevado a cabo, a mi padre que se sacrificó durante muchos años para darnos lo mejor a mis hermanos y a mí. También quiero dedicar este trabajo de forma muy especial al amor de mi vida, Adriana Otero, ya que sin su valioso apoyo incondicional este trabajo no hubiese sido posible.

Agradecimientos

Quiero agradecer primeramente a Dios por permitirme terminar este ciclo y comenzar uno nuevo, al doctor Rodrigo Cadena por su apoyo durante el desarrollo de la maestría en ingeniería de software.

Contenidos

Resumen ii
Abstract iii
Dedicatoria iv
Agradecimientos v
Capítulo 1. Marco introductorio 1
1.1. Antecedentes 1
1.2. Problema de investigación 2
1.3. Justificación 3
1.4. Establecimiento de hipótesis y objetivos 5
Capítulo 2. Marco teórico 7
2.1 Arquitectura de software 7
2.2 Ingeniería de software 7
2.3 Sistema 8
2.4 Sistematización de procesos 9
2.5 Mantenimiento 9
2.5.1. Mantenimiento preventivo 10
2.5.2. Mantenimiento correctivo 10
2.6 Mantenimiento computarizado 11
2.7 Modelo de vistas 4+1 11
2.8 Especificación de requisitos 13
2.9 Arquitectura orientada a servicios (SOA) 13
2.10 Patrón de diseño MVC (Modelo vista controlador) 14
2.11 Patrón MVVM (Model - View - ViewModel) 15
2.12 Estado del arte 16
2.12.1. Contexto internacional 17
2.12.1.1. Software para el mantenimiento de una empresa manufacturera 17
2.12.1.2. Software Prisma 17
Capítulo 3. Marco empírico 19
3.1. Metodología 19
3.1.1. Diseño y tipo de investigación 19
3.1.2. Población y muestra 19
3.1.3. Instrumentos de evaluación 20

3.1.4. Análisis de la información 23
3.1.4.1. Obtener información 23
3.1.4.2. Capturar, transcribir y ordenar la información 23
3.1.4.3. Codificar la información 24
3.1.4.4. Integrar la información 24
3.2. Resultados 25
3.2.1. Especificación de requisitos 26
3.2.2. Modificaciones a la especificación de requisitos 41
3.2.3. Diseño de la arquitectura del sistema 43
Capítulo 4. Conclusiones y recomendaciones 77
Referencias bibliográficas 78
Apéndices 84

Índice de tablas y figuras

Tablas

Tabla 1. *Requisito funcional 1* 26

Tabla 2. *Requisito funcional 2* 27

Tabla 3. *Requisito funcional 3* 27

Tabla 4. *Requisito funcional 4* 28

Tabla 5. *Requisito funcional 5* 28

Tabla 6. *Requisito funcional 6* 29

Tabla 7. *Requisito funcional 7* 30

Tabla 8. *Requisito funcional 8* 31

Tabla 9. *Requisito funcional 9* 32

Tabla 10. *Requisito funcional 10* 33

Tabla 11. *Requisito funcional 11* 34

Tabla 12. *Requisito funcional 12* 35

Tabla 13. *Requisito funcional 13* 36

Tabla 14. *Requisito funcional 14* 37

Tabla 15. *Requisito funcional 15* 38

Tabla 16. *Requisito funcional 16* 39

Tabla 17. *Requisito funcional 17* 39

Tabla 18. *Requisito funcional 18* 40

Tabla 19. *Requisito funcional 1 modificado* 41

Tabla 20. *Requisito funcional 16 modificado* 42

Tabla 21. *Requisito funcional 7 modificado* 43

Figuras

Figura 1. *Modelo de dominio del sistema* 44

Figura 2. *Vista de escenarios* 46

Figura 3. *Diagrama de actividades registrar cliente* 48

Figura 4. *Diagrama de actividades registrar vehículo* 49

Figura 5. *Diagrama de actividades registrar mecánico* 50

Figura 6. *Diagrama de actividades ver información del cliente* 51

Figura 7. *Diagrama de actividades ver información del vehículo* 53

Figura 8. *Diagrama de actividades ver la información del mecánico* 54

Figura 9. *Diagrama de actividades modificar la información del cliente* 55

Figura 10. *Diagrama de actividades modificar la información del vehículo* 56

Figura 11. *Diagrama de actividades modificar la información del mecánico* 57

Figura 12. *Diagrama de actividades registrar ingreso de vehículo* 59

Figura 13. *Diagrama de actividades registrar salida de vehículo* 61

Figura 14. *Diagrama de actividades iniciar mantenimiento* 62

Figura 15. *Diagrama de actividades pausar mantenimiento* 64

Figura 16. *Diagrama de actividades reanudar mantenimiento* 65

Figura 17. *Diagrama de actividades finalizar mantenimiento* 66

Figura 18. *Modelo de clases general* 68

Figura 19. *Modelo de clases backend* 70

Figura 20. *Modelo de clases frontend* 71

Figura 21. *Diagrama de componentes* 73

Figura 22. *Diagrama de despliegue* 75

Capítulo 1. Marco introductorio

1.1. Antecedentes

El desarrollo de la tecnología está afectando cada vez más industrias de forma positiva, dentro de las cuales se encuentra la industria de los talleres de mantenimiento de vehículos.

En la actualidad existen diferentes alternativas de *software* que permiten llevar un control sobre el histórico de los mantenimientos que se realizan a vehículos, así como los costes, programación de mantenimientos, gestión de piezas, etc. Sin embargo, hoy en día muchos de los talleres de mantenimiento de vehículos no cuentan con una solución de *software* que les permita llevar un control sobre los aspectos mencionados anteriormente para cada uno de los vehículos ingresados sobre los cuales se realizan mantenimientos.

Una de las razones por las que esta situación se presenta es porque muchas de las soluciones *software* existentes son de carácter privado, pertenecen a grandes talleres y talleres de grandes marcas de vehículos. Otra razón es que, a pesar de que en el mercado existen diferentes soluciones de *software* que permitan la gestión, monitoreo, manejo de históricos de los mantenimientos de vehículos, entre otros, muchas de estas están dirigidas no a empresas o talleres de mantenimiento, sino a empresas o personas particulares que son dueñas de flotas de vehículos para que lleven el control de los mantenimientos realizados a sus vehículos.

Hasta la fecha se han realizado diversos trabajos de investigación cuyo propósito es la implementación de diferentes *software* de mantenimiento de vehículos, pero acotados a poblaciones específicas, tal como lo es el trabajo titulado *Implementación de un software de gestión de mantenimiento automotriz para el control preventivo y correctivo de la flota vehicular del GADIP Municipio de Cayambe,* en el cual se implementó un *software* de mantenimiento automotriz para los vehículos pertenecientes

al Gobierno Autónomo Descentralizado Intercultural y Plurinacional (GADIP), a través del cual lograron una mejora en la organización y planificación, así como en la reducción del 12,81% en los gastos (Cuatucuago, 2022, p. 14).

Los investigadores Mozo Coronel y Méndez Garcés (2020) publicaron su trabajo titulado *Elaboración de un plan de mantenimiento preventivo mediante la implementación de un software para la flota vehicular del gobierno autónomo descentralizado del cantón Otavalo*, mediante el cual lograron reducir los costos en un 9,02% representados en un ahorro de $ 6.200,31.

Otro trabajo realizado para la implementación de un *software* para el manejo de los mantenimientos preventivos de vehículos fue el desarrollado en Ecuador por Quisilema (2018), el cual estuvo orientado a los mantenimientos realizados a los vehículos pertenecientes al Gobierno Autónomo Descentralizado del Cantón Espejo (GADME), y mediante el cual se logró una disminución del 12% en los gastos generales de los mantenimientos (p. 8).

En 2017, Cabrera y Suárez desarrollaron la *Implementación de un plan de mantenimiento automotriz con indicadores CMD mediante el diseño de un software especializado para la flota de vehículos livianos del GAD Provincial de Chimborazo*, trabajo mediante el cual realizan la medición de diferentes indicadores CMD como la disponibilidad, la confiabilidad en actividades correctivas y actividades programadas y la mantenibilidad de actividades correctivas y actividades programadas.

1.2. Problema de investigación

En la actualidad, muchos de los talleres de mantenimiento de vehículos no cuentan con un sistema de *software* que les permita tener un control sobre la gestión, así como de la información propia del detalle de cada mantenimiento, ni del registro de los

historiales de cada vehículo que ingresa al taller, dando cabida así a una falta de organización y una deficiencia en el control de los mantenimientos, lo cual, según Cuatucuago (2022), puede causar malestar en los usuarios (p. 16).

Al no contar con el apoyo de una herramienta de *software*, a los talleres se les dificulta o imposibilita la toma y análisis de indicadores de mantenimiento, a través de los que se puede tener una base para la toma de decisiones que ayuden mejorar los procesos de los mantenimientos que se realizan dentro del taller, así como para conseguir una disminución en los tiempos y costos de mantenimiento, lo que lleva, a su vez, a un mejor servicio y satisfacción del cliente.

Al analizar la situación previamente descrita se puede sintetizar la problemática a través de la siguiente pregunta de investigación:

¿Cómo plantear una arquitectura de *software* a través de la cual se representa un sistema que permita a los talleres de mantenimiento llevar un control sobre el registro, la gestión, el detalle, el manejo de históricos de los mantenimientos prestados a los vehículos que ingresan al taller?

1.3. Justificación

Con el desarrollo de la presente investigación se brinda una solución de arquitectura de un *software* que ayude a los talleres de mantenimiento de vehículos a realizar de forma sistematizada procesos que, sin esta solución, actualmente muchos de los talleres lo realizan de forma manual, lo que implica un consumo de tiempo y recursos que pueden ser optimizados para lograr una mayor productividad y un mejor manejo de los recursos.

Una de las razones por la cual se considera necesaria esta investigación es porque con su implementación se representa una disminución los tiempos de los procesos de los

talleres, lo que representa, a su vez, un incremento en la tasa de ingreso de vehículos para mantenimiento, llevando tanto a una mayor satisfacción para los clientes como al incremento de las ganancias del taller.

Otra de las razones por la que es útil la realización de este proyecto es porque su implementación representa un avance tecnológico para los talleres de mantenimiento de vehículos, ya que esta permite el cálculo automatizado de los indicadores de mantenimiento, y según García (2017), “una ventaja adicional es que, una vez automatizado, podemos generar informes con la periodicidad que queramos, con un esfuerzo mínimo” (p. 2).

En cuanto al impacto ambiental, esta investigación es necesaria porque ayuda a disminuir el uso de papel en los procesos que se realizan actualmente en muchos de los talleres de mantenimiento como el registro del ingreso de los vehículos y los datos del propietario, el registro del diagnóstico de los vehículos y, en algunos casos, el historial de mantenimiento.

El investigador contó con los recursos bibliográficos, de personal y de tiempo para la realización de la presente investigación, haciendo que esta sea viable desde el punto de vista financiero, ya que no implicó grandes gastos económicos por concepto de materiales o herramientas para su exitosa realización.

Desde el punto de vista académico es importante el desarrollo de este trabajo, ya que deja documentada toda la arquitectura del *software* diseñado durante el desarrollo de la misma, haciendo de este proyecto un aporte académico significativo para futuras investigaciones que pretendan realizar nuevos aportes y/o avances relacionados a la materia aquí tratada.

1.4. Establecimiento de hipótesis y objetivos

Hipótesis

Con la realización de la presente tesis se obtendrá como resultado una arquitectura *software* para la gestión de los procesos que se llevan a cabo actualmente dentro los talleres de mantenimiento de vehículos, desde el momento en el que se agenda la reparación, hasta el momento de darle de alta al vehículo, pasando por todos los procesos intermedios que esto conlleva como la entrada del vehículo al taller, la asignación del técnico que realizará el mantenimiento, la comunicación con el cliente, la solicitud de los repuestos que se lleguen a necesitar, el registro del histórico, entre otros. La implementación de esta tesis supone una mejora y agilización en los procesos mencionados anteriormente, debido a que al utilizar medios tecnológicos la información que se gestiona dentro del taller siempre estará al alcance de la mano, tanto de los técnicos, como del taller y de los clientes, de acuerdo con el nivel de acceso a la información de cada rol. El desarrollo de esta tesis también supone un avance en el campo ambiental, ya que al digitalizar los procesos se conlleva al ahorro en el gasto de papelería utilizada hoy en día para registrar los procesos de cada mantenimiento.

Objetivo general

Diseñar la arquitectura de un *software* que le permita a los talleres de mantenimiento y reparación de vehículos llevar de forma digital los procesos relacionados con la gestión, el monitoreo y el manejo de históricos de los mantenimientos realizados a los vehículos.

Objetivos específicos

1. Realizar un levantamiento de información en campo que sea útil para la especificación de requerimientos.

2. Especificar los requerimientos del *software* para determinar las características funcionales y no funcionales del sistema, aplicando el estándar IEEE 830.
3. Validar los requerimientos establecidos mediante una labor de campo.
4. Refinar los requerimientos de acuerdo con los resultados de la labor en campo.
5. Diseñar la arquitectura del *software* teniendo en cuenta los requerimientos establecidos.

Capítulo 2. Marco teórico

La tecnología juega un papel fundamental en el sector industrial y sus aplicaciones potenciales son muy amplias. En esta sección se presentarán las tecnologías y servicios a utilizar en este proyecto, junto con sus principales beneficios y ventajas.

2.1 Arquitectura de software

En ingeniería de *software* existen muchos términos con múltiples definiciones**,** lo mismo ocurre con el concepto de arquitectura de *software*, no existe una definición universal de este concepto. Sin embargo, existe una definición general que ofrece el Instituto de Ingeniería de Software (SEI, por sus siglas en inglés), la cual tiene gran relevancia y es ampliamente aceptada: la arquitectura de *software* de un sistema es un grupo de estructuras indispensables para inferir sobre el mismo, esto incluye elementos de *software*, que llevan relación y propiedades entre ellos (Bass et al., 2003).

Por último, según lo acordado por el IEEE Std 1471-2000 y adoptado por Microsoft, una de las definiciones más aceptadas es la siguiente: la arquitectura de *software* es la estructura fundamental de un sistema, incorporada un factor fundamental como son las relaciones entre ellos, el entorno y razones que guían su diseño y evolución (Reynoso, 2004).

2.2 Ingeniería de software

La definición de ingeniería de *software* es bastante amplia, ya que el campo abarca una gran variedad de actividades. Sin embargo, en términos generales, la ingeniería de *software* implica el proceso de diseñar, crear, probar y mantener aplicaciones de *software*. Es un campo importante porque ayuda a construir aplicaciones

útiles que hacen la vida más fácil. Además, la ingeniería de *software* se encarga de desarrollar las herramientas que necesitamos para resolver problemas complejos.

Para Sommerville (2004) la ingeniería de software es un campo amplio, complejo y en constante cambio que combina muchas disciplinas, incluidas las ciencias de la computación, matemáticas, el diseño, la administración y la interacción humano - computador.

Pressman (2010), en su libro *Ingeniería del software*, describe el concepto de ingeniería de *software* como una tecnología compuesta por varias capas en las cuales se encuentra el compromiso con la calidad,métodos, herramientas y proceso. Cualquier orientación de ingeniería (incluso el de *software*) debe estar basada en un compromiso obligatorio de calidad con la organización.

2.3 Sistema

La palabra sistema viene desde mucho tiempo atrás, sin embargo, actualmente es difícil obtener definiciones precisas y claras debido a los requisitos lógicos existentes que deben cumplirse en diferentes contextos del sistema terminológico. Es por esto que, a continuación, se muestran algunas definiciones de la palabra sistema:

Se puede pensar en un sistema como algún aspecto de la realidad que se puede describir donde se enuncia básicamente una serie de componentes y la forma de interacción entre ellos, lo que proporciona un vínculo organizado y unificado en un sistema (Duffua et al., 2000)

Por otro lado, Bustamante y Ramos (2009) afirman que el sistema es la idea de una colección de elementos que están conectados entre sí para formar el todo, de modo que representan propiedades del todo en lugar de partes

La definición de sistema que se acaba de presentar tiene una forma específica de ver el mundo. Al aplicar una visión de sistema, el problema no es solo investigar esa parte individualmente, sino comprender toda la realidad a través de un enfoque holístico en el que la relación entre esa parte y el concepto de la organización juega un papel importante.

2.4 Sistematización de procesos

Acosta (2005) afirma que la sistematización se define como el uso de sistemas y tecnologías de la información que permiten realizar las tareas de forma rápida y sencilla. La sistematización implica cambios en los procesos, actividades o controles con el fin de brindar a los usuarios herramientas que los ayuden en su trabajo, incluyendo ciertos cálculos, tareas y procedimientos para apoyar su trabajo en la empresa.

Jara (2011) sostiene que la sistematización es una interpretación importante de una o más experiencias para descubrirlas, desentrañar, reorganizar y reconstruir la lógica del proceso, los factores que interfirieron en el proceso, cómo se vinculan y por qué lo hicieron. Por su parte, Barnechea y Morgan (2007) entienden la sistematización como un proceso de reconstrucción y análisis de la experiencia, a través del cual se explica para comprender lo sucedido.

2.5 Mantenimiento

El concepto de mantenimiento se puede definir como actividades que aumentan la confiabilidad de los sistemas de producción, cuyo propósito es mantener la infraestructura como maquinarias y equipos en buenas condiciones para desarrollar las funcionalidades para la que fueron desarrolladas y alcanzar su máximo desempeño, lo que incluye servicio, pruebas, inspecciones, ajustes, reemplazos, entre otras.

Por otro lado, Botero (2009) argumenta que el mantenimiento es una serie de actuaciones que deben llevar a cabo los responsables de este servicio o de esta área, para que los equipos, máquinas, componentes e instalaciones que intervienen en una empresa industrial estén en las condiciones óptimas para lo que fueron diseñados, construidos, instalados y puestos en servicio.

Según lo expuesto anteriormente, existen diversas formas de realizar un mantenimiento a un equipo o maquinaria en producción y cada una de ellas tiene sus propias características, a continuación, se describen los dos principales tipos de mantenimiento industrial:

2.5.1. Mantenimiento preventivo

Este es un conjunto de actividades que detectan y solucionan problemas antes de que ocurran y brindan soporte a usuarios, operadores y mantenimiento con total previsibilidad y operación segura, al igual que se encargan de comprobar el funcionamiento de máquinas, equipos, vehículos, etc. (Osorio, 2012).

2.5.2. Mantenimiento correctivo

Mantenimiento correctivo, también conocido como mantenimiento predictivo, es el proceso por el cual se intenta reducir al mínimo las posibilidades de que ocurra un fallo en un equipo. Para lograr esto, se analiza el comportamiento de los componentes del mismo y se detectan aquellos que están mostrando signos de deterioro (Molina, 2007).

Asimismo, Torres (2010) sostiene que el mantenimiento preventivo se refiere a un conjunto de tareas programadas y periódicas que se aplican a todas las instalaciones, máquinas o equipos con el fin de prolongar su vida útil. Este tipo de mantenimiento se lleva a cabo para reducir la probabilidad de falla y, como tal, las actividades se realizan antes de que ocurra la falla.

2.6 Mantenimiento computarizado

El propósito de la gestión de mantenimiento computarizado, o CMMS, es utilizar sistemas computarizados para administrar todos los aspectos del mantenimiento, incluidos los aspectos técnicos, económicos y organizativos. Esto puede incluir mantenimiento preventivo, predictivo, correctivo y de modificación para cualquier tipo de equipo o instalaciones en uno o más negocios. El mantenimiento computarizado brinda a los administradores de TI la capacidad de supervisar y administrar el rendimiento de los equipos de TI, así como la planificación y ejecución de tareas de mantenimiento preventivo. Debido a su importancia, se han creado una gran cantidad de diferentes sistemas de CMS para satisfacer las necesidades de las organizaciones de todo el mundo. Estos sistemas varían en cuanto a la funcionalidad (Alulema, 2015).

Un sistema GMAO/CMMS consta de varios módulos que interactúan entre sí, lo que permite la ejecución y el control integral de las tareas típicas del departamento de mantenimiento, tales como:

- Realizar un seguimiento de las fallas y errores de las máquinas, así como generar un historial operativo para cada máquina o equipo.
- Programar tareas de mantenimiento preventivo como limpieza y engrase.
- Las existencias de repuestos deben administrarse cuidadosamente para mantener niveles óptimos; este proceso se conoce como gestión de almacenes.
- Transmisión y seguimiento de “ordenes de trabajo” para el control y realización del mantenimiento.

2.7 Modelo de vistas 4+1

Se basa en un modelo propuesto por Kruchten (1995), el cual recibe su nombre por el número de vistas que propone: 4 vistas que se consideran ortogonales (aunque no

del todo), y la quinta vista es utilizada para ilustrarlas y validarlas. En total, hay 5 vistas las cuales se describen a continuación:

Vista lógica: la vista explica que el diagrama dado tiene un enfoque en el dominio del problema y está orientado al análisis de los servicios que el sistema proporciona a los usuarios finales. Es por esto que las principales preocupaciones son los requisitos funcionales, que se pueden modelar mediante clases (si se utiliza la orientación a objetos).

Vista de procesos: esta vista documenta los procesos que están en el sistema, también especifica el flujo de trabajo enumerando cada uno de los pasos y el funcionamiento de los componentes que se cumplen de esos procesos.

Vista de desarrollo: en este caso, la perspectiva se centra en la organización de módulos de *software* que ya han sido diseñados e implementados dentro del entorno de desarrollo.

Vista física: según una definición establecida por Kruchten afirma que esta vista está basada principalmente en el *hardware* y *software* así como también cada una de las conexiones y componentes físicos a diversos servicios con el cual se crea el punto de vista del proyecto

Vista de escenario: esta última vista se ilustra como una guía arquitectónica utilizando un conjunto de casos de uso para descubrir elementos arquitectónicos, lo que genera secuencias de interacción durante el proceso de diseño. Esta vista no establece ningún vínculo estable que mantendría la relación entre las vistas, ya que la relación que se establece entre ellas tiene únicamente fines de validación y prueba.

2.8 Especificación de requisitos

De acuerdo con Farsimadan (2014), el propósito de un documento de especificación de requisitos de *software* es proporcionar una definición completa de un producto de *software* que se desarrollará, incluido su propósito, los procesos comerciales clave que admitirá, características, parámetros de rendimiento y comportamiento. Principalmente sirve para guiar el proceso de desarrollo manteniendo todo en el camino correcto.

Para la IEEE Std 1471-2000 (2008) la especificación de requisitos de *software* (SRS) es un documento que define en detalle las características y capacidades de un sistema de *software*. Los desarrolladores, usuarios, clientes y otras partes interesadas pueden utilizar un SRS para comprender qué debe hacer el sistema y cómo debe funcionar. También puede servir como modelo para el desarrollo, asegurando que el producto final sea satisfactorio para los participantes.

2.9 Arquitectura orientada a servicios (SOA)

Un SOA (arquitectura orientada a servicios) es una arquitectura de *software* basada en los conceptos clave de *front-end* de aplicación, servicio, repositorio de servicio y bus de servicio. Un servicio se compone de un contrato, una o más interfaces y una implementación. Los componentes básicos de SOA son los servicios, que están débilmente acoplados para promover la reutilización y son altamente interoperables a través de sus contratos (Maurizio et al., 2008)

Desde esta perspectiva, según Krafzig et al. (2005), la arquitectura orientada a servicios (SOA) define la interrelación de diferentes unidades funcionales de aplicaciones, que se llaman servicios, vistas a través de interfaces y contratos definidos previamente entre estos servicios, definiéndose de forma neutral volviéndose

independiente de la plataforma utilizada para *hardware*, el sistema operativo y el lenguaje de programación. Esto permite que los servicios creados en sistemas diferentes interactúen entre sí de manera homogénea y general.

De lo anterior se concluye que SOA se basa en un modelo de componentes que intercomunica todas las unidades funcionales que se complementa con las aplicaciones, denominadas servicios, a través de interfaces y contratos bien estructurados, entre estos servicios. Por lo tanto, esta interfaz se define de forma ecuánime y debe ser autónoma de la plataforma de *hardware*, el sistema operativo y el lenguaje de programación que se utiliza (González, 2007).

2.10 Patrón de diseño MVC (Modelo vista controlador)

El patrón arquitectónico MVC (*Model-View-Controller*) se aplicó originalmente a los modelos de interfaz gráfica de usuario, para entradas, procesamiento y salida. Esta arquitectura propone dividir las aplicaciones en tres diferentes capas, y cada capa hace alusión a los diversos elementos que componen el *software*. Estas capas, las cuales dividen al patrón MVC, son el modelo, la vista y el controlador y que a continuación de describen brevemente (Bodhuin et al., 2003).

MVC es un patrón de abstracción de desarrollo de *software* que separa los datos, la lógica comercial y la interfaz de usuario de una aplicación en tres componentes diferentes. Lo que se conoce como patrón de llamada y retorno de MVC se ve comúnmente en aplicaciones web, creando así una vista HTML y el código que proporciona datos dinámicos a la página (Guzmán & Tovar, 2013).

Modelo: el modelo representa los datos de un sistema o aplicación conteniendo toda la lógica para acceder a ellos y manipularlos. Los servicios que gestiona el modelo deben ser lo suficientemente generales para admitir todo tipo de clientes y debe ser fácil

comprender cómo controlar el comportamiento del modelo simplemente revisando paso a paso la lista de métodos (Alvarado, 2012).

Vista: la vista es responsable de permitir el acceso a los datos del modelo, especificar la manera en que se mostrarán esos datos y actualizar la presentación de esos datos cuando se produce una alteración en el modelo. La semántica de la presentación reside dentro de la capa de vista, es por esto que la información contenida en el modelo puede ser adaptable a diferentes tipos de vistas. Esta capa puede ser modificada cuando la capa de modelo se comunica directamente con ella y en su momento, la vista le envía información ingresada por el usuario directamente al controlador (Alvarado, 2012).

Controlador: determina el comportamiento de la aplicación. Envía solicitudes de usuarios y selecciona presentaciones posteriores en función de la información ingresada por el usuario y los resultados de las operaciones realizadas por el formulario. En otras palabras, interpreta la entrada del usuario y la relaciona con las acciones realizadas por el formulario (Martínez, 2014).

2.11 Patrón MVVM (Model - View - ViewModel)

Modelo-Vista-Modelo de Vista (MVVM) es la traducción más confiable de *Model View Controller*. Es una forma de *Model View Controller* diseñada para plataformas modernas de desarrollo de interfaz de usuario, donde la vista es responsabilidad del diseñador clásico, no del desarrollador. MVVM también se basa en enlace de datos común El mecanismo personalizado permite que la clase *View* se desarrolle fácilmente a partir del resto del modelo eliminando todo el código detrás de la clase *View* (Smith, 2009).

El concepto principal de MVVM es la separación del modelo con la vista al introducir una capa abstracta en medio ellos, llamada modelo de vista. La vista y el

modelo de vista normalmente son instalados por la aplicación contenedora. La vista contiene una referencia al modelo de vista. El modelo de vista expone comandos y entidades observables o enlazables a los que se puede enlazar la vista. Las interacciones del usuario con la vista disparan comandos contra el modelo de vista y de manera similar, las actualizaciones en el modelo de vista se propagan automáticamente a la vista a través del enlace de datos (Llorente et al., 2010)

Por último, según investigaciones realizadas por Ocsa et al. (s.f.) confirman que el modelo MVVM es un modelo de arquitectura basado en el modelo de vista controlador o MVC permitiendo la separación del código de interfaz de usuario con el código que o pertenece a la interfaz de usuario dentro de una aplicación multiplataforma.

2.12 Estado del arte

La visión, en lo que concierne a mantenimiento,ha sufrido muchos cambios, debido a que en la actualidad las industrias han implementado más equipos automatizados y complejos en su diseño, así como se han sido desarrolladas nuevas técnicas, herramientas de planificación, análisis y ejecución de los mantenimientos, todo ello con un enfoque completamente nuevo de las responsabilidades y de la organización de estos (López & Guamán, 2015). Si bien las computadoras han experimentado un gran crecimiento tecnológico, muchos sistemas de mantenimiento que existen en la actualidad han venido presentando ciertas limitaciones en cuanto a la prevención y diagnóstico de fallas, lo que, a su vez, exige que el personal sea cada vez más intenso y lo obliga a actuar y pensar de una forma diferente. Esto implica que se ha tenido que buscar nuevas opciones que nos permitan realizar la toma de mejores decisiones y estrategias basadas en modelos que incorporen nuevas técnicas o metodologías de mantenimiento que consigan el mayor beneficio posible para las empresas.

Existen una serie de investigaciones realizadas previas a este trabajo que se relacionan al contexto internacional, así como también al contexto nacional y regional expuestas a continuación:

2.12.1. Contexto internacional

2.12.1.1. Software para el mantenimiento de una empresa manufacturera

En la empresa Ambrake Manufacturing LTD, fue propuesto un sistema CMMS basado en el reconocido *software* AutoCAD de AutoDesk, este *software* es creado por estudiantes de la universidad de Louisville junto a empleados de la empresa anteriormente mencionada (Ardila et al., 2015).

A pesar de que esta propuesta es bastante prometedora en cuanto a la implementación del *software* de gestión de mantenimiento, esta no cuenta con un diseño *responsive* para adaptarlo a otros dispositivos como móvil o *tablet* ya que fue implementado de manera local y solo podrá ser utilizado en *desktop*.

2.12.1.2. Software Prisma

Por otro lado, Juez (2012) afirma haber implementado un *software* llamado Prisma, este *software* fue implementado en la empresa Sisteplant y está siendo utilizado para optimizar procesos de mantenimiento mediante la información estructurada. Ha tenido una implementación progresiva gracias a su modularidad, cumpliendo con los estándares mundiales establecidos. Por último, al *software* se le implementó un sistema de mensajería tipo alerta la cual es enviada a los dispositivos móviles en tiempo real.

Por las razones anteriores, encontrará que este poderoso *software* es costoso, difícil de implementar y el módulo no está disponible a pesar de estar limitado por la

política de mantenimiento de activos de su infraestructura comercial. Por esta razón, este *software* no es adecuado para las necesidades de la industria automotriz.

Capítulo 3. Marco empírico

3.1.Metodología

3.1.1. Diseño y tipo de investigación

Con el objetivo de cumplir con cada uno de los objetivos específicos, que a su vez contribuyen para alcanzar el objetivo general del proyecto, este trabajo de grado se llevó a cabo utilizando una metodología iterativa e incremental. El proyecto fue desarrollado entre el intervalo de tiempo entre el mes de febrero y el mes de mayo de 2022 en la ciudad de Cartagena de Indias, perteneciente al departamento Bolívar en Colombia.

De acuerdo con los objetivos del presente proyecto, y en aras de su cumplimiento, se llevó a cabo la presente investigación que es de tipo descriptiva, puesto que se requirió la indagación de información del área de estudio, la cual fue obtenida de acuerdo a los objetivos planteados y que fue analizada de forma posterior. Según la finalidad, la presente es considerada una investigación de tipo aplicada, debido a que poder desarrollar sus actividades fue necesario adquirir un conocimiento que se aplicó a lo largo de toda la investigación. De acuerdo con la fuente de datos de la cual fue obtenida la información requerida, es clasificada como una investigación de campo, puesto que se basa en la obtención de conocimiento mediante la aplicación de entrevistas a distintos talleres de mantenimiento de vehículos.

3.1.2. Población y muestra

La población utilizada para la presente investigación son los talleres de mantenimiento de vehículos de la ciudad de Cartagena, Bolívar en Colombia, a partir de la cual, mediante un muestreo no probabilístico por juicio se seleccionó una muestra de 5 talleres de mantenimiento que cumplen las siguientes características: Ser un taller autorizado por un distribuidor oficial, estar constituido legalmente ante el gobierno

colombiano, tener técnicos de mantenimiento certificados, no contar con un sistema de gestión de mantenimientos.

3.1.3. Instrumentos de evaluación

Como instrumento de evaluación principal se utilizó la entrevista, para la cual se diseñó un cuestionario con diferentes preguntas acerca de los procesos de los talleres desde que se ingresan los vehículos hasta su salida, y por medio de la cual se obtuvo información valiosa para el correcto desarrollo de la presente investigación.

Esta entrevista fue realizada a los jefes de mantenimiento de los talleres seleccionados para la presente investigación, con la participación eventual de diferentes técnicos de mantenimiento.

La aplicación de la entrevista como instrumento de evaluación es necesaria, ya que como lo dice Díaz et al. (2013), este instrumento permite recabar datos, pero debido a su flexibilidad permite obtener información más profunda, detallada, que incluso el entrevistado y entrevistador no tenían identificada, ya que se adapta al contexto y a las características del entrevistado.

A continuación, se presentan las preguntas contenidas en el formulario diseñado:

1. ¿El taller cuenta con algún mecanismo manual o digital para tener una gestión, monitoreo y manejo de históricos de los mantenimientos que se realizan a los vehículos?
2. ¿El taller cuenta actualmente con un *software* que le permita llevar manera sistematizada la gestión de los mantenimientos y/o de los procesos del taller?
3. En caso de utilizar una solución *software*, ¿cuál o cuáles se utilizan actualmente?
4. ¿Podría describir el(los) proceso(s) que se realiza(n) para darle ingreso a un vehículo?

5. ¿Podría por favor describir el proceso que se realiza y mencionar las variables que son consideradas al momento de asignar el personal técnico para realizar los mantenimientos de los vehículos?
6. ¿Qué procesos ocurren desde que el vehículo ingresa al taller hasta que sale?
7. ¿Antes de realizar un mantenimiento a un vehículo se revisa el historial de mantenimientos de este?
8. ¿Podría describir el(los) proceso(s) que se realiza(n) para darle salida a un vehículo?
9. ¿Cómo se realiza la comunicación con el cliente durante el tiempo en el que su vehículo se encuentra en el taller?
10. ¿El cliente tiene información 100% actualizada en todo momento sobre el estado del mantenimiento de su vehículo?
11. Mientras un vehículo se encuentra en mantenimiento, si el cliente desea conocer en qué estado o fase del mantenimiento se encuentra su vehículo, ¿cómo podría hacerlo?
12. Al momento de realizar el diagnóstico de un vehículo para su reparación, ¿se deja alguna constancia de este diagnóstico?
13. En caso de dejar constancia del diagnóstico, ¿se adjunta evidencia como fotografías o videos?
14. En caso de que se dejen plasmadas evidencias del diagnóstico, ¿cómo se le hacen llegar al cliente?
15. ¿Actualmente se maneja un historial de los mantenimientos realizados a cada vehículo con sus respectivos diagnósticos y evidencias?
16. En caso de manejar un historial de los mantenimientos de cada vehículo, ¿qué información se registra en cada mantenimiento?

17. En caso de manejar historial, cuando un vehículo que ya había recibido mantenimiento en el taller vuelve a ser ingresado, ¿se revisa el historial del vehículo?
18. En caso de manejar el historial de mantenimientos por cada vehículo, ¿en qué casos se revisan dichos historiales?
19. ¿Considera que un sistema de software sería de utilidad para gestionar, monitorear y llevar un histórico de los mantenimientos que se realizan dentro del taller?
20. En caso de que mientras se realiza un mantenimiento al vehículo hay que hacerse reparaciones o cambios de piezas que no se contemplaron inicialmente, ¿qué procedimiento se realiza?
21. Si al final de un mantenimiento se le da salida a un vehículo, pero por alguna razón quedan puntos pendientes por corregir, ¿dichos pendientes quedan registrados para ser revisados la próxima vez que ingrese el vehículo? ¿Dónde se registran estos pendientes?
22. En el caso de que queden puntos pendientes al darle salida a un vehículo, ¿el cliente tiene la opción de realizar un agendamiento para resolver los puntos pendientes en un próximo mantenimiento?
23. ¿Cuántas hojas de papel se utilizan por cada mantenimiento que se realiza en el taller?
24. ¿Actualmente cuánto espacio físico se utiliza para almacenar la documentación relacionada a cada vehículo? (Facturas, historial de mantenimientos, diagnósticos, etc.).
25. ¿Aproximadamente cuánto tiempo se demora actualmente realizar el proceso de ingreso de un vehículo en cuanto al registro de la información?

26. ¿Aproximadamente cuánto tiempo se demora actualmente realizar el proceso de salida de un vehículo en cuanto al registro de la información?

3.1.4. Análisis de la información

Teniendo en cuenta que la información que se recolectó durante el desarrollo de la presente investigación son datos de carácter cualitativo, se tomó la decisión de realizar el procesamiento y el análisis de dichos datos siguiendo una serie de pasos para analizar datos cualitativos que propone Fernández (2006), cuya implementación se describe a continuación:

3.1.4.1. Obtener información

Este paso fue realizado en dos etapas, la primera de ellas fue llevada a cabo a través de la realización de entrevistas a diferentes talleres de mantenimiento de vehículos de la ciudad de Cartagena, las cuales fueron estructuradas utilizando como guía el instrumento de evaluación descrito en el apartado anterior, dando así cumplimiento al primer objetivo específico de la investigación.

La segunda etapa, la cual ayudó al cumplimiento del tercer objetivo específico, consistió recibir una retroalimentación por parte de los talleres al presentarles la especificación de requisitos obtenida en la primera etapa del tercer paso.

3.1.4.2. Capturar, transcribir y ordenar la información

Al igual que el paso anterior, este paso fue llevado a cabo en dos etapas, la primera de las cuales fue realizada a través de plasmar las respuestas de cada pregunta

realizada durante las entrevistas realizadas en los talleres. Estas respuestas se pueden encontrar en el documento anexo “Respuestas de entrevistas.docx”.

Para llevar a cabo la segunda etapa de este paso, se tomaron notas de la retroalimentación brindada por los talleres al conocer la especificación de requisitos desarrollada en la primera etapa del paso 3.

3.1.4.3. Codificar la información

Tal como lo indica Fernández (2006), este proceso consiste en agrupar la información en categorías en las que se concentran los pasos o fases de un proceso, las ideas, conceptos o temas similares descubiertos por el investigador.

Con la finalidad de cumplir con este paso, y a la vez con el segundo objetivo específico de este trabajo investigativo, la información obtenida del paso anterior fue agrupada y organizada en una especificación de requisitos de *software*, la cual fue desarrollada siguiendo el estándar para la especificación de requisitos de *software* IEEE830.

Como segunda instancia se presentó la especificación de requisitos inicial a los mismos talleres que fueron entrevistados, y luego de recibir una retroalimentación por parte de estos, y con la finalidad de cumplir con el cuarto objetivo específico de la presente investigación, se realizó el refinamiento de dicha especificación de requisitos obteniendo como resultado una especificación de requisitos definitiva.

Esta especificación de requisitos se puede encontrar en el documento adjunto “Especificación de requisitos de software IEE830.docx”.

3.1.4.4. Integrar la información

En este paso se utilizó como insumo la especificación de requisitos desarrollada en el paso anterior, a partir de la cual se realizó la elaboración y especificación de los modelos y diagramas UML, que representa la arquitectura del *software* prometida como resultado del presente proyecto de investigación.

3.2.Resultados

En el presente apartado son presentados los resultados que fueron obtenidos por desarrollar la presente investigación. Estos resultados son presentados a continuación con base tanto en el objetivo general como en los objetivos específicos presentados en este documento, así como la metodología que se utilizó para lograr el cumplimiento de cada uno de dichos objetivos.

Como resultado principal de la presente investigación, se obtuvo la arquitectura de un *software* que le permite a los talleres de mantenimiento de vehículos tener un control sobre la gestión, el monitoreo y el manejo de históricos de los mantenimientos que se realizan a los vehículos que ingresan al taller.

En este capítulo se presenta la especificación de requisitos que se obtuvo como resultado del levantamiento de información, seguido de las modificaciones realizadas a la especificación de requisitos como producto de la retroalimentación de los talleres. Luego, con el fin de cumplir con el objetivo principal del presente trabajo de investigación, se expone la arquitectura del *software* que fue diseñada a partir de la especificación de requisitos utilizando como referencia el modelo de vistas 4+1, que comprende todos los diagramas mediante los que se representan tanto la arquitectura como los comportamientos del sistema *software*, y que se dividen y agrupan en 5 diferentes vistas. Por último, se mencionan los aspectos que diferencian el sistema diseñado en este trabajo frente a los sistemas mencionados en el estado del arte.

3.2.1. Especificación de requisitos

Teniendo en cuenta que la documentación para el proceso de ingeniería de *software* es un factor necesario para garantizar el mantenimiento y la calidad del sistema (Méndez, 2008), en esta fase se realizó la especificación de los requisitos del sistema a partir de analizar la información que se obtuvo de las entrevistas a los talleres haciendo uso del formato del estándar IEEE 830, facilita realizar la recolección de la información y brinda recomendaciones en relación al diseño de software. Este formato resultó útil para el investigador porque le permitió documentas de una manera adecuada cada uno de los requisitos funcionales del sistema diseñado. La especificación de requisitos se puede encontrar en el documento anexo "Especificación de requisitos de software IEE830.docx"

A continuación, en las tablas de la 1 a la 21, se presenta la lista de los requisitos, obtenida como resultado de esta fase del proyecto:

Tabla 1.

Requisito funcional 1

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito
RF1	Registro de usuarios	Requisito, restricción	El sistema debe permitir el registro de usuarios y definirle un rol entre los siguientes: - Jefe de taller - Mecánico - Cliente Para registrar el usuario el sistema debe solicitar la siguiente información: - *Nombres y apellidos completos - *Tipo de documento de identidad - *Número de documento de identidad	Alta/Esencial

			- *Contraseña - *Verificación de contraseña - *Teléfono de contacto - *Correo electrónico Los campos marcados con * son obligatorios	

Nota. Elaboración propia.

Tabla 2.

Requisito funcional 2

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito
RF2	Registro de tipos de vehículo	Requisito	El sistema debe permitirle al jefe de taller registrar los tipos de vehículos aceptados para mantenimiento, por ejemplo: Motocicleta Vehículo sedán Camioneta Etc. Para registrar un tipo de vehículo el sistema debe solicitar la siguiente información: Nombre del tipo de vehículo Descripción del tipo de vehículo.	Alta/Esencial

Nota. Elaboración propia.

Tabla 3.

Requisito funcional 3

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito

RF3	Estados de los vehículos	Requisito	El sistema debe tener almacenados los siguientes estados para los vehículos: - Por fuera del taller - Ingresado al taller - Diagnosticado - En mantenimiento - Mantenimiento en pausa - Mantenimiento finalizado	Alta/Esencial

Nota. Elaboración propia.

Tabla 4.

Requisito funcional 4

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito
RF4	Medios de envíos de notificaciones	Requisito, Restricción	El sistema debe permitir enviar notificaciones a los usuarios por medio de los siguientes canales: - Correo electrónico - WhatsApp - Mensaje de texto	Alta/Esencial

Nota. Elaboración propia.

Tabla 5.

Requisito funcional 5

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito
RF5	Establecer/Modificar las preferencias del usuario	Requisito	El sistema debe permitir a los usuarios establecer las siguientes preferencias: - Seleccionar el medio de envío de notificaciones (por defecto debe ser mensaje de texto). El usuario debe cumplir con los requisitos necesarios para el medio de envío seleccionado, por ejemplo, para el correo electrónico, el usuario debe tener registrado un correo. Si el usuario no cumple con los requisitos para un medio de envío, el sistema deberá mostrar la opción desactivada indicando con un mensaje que debe registrar la información necesaria para dicho medio de envío de notificaciones.	Alta/ Esencial

Nota. Elaboración propia.

Tabla 6.

Requisito funcional 6

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito

RF6	El sistema debe permitir el registro de los vehículos	Requisito	Para registrar un vehículo el sistema debe solicitar: - *Cliente propietario del vehículo - *Tipo de vehículo - *Número de placa/matrícula - Número de chasis - Número de motor - Fecha de matrícula - Color - Cilindraje - Marca - Modelo - Línea - Tipo de combustible - Cantidad de pasajeros - Organismo de tránsito donde se registró el vehículo Nota: Los ítems marcados con * son requeridos	Alta/Esencial

Nota. Elaboración propia.

Tabla 7.

Requisito funcional 7

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito

RF7	Agendamiento de cita para mantenimiento	Requisito	El sistema debe permitir agendar una cita para realizar el mantenimiento de un vehículo. Para esto el software debe solicitar la siguiente información: - Número de placa/matrícula del vehículo - Fecha y hora de la cita El vehículo debe estar registrado en el sistema. El sistema deberá mostrarle al usuario las fechas y horas disponibles y permitir agendar la cita para la fecha y hora escogidas.	Alta/Esencial

Nota. Elaboración propia.

Tabla 8.

Requisito funcional 8

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito

RF8	Ingreso de vehículo para mantenimiento	Requisito	El vehículo que se desea ingresar debe estar registrado en el sistema. El sistema le debe permitir al jefe de taller registrar el ingreso de los vehículos para su mantenimiento. Para esto el sistema debe solicitar: - Número de placa/matrícula del vehículo Si el vehículo tiene una cita programada para el día en el que se realiza el ingreso, el sistema realiza deberá: - Crear el registro de ingreso del vehículo. - Colocar la fecha y hora del momento en que se registra el ingreso de forma automática. - Cambiar el estado del vehículo a "Ingresado al taller" - Notificar al cliente el ingreso de su vehículo al taller por los medios configurados. En caso de no tener una cita programada, el sistema lo indicará a través de la pantalla, y permitirá la opción de ingresar el vehículo sin cita, en tal caso, el sistema realizará el procedimiento descrito anteriormente.	Alta/Esencial

Nota. Elaboración propia.

Tabla 9.

Requisito funcional 9

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito
RF9	Asignación de técnico al vehículo	Requisito	Una vez ingresado un vehículo al taller, el sistema debe permitir al jefe de taller asignar un mecánico para que realice el mantenimiento del vehículo. Esta asignación se podrá realizar de la siguientes formas: - El sistema elige un técnico que no tenga vehículos asignados en el momento. - El jefe de taller puede aceptar la sugerencia del sistema, o indicar un técnico en específico. - El jefe de taller decidirá en ese momento si el sistema debe enviar una notificación al técnico asignado por los medios configurados para el técnico.	Alta/Esencial

Nota. Elaboración propia.

Tabla 10.

Requisito funcional 10

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito

RF10	Cambio de estado por inicio del mantenimiento	Requisito	El sistema le debe permitir al técnico, cambiar el estado del vehículo de "Diagnosticado" a "En mantenimiento", indicando una breve descripción opcional. En el registro del cambio de estado se registra de forma automática la fecha y hora en la que este se realiza. Para realizar este cambio, el mecánico podrá buscar el vehículo por el número de placa/matrícula y seleccionar la opción de cambio de estado a “En mantenimiento”. El cliente propietario del vehículo es notificado (o no) según la configuración de las preferencias.	Alta/Esencial

Nota. Elaboración propia.

Tabla 11.

Requisito funcional 11

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito
RF11	Cambio de estado por pausa del mantenimiento	Requisito	El sistema le debe permitir al técnico, cambiar el estado del vehículo de "En mantenimiento" a "mantenimiento en pausa" indicando la razón del cambio de estado en formato de texto plano. En el registro del cambio de estado se registra de forma automática la fecha y hora en la que este se realiza. Para realizar este cambio, el mecánico podrá buscar el vehículo por el número de placa/matrícula y seleccionar la opción de cambio de estado a "mantenimiento en pausa". El cliente propietario del vehículo es notificado (o no)	Alta/Esencial

			según la configuración de las preferencias.	

Nota. Elaboración propia.

Tabla 12.

Requisito funcional 12

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito
RF12	Cambio de estado por fin del mantenimiento	Requisito	El sistema le debe permitir al técnico, cambiar el estado del vehículo de "Mantenimiento en pausa" a "Mantenimiento finalizado" indicando un mensaje opcional. En el registro del cambio de estado se registra de forma automática la fecha y hora en la que este se realiza. Para realizar este cambio, el mecánico podrá buscar el vehículo por el número de placa/matrícula y seleccionar la opción de cambio de estado a "Mantenimiento finalizado". El cliente propietario del vehículo es notificado (o no)	Alta/Esencial

	según la configuración de las preferencias.

Nota. Elaboración propia.

Tabla 13.

Requisito funcional 13

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito

RF13	Registrar la salida/entrega del vehículo	Requisito	El sistema debe permitir al jefe de taller registrar la entrega del vehículo al cliente. Para esto, el jefe de taller debe ingresar el número de placa/matrícula del vehículo para buscarlo en el sistema. Si la placa/matrícula no corresponde con un vehículo registrado, el sistema debe indicar que el vehículo no se encuentra registrado. En el caso contrario, el sistema debe mostrar la información del vehículo y brindarle al jefe de taller la opción para registrar la salida del vehículo. En caso de que el jefe de taller indique la opción de registrar la salida del vehículo, el sistema debe: - Generar un registro de salida con la fecha y hora correspondiente - Quitar la asignación del técnico (pero quedando registrada en el historial). - Cambiar el estado del vehículo a “Por fuera del taller". El cliente propietario del vehículo es notificado (o no) según la configuración de las preferencias.	Alta/Esencial

Nota. Elaboración propia.

Tabla 14.

Requisito funcional 14

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito

RF14	Registro del diagnóstico del vehículo	Requisito	El vehículo debe estar en estado “Ingresado al taller”. El sistema debe permitirle al técnico asignado al vehículo para su mantenimiento, registrar el diagnóstico del vehículo a través de una descripción en texto plano, y brindarle la posibilidad de adjuntar fotografías y/o videos como evidencia del diagnóstico. Este diagnóstico será adjunto al registro del mantenimiento abierto para el vehículo que se crea al momento de ingresar al taller. Una vez registrado el diagnóstico el sistema cambiará de forma automática el estado del vehículo a “Diagnosticado”. El cliente propietario del vehículo es notificado (o no) según la configuración de las preferencias.	Alta/Esencial

Nota. Elaboración propia.

Tabla 15.

Requisito funcional 15

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito

			El sistema debe permitir visualizar un historial de mantenimientos de un vehículo específico, mostrando de forma cronológica (desde la más reciente a la más antigua) la información de todos los mantenimientos que se le han realizado dentro del taller.	
RF15	Visualización del historial de servicios	Requisito	Lo siguientes usuarios tendrán permisos para ver el historial de mantenimientos de un vehículo. El propietario del vehículo, siempre que lo requiera. El jefe de taller, siempre que lo requiera. Los técnicos, siempre que lo requieran mientras tengan el vehículo asignado.	Alta/Esencial

Nota. Elaboración propia.

Tabla 16.

Requisito funcional 16

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito
RF16	Autenticación de usuarios	Requisito	El sistema debe permitir el inicio de sesión de los usuarios que se encuentren registrados. Para esto el sistema debe solicitar la siguiente información: - Correo electrónico - Contraseña	Alta/Esencial

Nota. Elaboración propia.

Tabla 17.

Requisito funcional 17

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito

RF17	Visualización de la información	Requisito	Cuando un usuario se identifica de manera exitosa ante el sistema, este debe ser capaz de brindarle al usuario toda la información que este tenga permisos de visualizar. Si el usuario autenticado es un cliente, el sistema le deberá proporcionar la siguiente información: - Información de todos los vehículos del cliente que hayan sido registrados en el sistema - La información personal del usuario - El histórico de mantenimientos de cada uno de sus vehículos - El estado en que se encuentra cada uno de sus vehículos En caso de que el usuario sea un mecánico, el sistema le permitirá ver los vehículos que tenga asignados para mantenimiento con toda su respectiva información e historial de mantenimientos. En caso de que el usuario autenticado sea jefe de taller, el sistema le debe permitir visualizar toda la información de todos los vehículos registrados en el sistema con sus respectivos históricos y toda la información de todos los clientes registrados en el sistema.	Alta/Esencial

Nota. Elaboración propia.

Tabla 18.

Requisito funcional 18

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito

RF18	Modificación y cancelación de citas	Requisito	El sistema debe permitir a los clientes modificar la fecha y hora de las citas programadas para sus vehículos, o en su defecto cancelarlas. El jefe de taller podrá modificar o cancelar las citas programadas de cualquier vehículo. Una vez que se realice un cambio en la cita de un vehículo o se cancele una cita, el sistema notificará al cliente del vehículo acerca del cambio o cancelación de esta por el medio configurado en las preferencias del usuario.	Media/Moderada

Nota. Elaboración propia.

3.2.2. Modificaciones a la especificación de requisitos

En esta etapa se presentan los requisitos que fueron cambiados de acuerdo con la retroalimentación de los talleres.

Como se puede observar en la siguiente tabla, en el RF1 se decidió colocar el correo electrónico como un campo opcional, ya que es posible que algunos clientes no cuenten con correo electrónico:

Tabla 19.

Requisito funcional 1 modificado

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito

RF1	Registro de usuarios	Requisito, restricción	El sistema debe permitir el registro de usuarios y definirle un rol entre los siguientes: - Jefe de taller - Mecánico - Cliente Para registrar el usuario el sistema debe solicitar la siguiente información: - *Nombres y apellidos completos - *Tipo de documento de identidad - *Número de documento de identidad - *Contraseña - *Verificación de contraseña - *Teléfono de contacto - Correo electrónico Los campos marcados con * son obligatorios	Alta/Esencial

Nota. Elaboración propia.

Debido a la modificación del RF1, fue necesario modificar el RF16, para que solicite el tipo y número de documento de identidad en lugar del correo electrónico (ver tabla 20):

Tabla 20.

Requisito funcional 16 modificado

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito
RF16	Autenticación de usuarios	Requisito	El sistema debe permitir el inicio de sesión de los usuarios que se encuentren registrados. Para esto el sistema debe solicitar la siguiente información: - Tipo de documento de identidad - Número de documento de identidad - Contraseña	Alta/Esencial

Nota. Elaboración propia.

Al agendar una cita para un vehículo, se desea que el cliente sea notificado, por lo que se modificó el RF7 quedando de la siguiente manera (ver tabla 21):

Tabla 21.

Requisito funcional 7 modificado

Número del requisito	Nombre del requisito	Tipo	Fuente del requisito	Prioridad del requisito
RF7	Agendamiento de cita para mantenimiento	Requisito	El sistema debe permitir agendar una cita para realizar el mantenimiento de un vehículo específico. Para esto el software debe solicitar la siguiente información: - Número de placa/matrícula del vehículo - Fecha y hora de la cita Para este requisito es necesario que el vehículo para el que se solicita la cita se encuentre registrado en el sistema. Si la placa no concuerda con un vehículo registrado, el sistema lo informará al usuario, en caso contrario, el sistema sugerirá la fecha y hora más cercana disponible, y el usuario debe decidir si aceptar la cita para dicha fecha y hora o indicar una diferente. El sistema deberá mostrarle al usuario las fechas y horas disponibles para que seleccione una. Una vez registrada la cita, el sistema notificará al cliente recibirá una notificación a través de medios configurados en sus preferencias.	Alta/Esencial

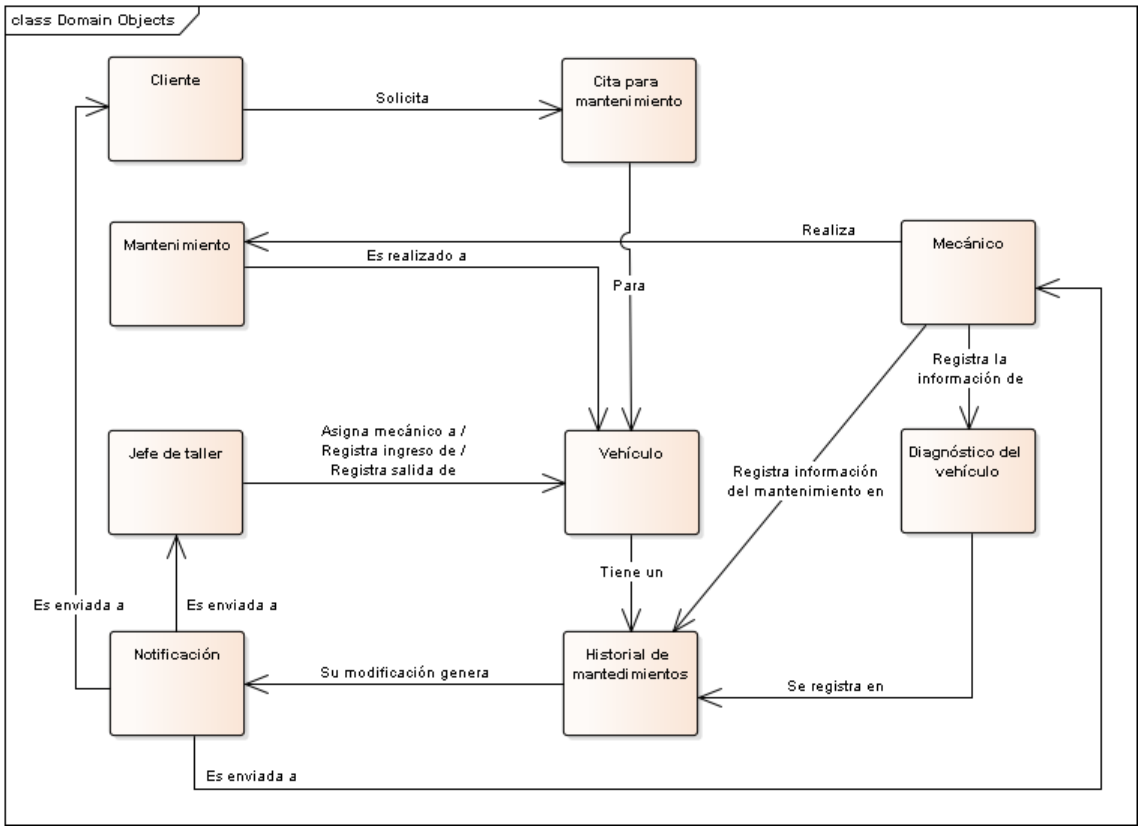
Nota. Elaboración propia.

3.2.3. Diseño de la arquitectura del sistema

La arquitectura del *software* fue diseñada a partir de los conceptos del mundo real aplicables al problema a resolver y obtenidos durante la labor de campo, creando en primera instancia el modelo de dominio del sistema que se ilustra a continuación (ver figura 1):

Figura 1.

Modelo d dominio del sistema



Nota. Elaboración propia.

Cada uno de los artefactos del diseño de la arquitectura se crearon a partir de la especificación de los requisitos de *software* y teniendo como base la propuesta de Kruchten en la que es planteado en modelo vistas 4+1 (Kruchten, 1995). Este modelo sugiere que los artefactos sean agrupados en las siguientes 5 categorías:

- **Vista de procesos:** Esta vista cubre los siguientes aspectos:
    - Concurrencia
    - Sincronización

- Distribución
- Integridad
- Tolerancia a fallos del sistema.

En esta vista se especifican los pasos y decisiones tomadas por el sistema en cada situación.

- **Vista lógica:** en esta vista se soportan los requerimientos funcionales. Aquí se descompone el sistema en diferentes entidades tomadas del dominio del problema, mencionando cada uno de sus atributos, indicando sus comportamientos y mencionando las relaciones entre cada una de las entidades
- **Vista de desarrollo:** esta vista se enfoca en representar el *software* como una agrupación de diferentes subsistemas que conforma un todo.
- **Vista física:** describe la forma en la que se distribuye el *software* dentro de la infraestructura de *hardware* y la comunicación con la misma con el fin de tener una instalación adecuada y un funcionamiento correcto del sistema. Aquí también se describe la relación entre los elementos de hardware necesarios para desplegar el sistema.
- **Vista de escenarios:** describe de manera abstracta los requisitos funcionales del sistema, que son el resultado de la interacción conjunta de las 4 vistas anteriores, brindando así una vista general de las funcionalidades, los actores y la relación entre ambos.

La etapa de diseño es una fase crucial en la creación de un sistema, ya que en esta se toman decisiones importantes de la solución que los afectan directamente, por lo que se procedió con el diseño del *software* tomando como base la información obtenida en la fase de análisis de requisitos de *software*, obteniendo como resultado todos los artefactos que conforman la arquitectura del sistema.

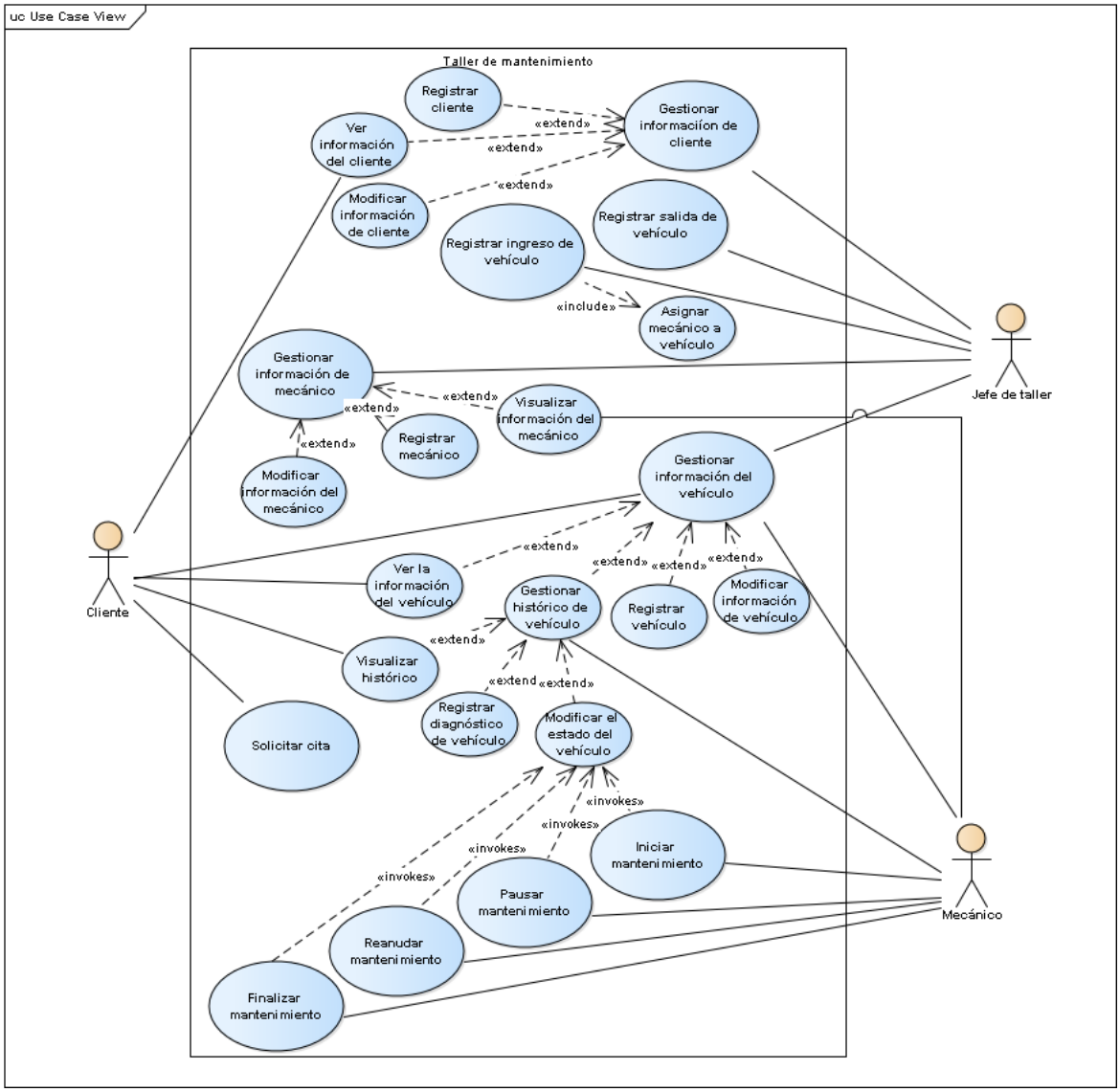
A continuación, se presentan los resultados obtenidos en la presente etapa bajo el modelo de vistas propuesto por Kruchten:

Vista de escenarios

Se identificaron los diferentes actores en la interacción con el sistema y cada una de las acciones que los involucran y se plasmó en el diagrama que se ilustra a continuación y que representa la vista de escenarios del sistema:

Figura 2.

Vista de escenarios



Nota. Elaboración propia.

En este modelo se pueden observar al cliente, al mecánico y al jefe de taller como los actores del sistema, cada uno de los cuales puede realizar diferentes acciones características de su rol. Los usuarios con rol de cliente pueden solicitar una cita para realizar el mantenimiento de su(s) vehículo(s) y pueden consultar la información relacionada a él y a su(s) vehículo(s) que se encuentre registrada en el sistema, incluyendo el histórico de mantenimientos. Por otro lado, el jefe de taller tiene la facultad de gestionar (involucra ingresar, visualizar y modificar) la información de los clientes, vehículos y mecánicos, también puede registrar tanto los ingresos como las salidas de los vehículos al taller y realizar la asignación de los técnicos a los vehículos para que estos realicen el mantenimiento. Los mecánicos, por su parte, se encuentran facultados para iniciar, pausar, reanudar y finalizar los mantenimientos de los vehículos que tienen asignados, así como registrar el diagnóstico de los mismos. Los mecánicos también pueden visualizar la información relacionada a los vehículos que tienen asignados incluyendo el historial de mantenimientos, el cual pueden modificar directamente al registrar el diagnóstico y cambiar el estado en el que se encuentra el vehículo durante el mantenimiento.

Vista de procesos

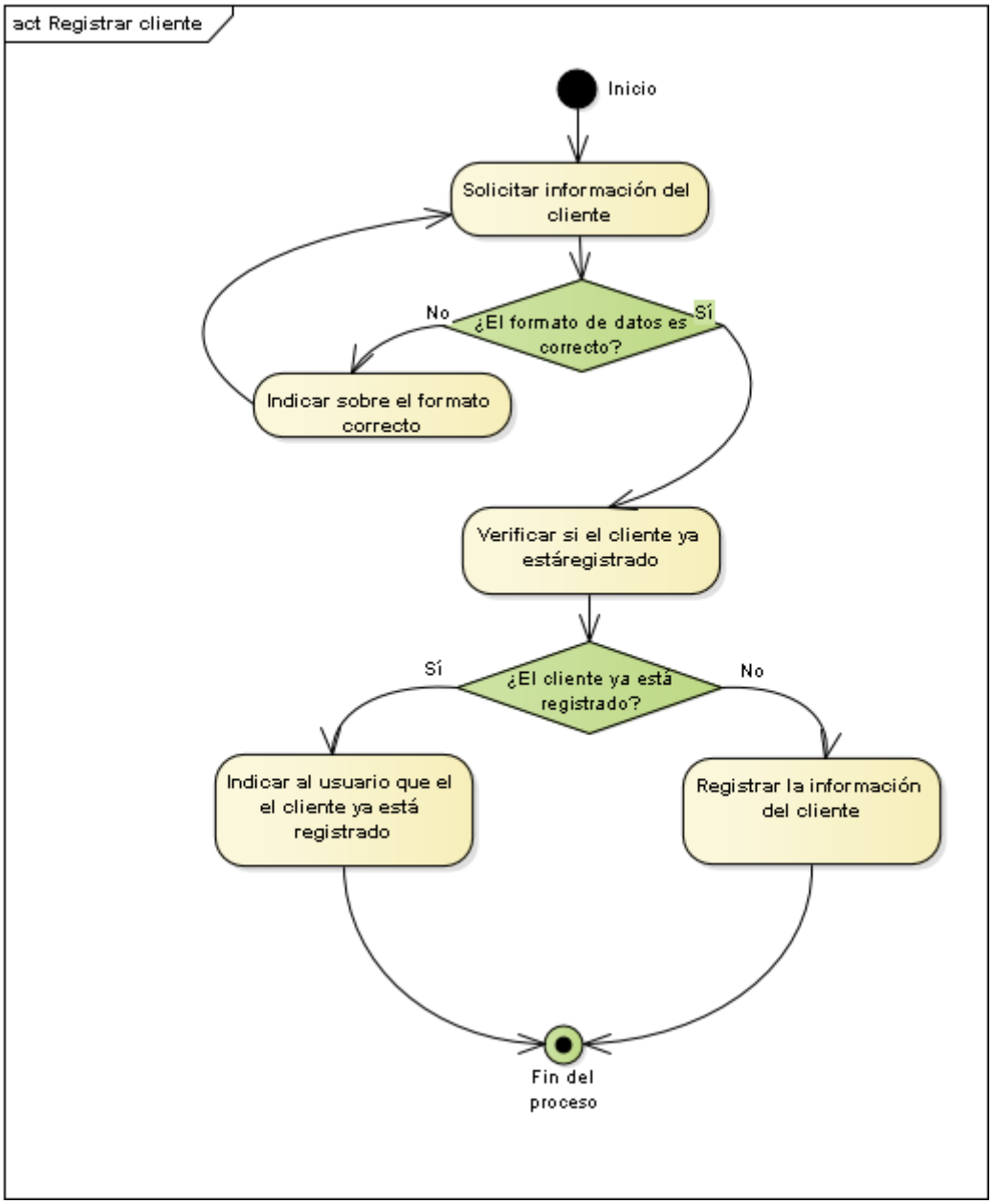
En esta fase se diseñaron diferentes diagramas de actividades que refleja el comportamiento del sistema en cada uno de sus casos de uso, con el fin de detallar el flujo de eventos que relacionado a cada uno de los procesos del sistema.
A continuación, seilustran y explican los diagramas de secuencia de cada uno de los procesos que se llevan a cabo en el sistema.

Como se observa en la figura 3, para registrar un cliente, el flujo deberá ser iniciado por el jefe de taller. Al iniciar el flujo, el sistema pedirá la información del cliente a registrar y validará que esta tenga un formato válido, de no ser así, el sistema le

indicará al usuario que el formato de los datos no es válido para que este vuelva a ingresar la información. En el caso de que los datos cumplan con el formato, el sistema procederá a verificar si el cliente ya se encuentra registrado en el sistema, y en caso de estar registrado, el sistema le indicará esto al usuario y el flujo terminará, de lo contrario, el sistema procederá con el registro del cliente.

Figura 3.

Diagrama de actividades registrar cliente

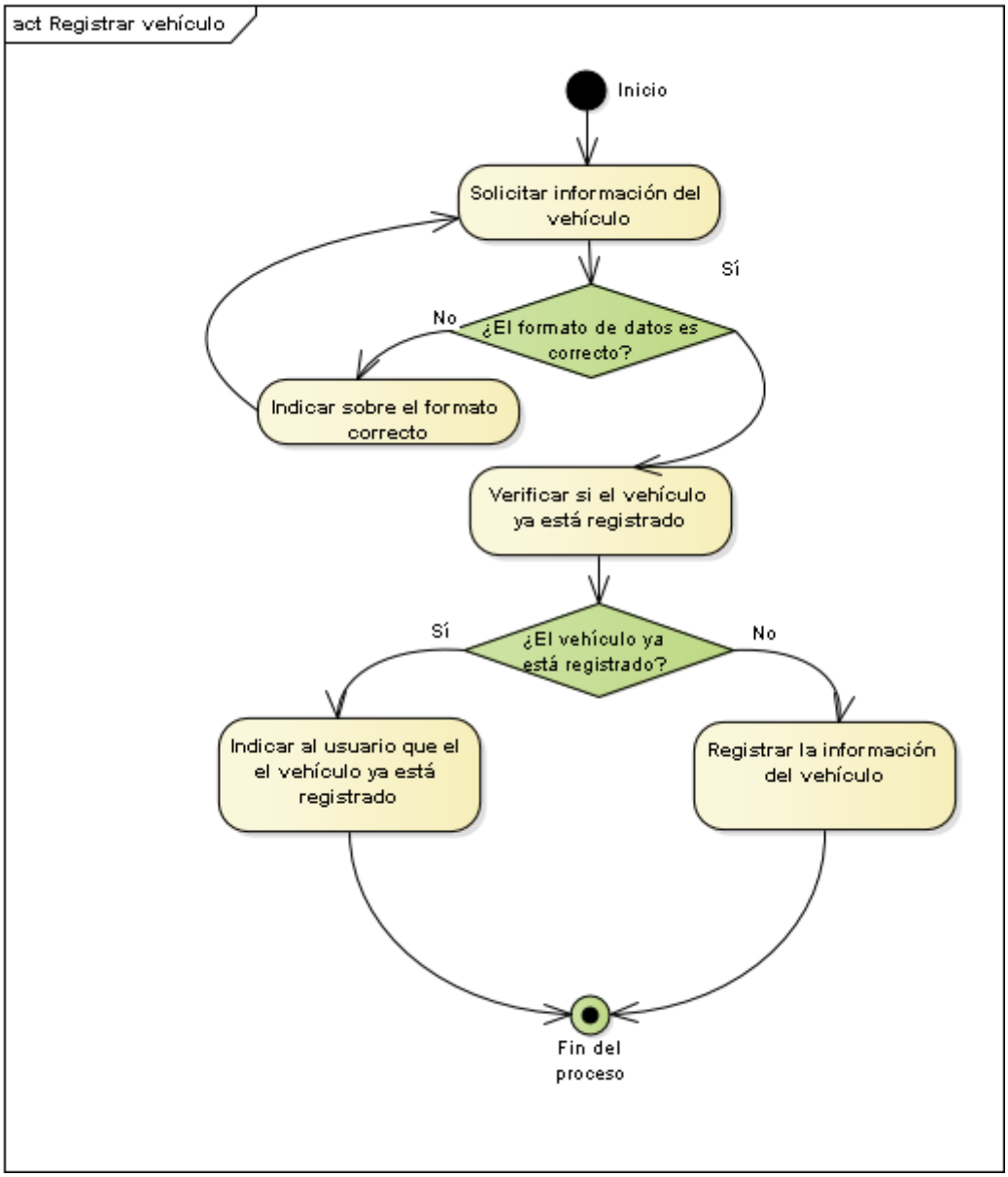


Nota. Elaboración propia.

El proceso para registrar un vehículo es bastante similar al proceso para registrar un cliente, pero con la variación de que los datos pedidos por el sistema corresponderán a los datos del vehículo, y que la validación de existencia se hace consultando sobre los vehículos existentes, tal como se ilustra en la figura 4.

Figura 4.

Diagrama de actividades registrar vehículo



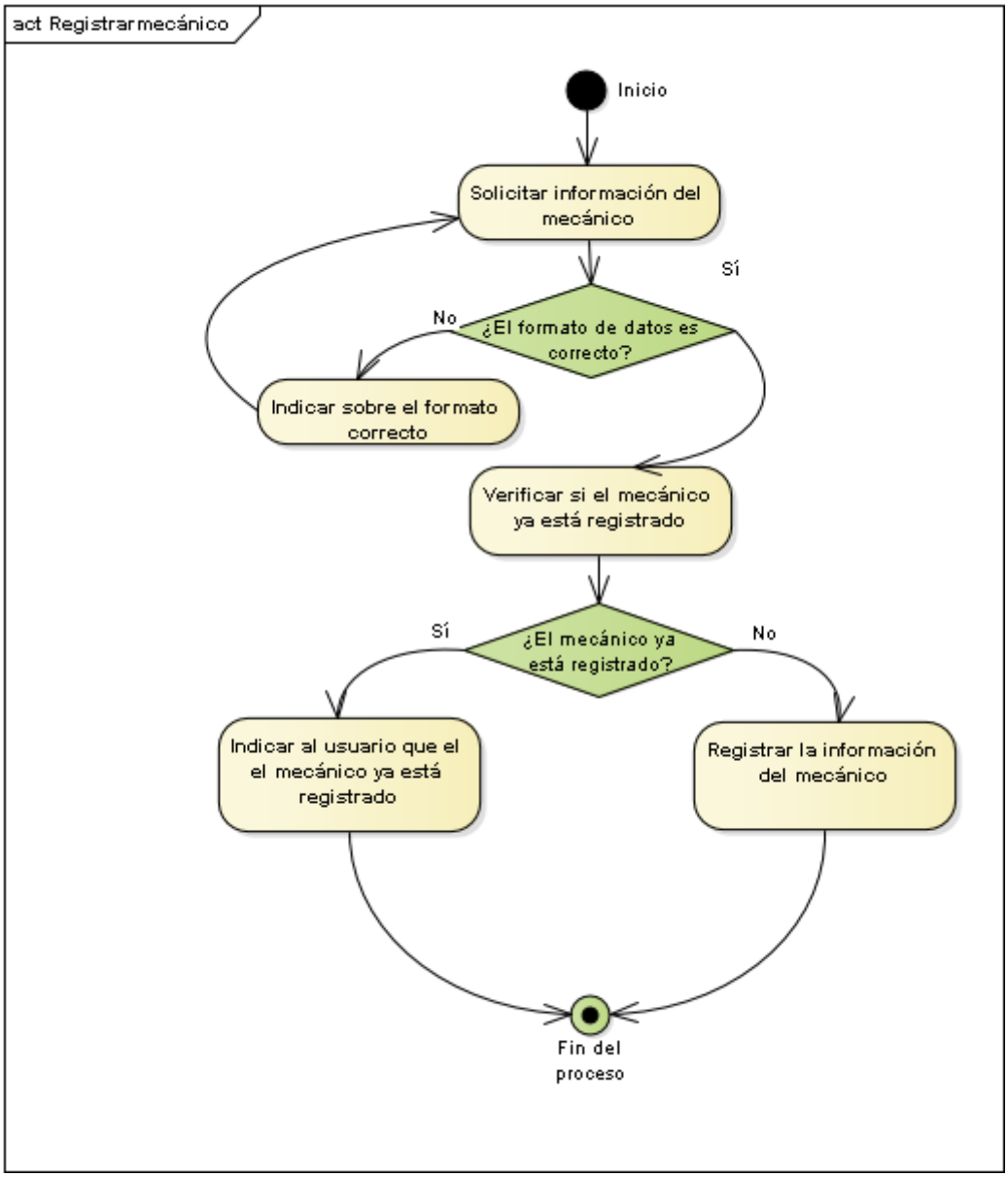
Nota. Elaboración propia.

En la figura 5 se ilustra el proceso para registrar un mecánico, que al igual que el proceso para registrar un cliente o un vehículo, este puede ser llevado a cabo por un jefe

de taller. El sistema solicitará los datos del mecánico, validará el formato de los datos, verificará si el mecánico ya está registrado y de no ser así, procederá con el registro. En el momento de ingresar los datos del mecánico, si estos no cuentan con el formato correcto el sistema se lo informará al usuario y este podrá volver a indicar los datos.

Figura 5.

Diagrama de actividades registrar mecánico



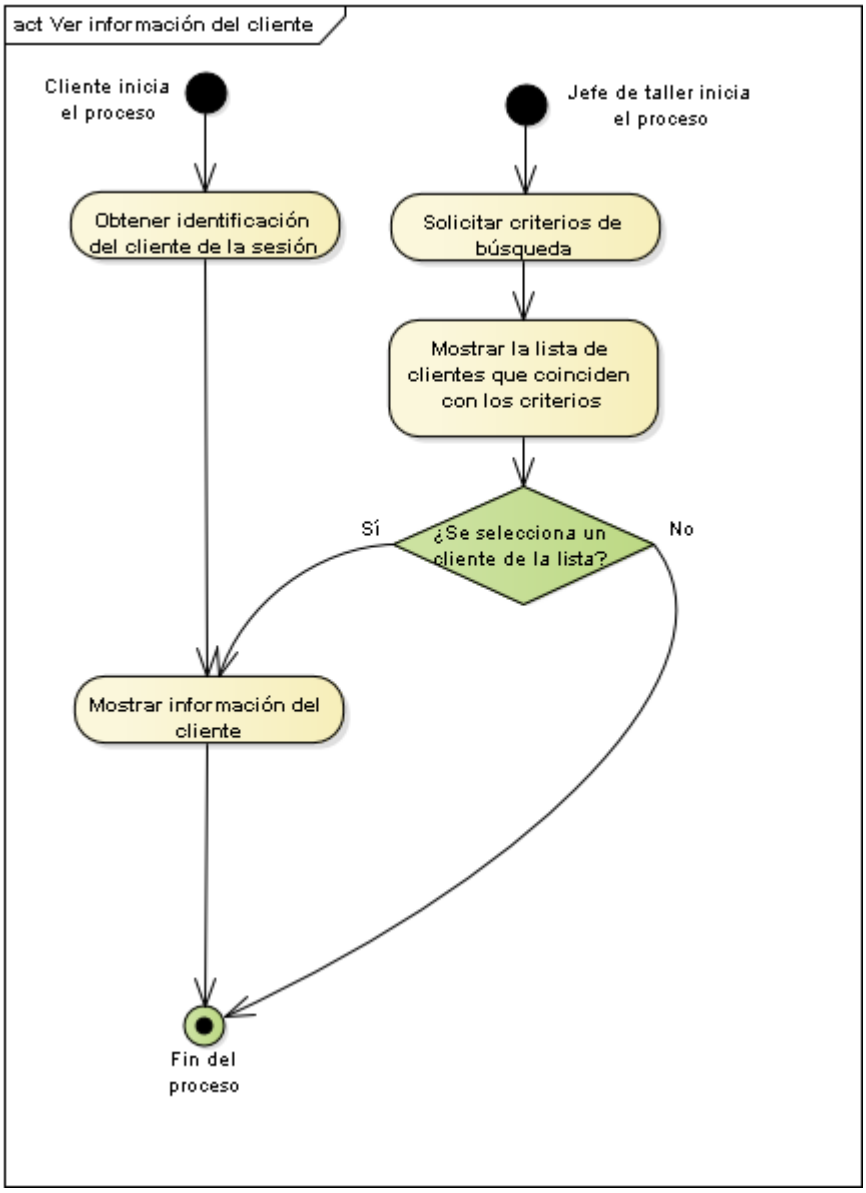
Nota. Elaboración propia.

Tal como se puede observar en la figura 6, el proceso “ver información del cliente” puede ser ejecutado tanto por el cliente, para ver su propia información, como

por el jefe de taller. En el caso de que el cliente inicie el flujo, se cargarán los datos del cliente de forma automática ya que para este flujo el cliente ha debido iniciar sesión en el sistema. Sin embargo, cuando el proceso sea iniciado por el jefe de taller, este deberá ingresar los criterios para buscar el cliente (nombre, apellido, número de identificación, teléfono, etc.), y aparecerá una lista con los clientes cuya información coincide con los criterios de búsqueda. Si el jefe de taller no selecciona ningún cliente el flujo se da por terminado, pero si selecciona un cliente de la lista, el sistema mostrará la información registrada para el cliente seleccionado y se finalizará el flujo.

Figura 6.

Diagrama de actividades ver información del cliente



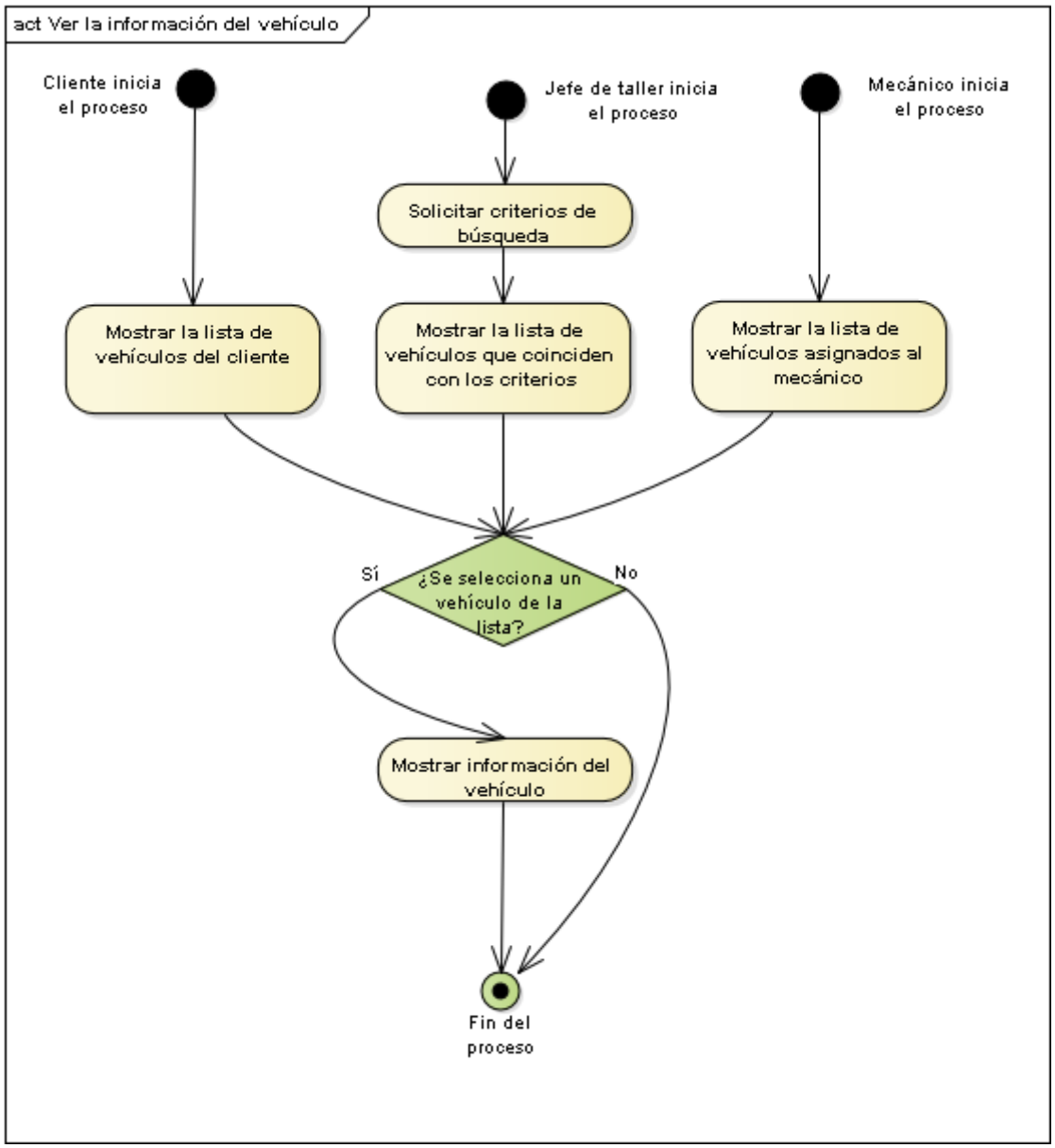
Nota. Elaboración propia.

El proceso para consultar la información de un vehículo es un poco distinto al proceso descrito anteriormente, ya que este puede ser iniciado tanto por el cliente, el mecánico o por el jefe de taller (ilustrado en la figura 7). Cuando el usuario sea el cliente, el proceso inicia mostrándole al usuario una lista con todos los vehículos que tiene, mientras que, si el usuario es el feje de taller, este debe ingresar unos criterios de búsqueda y el sistema le mostrará una lista de vehículos que cumplan con los criterios, y por otra parte, al mecánico se le mostrará una lista de todos los vehículos que tienen asignados. Si el usuario selecciona un vehículo de la lista, el sistema mostrará toda la

información del vehículo que se encuentre registrada, incluyendo una opción para visualizar el historial de mantenimientos del vehículo y se da por terminado el proceso, en el caso en que no se seleccione un vehículo de la lista, también se da por terminado el proceso.

Figura 7.

Diagrama de actividades ver información del vehículo



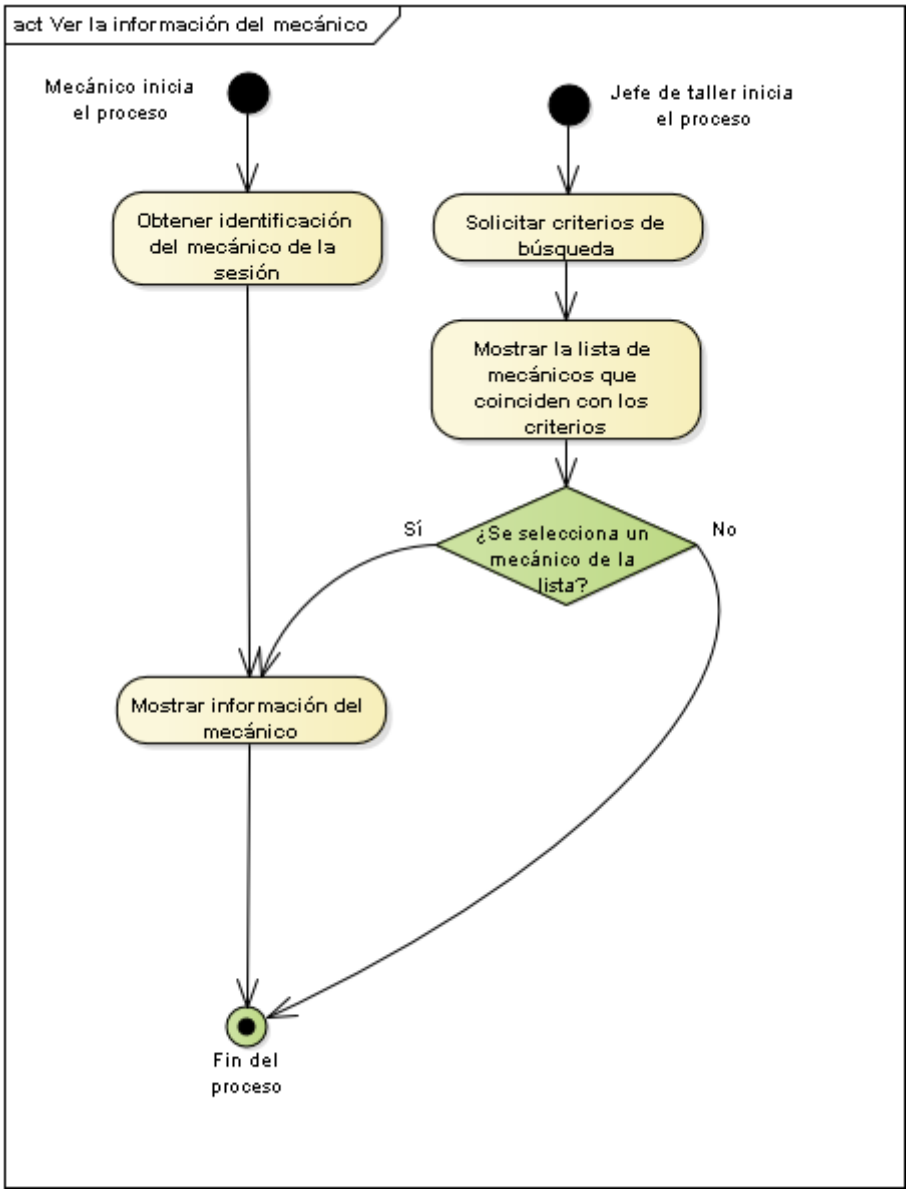
Nota. Elaboración propia.

Para el proceso que permite ver la información de un mecánico, si el usuario que está ejecutando el sistema es un mecánico, el sistema mostrará la información del

mecánico en cuestión, pero si el proceso es iniciado por un jefe de taller, este deberá primero consultar el mecánico a partir de unos criterios de búsqueda. Esta búsqueda dará como resultado una lista de mecánicos que cumplan con los criterios ingresados, y en caso de que se seleccione un mecánico de la lista el sistema mostrará la información de este y se finalizará el proceso. En caso de no seleccionar un mecánico de la lista se da por terminado el proceso (ver figura 8).

Figura 8.

Diagrama de actividades ver la información del mecánico

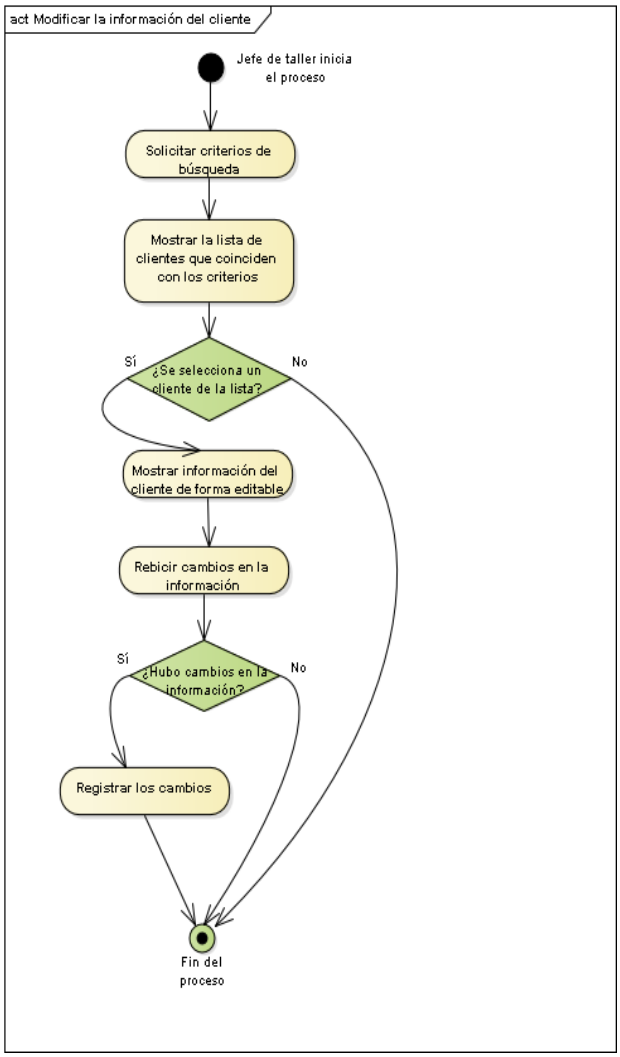


Nota. Elaboración propia.

Para modificar la información de un cliente, el usuario que se encuentre autenticado en el sistema debe ser un jefe de taller. Ese proceso inicia con jefe de taller ingresando los criterios de búsqueda para encontrar el cliente a modificar, luego, el sistema mostrará una lista de clientes cuya información coincida con los criterios de búsqueda, Al seleccionar un cliente de la lista, se debe mostrar la información del cliente en campos editables. Si el jefe de talle realiza cambios, estos se guardarán en el sistema cuando se seleccione la opción guardar y se finalizará el flujo. En caso contrario, no se registrarán los cambios y finalizará el flujo.

Figura 9.

Diagrama de actividades modificar la información del cliente

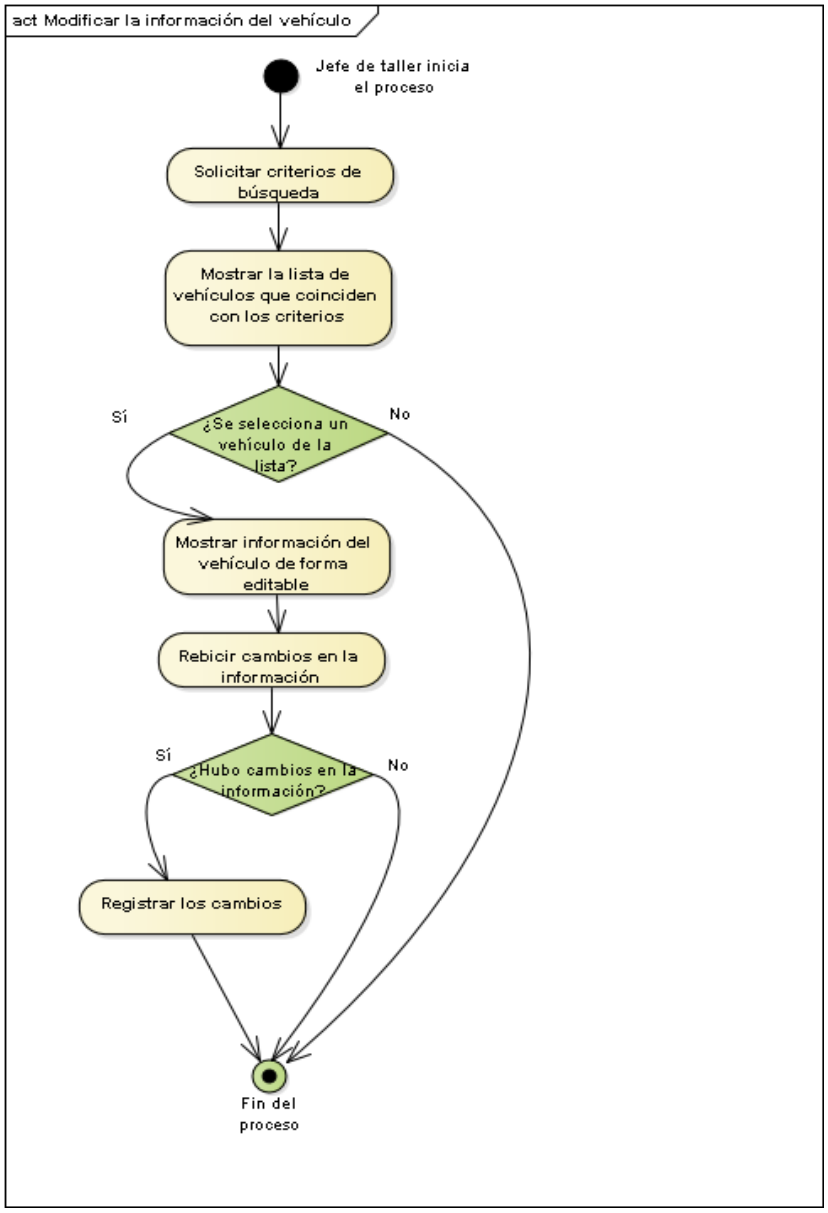


Nota. Elaboración propia.

Para llevar a cabo el proceso por medio del cual se puede modificar la información de un vehículo, es necesario que dicho proceso sea iniciado por el jefe de taller, a quién el sistema le solicitará los criterios de búsqueda para desplegar una lista con todos los vehículos que cumplan con los criterios, si el jefe de taller no selecciona un vehículo de la lista, el flujo se da por terminado, mientras que si se selecciona un vehículo el sistema mostrará su información de forma editable, y si el usuario (jefe de taller) realiza cambios sobre la misma se guardan los cambios y se finaliza el proceso, de lo contrario, se termina el proceso sin guardar las modificaciones.

Figura 10.

Diagrama de actividades modificar la información del vehículo

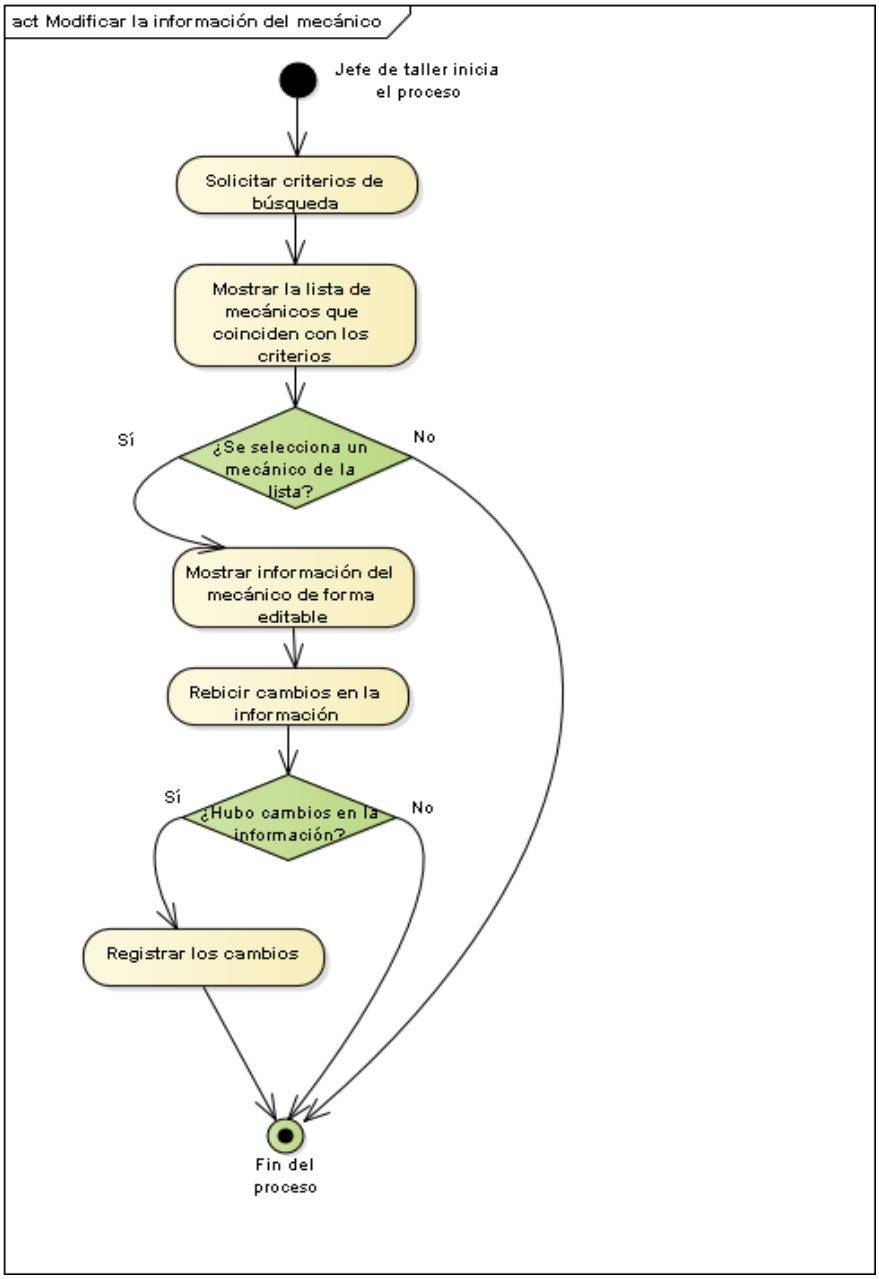


Nota. Elaboración propia.

Para modificar la información de un mecánico el proceso debe ser iniciado por el jefe de taller. En primera instancia el sistema solicitará los criterios de búsqueda y mostrará una lista de los mecánicos cuya información cumpla con los criterios ingresados, cuando se seleccione un mecánico de la lista, el sistema mostrará la información del mismo de forma editable, si el usuario realiza cambios en la información el sistema guardará los cambios y terminará el proceso, mientras que si no se realizan cambios en la información el proceso terminará sin guardar los cambios.

Figura 11.

Diagrama de actividades modificar la información del mecánico



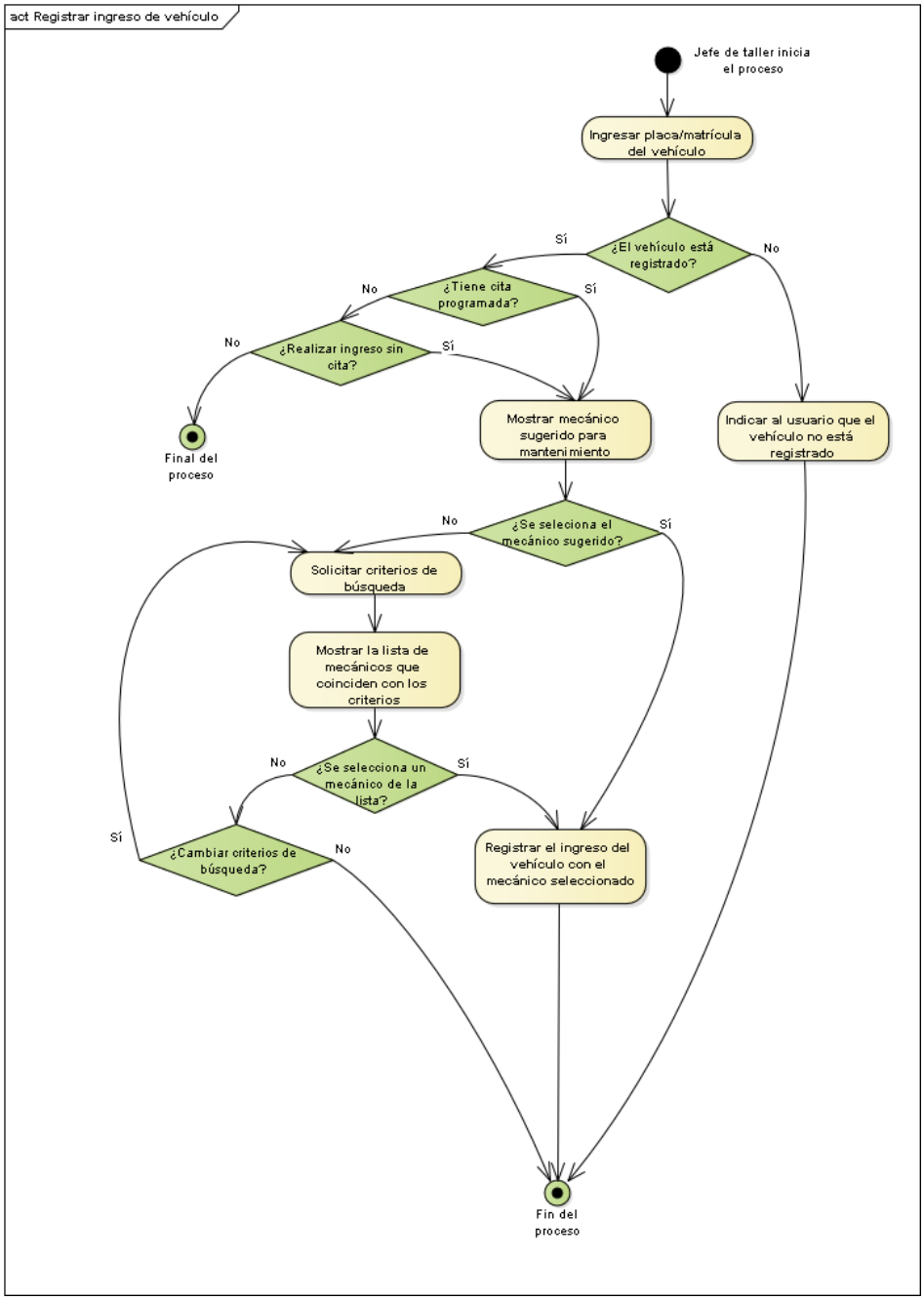
Nota. Elaboración propia.

El proceso para registrar el ingreso de un vehículo puede ser llevado a cabo por el jefe de taller ingresando el número de la placa o de la matrícula del vehículo en el sistema, en caso de no estar registrado el vehículo, el proceso terminará indicando al usuario que el vehículo no se encuentra registrado. Por otro lado, si el vehículo está registrado, el sistema verificará que haya una cita programada para este, si no tiene cita programada el sistema permitirá hacer el ingreso sin una cita previa, en este punto, si el

jefe de taller no desea ingresar el vehículo sin cita previa, el proceso terminará, mientras que si se quiere dar ingreso el vehículo ya sea con cita previa o sin ella, el sistema sugerirá un mecánico para que realice el mantenimiento del vehículo que será ingresado, y el jefe de taller podrá aceptar el mecánico sugerido o asignar uno diferente. Para asignar un mecánico distinto al sugerido el sistema solicitará los criterios de búsqueda a partir de los cuales mostrará una lista de mecánicos cuya información cumpla con los criterios ingresados por el usuario, luego, el usuario podrá seleccionar un usuario de la lista, cambiar los criterios de búsqueda o terminar el proceso. Una vez se tiene el mecánico que realizará el mantenimiento del vehículo (ya sea el sugerido o uno seleccionado de forma manual) el sistema registrará el ingreso del vehículo asignándolo al mecánico seleccionado.

Figura 12.

Diagrama de actividades registrar ingreso de vehículo



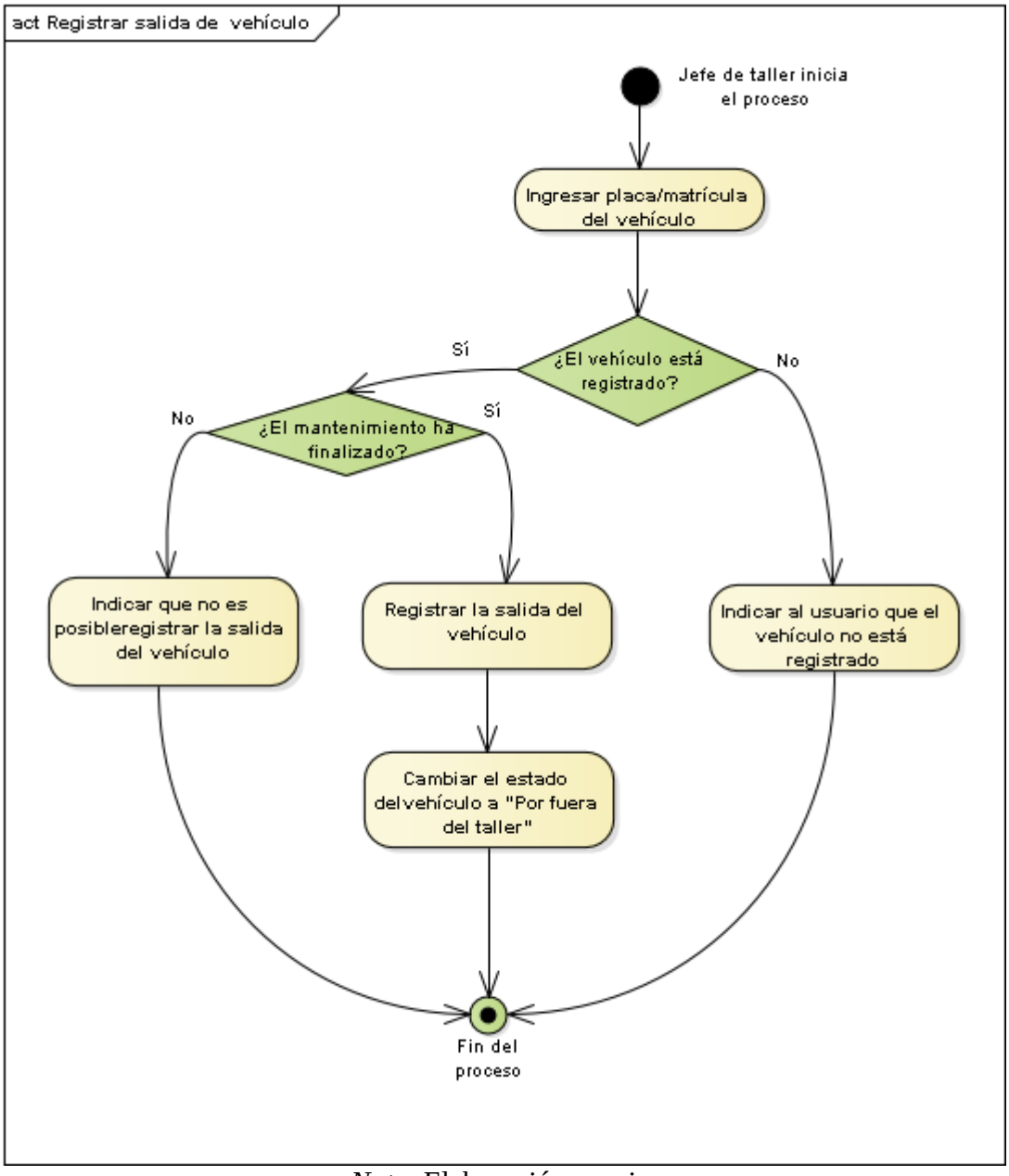
Nota. Elaboración propia.

Cuando un vehículo vaya a ser retirado del taller se llevará a cabo el proceso para registrar la salida del vehículo, el cual puede ser llevado a cabo únicamente por el jefe de taller, quien ingresará el número de placa o matrícula del vehículo. Si el vehículo no se encuentra registrado el sistema lo informará y se terminará el proceso, de lo contrario, el sistema verificará si el estado del vehículo es “Mantenimiento finalizado”, de no ser ese el estado actual del vehículo se indicará al usuario que no es posible registrar la salida del

vehículo porque el mantenimiento no ha finalizado y terminará el proceso, mientras que si el estado del vehículo corresponde al requerido para su salida, el sistema procederá a registrar la salida del vehículo y cambiará el estado del vehículo a “Por fuera del taller” finalizando así el proceso.

Figura 13.

Diagrama de actividades registrar salida de vehículo



Nota. Elaboración propia.

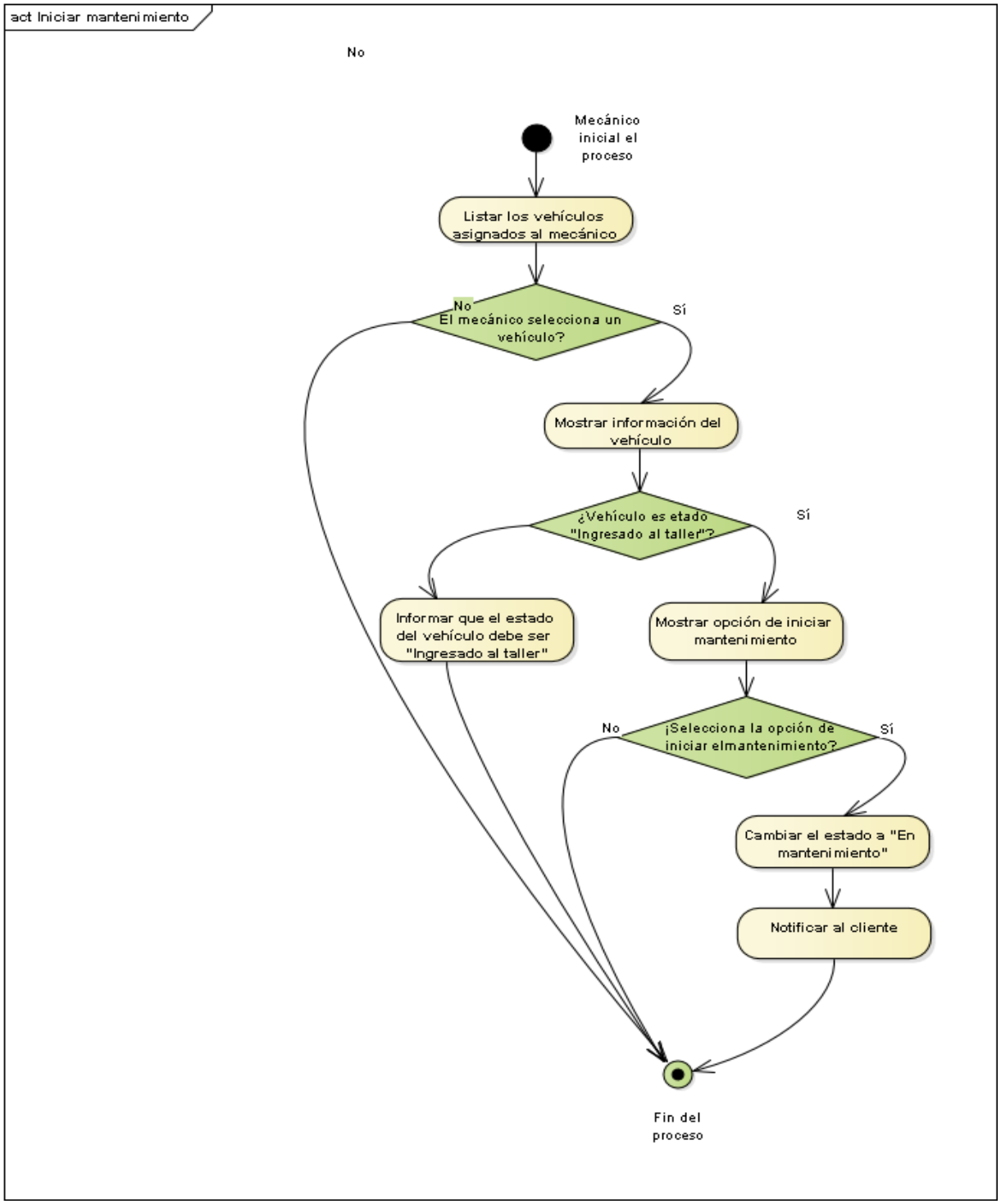
El proceso para iniciar un mantenimiento, como se ilustra en el diagrama anterior, puede ser llevado a cabo por un mecánico que tiene asignado del vehículo. Para ejecutar

este proceso el mecánico con una sesión activa en el sistema accede a la pantalla en la que se muestra una lista de todos los vehículos que tiene asignados para mantenimiento, y de la cual el mecánico podrá seleccionar un vehículo para iniciar su mantenimiento. Se estableció que no es necesario implementar una funcionalidad de búsqueda de vehículos sobre la lista que le es mostrada al usuario, ya que en dicha lista solo se mostrarán los vehículos que estén asignados en ese momento al mecánico la cual es relativamente pequeña.

Una vez seleccionado el vehículo, el sistema verifica que el vehículo seleccionado se encuentre en estado “Ingresado al taller”, de no ser así, el sistema le indicará al usuario que para iniciar el mantenimiento el vehículo debe estar en estado “Ingresado al taller”, en caso que el estado del vehículo sea el requerido para el proceso, el sistema le mostrará al usuario la opción de iniciar el mantenimiento del vehículo, si esta opción es seleccionada el sistema cambiará el estado del vehículo a “En mantenimiento” y notificará al cliente por el medio configurado en las preferencias. De no seleccionarse la opción de iniciar el mantenimiento no se cambiará el estado del vehículo ni se notificará al cliente, terminando el proceso.

Figura 14.

Diagrama de actividades iniciar mantenimiento



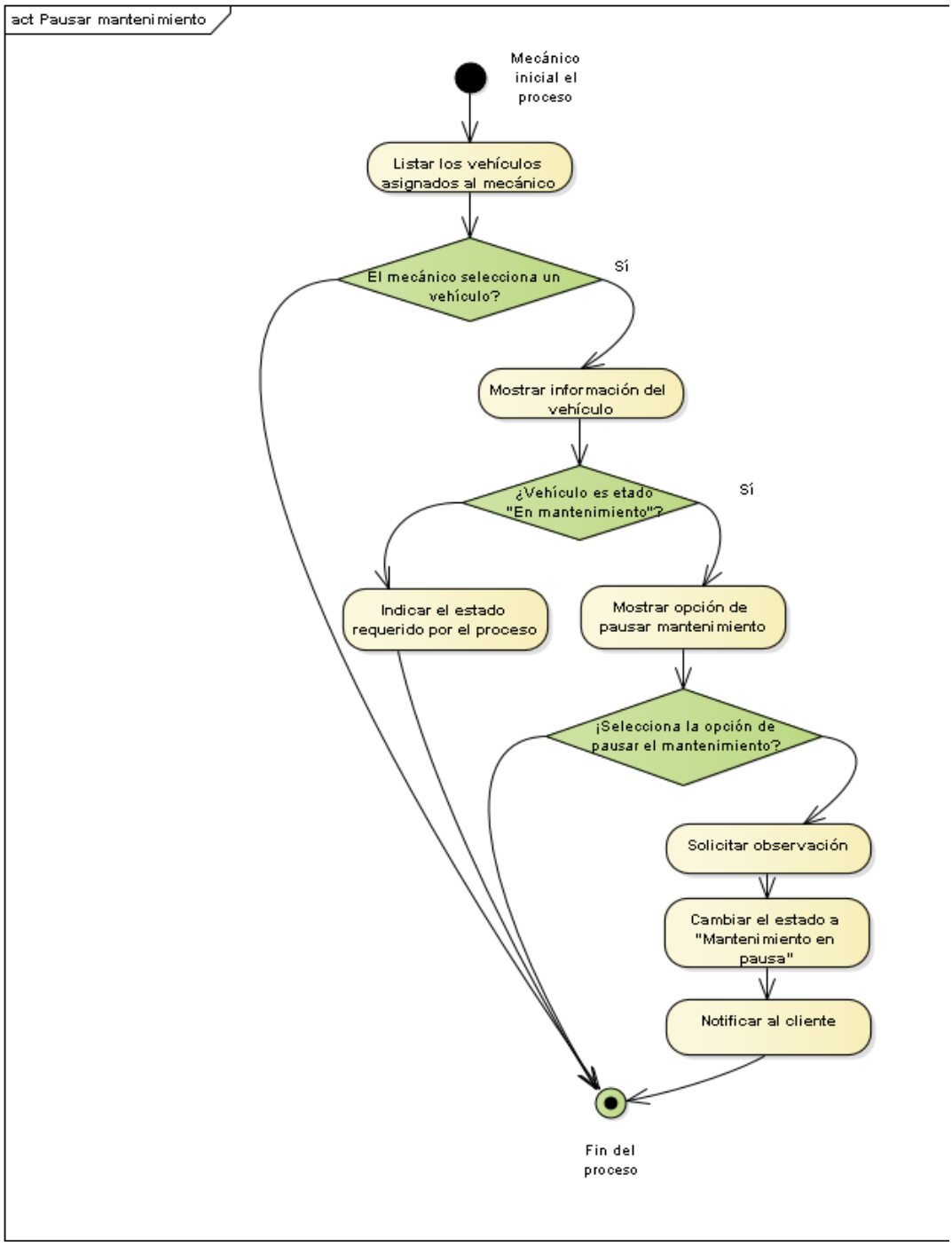
Nota. Elaboración propia.

El mecánico también tendrá la facultad de pausar un mantenimiento, proceso que iniciará mostrando la lista de los vehículos que el mecánico tenga asignados para mantenimientos, luego, si el mecánico selecciona un vehículo el sistema mostrará la información del mismo, y si el estado del vehículo es “En mantenimiento” el sistema mostrará la opción de pausar el mantenimiento. En el caso que el usuario seleccione la

opción de pausar el mantenimiento el sistema le solicitará una observación, cambiará el estado a “Mantenimiento en pausa” y notificará al cliente por el medio configurado, mientras que si no se selecciona la opción de pausar el mantenimiento el proceso se dará por finalizado.

Figura 15.

Diagrama de actividades pausar mantenimiento

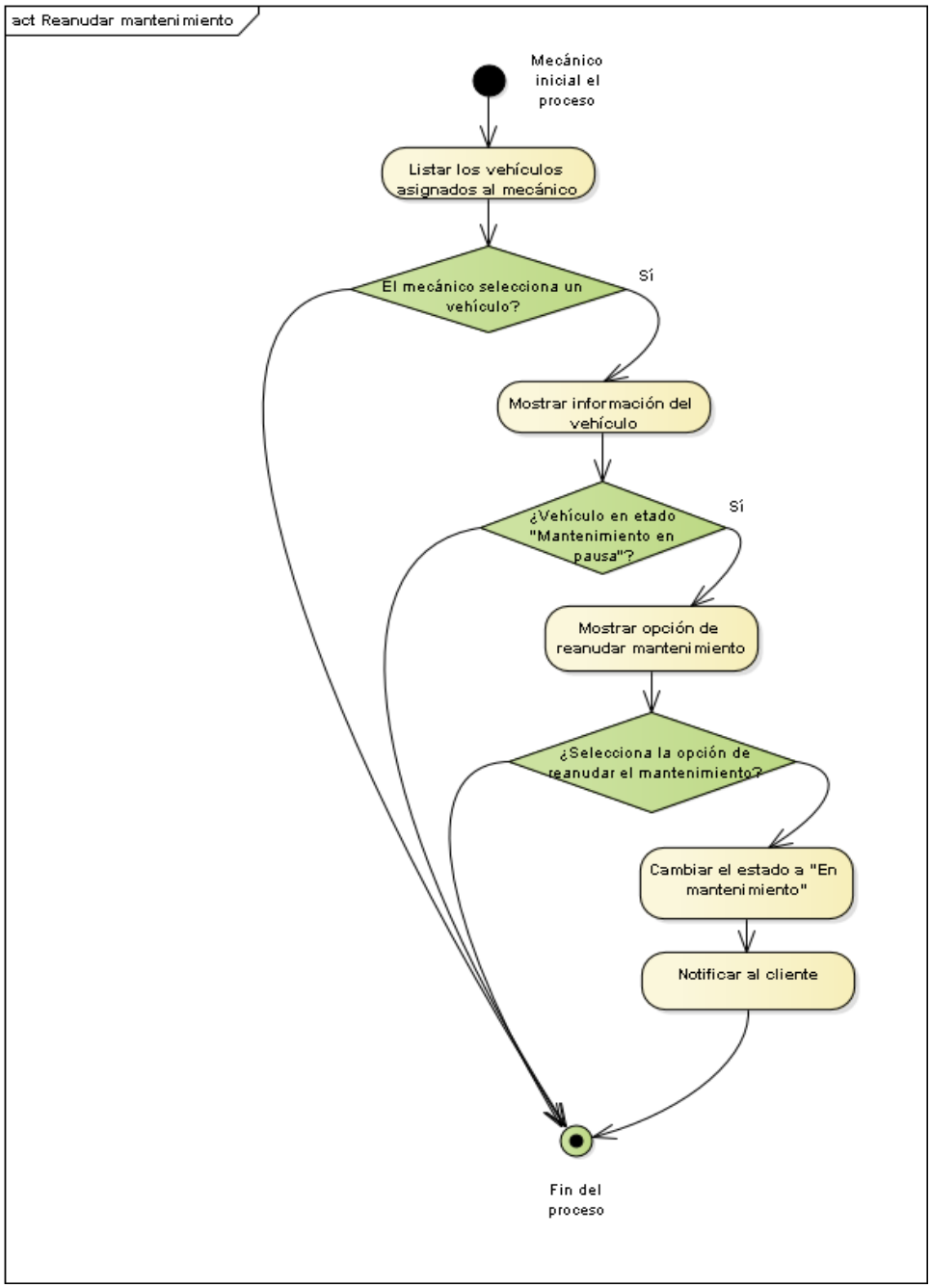


Nota. Elaboración propia.

Para reanudar un mantenimiento, es necesario que el estado del vehículo sea “Mantenimiento en pausa”, lo que causará que cuando el usuario seleccione un vehículo de la lista de los vehículos asignados le aparezca la opción de reanudar mantenimiento. Si el mecánico selecciona esta opción, el sistema cambiará el estado del vehículo a “En mantenimiento” y notificará al cliente por el medio configurado.

Figura 16.

Diagrama de actividades reanudar mantenimiento

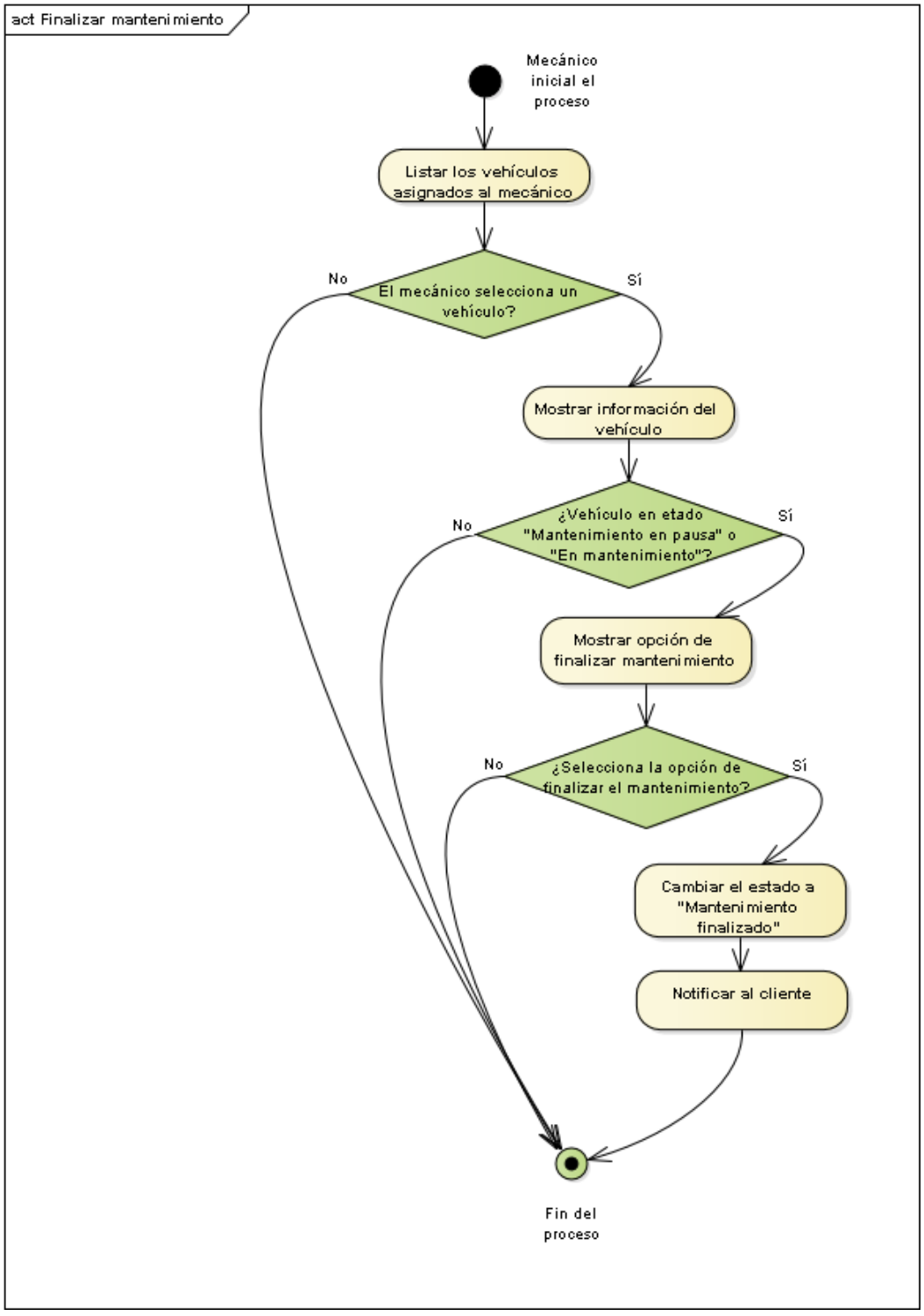


Nota. Elaboración propia.

Para finalizar un mantenimiento, se realiza un proceso similar a los procesos descritos anteriormente, teniendo en cuenta que para este proceso el vehículo debe estar en estado “En mantenimiento” para que el sistema le proporcione al mecánico la opción de finalizar el mantenimiento, y en caso de que esta opción sea seleccionada el sistema actualizará el estado del vehículo a “Mantenimiento finalizado” y notificará al cliente por los medios configurados.

Figura 17.

Diagrama de actividades finalizar mantenimiento



Nota. Elaboración propia.

Vista lógica

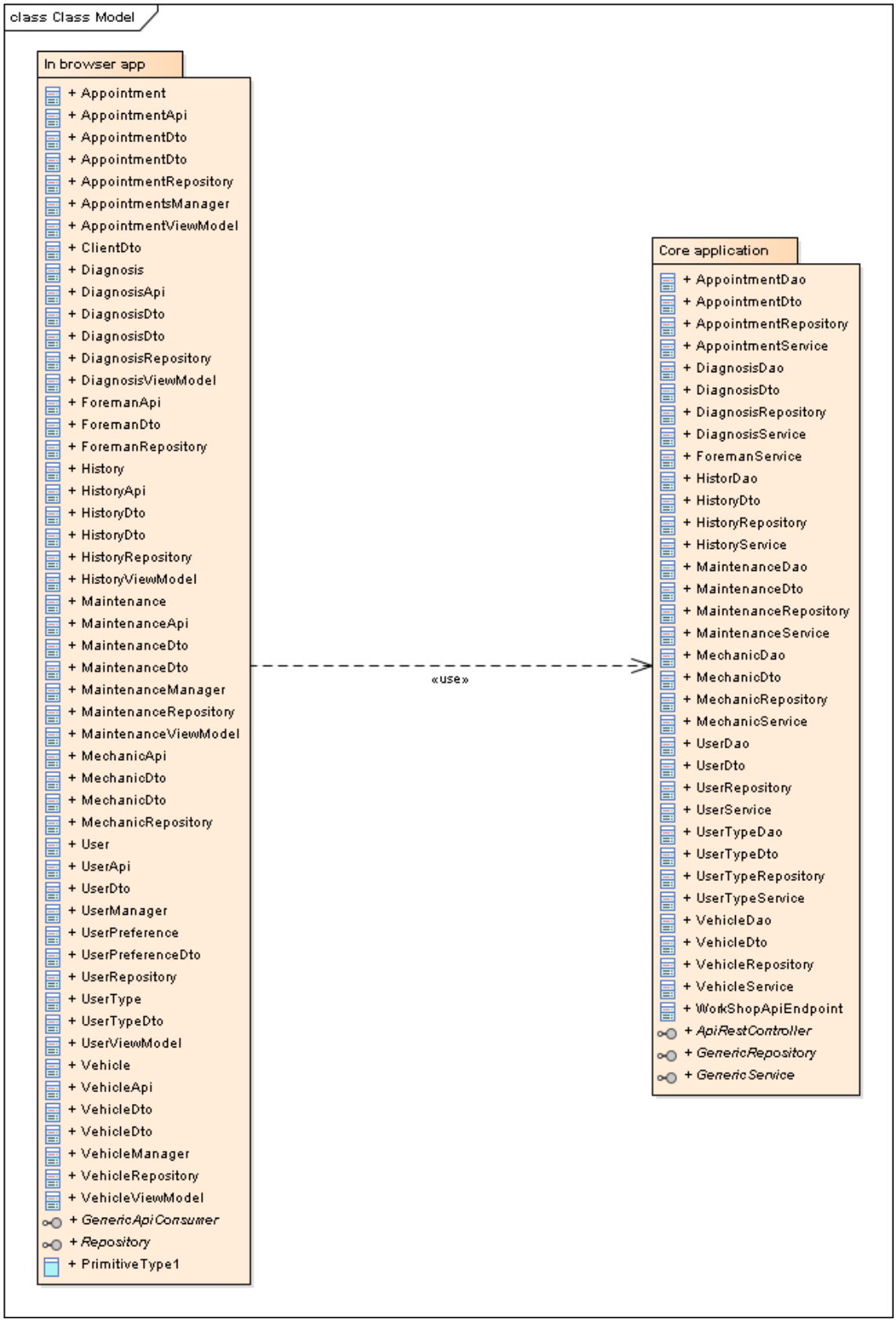
Teniendo en cuenta la naturaleza del sistema diseñado en el presente trabajo de investigación, se decidió abordar la solución utilizando el patrón de arquitectura SOA (Arquitectura orientada a servicios), el cual permite el diseño y el desarrollo de sistemas

distribuidos, permitiendo crear soluciones que satisfagan las necesidades del negocio ofreciendo facilidad para integrarse con sistemas externos y la alineación a los procesos de negocio reduciendo costos de implementación y permitiendo una adaptación fácil y rápidamente los cambios.

En el presente proyecto el patrón de arquitectura SOA se implementó a través de definición de dos grandes partes del sistema, donde una de ellas tiene que ver con los servicios relacionados al *backend* de la aplicación y la otra al *frontend*.

Figura 18.

Modelo de clases general

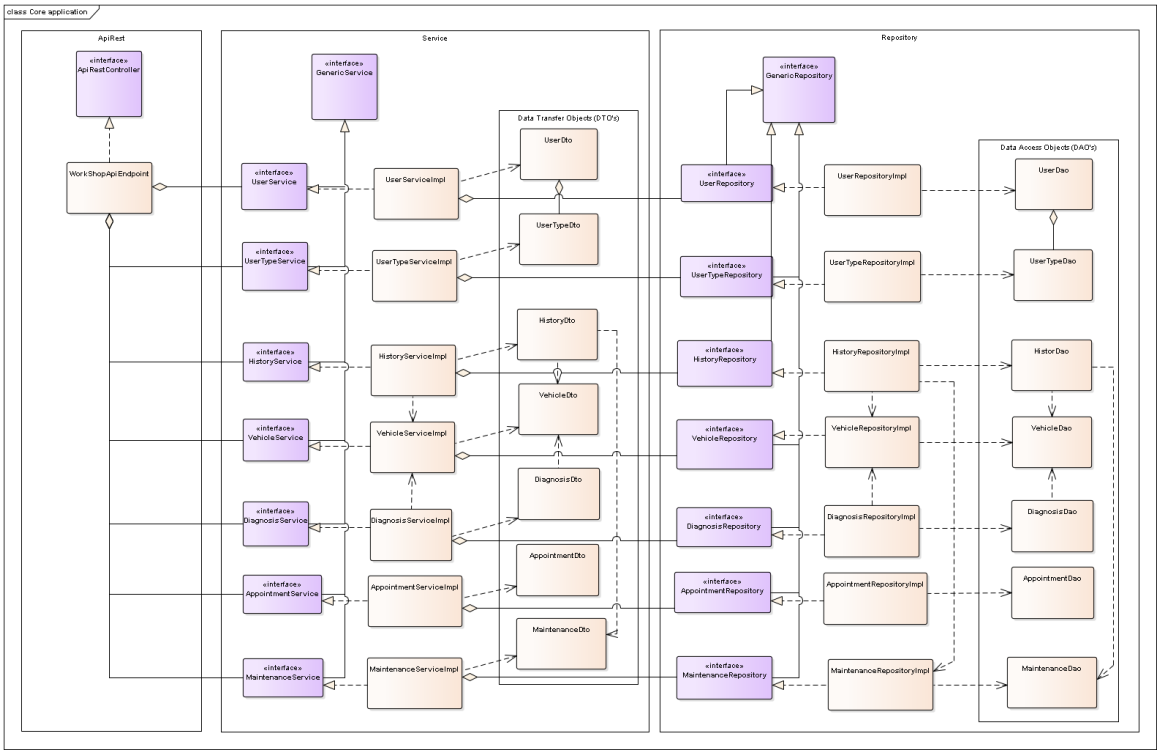


Nota. Elaboración propia.

Para diseñar la parte del sistema que corresponde a la lógica de *backend*, la cual se encarga de la validación y gestión de la información en el sistema y es el canal de comunicación del *frontend* con la persistencia, se utilizó el patrón de diseño MVC, el cual permite tener una reducción del esfuerzo de programación que se necesita para implementar sistemas múltiples y sincronizados de los mismos datos, partiendo de la estandarización del diseño de las aplicaciones (González & Romero, 2012).

Figura 19.

Modelo de clases backend



Nota. Elaboración propia.

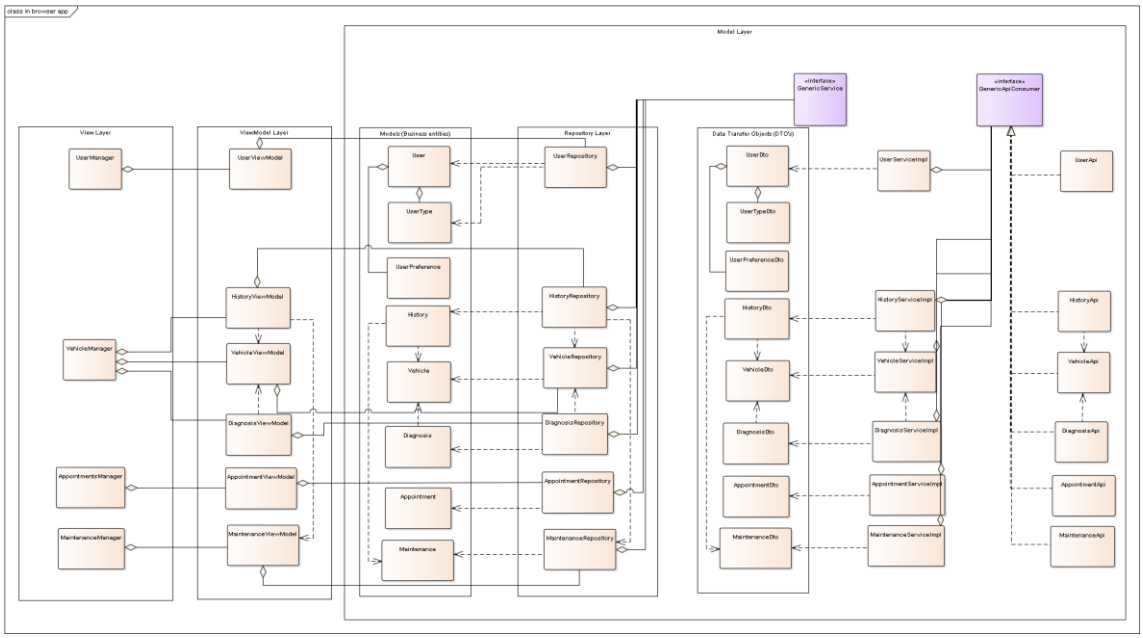
En el diagrama anterior se puede observar dentro del grupo *ApiRest* (que hace las veces de vista en el modelo MVC) una interfaz llamada *ApiRestController*, que es la encargada de exponer todos los servicios disponibles en *backend* para ser llamados desde el exterior, esta interfaz es el principal y único canal de comunicación hacia *backend* desde el exterior, y es a través de la cual los servicios del mismo son consumidos, teniendo coherencia y alineándose al patrón arquitectónico SOA. Se puede observar

también el módulo de servicios (*Service*), en el que se expone la lógica de cada servicio a través de interfaces, las cuales a su vez son especificaciones de la interfaz *GenericService* donde se establecen los comportamientos genéricos encargados de la lógica de validación y gestión de la información que será almacenada y consultada, utilizando para esta comunicación los objetos de transferencia de datos (DTOs) ilustrados en el diagrama. En este caso, la interfaz *GenericService* no es consumida directamente por *ApiRestController*, ya que además de los comportamientos genéricos establecidos por *GenericService*, cada servicio tiene comportamientos propios del mismo que deben ser expuestos por una interfaz especializada. El módulo *service* a su vez consume los servicios del módulo *Repository*, quien se encarga de la comunicación del sistema con la persistencia de datos, comunicación que se realiza haciendo uso de las entidades de acceso a datos (DAOs) presentadas en el diagrama.

Por otro lado, la división del sistema encargada de los servicios *frontend* del sistema está diseñado aplicando el patrón de diseño *Model-View-ViewModel* (MVVM) el cual, según Reyes (2018), permite separar la lógica de la vista, haciendo que sea posible aplicar diferentes vistas a un mismo *viewModel*, y/o reutilizar una misma vista para la implementación de diferentes funcionalidades con distintos *viewModels*, así como la capacidad de que el *software* pueda ser probado a través de pruebas unitarias, y que además provee un paso hacia la estructuración del proyecto para soportar principios SOLID como la inversión de control e inyección de dependencias.

Figura 20.

Modelo de clases frontend



Nota. Elaboración propia

En el anterior diagrama se ilustran los diferentes módulos de la parte *frontend* de sistema. Aquí se puede notar claramente la utilización del patrón MVVM. En este caso se explicará el diagrama iniciando desde el lado izquierdo hacia el derecho, observando inicialmente la interfaz "*GenericApiConsumer*" que es implementada por las clases encargadas de consumir los servicios *rest* del módulo de *backend*, utilizando para su comunicación las entidades de transferencia de datos (DTOs), y a su vez, interactuando con la capa de servicios, que es la encargada de agrupar los servicios que pueden ser consumidos por la capa del repositorio.

La capa de repositorio es la que se encarga de comunicarse con la capa de servicios y traducir las respuestas de esta a las entidades utilizadas por la capa de negocio de la aplicación (*Business entities*). Estas entidades son utilizadas por los *viewModels* de la aplicación, y estos, a su vez, contienen toda la lógica de la aplicación y, al mismo tiempo, se comunica con la vista realizando intercambios de información a través de las entidades de negocio mencionadas anteriormente. Finalmente, la capa visual de la

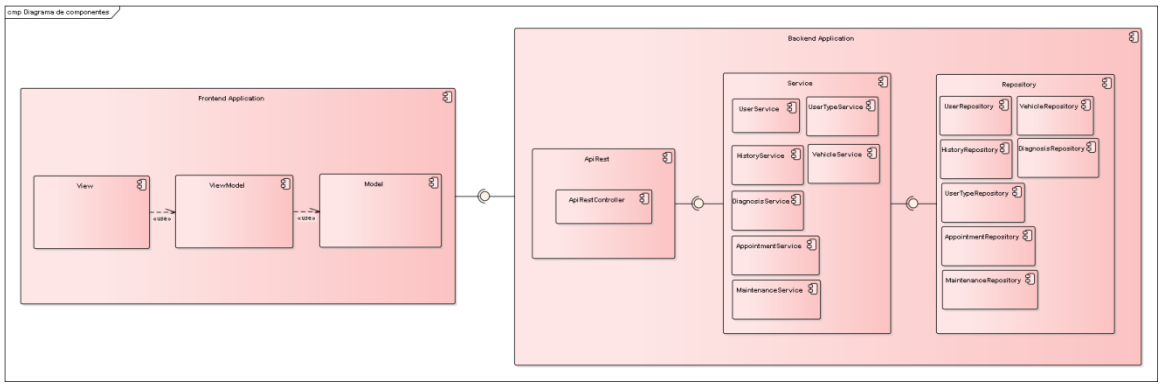
aplicación es la encargada de la interacción con el usuario proporcionándole la información que este necesite y recibiendo las solicitudes del mismo.

Vista de desarrollo

Para el diseño de la vista de desarrollo se descompuso el sistema en diferentes componentes que cumplen con funciones específicas y que mediante la interacción entre ellos logran el cumplimiento de cada uno de los requisitos funcionales del sistema.

Figura 21.

Diagrama de componentes



Nota. Elaboración propia

Como se ilustra en el diagrama anterior, el sistema dividió en dos grandes componentes principales llamados *Frontend Application* y *Backend Application*. El primero de estos es el encargado de cumplir con las funciones propias de la lógica del sistema del lado del navegador, esto va desde la interacción directa con el usuario, la lógica de validación del lado del navegador, y la comunicación con los servicios de comunicación con el módulo de *backend*. Por otro lado, el componente *Backend Application* se encarga de cumplir con las funciones que tienen que ver con la validación y gestión de la información del lado del servidor, esto incluye la exposición de las APIs hacia el exterior, la validación de la información recibida y la comunicación con el sistema de persistencia de datos.

Dentro de *Frontend Application* se establecieron tres componentes principales que representan las capas del patrón MVVM utilizado para aplicación del lado del cliente (o del navegador). El primero de estos es el módulo de la vista, que es el encargado de la presentación de la información al usuario de forma visual y de la interacción con el mismo a través de los distintos eventos que este pueda ejecutar en el sistema, interactuando con este primer elemento está el componente *ViewModel*, que es quien se encarga de ejecutar la lógica de los eventos que se disparan con la interacción del usuario, tomar las decisiones de qué información se debe mostrar y en qué momento, también se encarga de decidir qué información se requiere consultar en, los servicios *backend* que se deben invocar y la información que debe ser almacenada, y por último, se definió el componente *Model*, que es el encargado de conocer los servicios *backend* con los que la aplicación *frontend* puede interactuar y de la comunicación con los mismos, así como de la conversión de la información recibida por el módulo de *backend* (la cual es representada a través de los DTOs) a las entidades de la lógica de negocio utilizada por el módulo de *viewModel* y de la vista.

El componente *Backend Application* también se encuentra subdividido en diferentes módulos más pequeños que cumplen funciones específicas que ayudan al correcto funcionamiento del sistema. Dentro de este se encuentra el componente ApiRest, que a su vez cuenta con un único componente que es *ApiRestController*, y se encarga tanto de exponer los servicios que están disponibles para ser consumidos por el exterior (en este caso por *Frontend Application*), como de la comunicación con los sistemas externos que lo utilicen y redireccionar esa interacción a un módulo de *service* específico de pendiendo del API que sea invocada. El componente *service* es un componente que está compuesto por diferentes componentes *service* más pequeños, los cuales están encargados de realizar la lógica que permite validar y gestionar la información específica

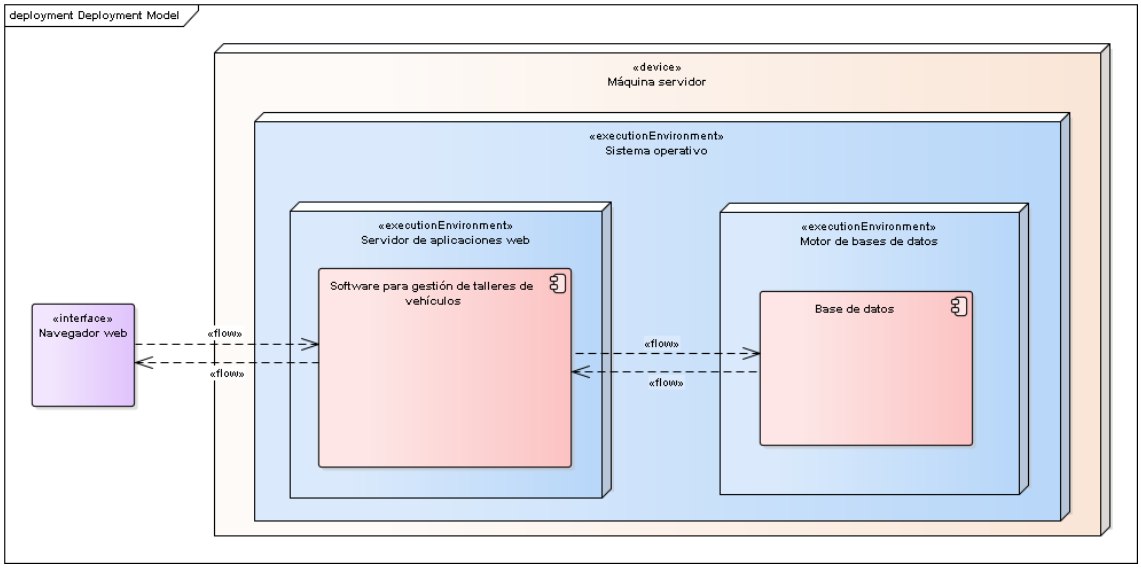
del sistema de acuerdo a su especialidad, por ejemplo, el componente *UserService* se encarga de validar la información de los usuarios que se quiera almacenar en el sistema y de invocar al módulo *Repository* para poder almacenar y de retornar una respuesta, ya sea positiva o negativa, al sistema externo que realizó la invocación mediante el API. Adicionalmente, también se encuentra el módulo *Repository*, que al igual que *Service* consiste en la agrupación de diversos componentes de repositorio, los cuales tienen el propósito de realizar la comunicación con la persistencia de datos y transformar los datos en entidades y viceversa para que el módulo *service* correspondiente pueda utilizarla.

Vista física

Las limitaciones físicas y visuales para la correcta ejecución del sistema necesitan ser definidas dentro de los artefactos de diseño de la arquitectura de *software*, por lo que en el siguiente diagrama se expone la forma en la que se debe desplegar el sistema indicando la infraestructura de hardware a utilizar y cómo se organizan los componentes de *hardware* y *software* entre sí. Teniendo en cuenta que el sistema diseñado en el presente trabajo debe ser ejecutado en un entorno web, se ejecutará en un servidor de aplicaciones web, el cual será desplegado sobre un sistema operativo que depende de los recursos físicos de una máquina que cumplirá las funciones de servidor físico, el *software* se comunicará con la base de datos que será gestionada por un motor de bases de datos, y este, a su vez, se ejecutará sobre el sistema operativo de una máquina física que cumplirá las funciones de servidor. Con el fin de que los usuarios sean capaces de interactuar con el sistema y dar ejecución a cada uno de los casos de uso, lo harán a través de un navegador web.

Figura 22.

Diagrama de despliegue



Nota. Elaboración propia

Capítulo 4. Conclusiones y recomendaciones

Como conclusión principal de este trabajo de investigación se puede decir que se obtuvo como resultado la arquitectura de *software* de un sistema que permite la gestión, el monitoreo y el manejo de históricos de servicios dentro de los talleres de mantenimiento de vehículos.

Una de las principales razones por la que la presente investigación es importante, es que con su desarrollo se contribuye en el proceso de creación de un sistema de *software* que ayude a los talleres de mantenimiento de vehículos a llevar de forma sistematizada y en línea toda la información relacionada a los clientes, mecánicos, jefes de taller, vehículos, históricos de mantenimientos, etc. Asimismo, a disminuir la cantidad de tiempo necesaria para realizar los procesos relacionados a la gestión de dicha información. Otra razón por la que el desarrollo de este proyecto es considerado importante es debido a que contribuye a la generación de un impacto ambiental positivo al disminuir el uso de papelería utilizada para gestionar la información y registrar los procesos dentro de los talleres.

Para futuros trabajos se recomienda realizar la codificación e implementación del sistema diseñado en la presente investigación teniendo en cuenta cada uno de los artefactos de diseño, ya que estos fueron creados a partir de las necesidades reales observadas en campo, pero sin limitarse, por el contrario realizando validaciones en campo de la utilidad del sistema y de los posibles incrementos que se puedan realizar para cubrir necesidades que a la fecha de publicación del presente trabajo puedan no existir, pero que puedan haber surgido en el tiempo.

Referencias bibliográficas

Acosta, L.A. (2005). *Guía práctica para la sistematización de proyectos y programas de cooperación técnica*. Oficina Regional de la FAO para América Latina y el Caribe.

Alvarado, C. (2012). *Sistema de información para apoyar el control de las actuaciones del consultorio jurídico de la UIS: SYSACJ-UIS 2.0*. [Trabajo de grado, Ingeniería de sistemas e informática, Universidad Industrial de Santander]. http://tangara.uis.edu.co/biblioweb/tesis/2012/146483.pdf

Anguisaca Ortega, W. A., & Peñafiel Vanegas, J. S. (2021). *Propuesta para la implementación de un software para la gestión de mantenimiento de la flota vehicular de la Empresa Pública Municipal de Movilidad, Tránsito y Transporte de Cuenca EMOV EP* [Trabajo de grado, Ingeniería mecánica automotriz, Universidad Politécnica Salesiana Sede Cuenca]. https://dspace.ups.edu.ec/bitstream/123456789/20680/1/UPS-CT009224.pdf

Ardila, J. G., Ardila, M. I., Rodríguez, D., & Hincapié, D. A. (2016). La gerencia del mantenimiento: una revisión. *Dimensión Empresarial, 14*(2), 129–144. https://doi.org/10.15665/rde.v14i2.480

Barnechea, M.M., y Morgan, M. (2007). *Propuesta de método de sistematización de experiencias*. [Tesis de maestría, Pontificia Universidad Católica del Perú]. https://cepalforja.org/sistem/documentos/Conocimiento_desde_practica.pdf

Bass, L., Clements, P., & Kazman, R. (2013). *Software Architecture in Practice* (3.a ed) Pearson India. https://edisciplinas.usp.br/pluginfile.php/5922722/mod_resource/content/1/2013%20-%20Book%20-

%20Bass%20%20KazmanSoftware%20Architecture%20in%20Practice%20%281%29.pdf

Bodhuin et al, M. (2003). Migrating COBOL systems to the Web by using the MVC design pattern. Ninth Working Conference on Reverse Engineering, 2002. Proceedings, 329–338. https://ieeexplore.ieee.org/document/1173090

Botero, C. (2009). Manual de mantenimiento. Parte I: ¿qué es el mantenimiento? *Informador Técnico, 47*, 35–37. https://revistas.sena.edu.co/index.php/inf_tec/article/view/1188/1302

Bustamante, L., y Ramos, J. (2009). *Diseño de un sistema de gestión de mantenimiento para una empresa de servicios en el área de telecomunicaciones* [Trabajo de grado, Universidad De Oriente Venezuela]. http://servicio.bc.uc.edu.ve/ingenieria/revista/IngenieriaySociedad/a10n2/art03.pdf

Castro-León, M. (2014). Arquitectura orientada a servicios, un enfoque basado en proyectos. *Enseñanza y Aprendizaje de Ingeniería de Computadores, 4,* 103-114. https://doi.org/10.30827/digibug.32208

Catucuago, H. (2022). *Implementación de un software de gestión de mantenimiento automotriz para el control preventivo y correctivo de la flota vehicular del GADIP Municipio de Cayambe* [Trabajo de grado, Ingeniería en mantenimiento automotriz, Universidad Técnica del Norte]. http://repositorio.utn.edu.ec/bitstream/123456789/12044/2/04%20MAUT%20172%20TRABAJO%20GRADO.pdf

Díaz, L., Torruco, U., Martínez, M., & Varela, M. (2013). *La entrevista, recurso flexible y dinámico. Investigación en educación médica*.

http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S2007-50572013000300009&lng=es&tlng=es

De la Torre Llorente, C., Castro, U. Z., Nelson, J. C., & Barroso, M. Á. R. (2010). *Guía de Arquitectura N Capas orientada al dominio con Net. 4.0.* https://resultadoslab.husi.org.co/resources/guia.pdf

Duffua, S., Raouf, A., & Cambel, J. (2000). *Sistema de Mantenimiento. Planeación y Control* (1ª Edición). Limusa.

Méndez, G. (2008). *Especificación de Requisitos según el estándar de IEEE 830.* https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

Farsimadan, K. (2014). *Vaatimusmääritys ALV-raportinautomatisointisovellukselle*. Haaga-Helia University of Applied Sciences

Fernández, L. (2006). *Cómo analizar datos cualitativos*. https://www.academia.edu/7572064/_C%C3%B3mo_analizar_datos_cualitativos

García, A. (2017). *Indicadores en mantenimiento*. http://www.mantenimientomundial.com/notas/indicadores-en-mantenimiento.pdf

Alulema, W. (enero-febrero 2015). Gestión de mantenimiento asistido por computador "CMMS". Industry tech magazine, (28), 4-7. https://documen.site/download/gestion-de-mantenimiento-asistido-por-computador-cmms_pdf

González, Y. D., & Romero, Y. F. (2012). Patrón Modelo-Vista-Controlador. *Telem@tica, 11*(1), 47–57. https://revistatelematica.cujae.edu.cu/index.php/tele/article/view/15/0

Guzmán, G., & Tovar, C. (2013). Implementacion del patron de mvc para el proceso de selección de personal [Trabajo de grado, Ingeniería de sistemas, Universidad

Libre]. https://repository.unilibre.edu.co/bitstream/handle/10901/8881/Trabajo%20de%20Grado%20Gelen%20Guzman%20y%20Natalia%20Tovar.pdf?sequence=1

Holliday, O. (2011). *Orientaciones teórico-prácticas para la sistematización de experiencias*. Alboan. http://centroderecursos.alboan.org/ebooks/0000/0788/6_JAR_ORI.pdf

Juez Gil, J. M. (2012). *Modelización y explotación de un GMAO*. Slideshare. https://es.slideshare.net/JoseMariaJuezGil/gestion-de-un-gmao-prisma3

Kruchten, P. (1995). The 4+1 View Model of architecture. *IEEE Software*, *12*(6), 42 -50. https://dblp.org/rec/journals/software/Kruchten95.html

López, G., Echeverría; Fierro, & Jeder, (2005). U*na propuesta de modelos de ciclo de vida (MCVS) para la integración de los procesos de negocio utilizando service oriented architecture(SOA).* https://core.ac.uk/download/pdf/15777267.pdf

López, L., & Guamán, R. (2015). *Implementación de una gestión de mantenimiento aistido por ordenador (GMAO) para la flota vehicular del GAD municipal de catamayo en la provincia de Joya*. [Tesis de pregrado, Ingeniería mecánica automotriz, Universidad Politécnica Salesiana]. https://dspace.ups.edu.ec/bitstream/123456789/7885/1/UPS-CT004734.pdf

Cup Coffe. (30 de marzo de 2012). El Patrón Model - View - ViewModel (MVVM). Cup Coffee. http://cup-coffe.blogspot.com/2012/03/el-patron-model-view-viewmodel-mvvm.html.

Medina Peña, D., Suárez Fragas, Y., & Hernández Alfonso, P. M. (2015). Sistema automatizado para la gestión del mantenimiento de equipos (módulos patrimonio y ordenes de trabajo). *Revista Ciencias Técnicas Agropecuarias*, *24*(S1).

https://link.gale.com/apps/doc/A467048648/IFME?u=anon~4a3c0b60&sid=googleScholar&xid=8a55ad0c

Montilla, C. (2016). Fundamentos de mantenimiento industrial. http://www.elprisma.com/apuntes/ingenieria_industrial/mantenimientoindustrial/

Mozo Coronel, P. A., & Méndez Garcés, B. A. (2020). *Elaboración de un plan de mantenimiento preventivo mediante la implementación de un software para la flota vehicular del gobierno autónomo descentralizado del cantón Otavalo* [Tesis de pregrado, Universidad Técnica del Norte]. http://repositorio.utn.edu.ec/handle/123456789/10720

Ocsa, A., Herrera, J., Suero, G., & Villalba, K. (s.f.). *Objetos de aprendizaje móviles para la educación básica regular: diseño, desarrollo y evaluación de un middleware.* https://recursos.educoas.org/sites/default/files/OBJETOS_DE_APRENDIZAJE_MOVILES_VIRTUALEDUCA_1.pdf

Quisilema Cuaical, K. A. (2018). *Implementación de un software para el manejo de mantenimiento preventivo del parque vehicular en el GAD municipal de Espejo* [Tesis de pregrado, Universidad Técnica del Norte]. http://repositorio.utn.edu.ec/handle/123456789/7909

Reyes, L. (21 de diciembre de 2018). Aplicando el patrón de diseño MVVM. *Medium*. https://medium.com/@reyes.leomaris/aplicando-el-patr%C3%B3n-de-dise%C3%B1o-mvvm-d4156e51bbe5

Reynoso, C. (2004). *Introducción a la arquitectura de software*. https://xdoc.mx/documents/introduccion-a-la-arquitectura-de-software-5cfc167e94b17

Sommerville, I. (2005). *Ingeniería de Software*. Pearson. http://zeus.inf.ucv.cl/~bcrawford/AULA_ICI_3242/Ingenieria%20del%20Software%207ma.%20Ed.%20-%20Ian%20Sommerville.pdf

Torres, L. (2004). *Mantenimiento, su implementación y gestión*. Universitas - Editorial Científica Universitaria

Vargas, C. (2010). *Ingeniería de software, enfoque práctico* (7ed.). Pressman. https://www.academia.edu/44770344/Ingenieria_de_software_enfoque_practico_7ed_Pressman_PDF

Apéndices



Especificación de requisitos de software