UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Migración del módulo Planificación de Mantenimiento del Sistema Orbita a la arquitectura de referencia en PHP Bosón

Autor: Yordan Bandera Rodríguez

Tutores: Ing. Yaniris Blanco Zamora
Ing. Ernesto Mató Roque

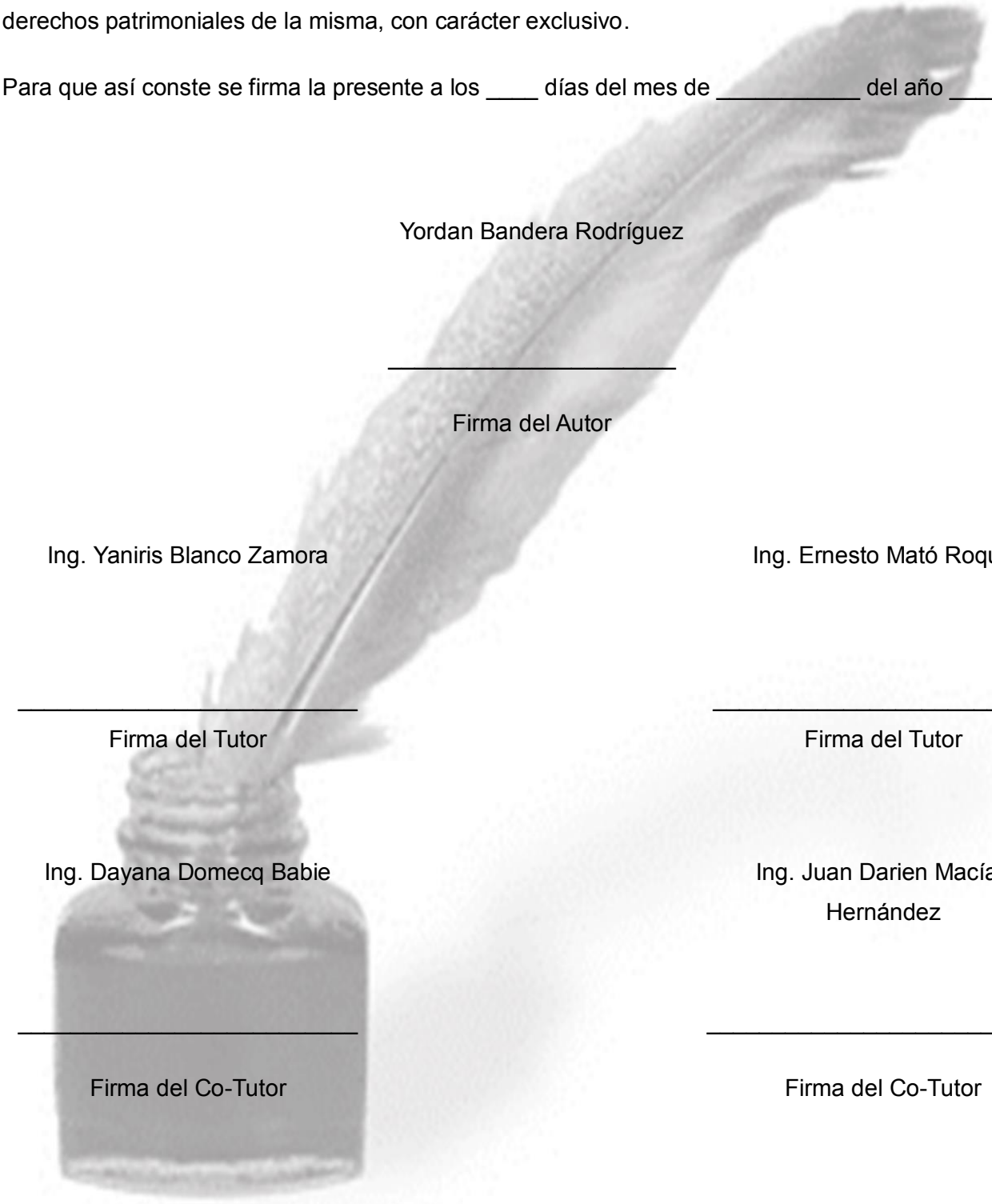
Co-Tutores: Ing. Dayana Domecq Babie
Ing. Juan Darien Macías Hernández

La Habana, 2016

Declaración de Autoría

Declaro ser el único autor de la presente y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los ____ días del mes de _____ del año _____.



Yordan Bandera Rodríguez

_____

Firma del Autor

Ing. Yaniris Blanco Zamora

_____

Firma del Tutor

Ing. Ernesto Mató Roque

_____

Firma del Tutor

Ing. Dayana Domecq Babie

_____

Firma del Co-Tutor

Ing. Juan Darien Macías Hernández

_____

Firma del Co-Tutor

Dedicatoria

Dedico el presente trabajo de diploma a mí mamá y papá por ser los mejores padres del mundo y darme las fuerzas para poder seguir adelante con mís sueños. A mí hermana por su apoyo incondicional. A mí familia por estar presente para mí siempre en las buenas y en las malas. A la familia que hoy no pudo estar aquí conmigo, mís abuelos Gisela, Jorge y Antonia, a mís tíos abuelos Raulito y Esther, que ellos estarían orgullosos de mí por haber llegado tan lejos.

Agradecimientos

Para llevar a cabo mis sueños fueron muchas las personas que influyeron a lo largo de mi carrera.

Quiero agradecer en primer lugar a mis padres, por ser mi guía, un ejemplo a seguir y por confiar en mí. A mi hermana y familia por darme el apoyo en las buenas y en las malas, que he pasado en esta universidad.

A mis hermanos del barrio o de la zona Rene y Dionys, que siempre han estado presente y me han tenido presente para cualquier momento de la vida. A todas mis amistades con las cuales he compartido durante este tiempo, en especial a Javier, mi compañero de cuarto, por ser un hermano para mí, a Elena por ser una amiga y hermana y darme buenos consejos, a Esther por ser una amiga y compañera que me ayudó en la universidad. Al piquete que siempre estaba ahí conmigo, Osvaldo, Ge, Yoandry, Kelly, el Jean Carlos, Alexei Ibáñez y Alexei Massabeaut, a Alejandro, Mebys, Neysi, Rachel, Dayanis y todo el personal del aula, en especial Eime. A mis amigos Carlos y Edel, que siempre me guardaban helado del bueno.

Un agradecimiento especial a Martha, por ser más que una amiga una hermana que ha confiado en mí para las buenas y las malas, a la pareja que siempre llevo en mi corazón a Dayana y Macías, que sin ellos no podía haber estado hoy aquí, a mis hermanos que no terminaron junto conmigo pero siempre están ahí que son Wilfredo, Michel, Elizabeth y Lisvany.

También quiero agradecer a los profesores que gracias a ellos soy un ingeniero, como son Ana Marys, la profesora Yaniesi, la profe de inglés Rosalina, la profe de tele Yinet y en especial al profe Leandro, que gracias a la ayuda que me ha

brindado he terminado con buenos resultados estos últimos años y a todos los profes por haberme brindado sus conocimientos.

Dar gracias a los tutores y al tribunal de tesis por su preocupación y paciencia.

A todas aquellas personas que de una forma u otra hicieron posible este momento, a las que he conocido y compartido durante toda la universidad y que de una forma u otra me ayudaron y me apoyaron quiero darle MUCHAS GRACIAS…

Resumen

El presente trabajo emprende la migración del módulo Planificación de Mantenimiento del Sistema Orbita a la arquitectura de referencia PHP Bosón. Permite realizar la gestión de grupos de vehículos con sus propiedades, sus documentos técnicos, las actividades que presentan, los tipos de mantenimientos y las fallas que se puedan presentar. La migración de este módulo permite corregir las deficiencias que presenta el Sistema Orbita en cuanto a la obsolescencia de las tecnologías utilizadas. Se abordan los conceptos fundamentales de la investigación como rendimiento y migración y se realiza un estudio de las diferentes tecnologías, lenguajes y herramientas a utilizar en el desarrollo del módulo Planificación. Se realizan pruebas de rendimiento para la comprobación del nuevo módulo, para mejorar el rendimiento con respecto al módulo anterior. En el transcurso del progreso del sistema se utiliza la metodología de desarrollo para la actividad productiva en la UCI que combina las buenas prácticas del nivel 2 del Modelo de Madurez de la Capacidad Integrado. La implementación del módulo Planificación para el nuevo Sistema Orbita contribuirá a mejorar su desempeño debido a las actualizaciones que incluyen las tecnologías que conforman Bosón.

Palabras clave: Bosón, mantenimiento, migración, planificación, rendimiento

Abstract

This paper undertakes migration module Maintenance Planning System Orbita reference architecture PHP Boson. It allows management of vehicle groups with their properties, technical documents, presenting activities, the types of maintenance and failures that may occur. The migration of this module to correct the deficiencies in the Orbita system regarding the obsolescence of the technologies used. The fundamental concepts of research as performance and migration are addressed and a study of different technologies, languages and tools is made to be used in the development of the Planning module. Performance tests to check the new module are made to improve performance over the previous module. During the progress of system development methodology for productive activity in the ICU that combines the best practices of Level 2 Maturity Model Integrated Capacity it is used. Planning module implementation for the new Orbita system will help improve performance because the updates that include technologies that make Boson.

Keywords: Boson, maintenance, migration, planning, performance

Índice

Índice de Figuras

Índice de Tablas

Índice de Anexos

Introducción

El hombre en el transcurso de la historia ha buscado la forma de desarrollar sus herramientas para la realización de diversos trabajos. Esta búsqueda se ha llevado a la necesidad de enfrentar múltiples retos que plantea el día a día, los cambios tecnológicos, el acceso a la información; lo cual le ha permitido ir perfeccionando sistemáticamente las herramientas con las que cuenta, hasta llegar a las computadoras, las cuales juegan un importante papel en el desarrollo del mundo actual (Cruz, 2006).

Estos avances tecnológicos han permitido un trascendental cambio en todas las esferas de nuestra sociedad proporcionado facilidades a la hora de la realización de un control y en el desarrollo de sistemas informáticos. Diversas empresas y organismos se han motivado en el uso de los sistemas, en aras de optimizar los procesos que realizan en función del tiempo y del costo de operación (González, 2011).

Un elemento indispensable para el exitoso funcionamiento de una empresa lo constituye el tener una eficiente administración de los recursos con que cuenta (Conocimiento organizacional: la gestión de los recursos y el capital humano, 2006). En este sentido se puede decir que para empresas con flotas de vehículos, el transporte constituye una actividad importante que le aporta beneficios, no solo por los ahorros que genera, sino también por la calidad de los servicios que brinda de producción y de distribución.

El Centro de Informatización de Entidades (CEIGE), ubicado en la Universidad de las Ciencias Informáticas (UCI), tiene dentro de sus líneas de investigación una que está enfocada al desarrollo de sistemas de gestión de las flotas de vehículos. Actualmente en el departamento de Aplicaciones de Gestión Empresarial (AGE) de CEIGE, se lleva a cabo la ejecución del proyecto Sistema de Control de Flota y Mantenimiento Orbita. Este sistema tiene como finalidad gestionar los procesos relacionado con los mantenimientos en una base de transporte.

El Sistema de Control de Flota y Mantenimiento Orbita emplea está desarrollado sobre la tecnología Sauxe en su versión 2.0, la cual en su implementación ha presentado insuficiencias en algunas características que presentan las capas de la tecnología Sauxe:

- En la capa de presentación se utiliza ExtJs en su versión 3.4, versión que presenta problemas de rendimiento y se encuentra atada a una licencia de pago para su uso comercial y fue utilizada a solicitud del cliente.
- En la capa de negocio se hace uso de Zend Framework 1.2, marco de trabajo desarrollado en una versión obsoleta de PHP 5.3.10, que no presenta la generación de código fuente mediante

pueden líneas de comandos para la creación y mantenimiento de aplicaciones, ni el almacenamiento en caché de las vistas. Además existe una mejora respecto al rendimiento de las versiones superiores a PHP 5.3.x (Ltd., 2016).

- En la capa de acceso a datos se hace uso de Doctrine 1.2.2, lo cual provoca bajo rendimiento y un mayor acoplamiento entre sus clases por lo que se pierde independencia y sería más costoso a la hora del cambio hacia otra tecnología. Otro inconveniente que presenta el uso de esta versión de Doctrine es que a partir del 1ro de junio del 2011 se dejó de dar soporte, trayendo consigo que no se corrijan posible errores y vulnerabilidades que atentan contra su estabilidad y rendimiento (Doctrine, 2016).

Además de los aspectos anteriormente expuestos, se identificaron varios factores subjetivos mediante una encuesta (Anexo 1) que influyeron negativamente en el desarrollo del sistema objeto de estudio tales como:

- Poca gestión de documentación relacionada con la arquitectura y funcionamiento de Sauxe, demostrado que el 100% del personal encuestado coinciden con este problema.
- No hubo un correcto uso de los estándares de codificación, opinión dada por el 75% del personal que se le realizó la encuesta.
- El 100% de los especialistas encuestados coinciden que el tiempo de respuesta de la página inicial o página de acceso al sistema oscila entre 0 a 12 segundos, influyendo negativamente en el trabajo con el sistema por parte de los usuarios.
- La no puesta en funcionamiento de un sistema de auditorías al código fuente o al cumplimiento de las definiciones arquitectónicas.
- Al determinar el rendimiento a través de la herramienta JMETER (Anexo 2) da un resultado de 19300 peticiones de muestras que se le realizaron para un total de 10044 milisegundos de ejecución, dando consigo que la el 90% de las páginas respondió en 875 milisegundos demostrando que su rendimiento está en 227.5 segundos por cada petición a realizar.

Partiendo de los problemas antes planteado se propone el siguiente **Problema a resolver:** ¿Cómo contribuir a mejorar el rendimiento del módulo Planificación de Mantenimiento del Sistema Orbita?

Objeto de estudio: Migración de tecnologías de desarrollo en sistemas de gestión.

Objetivo general: Migrar el módulo Planificación de Mantenimiento del Sistema Orbita a la arquitectura de referencia en PHP Bosón, contribuyendo a mejorar el rendimiento del sistema.

Campo de acción: Migración de tecnologías de desarrollo en sistemas de mantenimiento vehicular.

Objetivos específicos:

1. Realizar un estudio del estado del arte para la elaboración del marco teórico en torno al objeto de estudio.

2. Analizar la ingeniería de requisitos del módulo planificación de mantenimiento del sistema Orbita.

3. Rediseñar la estructura de componentes y clases, aplicando los patrones de diseños, algoritmos y técnicas consideradas como necesarias en el estudio.

4. Implementar el módulo Planificación de Mantenimiento del Sistema Orbita, siguiendo las técnicas de programación estudiadas en la tecnología de desarrollo selecciona para la migración.

5. Validar la solución propuesta.

Idea a defender: Si se realiza la migración del módulo de planificación de mantenimiento del Sistema Orbita a la arquitectura de referencia en PHP Bosón, se contribuye a mejorar su rendimiento.

Métodos científicos

Métodos teóricos

- **Analítico–Sintético**: se utiliza con el fin de analizar bibliografías para sintetizar los elementos importantes que se relacionan con esta investigación. Este método se utiliza también para tomar decisiones en cuanto a los resultados arrojados por las encuestas realizadas, y para sentar las bases teóricas para el desarrollo de la tesis.

- **Modelación**: se utiliza con el objetivo de crear modelos y diagramas que son abstracciones del producto final, permitiendo tener un dominio inicial de la información que se va a modelar, representar de forma estática los requisitos y obtener una versión del sistema original para validar los requisitos con el cliente.

Métodos empíricos

- **Encuesta**: se utiliza con el objetivo de obtener información valiosa para el desarrollo de la investigación para ser analizada posteriormente a partir de un listado de preguntas previamente elaboradas. También se utiliza para conocer el criterio de los trabajadores del CEIGE con respecto a la propuesta de solución.

Estructura del documento:

Para el cumplimiento de los objetivos propuestos, el trabajo de tesis se ha estructurado de la siguiente forma: introducción, tres capítulos, conclusiones, recomendaciones, bibliografía y anexos.

Capítulo 1. Fundamentación teórica

Se analiza la gestión de los procesos relacionados con los mantenimientos de una base de transporte. Además se justifica el empleo de cada una de las herramientas, metodologías, lenguajes de desarrollos, marcos de trabajo y tecnologías en el desarrollo del módulo Planificación de Mantenimiento del Sistema Orbita a la arquitectura de referencia en PHP Bosón.

Capítulo 2. Propuesta de solución

En este capítulo se realiza la modelación del módulo Planificación de Mantenimiento del Sistema Orbita a la arquitectura de referencia en PHP Bosón y estudio de requisitos. Se expone el diseño arquitectónico presente en la solución, los diagramas de clases del diseño, así como los patrones de diseño utilizados. Se presenta además, el mapeo de datos y los diagramas de componente y despliegue.

Capítulo 3. Validación de la investigación

En este capítulo se exponen los resultados obtenidos luego de haber realizado pruebas de software para determinar la calidad del sistema desarrollado. Asegurando el correcto funcionamiento de las funcionalidades implementadas para apoyar el proceso de migración del módulo Planificación de Mantenimiento del Sistema Orbita a la arquitectura de referencia en PHP Bosón.

Capítulo 1. Fundamentación teórica

Introducción

En este capítulo se exponen los principales aspectos de la investigación con vista a conocer la forma en que se va a migrar el Sistema Orbita a un marco de trabajo más avanzado. Bajo este marco teórico se identificarán los principales elementos que la sustentan y que forman parte del sistema, se explicarán cada uno de ellos a partir de sus conceptos y aplicación, con el propósito de comprender sus significados; así como su apreciación y valoración en la práctica. El capítulo culmina argumentando la metodología de desarrollo, la arquitectura de software, las tecnologías, el marco de trabajo y las herramientas a utilizar en el desarrollo de la solución. También se presentan las técnicas a utilizar para la validación de la investigación.

1.1 Conceptos asociados a la investigación

La historia del mantenimiento acompaña el desarrollo Técnico-Industrial de la humanidad. Al concluir el siglo XIX, con la mecanización de las industrias, surgió la necesidad de las primeras reparaciones (Tavares, 2000). Con la evolución de las reparaciones, surgieron algunas definiciones con respecto al mantenimiento.

1.1.1 Mantenimiento

Conjunto de actividades que deben realizarse a instalaciones y equipos, con el fin de corregir o prevenir fallas, buscando que estos continúen prestando el servicio para el cual fueron diseñados (Pérez, 2008).

Al igual que las reparaciones, el mantenimiento ha evolucionado y se ha clasificado en diversos tipos:

- **Mantenimiento Correctivo**: Mantenimiento Correctivo o por Rotura consiste en la corrección de las averías, roturas o fallas cuando se presentan, impidiendo el diagnóstico fiable de las causas que provocan las fallas (Garrido, 2009).
- **Mantenimiento Preventivo Planificado:** Es la programación de actividades de inspección de los equipos, tanto de funcionamiento como de limpieza y calibración, que deben llevarse a cabo en forma periódica con base en un plan de aseguramiento y control de calidad. Su propósito es prevenir las fallas, manteniendo los equipos en óptima operación (Garrido, 2010). La característica principal de este tipo de mantenimiento es la de inspeccionar los equipos, detectar las fallas en su fase inicial y corregirlas en el momento oportuno (Renove Tecnología, 2009-2016).

Desde el punto de vista informático la función principal del **mantenimiento Preventivo** es evitar una posible avería/reparación del equipo o pérdida de datos, mientras que la del **mantenimiento Correctivo** es arreglar una avería/reparación del equipo una vez ocurrida (Armero Kreisberger, 2011).

Partiendo de los conceptos antes estudiados sobre el mantenimiento se escoge como concepto general el de Luis Alberto Cuarta Pérez, el cual plantea que el mantenimiento es el conjunto de actividades que deben realizarse a instalaciones y equipos, con el fin de corregir o prevenir fallas, buscando que estos continúen prestando el servicio para el cual fueron diseñados. Además, de los conceptos estudiados de los tipos de mantenimientos, para la planificación se escoge el de mantenimiento preventivo expuesto por Armero Kreisberger. El mismo plantea que es la función principal de este tipo de mantenimiento es evitar una posible avería/reparación del equipo o perdida de datos, como por ejemplo un diagnóstico de hardware periódicamente, una limpieza a los equipos de hardware o un cambio de pasta térmica en los disipadores para ayudar a reducir la temperatura de los componentes del equipo.

1.1.2 Planificación de Mantenimiento

Existen diferentes criterios sobre la planificación de mantenimiento, al respecto diferentes autores como Yolanda Gil Ojeda, Eva Vallejo García y Alicia Arias Coello se han pronunciado, pero todas tienen un punto coincidente: “un proceso es una secuencia de actividades, tareas o pasos y que llevan al cumplimiento de algún resultado” (Ojeda, 2008). En la presente investigación se analizó el proceso general del módulo de Planificación, comenzando por la definición de planificación.

La Planificación consiste en una exhaustiva preparación de las tareas de mantenimiento, definiendo las acciones o proyectos que se van a realizar como parte de la política de la organización y se determinan los recursos que están involucrados en los mismos con vistas a su utilización (Fernández Rodríguez, 2002).

Las actividades de planificación son definidas para una empresa en su totalidad, aparecen dentro de las que la Gestión del Mantenimiento determina. La característica fundamental de la planificación es la relación entre las actividades que se van a realizar, teniendo como finalidad la ejecución de acciones que evitan las fallas de los equipos, siempre que su costo sea justificable en comparación con las afectaciones inducidas o por la ocurrencia de una falla. Si bien el objetivo de la planificación es lograr un mantenimiento con el mínimo coste, el mayor tiempo de servicio de las instalaciones y maquinarias productivas, su finalidad radica en conseguir la máxima disponibilidad, aportando el mayor rendimiento, calidad del producto y máxima seguridad de funcionamiento, y permitiendo tener una visión global y concreta de todas las acciones previstas para una instalación determinada (Davenport, 2013).

Partiendo del concepto referente a la planificación de mantenimiento ofrecido por Daverport, se puede afirmar que el Sistema Orbita refleja las características antes expuestas, ya que las actividades que realiza tienen como finalidad prevenir las fallas que puedan presentar los vehículos. Así como determinar las propiedades que presentan y dar a conocer qué tipo de mantenimiento es el adecuado para cada vehículo. Propiciando una solución con calidad y segura que permita tener una visión de todas las acciones que se van a realizar.

1.1.3 Sistema de gestión de flota y mantenimiento

Dada la creciente necesidad de controlar el mantenimiento de los equipos e instalaciones, se ha tomado como estrategia el desarrollo de sistemas informáticos para automatizar los procesos de las empresas que se dediquen a la gestión del mantenimiento. Logrando así un ahorro de presupuesto, recursos materiales y humanos, así como alcanzar resultados satisfactorios en un menor tiempo posible (Revista Venezolana de Gerencia, 2010).

Los sistemas de gestión de flota están diseñados para realizar un eficaz control sobre el estado de los vehículos de una flota. Se puede almacenar información, realizar un control de mantenimiento preventivo, consumo de combustible, consumo de ruedas, viajes, entre otras funcionalidades (TelFo Networks S.L, 2003-2012).

1.1.4 Migración de sistemas informáticos

El término migración en el mundo de la informática es el “proceso consistente en hacer que los datos y las aplicaciones existentes funcionen en una computadora, software o sistema operativo distinto al origen en que se encuentran” (mipatente, 2016).

En la actualidad este término se ha utilizado en muchas ocasiones entorno al auge de las migraciones a software libre y al hecho de que instituciones públicas a nivel mundial han realizado este proceso de migración exitosamente. La migración se puede encontrar en muchas versiones en el ámbito de la informática, por lo que consiste en actualizar un software a una versión superior o mejor, que remplaza a una versión antigua u obsoleta instalada de un producto, no alterando los datos, ni la preferencia del usuario (Entonado, 2001).

Las ventajas y desventajas que presenta la migración de versiones se muestran a continuación:

Ventajas

- Ocurrencia de incorporación de nuevas tecnologías (Mateu, 2004).
- Eliminación de errores presentes en las versiones anteriores (Mateu, 2004).
- Desarrollo de nuevas funcionalidades que enriquecen el manejo de las herramientas (Mateu,

2004).

- Soporte técnico actualizado sobre la versión actualizada brindada por los proveedores de la herramienta (Mateu, 2004).

Desventajas

- En algunas ocasiones se debe pagar para mantener actualizado un sistema (Mateu, 2004).
- No todas las versiones nuevas que se realizan son compatible con sus versiones anteriores (Mateu, 2004).
- Mucho tiempo entre una versión y otra (Mateu, 2004).

Con el objetivo de mejorar el rendimiento del Sistema Orbita se decide realizar una migración tecnológica, actualizando las tecnologías con que está desarrollado. Previniendo futuros problemas de funcionamiento y mantenimiento.

1.1.5 Rendimiento de sistemas informáticos

Según la norma ISO 9129, rendimiento es la capacidad del producto de software para proporcionar apropiados tiempos de respuesta y procesamiento, así como tasas de producción de resultados, al realizar su función bajo condiciones establecidas (ISO, 2001).

El rendimiento como atributo de calidad posibilita caracterizar el comportamiento de un sistema en su entorno de producción. Medir el rendimiento de una aplicación, permite a las organizaciones tener una percepción de cuántos recursos deben dedicar para que el software realice sus funciones, determinar cuánta carga de trabajo puede procesar el sistema por unidad de tiempo y por otra parte, al equipo de desarrollo le permite realizar una valoración crítica sobre los problemas que pudieron afectar la calidad del producto durante su implementación (Guasch, 2006).

El Instituto de Ingeniería de Software define que el rendimiento caracteriza la eficacia del servicio que provee el sistema (SEI, 2016). Se clasifican dentro de este atributo como requerimientos de rendimiento:

- **Latencia**: Intervalo de tiempo que transcurre desde que se produce un evento hasta que se ejecuta una respuesta del sistema. Este intervalo está dado por un tiempo de inicio (latencia mínima) y un tiempo final (latencia máxima) (Collard, 2005).
- **Capacidad de procesamiento**: Define el número de respuestas de eventos que deben ser completadas totalmente en un intervalo de tiempo especificado (Collard, 2005).
- **Capacidad**: Es la medida de la cantidad de trabajo máximo que un sistema puede realizar. Una definición más práctica puede ser la máxima capacidad de procesamiento que se puede lograr

sin violar los requerimientos de latencia especificados (Collard, 2005).

- **Modo**: Puede ser caracterizado como el estado de la demanda del sistema y el estado del sistema, la configuración de los recursos utilizados para satisfacer la demanda (Collard, 2005).

Herramientas para medir rendimiento

En la actualidad la utilización de herramientas se ha hecho imprescindible para realizar disímiles procesos. Las pruebas son un flujo importante dentro del proceso de desarrollo de software y en ocasiones se requiere utilizar determinadas herramientas para mejorar la calidad y eficiencia de dicho flujo. Existe una gran variedad de soluciones informáticas enfocadas a medir uno o varios objetivos de rendimiento en un sistema, en general, concebidas para ejecutar pruebas de rendimiento. En la presente investigación se tuvo en cuenta el estudio de JMeter como herramienta de rendimiento (Vázquez, 2007).

JMeter es una aplicación de escritorio desarrollada en Java, diseñada para medir el rendimiento y el comportamiento de los sistemas ante las pruebas de sobrecarga. Se puede utilizar para probar el rendimiento tanto de los recursos estáticos como dinámicos. También para simular una sobrecarga en un servidor, una red o un objeto, para poner a prueba su resistencia o para analizar el rendimiento global para diferentes tipos de carga. Además, permite hacer un análisis gráfico de rendimiento o probar el comportamiento del objeto con sobrecargas concurrentes (Apache Software Foundation, 1999-2016).

1.2 Metodología de desarrollo de software

“Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo que proporciona un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito” (INTCO, 2009).

AUP- UCI

Al encontrarse la UCI inmersa en un programa de mejora de desarrollo de software, decide realizar una variación de la metodología Proceso Unificado Ágil (AUP por sus siglas en inglés) en unión con el Modelo de Integración de Madurez de Capacidades para el Desarrollo en su versión 1.3 (CMMI-DEV por sus siglas en inglés), con el objetivo de que sea aplicable a todos los proyectos productivos que se realicen en la universidad (Sánchez, 2015).

AUP es una versión simplificada del Proceso Unificado del Rational (RUP por sus siglas en inglés). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP (Scott W. Ambler, Ambysoft Inc, 2005-2014).

Esta metodología consta de cuatro fases (Inicio, Elaboración, Construcción y Transición), para el ciclo de vida de los proyectos de la UCI se decide mantener la fase de Inicio y se unifican las restantes tres fases de AUP en una sola llamada Ejecución y se agrega una fase de Cierre (Sánchez, 2015).

1. **Inicio**: Durante la fase de inicio se realizan las actividades relacionadas con la planeación del proyecto. En esta fase se obtiene la información fundamental acerca del alcance del proyecto, se realizan estimaciones de tiempo, esfuerzo y costo y se decide si se ejecuta o no el proyecto (Sánchez, 2015).
2. **Ejecución**: En esta fase se ejecutan las actividades requeridas para desarrollar el software. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente, además, en la transición se capacita a los usuarios finales sobre la utilización del software (Sánchez, 2015).
3. **Cierre**: Durante esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto (Sánchez, 2015).

La presente investigación basa su desarrollo en las fases de Ejecución y Cierre. La fase de Inicio no se ejecuta debido a que sus actividades no se realizan durante esta investigación.

AUP propone siete disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener ocho disciplinas (Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas, Pruebas de liberación, Pruebas de aceptación y Despliegue) (Sánchez, 2015). Durante la presente investigación se desarrollan las siguientes disciplinas:

1. **Requisitos**: Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. En la presente investigación no se modela el negocio, por lo que se realizan Historias de Usuario (HU) para modelar el sistema, facilitando la comprensión de su funcionamiento (Sánchez, 2015).
2. **Análisis y diseño**: En esta disciplina se modela el sistema, su forma y arquitectura para que soporte todos los requisitos, incluyendo los requisitos no funcionales (Sánchez, 2015).

3. **Implementación**: Se construye el sistema a partir de los resultados del Análisis y Diseño (Sánchez, 2015).
4. **Pruebas internas**: Se verifica el resultado de la implementación. Se desarrollan artefactos de prueba como: diseños de casos de prueba, listas de chequeo y si es posible componentes de prueba ejecutables para automatizar las pruebas (Sánchez, 2015).

1.3 Arquitectura de software

Según el estándar IEEE 1471 "la Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el medio ambiente y los principios que orientan su diseño y evolución (IEEE, 2000)."

1.3.1 Patrones arquitectónicos

Un patrón arquitectónico es una transformación impuesta al diseño de todo un sistema. El objetivo es establecer una estructura y funcionalidad para todos los componentes del sistema. Un software o sistema de cómputo muestra uno o varios patrones arquitectónicos, compuestos por un conjunto de componentes que realizan las funciones requeridas por el sistema, un conjunto de conectores que permiten la comunicación, coordinación y cooperación entre los componentes y restricciones que definen cómo se integran los componentes para formar el sistema (Pressman, 2010).

La solución propuesta muestra la combinación del patrón arquitectónico Modelo-Vista-Controlador (MVC) y la arquitectura basada en componentes. El patrón arquitectónico MVC separa la lógica del negocio de la interfaz de usuario en tres partes diferentes. El modelo administra el comportamiento y los datos del dominio de aplicación, la vista maneja la visualización de la información y el controlador es el encargado del funcionamiento del sistema y el intermediario en la comunicación entre el modelo y la vista (Frank Buschmann, 1996).

"La arquitectura basada en componentes se trata de la descomposición del software en componentes funcionales. Es completamente modular y favorece la reutilización de todos sus elementos, debido a que el sistema será dividido completamente en componentes, facilitando un mejor entendimiento del desarrollo y logrando una mejor integración entre ellos" (Frank Buschmann, 1996). La arquitectura basada en componentes permite alcanzar un mayor nivel de reutilización de software, donde las pruebas podrán ser ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados. Además, en caso de existir un débil acoplamiento entre componentes, el desarrollador es libre de actualizar o agregarlos según sea necesario, sin afectar otras partes del sistema (Isidro Ramos Salavert, 2000).

1.3.2 Patrones de diseño

Los patrones de diseño proveen un esquema para refinar los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Describen la estructura recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto determinado. (Gamma, y otros, 1994)

Existen distintos patrones de diseño utilizados durante el proceso de desarrollo de software. Estos patrones son agrupados en dos grupos conocidos como Patrones Generales de Software para la asignación de Responsabilidades (GRASP, por sus siglas en inglés) y "La Banda de los cuatro" (GOF, por sus siglas en inglés).

Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, son una serie de buenas prácticas de aplicación recomendable en el diseño de software (Larman, 1999). Existen nueve patrones GRASP, de estos se utilizarán cinco en el desarrollo de la investigación: el experto, el creador, alta cohesión, bajo acoplamiento y el controlador.

Experto: Consiste en asignar una responsabilidad al experto en información, la clase que posee la información necesaria para cumplir con la responsabilidad (Larman, 1999).

Creador: Consiste en la asignación de responsabilidades relacionadas con la creación de objetos, permite identificar quién debe ser el responsable de la creación de nuevos objetos de una clase.

Controlador: Permite que a través de un archivo se procesen todas las peticiones de manipulación por analizar a través de un objeto de controlador único (Larman, 1999).

Alta cohesión: Es un patrón evaluativo que determina cuán relacionadas y adecuadas están las responsabilidades de una clase, de manera que no se realice un trabajo excesivo, y permita un fácil mantenimiento y un aumento de la reutilización (Larman, 1999).

Bajo acoplamiento: Es un patrón evaluativo que permite la creación de clases más independientes que reducen el impacto de los cambios, y también más reutilizables (Larman, 1999).

Patrones GOF

Los patrones de diseño GOF se clasifican en tres grupos: creacionales, estructurales y de comportamiento, y comprenden un total de 23 patrones (Gamma, y otros, 1994). A continuación se muestran los que se utilizarán en la solución.

Mediador: Es un patrón de comportamiento que define un objeto que hace de procesador central, coordinando las relaciones entre sus asociados o participantes. Permite la interacción de varios objetos, sin generar acoples fuertes en esas relaciones. Todos los objetos se comunican con un mediador y es éste quién realiza la comunicación con el resto (Gamma, y otros, 1994).

Patrón Inyección de Dependencias

Este patrón consiste en suministrar objetos a una clase en lugar de ser la propia clase quien cree los objetos, puesto que permite instanciar objetos sin tener que conocer nada sobre sus dependencias, tan solo hay que pedírselo al contenedor de dependencias. Contribuye a la velocidad y flexibilidad del marco de trabajo, pues permite estandarizar y centralizar la forma en que se construyen los objetos en la aplicación (Eguiluz, 2013).

1.4 Tecnologías, Herramientas y Lenguaje de modelado

La utilización de tecnologías y herramientas es fundamental para el desarrollo de software. La solución propuesta en esta investigación es desarrollada usando las tecnologías que a continuación se describen.

Bootstrap 3.0 es conocido como un marco de trabajo o biblioteca CSS. En sus inicios solo contenía código CSS para mejorar la experiencia de usuario, pero al tener tanto éxito en la comunidad de desarrolladores se le incorporaron funciones JavaScript apoyadas en el uso de jQuery, actualmente también incorpora código Syntactically Awesome Stylesheets (SASS por sus siglas en inglés). En la versión 2 de Bootstrap se le adiciona la característica adaptable que permite a una aplicación ajustarse fácilmente a distintas pantallas (televisores, computadoras, tabletas y teléfonos) (Bootstrap, 2015).

jQuery 1.9.1 es una biblioteca de JavaScript, permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica Ajax a páginas web (jQuery Foundation, 2015).

JavaScript 1.8.5 es un lenguaje de programación que se utiliza principalmente del lado del cliente. Es dinámico, responde a eventos en tiempo real, eventos como presionar un botón, pasar el puntero del mouse sobre un determinado texto o el simple hecho de cargar la página. JavaScript contiene una librería estándar de objetos, tales como Array, Date, y Math, y un conjunto central de elementos del lenguaje, tales como operadores, estructuras de control, y sentencias (W3C, 2012).

HTML5 es la quinta versión de HyperText Markup Language (HTML por sus siglas en inglés), es un lenguaje para escribir las páginas web. Es un lenguaje de hipertexto que permite escribir texto de forma estructurada, y que está compuesto por etiquetas, que marcan el inicio y el fin de cada elemento del documento. El lenguaje HTML es extensible, se le pueden añadir características, etiquetas y funciones

adicionales para el diseño de páginas web, generando un producto vistoso, rápido y sencillo (W3C, 2012).

CSS3 es el nivel 3 de Cascading Style Sheets (CSS por sus siglas en inglés), es un lenguaje usado para definir la presentación de un documento escrito en HTML. La idea principal de CSS es separar el contenido de la presentación de la página, aunque pueden estar juntos en el mismo documento. El nivel 3 de CSS incorpora opciones de sombras, rotación en 3 dimensiones y transiciones que hacen más atractivo el diseño (W3C, 2016).

Ajax que proviene de Asynchronous JavaScript And XML es una técnica para el desarrollo web de aplicaciones, es un código JavaScript que se ejecuta en el navegador del cliente o usuario. Su principal característica es que permite hacer múltiples llamadas al servidor web sin necesidad de recargar la página (Eguíluz, 2012).

PHP 5.5 es un acrónimo recursivo de PHP Hypertext Pre-processor (PHP por sus siglas en inglés) es un lenguaje de programación de uso general de código abierto y forma parte del software libre publicado bajo la licencia PHP. El código PHP se ejecuta del lado de servidor y fue diseñado para el desarrollo web de contenido dinámico. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas (PHP Team, 2015).

Twig 1.18.2 es conocido como el flexible, rápido y seguro motor de plantillas para PHP. Con Twig se pueden crear plantillas muy concisas y fáciles de leer, por lo que además son fáciles de entender por parte de los diseñadores web. Permite controlar los espacios en blanco generados por el código, renderizar las plantillas dentro de un entorno de ejecución seguro y controlado (llamado sandbox) y la aplicación automática del mecanismo de escape. En pruebas hechas a su rendimiento se puede destacar como el motor de plantillas más rápido del lenguaje PHP (Potencier, 2009).

Doctrine 2.0 es un mapeador objeto-relacional (ORM por sus siglas en inglés) escrito en PHP que proporciona persistencia para objetos PHP. Está por encima de la capa de abstracción a la base de datos, una de sus características es la posibilidad de escribir consultas a la base de datos a partir del tratamiento con objetos en PHP llamado Doctrine Query Language (Doctrine Team, 2006-2014).

1.4.1 Bosón como marco de trabajo

Bosón es una arquitectura de referencia en PHP (Hypertext Pre-processor por su nombre en inglés) que al haber sido desarrollada haciendo uso de las tecnologías libres, garantiza que todas las soluciones que se desarrollen sobre PHP cumplan con la versatilidad necesaria para la creación de sistemas de diferentes dominios y proporcione la capacidad de integración con un mínimo esfuerzo. Al utilizar como marco base a Symfony 2 garantiza la presencia de una comunidad mundial brindando

soporte y actualizaciones a diario, permitiendo que los equipos de desarrollo se centren en la implementación lógica de su negocio abstrayéndose de elementos con la gestión de la caché, el manejo de estructuras de datos, el registro de trazas, la seguridad de los sistemas, la gestión de excepciones, entre otros elementos (Amat, 2016).

El marco de trabajo Bosón materializa requisitos comunes de las arquitecturas base, para sistemas de gestión web a partir del desarrollo de varios componentes.

- **Componente Caché**: Permite a las aplicaciones que se están desarrollando la gestión de la caché, el almacenamiento y la recuperación de la información almacenada de forma rápida y sencilla. Permite guardar información en distintos formatos, textos planos, arreglos o tablas hash, objetos o instancias de clases, estructuras complejas como listas, pilas, colas, árboles y otras estructuras de mayor complejidad (Calás, 2014).
- **Componente Estructuras de Datos**: Homogeneiza el uso de las estructuras de datos como nodos, árboles, pilas, colas, listas y grafos, en el lenguaje de PHP; siendo de esta forma minimizar los tiempos de accesos y lograr formar efectivas de inserción y eliminación de datos en estructuras de almacenamiento (Calás, 2014).
- **Componente Excepciones**: Permite el control de forma centralizada en diferentes tipos de excepciones y tratarlas según su tipo especificado. Garantiza la internacionalización de los mensajes de las excepciones, el manejo declarativo del comportamiento, el mecanismo extensible de creación de nuevos tipos y la codificación (Calás, 2014).
- **Componente Estructuras dinámicas**: Modela y gestiona estructuras y nomencladores en toda la amalgama de probabilidades que integran un negocio (Calás, 2014).
- **Componente Trazas**: Registra y detecta trazas que son generadas por un sistema a partir de la captura de eventos por lo cual resulta útil para las auditorías (Calás, 2014).
- **Componente Integración**: Brinda un nuevo mecanismo de integración en Symfony2 haciendo uso de servicios REST. Se hace transparente para el usuario la implementación de la lógica, la creación de rutas y manejos de códigos de errores (Calás, 2014).
- **Componente Seguridad**: Provee un mecanismo de autenticación haciendo uso de estándar abierto SAML (Security Assertion Markup Lenguage por su nombre en inglés), así como un mecanismo de autorización mediante la extensión del patrón RBAC (Control de Acceso con Base de Roles). Permite además la autenticación a partir de fuentes como base de datos, servidores LDAP, entre otras (Calás, 2014).
- **Componente IUX**: Permite la creación de interfaces utilizando el proyecto de Interfaz Única (IUX), permitiendo definir estándares de diseño para cada una de las líneas temáticas de la UCI.

Proporciona a los desarrolladores la posibilidad de elegir la línea a utilizar, para así generar automáticamente elementos gráficos ajustados a la línea temática (Calás, 2014).

- **Componente Portal**: Brinda un área de trabajo única, taxonómicamente ordenada, que se integre con los mecanismos de autenticación y autorización de la plataforma y con el componente de interfaz única. Soporta las tecnologías de presentación HTML5, CSS3, JS y mecanismos de comunicación asíncrona con el servidor como Ajax (Calás, 2014).
- **Componente Aspecto**: Permite a las aplicaciones que se están desarrollando gestionar acciones y métodos (aspectos) que se ejecutarán antes o después de la ejecución de un controlador, permitiendo al usuario configurar el orden con que se realizarán estos métodos. También el servicio y método que se ejecutará para el controlador y acción. Si el método o procedimiento se debe ejecutar antes o después del controlador y acción. Además del controlador y acción en contexto (Calás, 2014).

1.4.2 Entorno de desarrollo integrado

Un entorno de desarrollo integrado o entorno de desarrollo interactivo (IDE por sus siglas en inglés) es una aplicación de software que proporciona servicios integrales a los programadores informáticos para el desarrollo de software (Microsoft, 2016).

PhpStorm 9.0 de programación desarrollado por JetBrains. Es uno de los entornos de programación más completos de la actualidad, permite editar código no sólo del lenguaje de programación php como lo indica su nombre. Proporciona un fácil autocompletado de código. Es compatible con Sistemas Operativos Windows, Linux y Mac OS (JetBrains s.r.o, 2000-2016).

1.4.3 Sistema gestor de base de datos

Un Sistema Gestor de Base de Datos es un conjunto de programas que administran y gestionan la información contenida en una base de datos. Permite la definición, control de la seguridad, privacidad y manipulación de los datos (Desarrolloweb.com, 2007).

PostgreSQL 9.2.4 es un sistema de gestión de bases de datos objeto-relacional, está distribuido bajo licencia BSD y con su código fuente se encuentra disponible libremente. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Actualmente se usa en todo el mundo y de los sistemas de gestión de bases de datos de código abierto es el más potente del mercado (PostgreSQL, 2013).

PgAdmin III 1.18.1 es un software libre para la administración de bases de datos PostgreSQL y se encuentra liberado bajo la licencia Artistic Licence. Surge como respuesta a las limitantes detectadas a las versiones anteriores de PgAdmin (pgAdmin Development Team, 2013).

1.4.4 Servidor de aplicaciones

Un servidor de aplicaciones es una tecnología básica que proporciona la infraestructura y servicios clave a las aplicaciones alojadas en un sistema. (Microsoft, 2016)

Apache 2.2.21 es un servidor para aplicaciones web, su capacidad de configuración, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa para publicar sus aplicaciones, tanto en Internet como en Intranet, también puede ser usado en equipos personales. (The Apache HTTP Server Project, 2015)

1.4.5 Herramientas CASE

Las herramientas CASE comprenden un amplio grupo de programas que se utilizan para ayudar a las actividades del proceso del software, como el análisis de requisitos, el modelado de sistemas, la depuración y las pruebas. (Sommerville, 2005)

Visual Paradigm for UML 8.0 Enterprise Edition es una herramienta CASE que soporta el ciclo de vida del desarrollo de software. Permite la generación automática de reportes, además de exportar los contenidos en diversos formatos incluyendo imágenes. Implementa una actualización automática del modelo de diseño y código permitiendo mantener la documentación de ambos modelos actualizadas con los cambios que ocurran en ambos sentidos, optimizando la descripción textual de elementos de código a partir de la descripción visual (Visual Paradigm, 2013).

Visual Paradigm utiliza UML (Unified Modeling Language o por su traducción al español Lenguaje Unificado de Modelado) para realizar el modelado del sistema. La metodología utilizada para el desarrollo del sistema define a UML como lenguaje de modelado. En el epígrafe se detallan algunas de las características de este lenguaje de modelado.

1.4.6 Lenguaje de modelado UML 2.0

UML es un lenguaje de modelado visual que se emplea para especificar, visualizar, construir y documentar artefactos de un sistema de software orientado a objetos. Captura decisiones y conocimiento sobre los sistemas que se deben construir, transformando en un lenguaje estándar los componentes del proceso de desarrollo de aplicaciones. Tiene como objetivo principal, entregar un material de apoyo que permita definir diagramas propios, como también entender la modelación de

diagramas existentes. Permite a los desarrolladores visualizar los resultados de su trabajo en esquemas o diagramas estandarizados (Jacobson, y otros, 2000).

1.5 Validación de la investigación

Las pruebas de software son el último bastión para la evaluación de la calidad y el descubrimiento de errores. Verifican que el software implemente correctamente una función específica y validan que el sistema construido corresponde con los requisitos del cliente. Una vez generado el código fuente, el software debe ser probado para descubrir y corregir, el máximo de errores posibles antes de su entrega al cliente (Pressman, 2010). En esta investigación se utilizan los métodos de pruebas de caja blanca (PCB) y pruebas de caja negra (PCN).

1.5.1 Pruebas de Caja blanca

Las PCB o pruebas estructurales se basan en un examen cercano al detalle procedimental del código a evaluar (Pressman, 2005). Para efectuar este método se aplicará la Prueba del Camino Básico, que permite obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como una guía para la definición de un conjunto básico de caminos de ejecución, garantizando con estos que durante la prueba se ejecute por lo menos una vez cada sentencia del programa (Pressman, 2005).

1.5.2 Pruebas de Caja negra

Las PCN también conocidas como pruebas de comportamiento, no son una alternativa a las técnicas de PCB, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados con las PCB. Se concentran en los requisitos funcionales del sistema, ejercitando cada requisito a partir de un conjunto de condiciones de entrada en sus valores válidos e inválidos. Las PCN intentan encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento, de inicialización y de terminación (Pressman, 2005). Para efectuar este método se utiliza la técnica partición equivalente y el análisis de valores límite (AVL).

Partición equivalente

La técnica partición equivalente divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba que descubra de forma inmediata una clase de errores que de otro modo requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (Pressman, 2005).

Análisis de valores límite

Los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el AVL como técnica de prueba. El AVL es una técnica de diseño de casos de prueba que complementa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los extremos de las clases. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida (Pressman, 2005).

1.5.3 Pruebas de aceptación

Las pruebas de aceptación o pruebas de validación son pruebas del sistema en el entorno real de trabajo con intervención del usuario final. La validación del software se logra mediante pruebas que demuestren que se cumple con los requisitos. Al construir software personalizado para un cliente se aplican pruebas de aceptación que permiten al cliente validar los requisitos (Pressman, 2010). La prueba alfa se lleva a cabo en el lugar de desarrollo pero por un cliente. Se usa el sistema de forma natural con el desarrollador como observador del usuario y registrando los errores y los problemas de uso (Pressman, 2010).

Estas pruebas se realizan bajo un ambiente controlado para determinar si el software cumple con lo requerido y establecido en la historia de usuario por el cliente. Terminadas estas pruebas y corregidas las no conformidades detectadas, el cliente avala la solución emitiendo el Certificado de Aceptación del Producto.

1.6 Conclusiones del capítulo

Al finalizar este capítulo se concluye que:

- Los problemas en el rendimiento y la desactualización de las tecnologías permitió identificar una necesidad de migración del módulo Planificación de Mantenimiento del sistema Orbita para mejorar el rendimiento del sistema en general.
- El proceso de desarrollo de la solución se regirá por la metodología AUP-UCI, además, la solución mostrará en su arquitectura una combinación del patrón arquitectónico MVC y la arquitectura basada en componente.
- Fueron seleccionadas las herramientas y tecnologías necesarias para la realización de la propuesta de solución, definiendo la arquitectura de referencia en PHP Bosón como marco de trabajo.
- La validación de la investigación se realizará mediante la ejecución de pruebas de caja blanca, caja negra y aceptación.

Capítulo 2. Propuesta de solución

Introducción

En el presente capítulo se hace una descripción del análisis y de la solución propuesta del módulo Planificación de Mantenimiento del Sistema Orbita, teniendo en cuenta las fases que presenta la metodología AUP-UCI. Se identifican y organizan las clases relevantes para las funcionalidades del sistema, así como los patrones arquitectónicos y de diseño utilizados para la realización de la aplicación. Se modela la estructura de los datos mediante el modelo de datos y se realizan las actividades de la implementación del sistema.

2.1 Estudio del módulo Planificación de Mantenimiento del Sistema Orbita existente

El Sistema Orbita, es un Sistema de Control de flota y Mantenimiento, el cual gestiona los procesos relacionados con los mantenimientos en una base de transporte. Está enfocado a la aplicación del mantenimiento preventivo y correctivo partiendo de la generación de una orden de trabajo y registrando los recursos tanto humanos como materiales utilizados en la realización de estos mantenimientos. El Sistema Orbita está compuesto por varios módulos, encargados de realizar sus funcionalidades y determinando el uso total del sistema. En la presente investigación se hace referencia al módulo Planificación de Mantenimiento específicamente.

El sistema para la gestión de grupos de vehículos cuenta con una interfaz, proporcionando a los usuarios información de dichos grupos. Muestra una serie de elementos que deben tenerse en cuenta a la hora de realizar cualquier tipo de gestión a un grupo de vehículos, contando con un conjunto de funcionalidades encargadas de dar a conocer todo lo referente a cada grupo en específico. El sistema permite una selección de propiedades, para determinar las características que pueden presentar los grupos, conjuntamente con las características se puede asociar o no un conjunto de documentos técnicos del vehículo que influyen en la planificación. Muestra además otras funcionalidades encargadas de manipular el tipo de mantenimiento que presentan, las actividades generadas para darle un seguimiento y un control a través de una planificación a esos grupos de vehículos. Además brinda la posibilidad de especificar los tipos de fallas que pueden existir en un grupo de vehículos al ser creado.

2.2 Requisitos

Un requisito es una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de este (Sommerville, 2005). Puede dividirse en dos categorías:

Requisitos Funcionales: Definen acciones o funciones que el sistema debe ser capaz de realizar, describen transformaciones que el sistema realiza sobre las entradas para generar salidas (Sommerville, 2005).

Requisitos No Funcionales: Características que de una forma u otra pueden limitar el sistema, como por ejemplo rendimiento, fiabilidad y seguridad (Sommerville, 2005).

2.2.1 Requisitos funcionales

Durante la presente investigación se analizaron 28 Requisitos Funcionales (RF), a continuación se nombran y describen.

Tabla 1: Requisitos funcionales del sistema.

No	Nombre	Descripción	Prioridad para el cliente	Complejidad de implementación
RF 1	Adicionar grupo de vehículos.	Se adiciona un grupo de vehículos al Sistema Orbita.	Alta	Baja
RF 2	Modificar grupo de vehículos.	Se modifica un grupo de vehículos existente en el Sistema Orbita.	Media	Baja
RF 3	Eliminar grupo de vehículos.	Se elimina un grupo de vehículos existente en el Sistema Orbita.	Media	Baja
RF 4	Imprimir grupo de vehículos.	Se imprime un grupo de vehículos existente en el Sistema Orbita	Media	Baja
RF 5	Buscar grupo de vehículos.	Se busca un grupo de vehículos existente en el Sistema Orbita.	Media	Baja
RF 6	Adicionar propiedades 1.	Se adiciona una propiedad 1 al grupo de vehículos.	Alta	Baja

RF 7	Adicionar propiedades 2.	Se adiciona una propiedad 2 al grupo de vehículos.	Alta	Baja
RF 8	Adicionar valor.	Se adiciona un valor al grupo de vehículos.	Alta	Baja
RF 9	Modificar valor.	Se modifica el valor al grupo de vehículos.	Media	Baja
RF 10	Eliminar valor.	Se elimina el valor al grupo de vehículos.	Media	Baja
RF 11	Modificar propiedades.	Se modifica una propiedad al grupo de vehículos.	Media	Baja
RF 12	Eliminar propiedades 1.	Se elimina una propiedad 1 al grupo de vehículos.	Media	Baja
RF 13	Eliminar propiedades 2.	Se elimina una propiedad 2 al grupo de vehículos.	Media	Baja
RF 14	Asociar documentos técnicos.	Se asocia un documento técnico al grupo de vehículos.	Alta	Baja
RF 15	Quitar documentos técnicos.	Se quita un documento técnico al grupo de vehículos.	Media	Baja
RF 16	Descargar documentos técnicos.	Se descarga un documento técnico al grupo de vehículos.	Media	Baja
RF 17	Adicionar tipos de mantenimientos.	Se adiciona un tipo de mantenimiento al grupo de vehículos.	Alta	Baja

RF 18	Modificar tipos de mantenimientos.	Se modifica un tipo de mantenimiento al grupo de vehículos.	Media	Baja
RF 19	Eliminar tipos de mantenimientos.	Se elimina un tipo de mantenimiento al grupo de vehículos.	Media	Baja
RF 20	Incluir tipos de mantenimientos.	Se incluye un tipo de mantenimiento al grupo de vehículos.	Alta	Baja
RF 21	Adicionar actividades.	Se adiciona una actividad al grupo de vehículos.	Alta	Baja
RF 22	Asociar repuesto.	Se adiciona un repuesto al grupo de vehículos.	Alta	Baja
RF 23	Quitar repuesto.	Se quita un repuesto al grupo de vehículos.	Media	Baja
RF 24	Modificar actividades.	Se modifica una actividad al grupo de vehículos.	Media	Baja
RF 25	Eliminar actividades.	Se elimina una actividad al grupo de vehículos.	Media	Baja
RF 26	Adicionar tipo de fallas.	Se adiciona una falla al grupo de vehículos.	Alta	Baja
RF 27	Modificar tipo de fallas.	Se modifica una falla al grupo de vehículos.	Media	Baja
RF 28	Eliminar tipo de fallas.	Se elimina una falla al grupo de vehículos.	Media	Baja

Para describir el funcionamiento de los requisitos funcionales del sistema, se realiza el artefacto Historias de Usuarios. A continuación se muestra la descripción de los requisitos Adicionar grupo de vehículos y Modificar grupo de vehículos. En el (Anexo 3) se muestran las historias de usuarios de todos los requisitos del sistema.

Tabla 2: Historia de Usuario Adicionar grupo de vehículos.

Número: 1	**Nombre del requisito:** Adicionar grupo de vehículos.
Programador: Yordan Bandera Rodrígue	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 10 horas

Descripción:

El sistema debe brindar la posibilidad al usuario de gestionar grupos de vehículos determinados. La primera vista **Gestionar grupo de vehículos** contendrá los campos **Nombre** el cual representa el nombre, la marca y el modelo de ese grupo de vehículos, el segundo campo será el **Régimen** identificado por 1 o por 2. Además, en la primera vista se muestra un **Buscar** este elemento permitirá realizar la búsqueda de cualquier campo de los antes mencionados introduciendo el nombre, o la marca, o el modelo, o el tipo de vehículo, con sus campos correspondientes, además esta vista muestra las funcionalidades **Adicionar, Modificar, Eliminar e Imprimir**.

La segunda vista **Adicionar grupo de vehículos** permitirá al usuario añadir un grupo de vehículo determinado. La misma contará con los campos **Nombre** (se rellena solo con el tipo de vehículo, la marca y el modelo), el segundo campo será **Tipo de vehículo** (seleccionar algún elemento que se encuentra dentro de las opciones que se dan a conocer), el tercer campo **Marca** (cadena de caracteres que solo admite letras)**,** el cuarto campo será **Modelo** (cadena de caracteres que solo admiten letras y números), el quinto campo será el **Régimen de mantenimiento** el mismo admitirá la selección de escoger el régimen de entre 1 y 2 y por último el campo de **Unidad de Medida** (seleccionar algún elemento que se encuentra dentro de las opciones que se dan a conocer),. Cuenta con varias pestañas las cuales se le asigna a cada grupo de vehículos los elementos que se le pueden adicionar de cada pestaña, además esta vista cuenta con tres botones para realizar la acción ya sea de **Cancelar, Aplicar** o **Aceptar** la nueva solicitud de gestionar un grupo de vehículos que se esté creando.

Observaciones:

Si no se activa uno de los componentes de selección, si no se rellenan los componentes de marca y modelo y no se selecciona un tipo de vehículo, no se puede completar la operación. En el caso de activar lectura y no seleccionar un tipo de unidad, entonces no se puede completar la operación.

Prototipo elemental de interfaz gráfica de usuario:

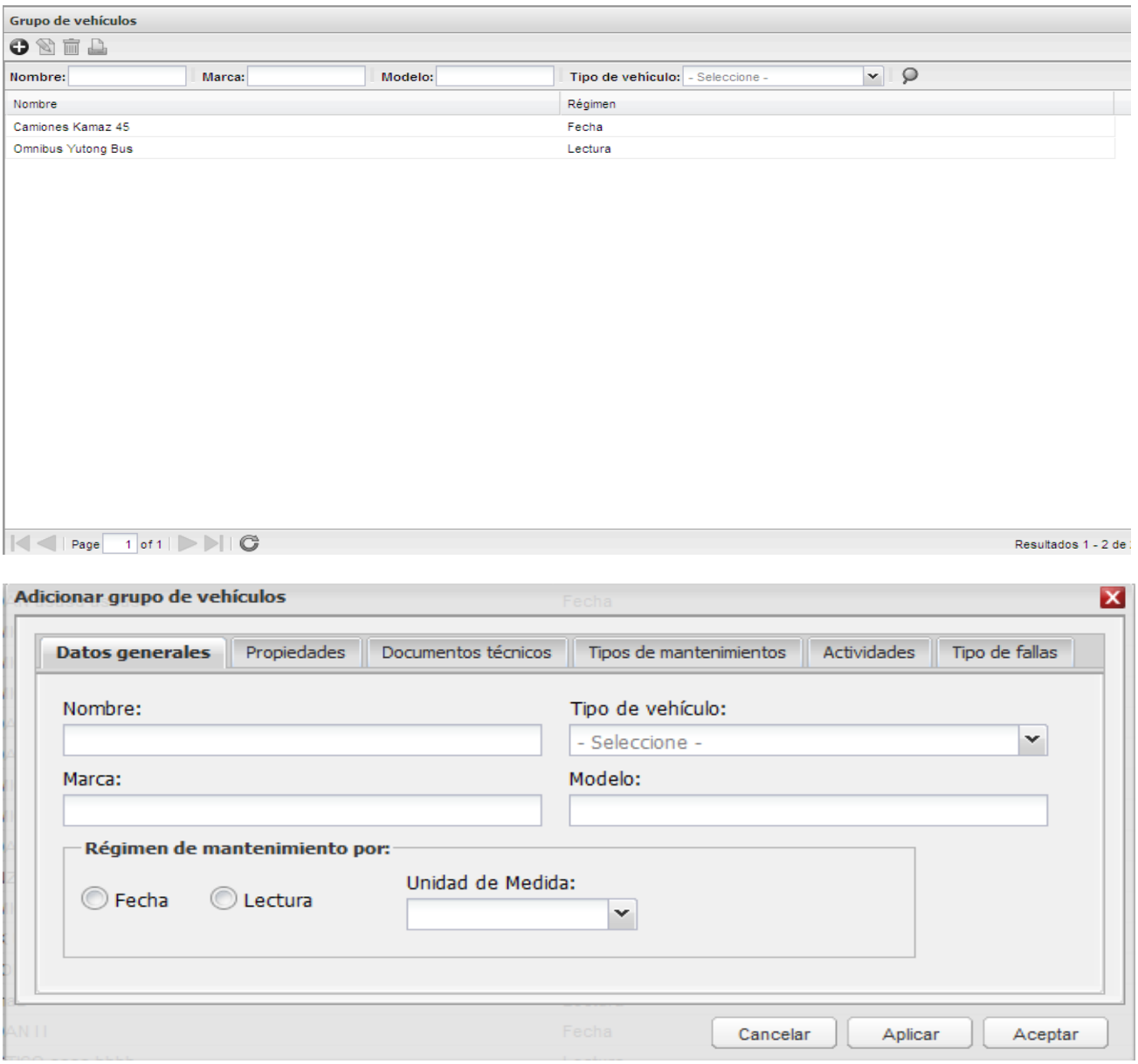


Tabla 3: Historia de Usuario Modificar grupo de vehículos.

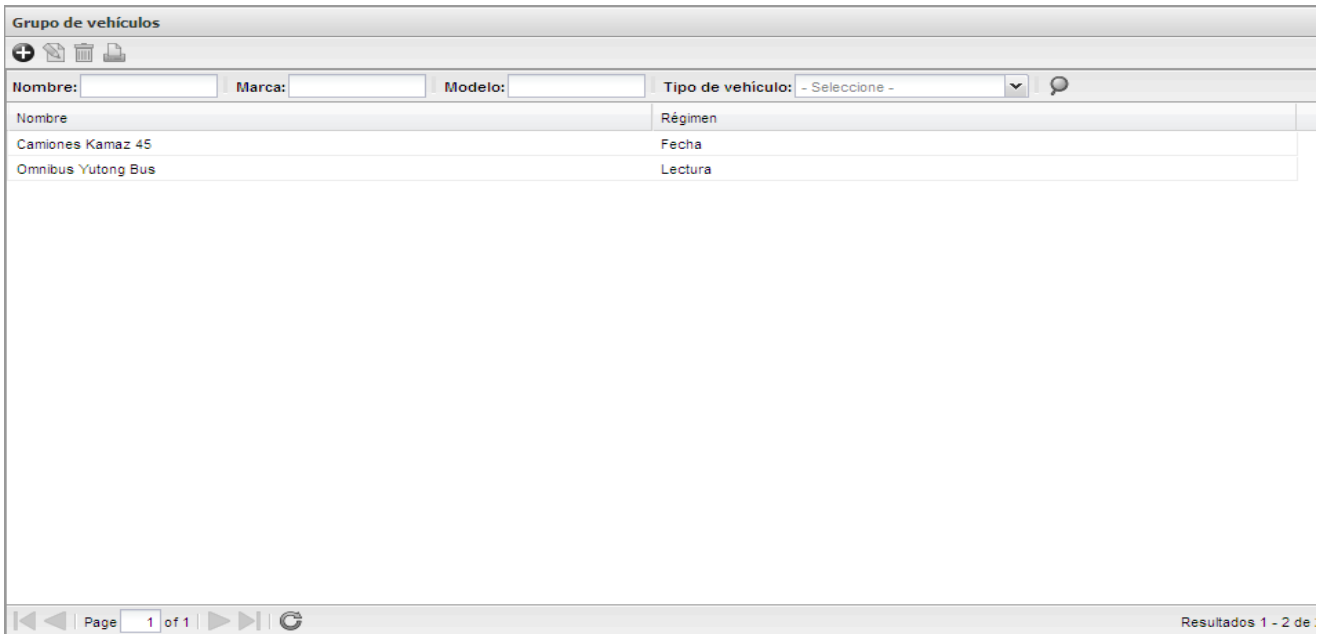
Número: 2	**Nombre del requisito:** Modificar grupo de vehículos.
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 6 horas
Descripción:	

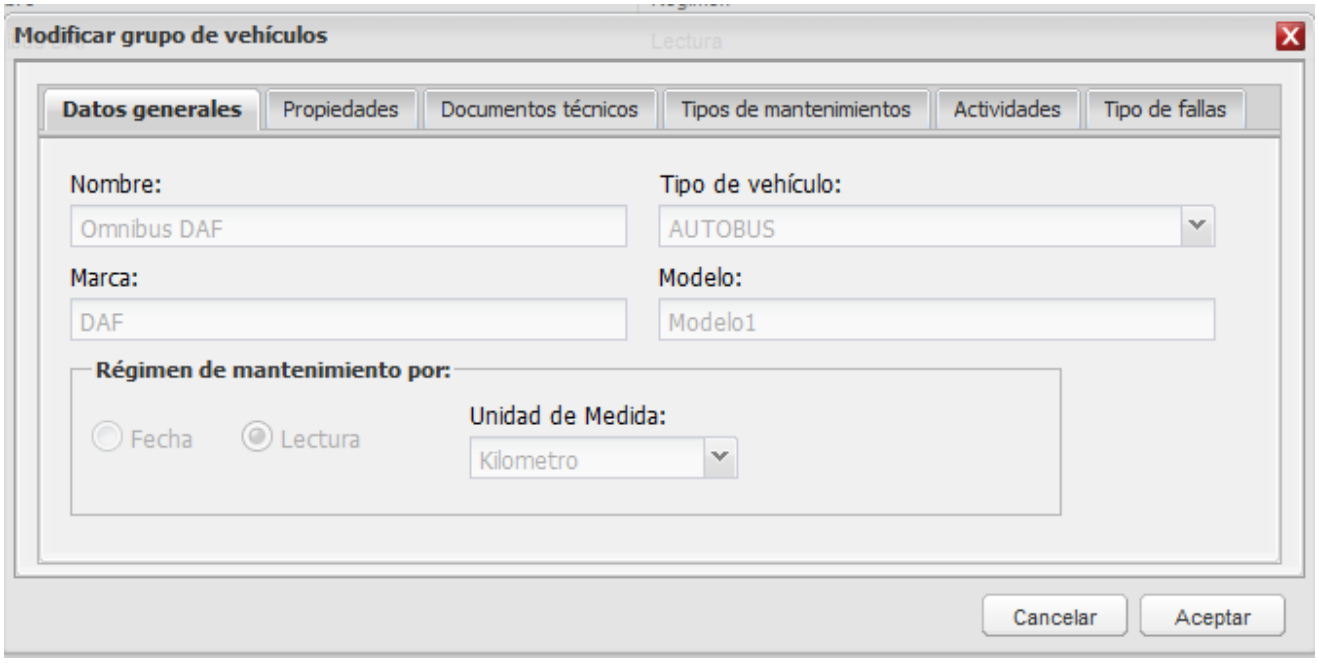
Permite modificar los datos de un grupo de vehículos en específico mostrando los mismos campos de la historia de usuario 1 con los datos originales.

Observaciones:

Si deja alguno de los campos en blanco, el sistema se lo marcara en rojo.

Prototipo elemental de interfaz gráfica de usuario:





2.2.2 Requisitos no funcionales

Durante la presente investigación se definen cinco Requisitos No Funcionales (RNF) que cumplen con las características de usabilidad, eficiencia, soporte, seguridad y portabilidad.

Usabilidad: El sistema podrá ser usado por personas con conocimientos básicos en el manejo de computadoras, sin necesidad de tener conocimiento de la forma en que se realizan los procesos que maneja el sistema.

Eficiencia: El promedio de las peticiones que se realizan al servidor no deben ser mayor de 3 segundos y en el caso de información que involucre consultas a las bases de datos los tiempos de respuestas no deben exceder los 10 segundos.

Soporte: La aplicación debe estar bien documentada y proveer el código fuente, previendo futuras modificaciones o migraciones en el mismo para potenciar su alcance o eficiencia.

Seguridad: La seguridad está a nivel de gestión de roles con el fin de mantener la integridad de los datos, por el cual el acceso a la herramienta será a través de estos roles, trayendo consigo además la protección de la información.

Portabilidad: El sistema debe ser multiplataforma con enfoque en tecnologías basadas en software no propietario.

2.3 Análisis y diseño

Durante el desarrollo de esta disciplina se modela el sistema, teniendo en cuenta la arquitectura, para que soporte todos los requisitos tanto funcionales como no funcionales. De este modo se propicia el desarrollo de una arquitectura sólida para el sistema y la adaptación del diseño para que sirva de base a la etapa de implementación. (OpenUP Copyright -Eclipse, 2011)

2.3.1 Diseño arquitectónico

El sistema a desarrollar muestra la combinación del patrón arquitectónico MVC y la arquitectura basada en componentes. A continuación se evidencia el patrón MVC en la estructura por carpetas que propone Symfony 2 como marco base de la arquitectura Bosón.

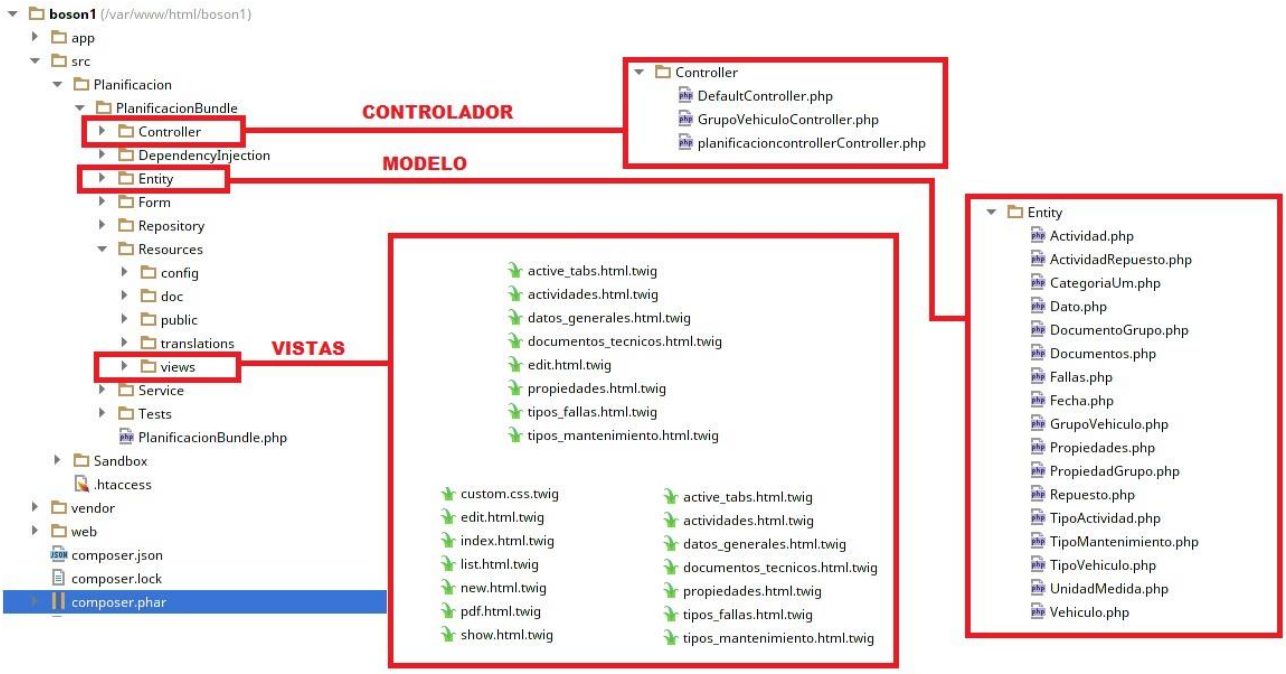


Figura 1: Patrón arquitectónico MVC en la estructura por carpetas propuesta por Symfony 2 como marco base de la arquitectura Bosón.

1. Todas las clase pertenecientes al modelo se pueden encontrar en el directorio /src/Planificacion/PlanificacionBundle/Entity. Esta carpeta contiene las entidades que representan las tablas de la base de datos. También contiene las clases Repository que es donde se encuentran las consultas que el desarrollador realiza a la base de datos.

2. Las vistas ubicadas en el directorio /src/Planificacion/PlanificacionBundle/Resources/views, conforman las vistas del sistema. Dentro de esta dirección se encuentra la vista frontal index.html.twig que contiene la estructura general de la vista frontal. A través de las vistas es mostrada la información para que el usuario interactúe con el sistema.

3. En el directorio /src/Planificacion/PlanificacionBundle/Controller se encuentra la clase controladora. Esta recibe las solicitudes o peticiones realizadas por el usuario desde la vista, ejecutando las acciones necesarias y devuelven una respuesta a la interfaz con los resultados de las operaciones realizadas.

2.3.2 Diseño de clases

La realización del diagrama de clases permite obtener de forma estática la representación de los requisitos a través de las clases del sistema y sus relaciones. Se emplea como una entrada fundamental para las actividades de implementación.

La siguiente figura muestra el diagrama de clases del diseño para el requisito Adicionar grupo de vehículos, en el que se reflejan las clases relacionadas con Grupo Vehículo.

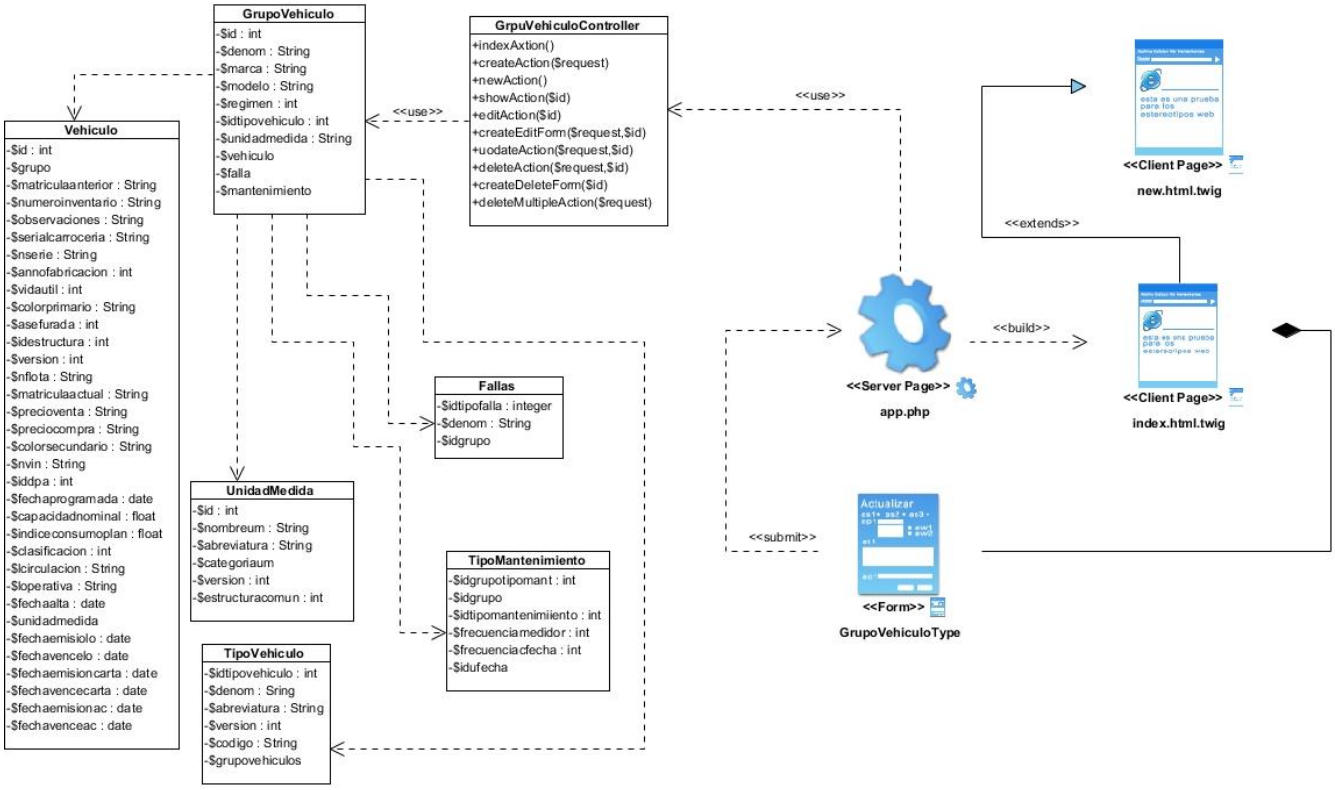


Figura 2: Diagrama de clases del diseño del requisito Adicionar grupo de vehículos.

Las clases Vehículo, UnidadMedida, TipoVehículo, Fallas, TipoMantenimiento y GrupoVehículo son las que almacenan la información necesaria para crear un grupo de vehículos. La clase Grupo Vehículo Controller es la encargada de gestionar la información contenida en las clases antes mencionadas. El controlador principal app.php construye la página index.html.twig que extiende de new.html.twig y contiene el formulario Grupo Vehículo Type, que es mediante el cual el usuario introduce los datos necesarios para crear un grupo de vehículos.

2.3.3 Patrones de diseño

A continuación se evidencia el uso de los patrones de diseño en las clases del sistema.

Patrones GRASP

Experto: Se pone en práctica en la clase GrupoVehiculoController.php, que es la que contiene la información necesaria para cumplir la responsabilidad de generar el listado, las modificaciones, las creaciones y las eliminaciones de los grupos de vehículos que se encuentran en el sistema.

Creador: Se refleja en la clase controladora ubicada en el directorio /src/Planificacion/PlanificacionBundle/Controller, donde se encuentran las acciones definidas para las operaciones lógicas del negocio referentes a las entidades y se ejecutan cada una de ellas.

Controlador: Permite que a través de un archivo se procesen todas las peticiones de manipulación por analizar a trasvés de un objeto de controlador único.

Alta cohesión: Se evidencia al definir una clase controladora para las entidades, donde maneja la información perteneciente a las entidades, de esta forma las responsabilidades están fuertemente ligadas a las entidades, en un sentido lógico, permitiendo que estas clases sean fáciles de comprender, reutilizar y conservar.

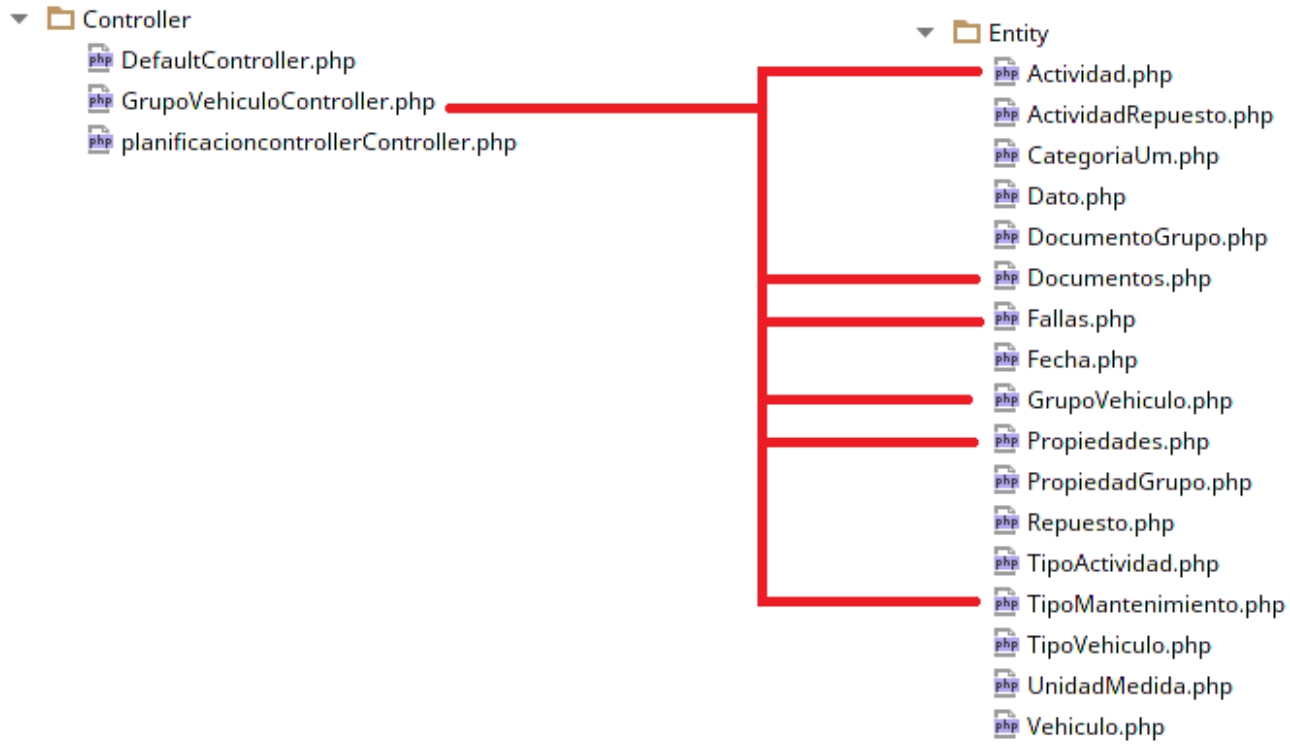


Figura 3: Ejemplo de uso del patrón Alta Cohesión.

Bajo acoplamiento: El uso de este patrón en el diseño facilita la existencia solo de las relaciones entre clases necesarias, de forma tal que pequeños cambios tengan la mínima repercusión en otras clases. También permite que las clases sean fáciles de entender por separado y de reutilizar.

Patrones GOF

Mediador: Se ve reflejado en la clase GrupoVehiculo, permitiendo relaciones de uno a muchos entre las clases Vehiculo, Fallas y TipoMantenimiento, y una relación de muchos a uno entre las clases

TipoVehiculo y UnidadMedida. La clase GrupoVehicuolo funciona como objeto mediador, coordinando la comunicación entre objetos de distintas clases.

Patrón Inyección de Dependencias: Se puede apreciar en el objeto especial entitymanager (em) a través del cual Doctrine2 realiza la manipulación de la información (modificar, eliminar e insertar registros en las tablas) sin que el programador tenga que escribir sentencias SQL.

2.3.4 Mapeo de la Base de Datos

Para la migración del módulo Planificación de Mantenimiento es necesario conservar todos los datos que han sido creados con el uso del sistema actual, por lo que se necesita la misma base de datos del Sistema Orbita. La primera acción a realizar en este caso es mapear todas las tablas utilizando los mecanismos de Symfony2 mediante la librería Doctrine 2.0.

Se mapearon un total de 14 tablas, utilizando el formato anotación, a su vez fueron creadas las Entitys con sus respectivos ficheros Repository que son los encargados de hacer las consultas. Con la nueva versión del ORM Doctrine 2.0 el funcionamiento interno transita desde las clases controladoras, accediendo a las entidades, hasta los ficheros Repository. En este archivo se establece una conexión a la base de datos, se realizan las peticiones y se envían los objetos necesarios o que fueron solicitados (Eguiluz, 2014).

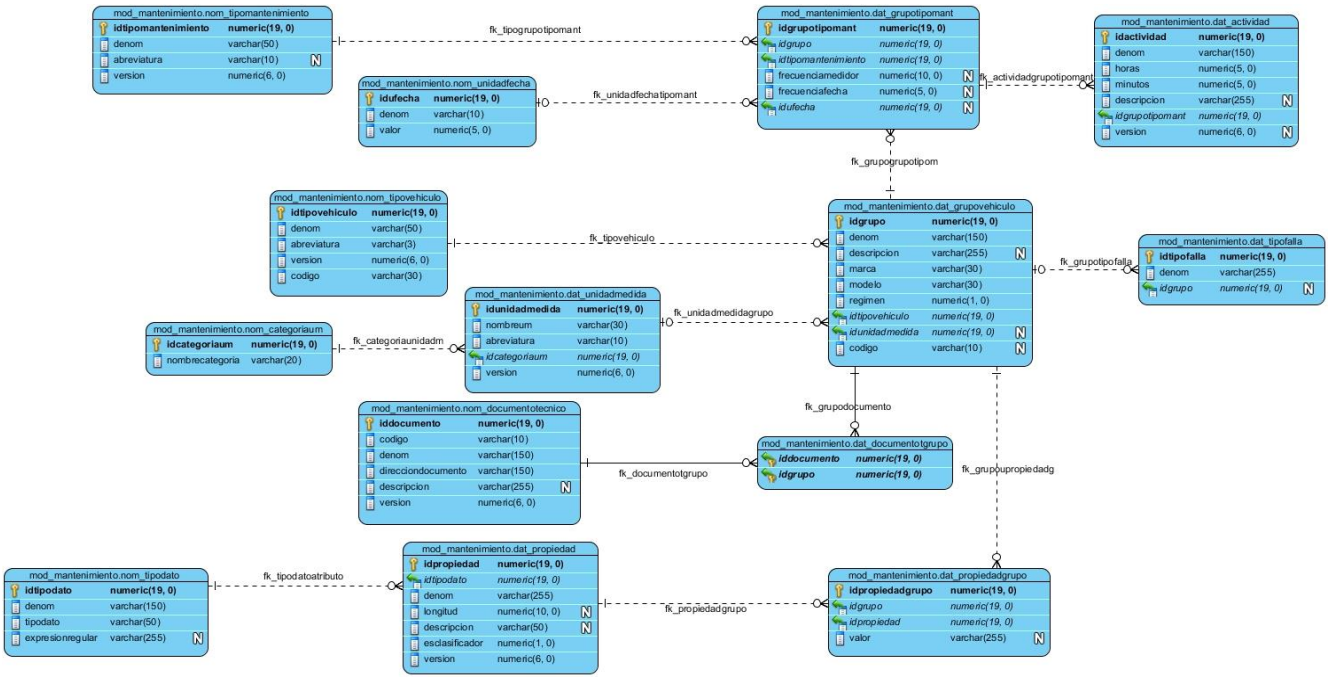


Figura 4: Modelo de datos del módulo Planificación de Mantenimiento del Sistema Orbita.

Con el fin de marcar una clase para ser persistida en la base de datos, se utiliza la anotación @ORM\Entity. Por defecto la entidad será persistida con el mismo nombre de la clase, pero no puede modificar utilizando la anotación @ORM\Table.

```
/**
 * Actividad
 *
 * @ORM\Table(name="mod_mantenimiento.dat_actividad")
 * @ORM\Entity(repositoryClass="Planificacion\PlanificacionBundle\Repository\ActividadRepository")
 */
```

Figura 5: Ejemplo del uso de la anotación @ORM\Entity en la clase Actividad.php.

Las instancias de Actividad serán persistidas en una tabla llamada dat_actividad en el esquema mod_mantenimiento en la base de datos.

Cada clase Entity necesita una clave primaria que identifique a la entidad. Se designa la propiedad que se usará como identificador con la notación @ORM\Id. Si es un valor generado automáticamente se utiliza la anotación @ORM\GeneratedValue y como estrategia SEQUENCE a través de @ORM\SequenceGenerator.

```
    /**
     * @var integer
     *
     * @ORM\Column(name="idactividad", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     * @ORM\SequenceGenerator(sequenceName="mod_mantenimiento.sec_actividad_seq", allocationSize=1, initialValue=1)
     */
```

Figura 6: Ejemplo del uso de las anotaciones para definir una clave primaria.

Para que exista una relación donde se obtenga una única instancia de un objeto se utilizan @OneToMany y @JoinColumn. Doctrne 2.0 soporta las relaciones uno-a-uno, uno-a-muchos, muchos-a-uno y muchos-a-muchos.

```
/**
 *
 * @ORM\ManyToOne(targetEntity="Planificacion\PlanificacionBundle\Entity\GrupoVehiculo", inversedBy="falla", cascade="persist")
 * @ORM\JoinColumn(name="idgrupo", referencedColumnName="idgrupo")
 */
```

Figura 7: Ejemplo del uso de anotacio0nes para determinar relaciones entre clases.

La manipulación de la información de Doctrine 2.0 se realiza a través de un objeto especial llamado getManager. El uso de un ORM es una alternativa efectiva a la hora de trasladar el modelo conceptual al esquema relacional nativo de la base de datos SQL.

2.4 Implementación

A partir de los resultados del análisis y el diseño se realiza la implementación del sistema, generándose el modelo de implementación como artefacto de esta disciplina. Este modelo está conformado por el modelo de componentes y el modelo de despliegue.

2.4.1 Modelo de componentes

“Un componente es una unidad física de implementación con interfaces bien definidas pensada para ser utilizada como parte reemplazable de un sistema. Cada componente incorpora la implementación de ciertas clases del diseño del sistema” (Jacobson, y otros, 2000). Pressman lo define como “un elemento funcional de un programa que incorpora la lógica del procesamiento, las estructuras internas de los datos para implementar dicha lógica, y una interfaz que permita la invocación del componente y el paso de los datos” (Pressman, 2010).

En la siguiente figura se muestra el modelo de componentes del módulo Planificación de Mantenimiento del Sistema Orbita. En la misma se puede apreciar que el componente GrupoVehiculoController es el encargado de facilitar todos los datos relacionados con los grupos de vehículos existentes en la Base de Datos.

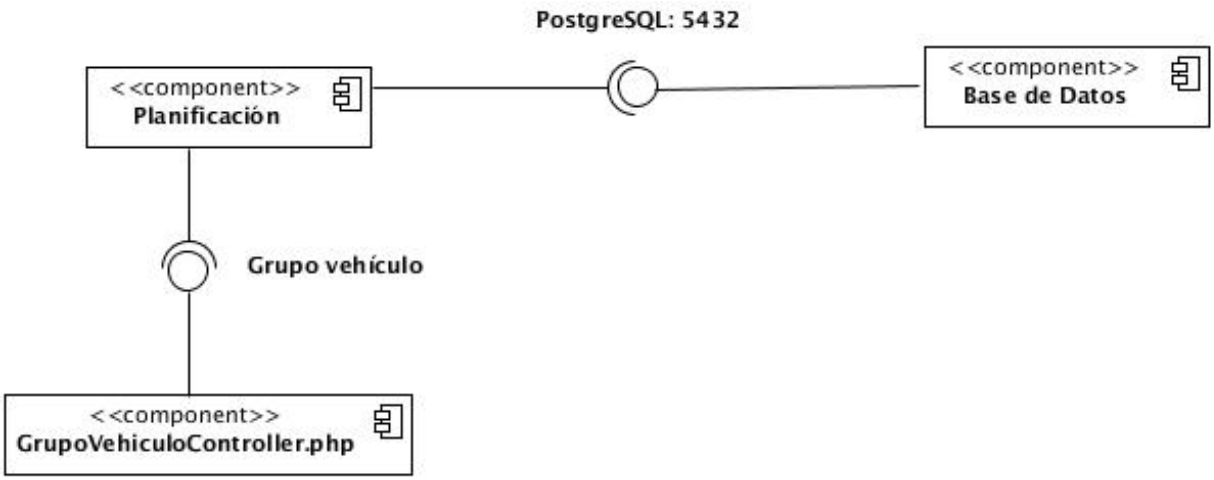


Figura 8: Modelo de componentes del módulo Planificación de Mantenimiento del Sistema Orbita.

2.4.2 Modelo de despliegue

El modelo de despliegue se realiza como parte de la implementación para describir la distribución física del sistema. Establece la correspondencia entre la arquitectura lógica, los procesos y nodos. Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware similar. Los nodos poseen relaciones que representan medios de comunicación entre ellos (Jacobson, y otros,

2000). Se refleja en este artefacto los protocolos de comunicación mediante los cuales se comunican los nodos respectivos.

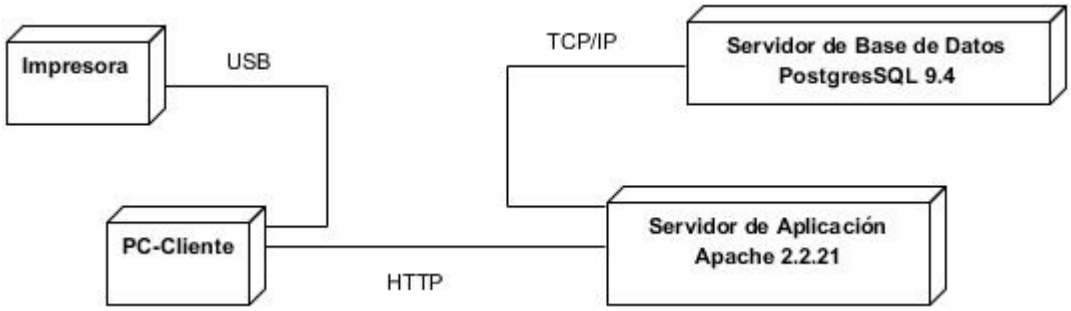


Figura 9: Modelo de despliegue del módulo Planificación de Mantenimiento del Sistema Orbita.

PC cliente: es desde donde se accede al sistema a través del navegador web Mozilla Firefox v30.0 o superior, Chrome v35.0 o superior, Opera v12.0 o superior sin importar el sistema operativo.

Servidor de aplicación: es donde radica la lógica de negocio de la aplicación, compuesto por el Servidor Web Apache 2.2.21 y como lenguaje PHP 5.5.

Servidor de base de datos: utilizando PostgreSQL en su versión 9.2.4 como sistema gestor de base de datos es el encargado de almacenar los datos del sistema.

2.5 Conclusiones del capítulo

Al finalizar este capítulo se concluye que:

- Fueron identificadas las funcionalidades del sistema mediante el estudio realizado al sistema existente.
- La realización de las historias de usuarios permitió definir la especificación de requisitos logrando un mayor entendimiento de las necesidades específicas del cliente.
- Se obtuvo el diseño del sistema a partir de la definición de los patrones de diseño y el mapeo de la base de datos.
- Se obtuvo el nuevo módulo Planificación de Mantenimiento del Sistema Orbita a partir de la implementación de los componentes identificados.

Capítulo 3. Validación de la investigación

Introducción

En este capítulo se exponen los resultados obtenidos mediante las pruebas realizadas al sistema para garantizar el cumplimiento del objetivo general planteado, se realizan las validaciones de los resultados para así reducir al máximo los errores detectados. Las pruebas y validaciones se llevan a cabo en cada una de las etapas de desarrollo para identificar a tiempo las imperfecciones e irregularidades y proporcionar una visión objetiva de la madurez y calidad de los procesos asociados.

3.1 Prueba de Caja blanca

Para desarrollar este tipo de prueba se utilizó la técnica del camino básico, cuyo objetivo principal se enmarcó en probar que todos los caminos del código están correctos. Fue necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se muestra el método deleteMultipleAction al cual se le realiza el grafo.

Método deleteMultipleAction

```
public function deleteMultipleAction(Request $request){
    $em = $this->getDoctrine()->getManager();
    $choices = $request->request->get('choose');
    foreach ($choices as $id => $value){
        $entity = false;
        $valor = $em->getRepository('PlanificacionBundleropiedades')->find($value);
        if ($entity){
            $em->remove($entity);
            $em->flush();
        }elseif($valor){
            $em->remove($valor);
            $em->flush();
            return $this->redirect($this->generateUrl('planificacion_propiedades'));
        }else {
           throw $this->createNotFoundException('Unable to find');
        }
    }
    return $this->redirect($this->generateUrl('planificacion'));
}
```

La complejidad ciclomática es una métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición. El camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente (Pressman, 2010). A continuación se muestra el grafo de flujo asociado al método deleteMultipleAction.

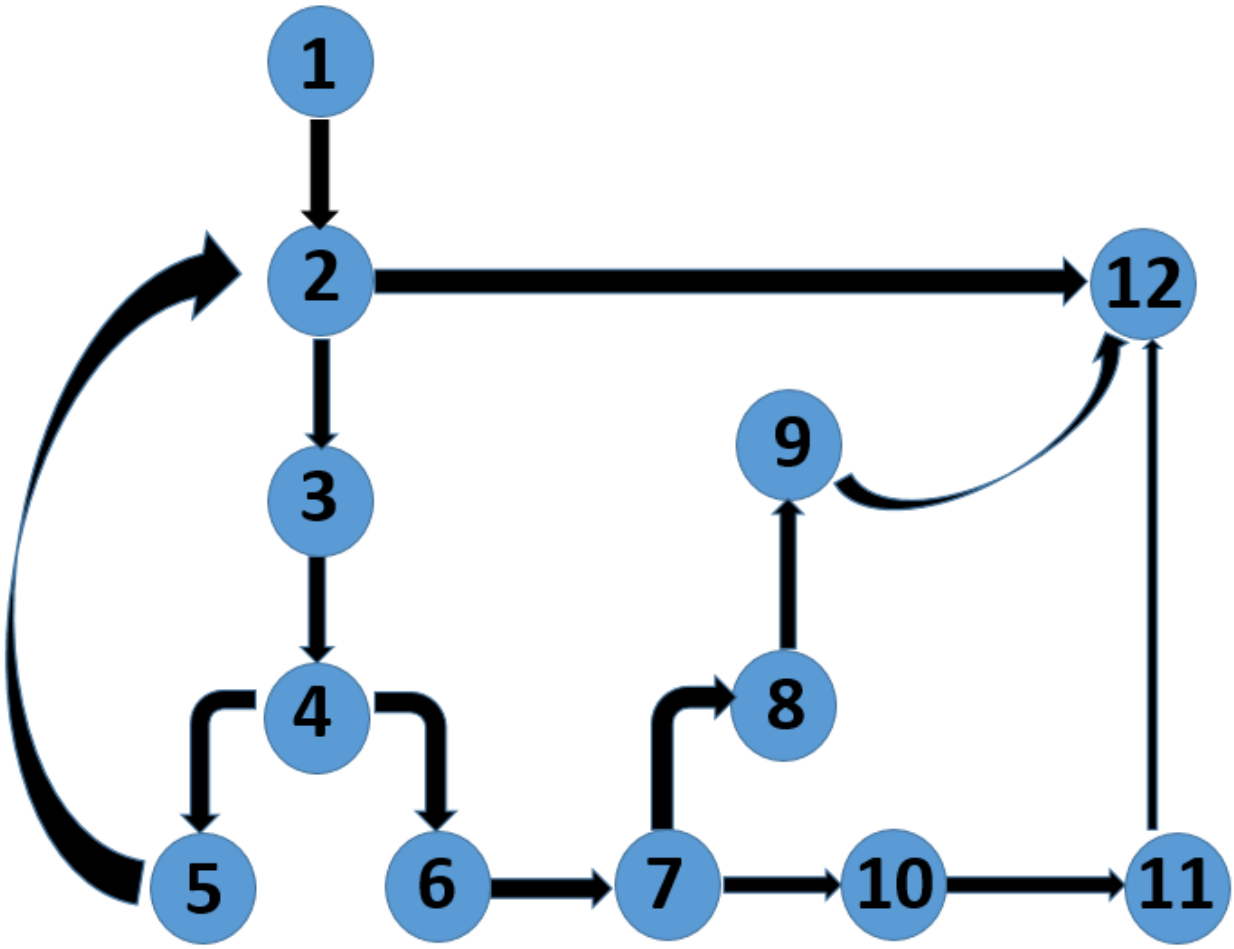


Figura 10: Ejemplo de la técnica Camino Básico aplicada al método deleteMultipleAction.

Cálculo de la complejidad ciclomática

V(G) = 4 regiones

V(G) = 14 aristas – 12 nodos + 2 = 4

V(G) = 3 nodos predicados + 1 = 4

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es de 4. Esto significa que existen 4 posibles caminos por donde el flujo puede circular.

Camino 1: 1 – 2 – 3 – 4 – 5 – 2 – 12

Camino 2: 1 – 2 – 3 – 4 – 6 – 7 – 8 – 9 – 12

Camino 3: 1 – 2 – 3 – 4 – 6 – 7 – 10 – 11 – 12

Camino 4: 1 – 2 – 12

Luego de aplicar la técnica del camino básico se constató que el método deleteMultipleAction se comportó de la manera esperada ya que los datos se corresponden y se mostraron los resultados de una manera correcta.

3.2 Pruebas de Caja negra

3.2.1 Partición equivalente

Para la ejecución de esta técnica se realiza un conjunto de diseños de casos de pruebas (DCP), cuyo principal objetivo es comprobar que el sistema se encuentra de acorde a los requisitos. Un caso de prueba incluye un conjunto de entradas, condiciones de ejecución y resultados esperados para conseguir un objetivo particular o condición de prueba, por ejemplo verificar el cumplimiento de un requisito específico (Aristegui, 2010). A continuación se exponen los DCP de las funcionalidades Adicionar Grupo de Vehículos y Adicionar Falla, especificando los datos de entrada, los resultados esperados y los resultados obtenidos.

Tabla 4: DCP Adicionar grupo de Vehículos.

Escenario	Descripción	Nombre	Tipo de vehículo	Marca	Modelo
EC 1.1: Adicionar grupo de vehículo.	Se adiciona correctamente un grupo de vehículo.	V	V	V AUDI, FIAT	V B45
EC 1.2: Adicionar grupo de vehículo.	Se aplica correctamente un grupo de vehículo.	V	V	V AUDI, FIAT	V B45

EC 1.3: Datos incompletos	Se muestra un mensaje indicando que no se han introducido los datos obligatorios.	V	NA	V AUDI, FIAT	V B45
		V	V	I Vacío	V B45
		V	V	V AUDI, FIAT	I Vacío
		V	V	V AUDI, FIAT	V B45
		V	V	V AUDI, FIAT	V B45
		V	V	V AUDI, FIAT	V B45
EC 1.4: Datos incorrectos.	Se muestra un mensaje de error indicando que se han introducido datos incorrectos.	V	NA	I {/@#~½	I {/@#~½
EC 1.5: Cancelar.	Cancela la acción de Adicionar grupo de vehículo y retorna a la página que le dio origen.	NA	NA	NA	NA

Fecha	Lectura	Unidad de Medida	Respuesta del Sistema	Flujo Central
V	V	V	Retorna a la página que le dio origen a la acción y se muestra un mensaje de que	1. Seleccionar del menú la opción “Adicionar”. 2. Introducir los datos solicitados.

			se ha creado el grupo correctamente.	3. Seleccionar la opción “Aceptar”. 4. Seleccionar la opción “Cancelar” del mensaje.
V	V	V	Se queda en la misma página y se muestra un mensaje de que se ha creado el grupo correctamente.	1. Seleccionar del menú la opción “Adicionar”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Aplicar”. 4. Seleccionar la opción “Cancelar” del mensaje.
V	V	V	Muestra un mensaje indicando que faltan datos por introducir y señala los datos que faltan por introducir. Los datos obligatorios son: Tipo de vehículo, Marca, Modelo, en caso de elegir Fecha o Lectura y Unidad de medida en caso de que se escoja Fecha.	1. Seleccionar del menú la opción “Adicionar”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Aplicar” o “Aceptar”. 4. Seleccionar la opción “Cancelar” del mensaje.
V	V	V		
V	V	V		
NA	V	V		
V	NA	V		
V	V	NA		
NA	NA	NA	Muestra un mensaje indicando que existen datos incorrectos, señala los datos que están incorrectos. Los datos se mantienen para que el usuario los modifique si lo desea.	1. Seleccionar del menú la opción “Adicionar”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Aplicar” o “Aceptar”. 4. Seleccionar la opción “Cancelar” del mensaje.
NA	NA	NA	Cancela la acción de Adicionar grupo y retorna a la página que le dio origen.	1. Seleccionar del menú la opción “Adicionar”. 2. Introducir los datos

				solicitados. 3. Seleccionar la opción “Cancelar”.

Tabla 5: Descripción de las variables del DCP Adicionar grupo de Vehículos.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto	Si	Texto que se rellena con el tipo de vehículo, la marca y el modelo.
2	Tipo de vehículo	Lista de selección	No	Permite una de las opciones del listado.
3	Marca	Campo de texto	No	Texto que admite solo letras mayúsculas y no admite espacios en cualquier lugar.
4	Modelo	Campo de texto	No	Texto que admite solo letras mayúsculas, números y no admite espacios en cualquier lugar.
5	Fecha	Campo de selección	Si	Permite seleccionarse.
6	Lectura	Campo de selección	Si	Permite seleccionarse.
7	Unidad de Medida	Lista de selección	No	Permite una de las opciones del listado.

Tabla 6: DCP Adicionar fallas.

Escenario	Descripción	Tipo de falla	Respuesta del sistema	Flujo central
EC 1.1: Adicionar falla.	Se adiciona correctamente una falla.	V Sin combustible	Retorna a la página que le dio origen a la acción y se muestra un mensaje de que se ha creado la falla correctamente.	1. Seleccionar del menú la opción “Adicionar”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Aceptar”. 4. Seleccionar la opción “Cancelar” del mensaje.
EC 1.2: Adicionar falla.	Se aplica correctamente una falla.	V Sin combustible	Se queda en la misma página y se muestra un mensaje de que se ha creado la falla correctamente.	1. Seleccionar del menú la opción “Adicionar”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Aplicar”. 4. Seleccionar la opción “Cancelar” del mensaje.

EC 1.3: Datos incompletos	Se muestra un mensaje indicando que no se han introducido los datos obligatorios.	I Vacío I Sin combustible	Muestra un mensaje indicando que falta dato por introducir o que se debe comenzar por mayúscula y señala el dato que falta por introducir o el dato incorrecto. El dato obligatorio es: Tipo de falla.	1. Seleccionar del menú la opción “Adicionar”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Aceptar” o “Aplicar”.
EC 1.4: Datos incorrectos.	Se muestra un mensaje de error indicando que se han introducido datos incorrectos.	I Sin combustible @#$%&/*-	Muestra un mensaje indicando que existe un dato incorrecto, señala el dato que está incorrecto. El dato se mantiene para que el usuario lo modifique si lo desea.	1. Seleccionar del menú la opción “Adicionar”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Aceptar” o “Aplicar”.
EC 1.5: Cancelar.	Cancela la acción de Adicionar falla y retorna a la página que le dio origen.	NA	Cancela la acción de Adicionar falla y retorna a la página que le dio origen.	1. Seleccionar del menú la opción “Adicionar”. 2. Introducir los datos solicitados. 3. Seleccionar la opción “Cancelar”.

Tabla 7: Descripción de las variables del DCP Adicionar de fallas.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Tipo de falla	Campo de texto	No	Texto que admite solo letras y espacios en cualquier lugar y la primera letra debe empezar con mayúscula.

Resultado de la prueba

Se realizó un DCP por cada funcionalidad, para un total de 28. Este método demostró resultados satisfactorios desde el punto de vista funcional. Las no conformidades detectadas durante las pruebas fueron analizadas y corregidas debidamente, logrando un correcto comportamiento de la funcionalidad ante diferentes situaciones (entradas válidas y no válidas). A continuación se muestra una tabla con el número de no conformidades detectadas tras cada iteración de prueba realizada.

Tabla 8: Resultados de las pruebas.

Iteraciones	Cantidad de no conformidades	Cantidad de no conformidades resueltas
1ra Iteración	30	15
2da iteración	15	10
3ra Iteración	5	0
4ta Iteración	0	

3.2.2 Análisis de valores límites

Las pruebas ya efectuadas se centran en el código, por lo que surge la necesidad de comprobar el rendimiento y la respuesta del sistema ante peticiones. Para la realización de este tipo de pruebas se utiliza la herramienta Apache JMeter, específicamente para realizar pruebas de rendimiento al módulo Planificación de Mantenimiento del Sistema Orbita. Se comparan los resultados arrojados por la herramienta una vez analizado el sistema implementado con marco de trabajo Sauxe e implementado con marco de trabajo Symfony 2 como marco base de la arquitectura Bosón.

Tabla 9: Comparación entre el Sistema Orbita en Sauxe y en Symfony 2 como marco base de la arquitectura Bosón.

Indicadores	Sistema Orbita en Sauxe	Sistema Orbita en Symfony 2 como marco base de la arquitectura Bosón
	Tiempo en milisegundo (ms)	
Tiempo medio ejecución promedio	875 ms	400 ms
Tiempo máximo ejecución promedio	10044 ms	2025 ms
Rendimiento	227.5 ms por petición	120.7 ms por petición

En la tabla se muestran los tiempos de respuestas del Sistema Orbita desarrollado en el marco de trabajo Sauxe y Symfony 2 como marco base de la arquitectura Bosón. Se puede constatar que existe una disminución de 0.5 segundos aproximadamente en cuanto al tiempo medio de respuesta de las funcionalidades. Se comprueba una diferencia en cuanto al tiempo máximo de respuesta de casi 9

segundos. Por último se evidencia que existe un aumento de casi el 50% en la velocidad de respuesta del sistema, ya que con Sauxe según la herramienta la velocidad era de 0.23 segundos y una vez realizada la migración a Symfony 2 como marco base de la arquitectura Bosón es de 0.12 segundos. De este modo se contribuye a un aumento en la velocidad de respuesta del sistema y un incremento de velocidad del módulo Planificación de Mantenimiento. Aunque se observa una mejora en el rendimiento del módulo, es necesario integrar todos los módulos del Sistema Orbita para comprobar sio existe una mejora en el rendimiento.

3.3 Pruebas de aceptación

Las pruebas de aceptación son realizadas principalmente por el cliente con el apoyo del equipo de desarrollo. El elemento fundamental de esta prueba es confirmar que el sistema está terminado, que cumple con las necesidades de la organización y que es aceptado por el usuario final.

Luego de la culminación de las pruebas y corregidas las no conformidades encontradas en las distintas iteraciones realizadas, el cliente comprobó el correcto funcionamiento del sistema y el cumplimiento de todas las funcionalidades. Avalando la solución al emitir el Certificado de Aceptación del Producto.

3.4 Conclusiones del capítulo

Al finalizar este capítulo se concluye que:

- A través de las pruebas de caja blanca y caja negra realizadas se identificaron y corrigieron errores en el código y en las funcionalidades del sistema, proporcionando el correcto funcionamiento de éste.
- A partir de las pruebas de aceptación se determinó que funcionalmente el módulo satisface las necesidades del cliente y se obtuvo la carta de aceptación por parte del mismo.
- Se cumplió el objetivo general de la investigación al mejorar el rendimiento del módulo Planificación de Mantenimiento del Sistema Orbita en casi el 50%.

Conclusiones

La investigación realizada cumple los objetivos planteados inicialmente mediante la migración del módulo Planificación de Mantenimiento del Sistema Orbita a la arquitectura de referencia en PHP Bosón, permitiendo arribar a las siguientes conclusiones:

- Se realizó el estudio del estado del arte para la elaboración del marco teórico en torno al objeto de estudio obteniéndose como resultado la metodología y las herramientas con que se llevó a cabo la migración del módulo Planificación de Mantenimiento del Sistema Orbita.
- Efectuando el estudio de los requisitos del módulo Planificación de Mantenimiento del Sistema Orbita, se obtuvieron artefactos para familiarizar el cliente con el sistema.
- Con la utilización del patrón arquitectónico y la arquitectura se obtuvo un buen orden entre clases para determinar el correcto funcionamiento del Sistema Orbita.
- El proceso de implementación permitió desarrollar el producto a partir de los resultados del análisis y diseño donde quedó establecida la nueva estructura del software.
- A través de las pruebas realizadas se validó el correcto funcionamiento del sistema y se comprobó un aumento en el rendimiento del mismo de casi el 50%.

Recomendaciones

Después de alcanzados los objetivos trazados al inicio de la investigación el autor propone las siguientes recomendaciones:

- Realizar la integración del módulo Planificación de Mantenimiento con los otros módulos del Sistema Orbita implementados sobre la arquitectura de referencia en PHP Bosón, para que el sistema brinde los servicios en el área de la dirección de Transporte de la UCI.
- Realizar las pruebas de rendimiento al Sistema Orbita una vez integrados los módulos que lo componen con el fin de comprobar si mejoró el rendimiento del mismo.

Referencias Bibliográficas

Amat, Daniel Arturo Casals. 2016. Comenzando con Bosón. *PHPuba, Comunidad Cubana de PHP.* [En línea] 2016. https://php.uci.cu/comenzando-con-boson/.

Apache Software Foundation. 1999-2016. APACHE JMeter. [En línea] 1999-2016. http://jmeter.apache.org/usermanual/best-practices.html#lean_mean.

Aristegui, José Luis. 2010. *LOS CASOS DE PRUEBA EN LA PRUEBA DE SOFTWARE.* s.l. : Revista Digital Lámpsakos, 2010.

Armero Kreisberger, Stiven. 2011. *Mantenimiento de Computadores.* 2011.

Bootstrap. 2015. Bootstrap. [En línea] 2015. http://getbootstrap.com.

Calás, Abraham. 2014. *Boson 0.1 documentation.* 2014. SeguridadBundle.

—. 2014. *Boson 0.1 documentation.* 2014. CacheBundle.

—. 2014. *Boson 0.1 documentation.* 2014. ExcepcionesBundle.

—. 2014. *Boson 0.1 documentation.* 2014. TrazasBundle.

—. 2014. *Boson 0.1 documentation.* 2014. IUXBundle.

—. 2014. *Boson 0.1 documentation.* 2014. PortalBundle.

—. 2014. *Boson 0.1 documentation.* 2014. IntegratorBundle.

—. 2014. *Boson 0.1 documentation.* 2014. EyCBundle.

—. 2014. *Boson 0.1 documentation.* 2014. ADTBundle.

—. 2014. *Boson 0.1 documentation.* 2014. AspectBundle.

Collard, Ross. 2005. *Developing the Test Strategy.* New York : s.n., 2005.

Conocimiento organizacional: la gestión de los recursos y el capital humano. **Silva, Frank E. Hernández. 2006.** La Habana : s.n., 2006, Vol. vol14_1_06.

Cruz, Emilio Gautier. 2006. *Modelos innovadores en la formación inicial docente.* Santiago de Chile : UNESCO, 2006.

Davenport, Sr. Thomas H. 2013. *Process Innovation: Reengineering Work Through Information Technology.* 2013.

Desarrolloweb.com. 2007. Desarrolloweb.com. [En línea] 2007. http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

Doctrine. 2016. Doctrine. [En línea] 15 de Febrero de 2016. http://docs.doctrine-project.org/projects/doctrine1/en/latest/en/index.html.

Doctrine Team. 2006-2014. Dcotrine Project. [En línea] 2006-2014. http://www.doctrine-project.org/projects/orm.html.

Eguiluz, Javier. 2013. *Desarrollo web ágil con Symfony2.* 2013.

—. 2014. *Desarrollo web ágil con Symfony2.* 2014.

Eguíluz, Javier. 2012. *Introducción a AJAX.* 2012.

Entonado, Florentino Blázquez. 2001. *Sociedad de la información y la educación.* Mérida : JAVIER FELIPE S.L., 2001.

Fernández Rodríguez, Néstor. 2002. *Manual de Proyectos.* s.l. : Junta de Andalucía, 2002.

Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerland, Michael Stal. 1996. *Pattern-Oriented Software Architecture.* 1ra. 1996.

Gamma, Erich, y otros. 1994. *Design Patterns: Eelemnts of Reusable Object-Oriented Software.* 1ra. 1994.

González, Rafael Herrera. 2011. *Conocimiento, innovación y desarrollo.* San Juan, Costa Rica : Gráfica del Este, 2011.

Guasch, Joan Antonio Ros. 2006. *Análisis de roles de trabajo en equipo: Un enfoque centrado en comportamientos.* 2006.

IEEE. 2000. *IEEE Std 1471, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.* 2000.

INTCO, Laboratorio Nacional de Calidad del Software del. 2009. *Ingeniería de Software: Metodologías y Ciclos de vida.* 1ra. 2009.

Isidro Ramos Salavert, María Dolores Lozano Pérez. 2000. *Ingeniería de Software y Base de Datos. Tendencias actuales.* s.l. : Ediciones de la Universidad de Castilla-La Mancha, 2000.

ISO, Organización Internacional para la Estandarización. 2001. *ISO/IEC 9126-1: 2001.* 2001.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. El Lenguaje Unificado de Modelado, Manual de referencia. 1ra. 2000.

Jacobson, Ivar, Rumbaugh, James y Booch, Grady. 2000. El Lenguaje Unificado de Modelado, Manual de referencia. 1ra. 2000, 9: Vistas físicas.

JetBrains s.r.o. 2000-2016. JetBrains . [En línea] 2000-2016. https://www.jetbrains.com/phpstorm/?fromMenu.

jQuery Foundation. 2015. Official jQuery Blog. [En línea] 2015. http://blog.jquery.com/.

Larman, Craig. 1999. *UML y Patrones, Introducción al análisis y diseño orientado a objetos.* 1ra. 1999.

Ltd., Zend Technologies. 2016. Zend Technologies Ltd. [En línea] 25 de Enero de 2016. http://framework.zend.com/.

Mateu, Carlos. 2004. *Desarrollo de aplicaciones web.* Barcelona : s.n., 2004.

Microsoft. 2016. MSDN, Developer Network. [En línea] 2016. https://msdn.microsoft.com/es-es/library/ms165088%28v=vs.80%29.aspx.

—. 2016. MSDN, Developer Network. [En línea] 2016. https://msdn.microsoft.com/es-es/library/cc728199%28v=ws.10%29.aspx.

mipatente. 2016. Mi Patente, Revista Digital sobre Patentes, Marcas y Propiedad Intelectual. [En línea] 2016. http://www.mipatente.com/migracion-tecnologica-por-un-futuro-mas-actualizado/.

Ojeda, Yolanda Gil. 2008. *Guía para la identificación y análisis de los procesos de la Universidad de Málaga.* Universidad de Málaga. 2008.

OpenUP Copyright -Eclipse. 2011. Disciplina: Analysis & Design. [En línea] 2011. http://epf.eclipse.org/wikis/openupsp/openup_basic/disciplines/analysis_and_design,_0TX9AMlgEdmt3adZL5Dmdw.html.

Pérez, Luis Alberto Cuarta. 2008. *Qué es el Mantenimiento Mecánico.* 2008.

pgAdmin Development Team. 2013. pgAdmin. [En línea] 2013. http://www.pgadmin.org.

PHP Team. 2015. PHP Hypertext Pre-processor. [En línea] 2015. http://php.net/.

PostgreSQL. 2013. Sobre PostgreSQL - es. [En línea] 2013. http://www.postgresql.org.es.

Potencier, Fabien. 2009. Templating Engines in PHP - Fabien Potencier. [En línea] 2009. http://fabien.potencier.org/article/34/templating-engines-in-php.

Pressman, Roger S. 2005. Ingeniería del Software. Un enfoque práctico. 6ta. 2005, 14.

—. 2005. Ingeniería del Software. Un enfoque práctico. 6ta. 2005, 15.

—. 2010. Software Engineering a Practitioner´s Approach. 7ma. 2010, 17.

—. 2010. *Software Engineering a Practitioner´s Approach.* 7ma. 2010.

—. 2010. Software Engineering, a Practitioner´s Approach. 7ma. 2010, 29.

Revista Venezolana de Gerencia. **2010.** 49, Maracaibo, Venezuela : s.n., 2010.

Sánchez, Tamara Rodríguez. 2015. *Metodología de desarrollo para la Actividad Productiva de la UCI v1.2.* 2015.

Scott W. Ambler, Ambysoft Inc. 2005-2014. The Agile Unified Process (AUP). [En línea] 2005-2014. http://www.ambysoft.com/unifiedprocess/agileUP.html.

SEI. 2016. Software Engineering Institute, Carnegie Mellon University. [En línea] 2016. http://www.sei.cmu.edu/.

Sommerville, Ian. 2005. 2005.

—. 2005. Ingeniería del Software. 7ma. 2005, Parte II.

—. 2005. Ingeniería del Software. 7ma. 2005, Parte I.

—. 2005. Ingeniería del Software. 2005.

—. 2005. *Ingeniería del Software.* 2005.

—. 2007. *Software Engineering.* 8va. 2007.

Symfony2 Spanish Translation Team. 2011. Symfony2 Spanish Documentation. [En línea] 2011. https://test-sf-doc-es.readthedocs.org/en/latest/contributing/code/standards.html.

TelFo Networks S.L. 2003-2012. Softbull. [En línea] 2003-2012. http://mantenimiento-de-flotas-de-vehiculos.softbull.com/.

The Apache HTTP Server Project. 2015. About the Apache HTTP Server Project. [En línea] 2015. http://httpd.apache.org/ABOUT_APACHE.html.

Universidad de las Ciencias Informáticas. 2012. Portal de la Universidad de las Ciencias Informáticas. [En línea] 2012. http://www.uci.cu/?q=entorno-productivo.

Vázquez, María Inés. 2007. *La Gestión Educativa en Acción: La metodología en casos.* s.l. : Gráfica Don Bosco, 2007.

Visual Paradigm. 2013. Visual Paradigm. [En línea] 2013. http://www.visual-paradigm.com.

W3C. 2016. W3C España. [En línea] 2016. http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo.

—. 2012. World Wide Web Consortium. [En línea] 2012. http://www.w3.org/.

Anexos

Anexo 1: Encuesta para determinar el rendimiento del Sistema Orbita.

La siguiente encuesta tiene como objetivo caracterizar el Sistema de Control de Flota y Mantenimiento Orbita al cual se adhieren los módulos Ejecución del mantenimiento vehicular, Vehículos, Planeación, Recursos Humanos entre otros. En la actualidad este sistema se encuentra en un proceso de migración por lo cual se realiza dicha encuesta con el objetivo de medir el estado real del funcionamiento del mismo en la Universidad de las Ciencias Informáticas (UCI).

Por ello le pedimos su contribución y que sea lo más sincero posible en sus planteamientos. Le garantizamos confidencialidad y anonimato. Gracias de antemano por su colaboración.

Datos de interés:

Área a la que pertenece: _____ Rol: _____

Preguntas:

1. ¿Los módulos asociados al Sistema de Control de Flota y Mantenimiento Orbita se encuentran integrados?

___ Sí No___

2. ¿La página que representa dicho sistema refleja la identidad de la empresa logos, compañía...)?

__Sí __No

3. ¿La información que se presenta en la aplicación es fácil de entender y memorizar?

__Sí __Más o menos __No

4. ¿Utiliza los conceptos establecidos para las funciones estándar? (“buscar" para las búsquedas, etc.)

__Sí __No

5. ¿Tras una acción relevante hay una opción de vuelta atrás?

__Sí __No

6. ¿La página de resultados de una búsqueda no muestra resultados duplicados (ni duplicados reales ni duplicados muy parecidos)? ¿En caso de Si, ponga ejemplos de ellos?

__Sí __No

7. ¿La interfaz de búsqueda está ubicada donde los usuarios esperan encontrarla (en la parte superior derecha de la página)?

__Sí __No

8. ¿El nombre de los botones de un formulario es adecuado, aplicado a la acción, no general (Ej.: utilizar “Enviar” en vez de “OK”)?

__Sí __No

9. ¿Especifica el tamaño de los archivos PDF?

__Sí __No

10. ¿El sitio tiene una URL correcta, clara y fácil de recordar?

__Sí __No

11. ¿Tiene un tiempo de respuesta rápida?

__Sí __No

12. ¿Cumple el sitio con el principio de usabilidad de realizar las operaciones con un máximo de tres click?

__Sí __No

13. Existe suficiente espacio entre los elementos de acción (links, botones, etc.) para prevenir que el usuario haga click en el elemento incorrecto?

__Sí __No

14. ¿El buscador (si existe) permite errores tipográficos y ortográficos (tildes)?

__Sí __No

15. ¿Existen aceleradores, accesos rápidos a operaciones frecuentes

__Sí __No

16. Tiempo de respuesta de la página principal.

__de 0 a 12segs __de 13 a 27 segs __más de 27segs

17. Tiempo de respuesta de las demás páginas.

__de 0 a 7 segs __de 8 a 12 segs __más de 13 segs

18. ¿Se cargan correctamente todos los componentes visuales del sistema? En caso de NO, ¿Cuáles no se cargan?

__Si __No

Anexo 2: Resultados arrojados por el Jmeter al módulo Planificación de Mantenimiento desarrollado con Sauxe.

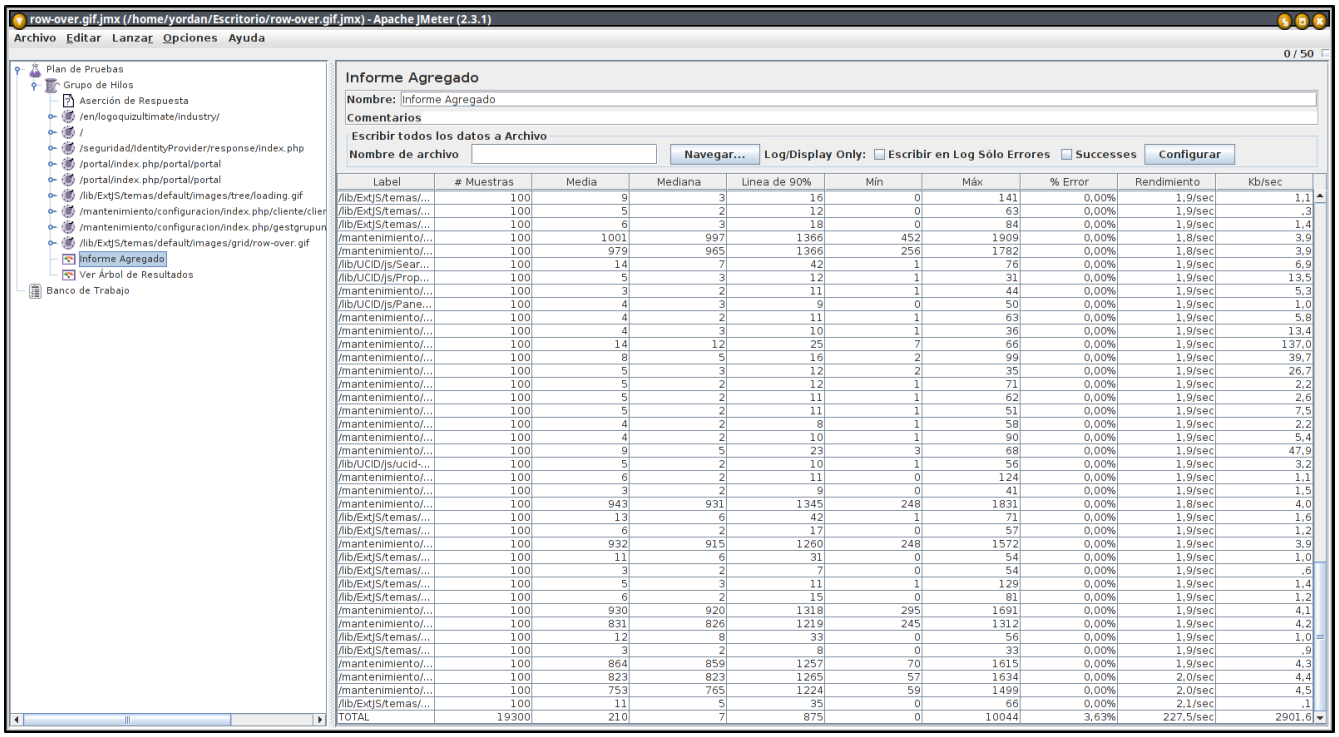


Figura 11: Resultados arrojados por el Jmeter al módulo Planificación de Mantenimiento desarrollado con Sauxe.

Anexo 3: Historias de Usuario (HU) del módulo Planificación de Mantenimiento.

Tabla 10: HU Adicionar grupo de vehículos.

Número: 1	**Nombre del requisito:** Adicionar grupo de vehículos.

Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 10 horas

Descripción:

El sistema debe brindar la posibilidad al usuario de gestionar grupos de vehículos determinados. La primera vista **Gestionar grupo de vehículos** contendrá los campos **Nombre** el cual representa el nombre, la marca y el modelo de ese grupo de vehículos, el segundo campo será el **Régimen** identificado por 1 o por 2. Además, en la primera vista se muestra un **Buscar** este elemento permitirá realizar la búsqueda de cualquier campo de los antes mencionados introduciendo el nombre, o la marca, o el modelo, o el tipo de vehículo, con sus campos correspondientes, además esta vista muestra las funcionalidades **Adicionar, Modificar, Eliminar e Imprimir**.

La segunda vista **Adicionar grupo de vehículos, pestaña Datos generales** permitirá al usuario añadir un grupo de vehículo determinado. La misma contará con los campos **Nombre** (se rellena solo con el tipo de vehículo, la marca y el modelo), el segundo campo será **Tipo de vehículo** (seleccionar algún elemento que se encuentra dentro de las opciones que se dan a conocer), el tercer campo **Marca** (cadena de caracteres que solo admite letras)**,** el cuarto campo será **Modelo** (cadena de caracteres que solo admiten letras y números), el quinto campo será el **Régimen de mantenimiento** el mismo admitirá la selección de escoger el régimen de entre 1 y 2 y por último el campo de **Unidad de Medida** (seleccionar algún elemento que se encuentra dentro de las opciones que se dan a conocer),. Cuenta con tres botones para realizar la acción ya sea de **Cancelar, Aplicar** o **Aceptar** la nueva solicitud de gestionar un grupo de vehículos que se esté creando.

Observaciones:

Si no se activa uno de los componentes de selección, si no se rellenan los componentes de marca y modelo y no se selecciona un tipo de vehículo, no se puede completar la operación. En el caso de activar lectura y no seleccionar un tipo de unidad, entonces no se puede completar la operación.

Prototipo elemental de interfaz gráfica de usuario:

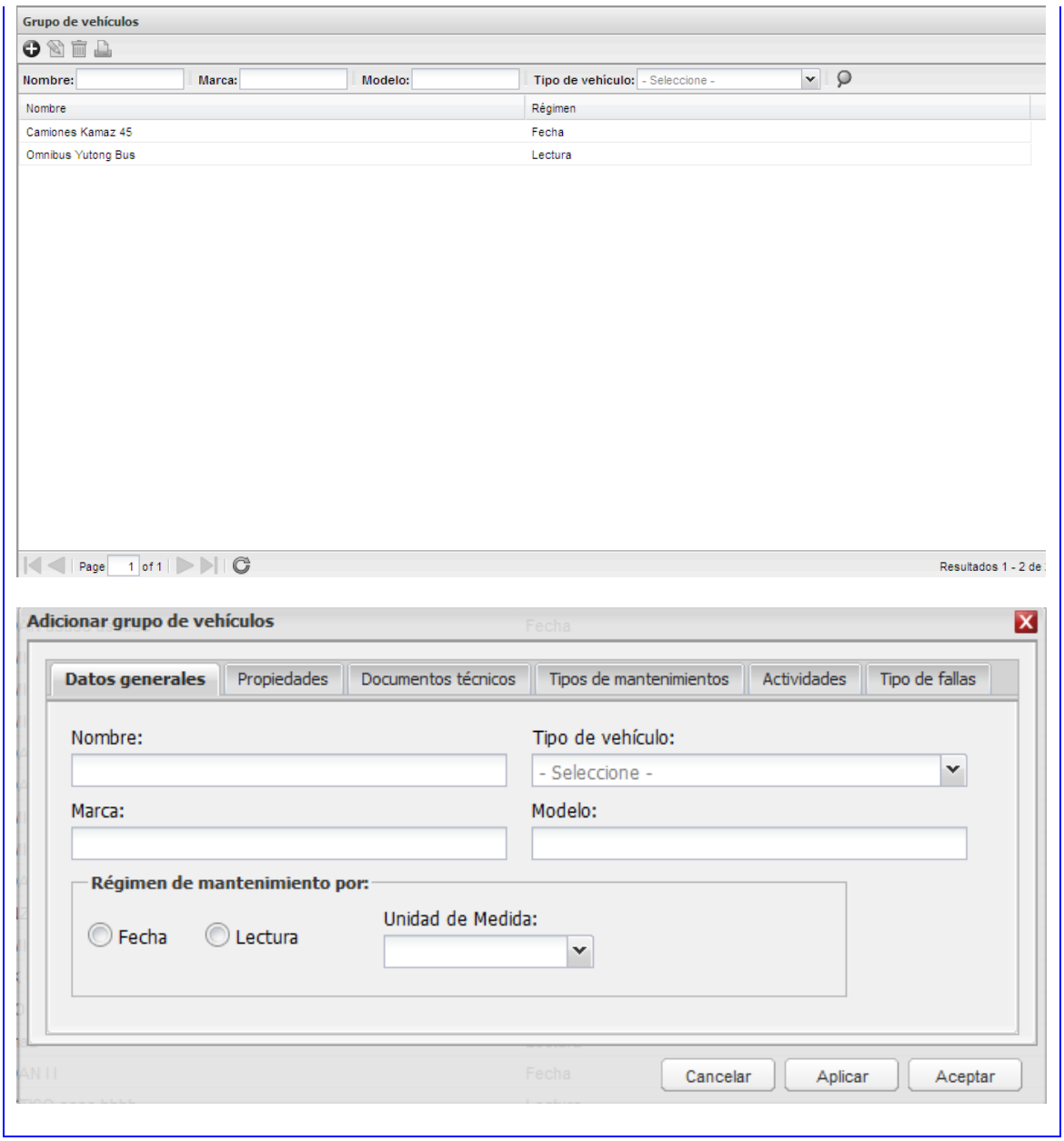


Tabla 11: HU Modificar grupo de vehículos.

Número: 2	**Nombre del requisito:** Modificar grupo de vehículos.
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 6 horas

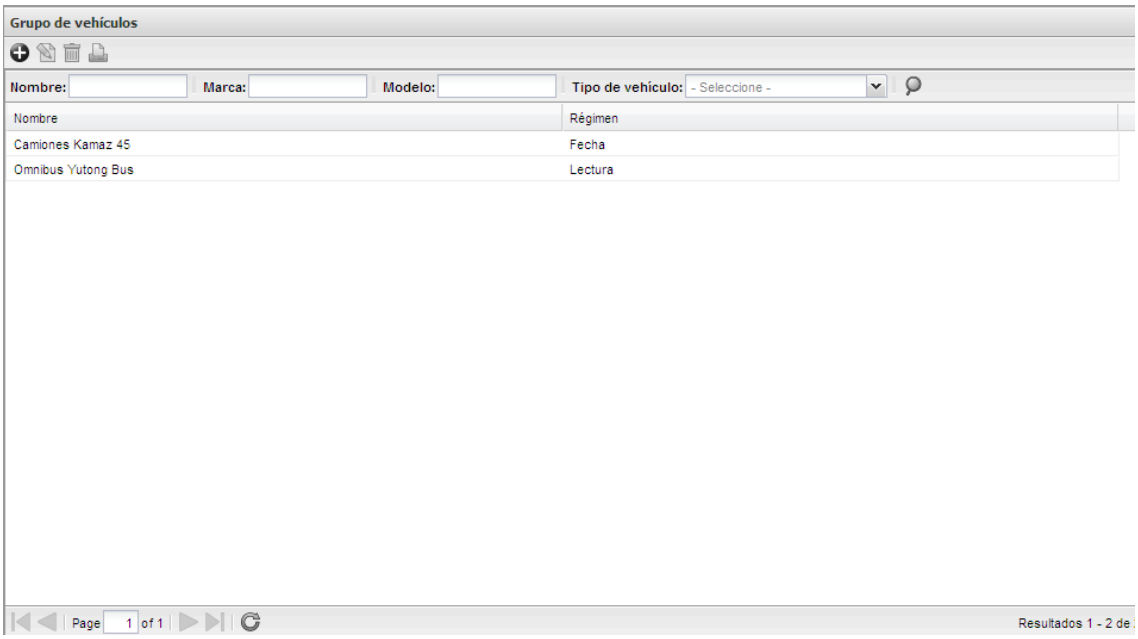
Descripción:

Permite modificar los datos de un grupo de vehículos en específico mostrando los mismos campos de la historia de usuario 1 con los datos originales.

Observaciones:

Si deja alguno de los campos en blanco, el sistema se lo marcara en rojo.

Prototipo elemental de interfaz gráfica de usuario:



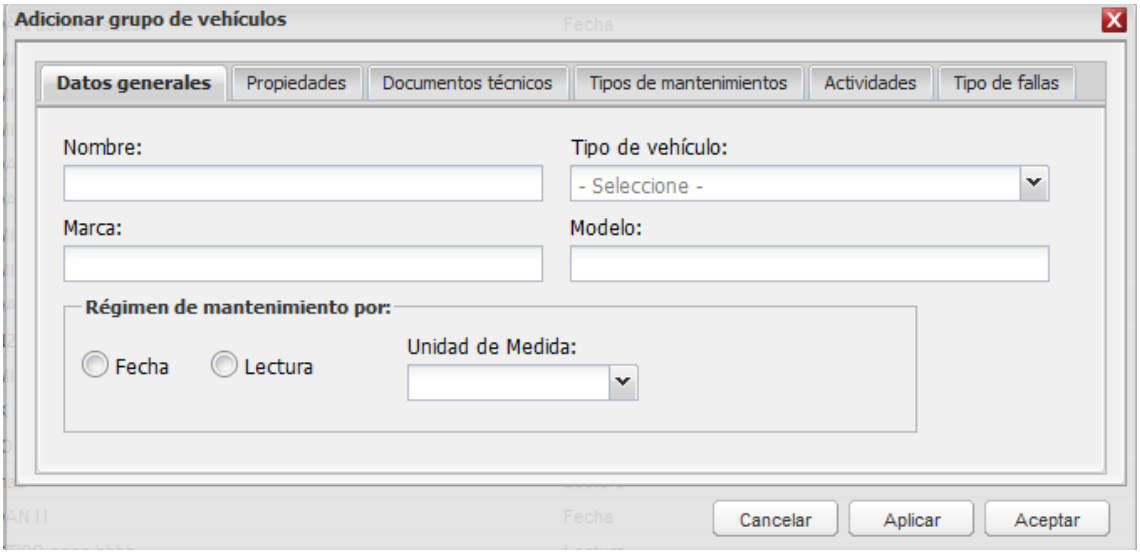


Tabla 12: HU Eliminar grupo de vehículos.

Número: 3	**Nombre del requisito:** Eliminar grupo de vehículos.
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 6 horas

Descripción:

El sistema debe brindar la posibilidad al usuario de eliminar grupos de vehículos determinados.

Observaciones:

Si no se selecciona un grupo de vehículo o varios grupos de vehículos, no se puede completar la operación.

Prototipo elemental de interfaz gráfica de usuario:

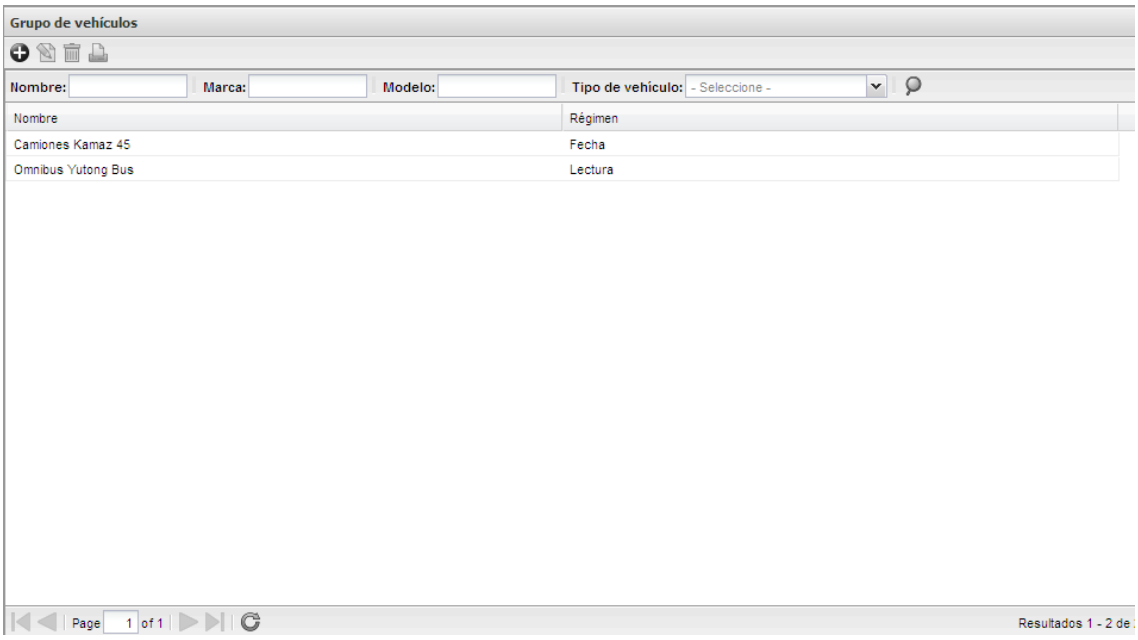


Tabla 13: HU Imprimir grupo de vehículos.

Número: 4	**Nombre del requisito:** Imprimir grupo de vehículos.
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Baja	**Tiempo Real:** 4 horas

Descripción:

Permite imprimir un grupo de vehículos seleccionado o varios grupos de vehículos seleccionados.

Observaciones:

Si no se selecciona lo que se desea imprimir, no se puede completar la operación.

Prototipo elemental de interfaz gráfica de usuario:

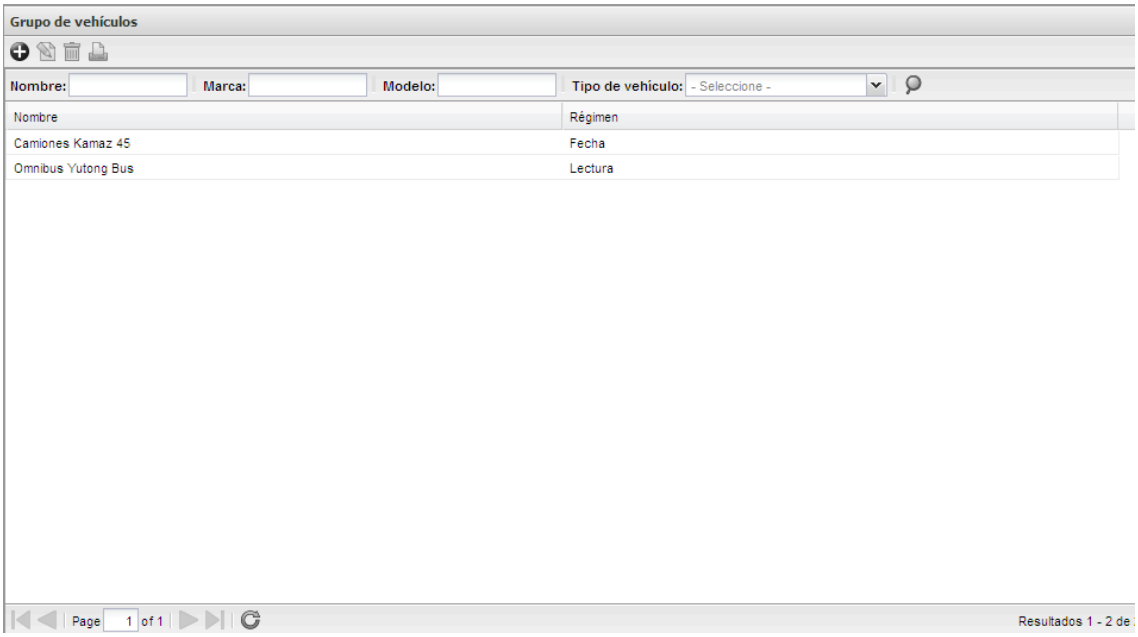


Tabla 14: HU Buscar grupo de vehículos.

Número: 5	**Nombre del requisito:** Buscar grupo de vehículos.
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 2 horas
Descripción: Permite buscar un grupo de vehículos seleccionado.	
Observaciones: Si no se escribe lo que se desea buscar, no se puede completar la operación.	
Prototipo elemental de interfaz gráfica de usuario:	

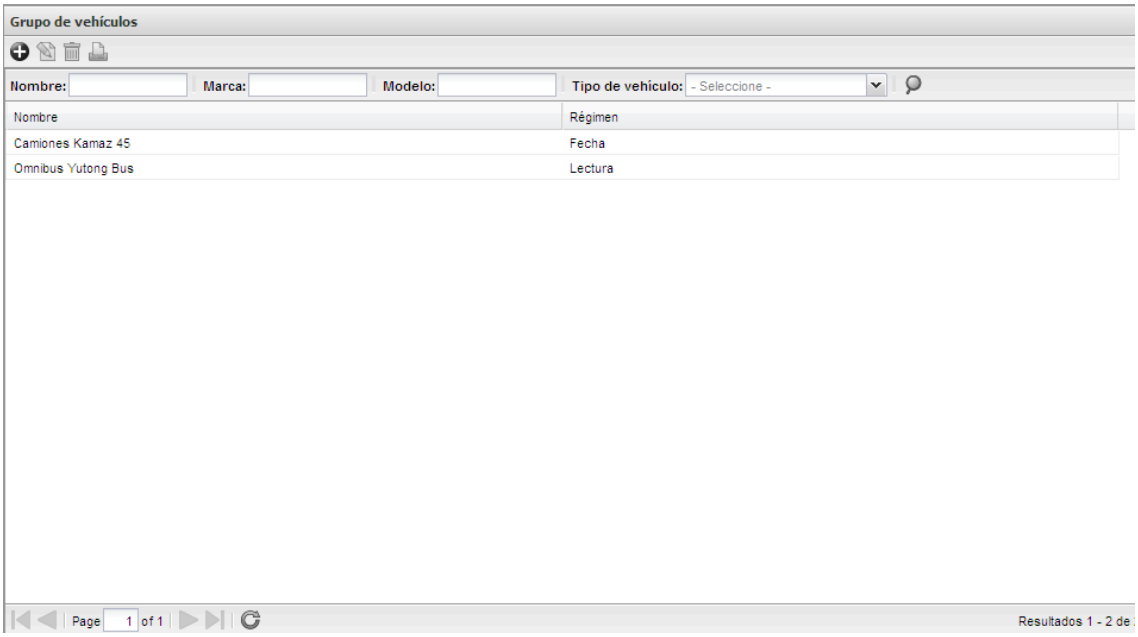


Tabla 15: HU Adicionar propiedades 1.

Número: 6	**Nombre del requisito:** Adicionar propiedades 1.
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 14 horas

Descripción:

El sistema debe brindar la posibilidad al usuario de gestionar grupos de vehículos determinados. La primera vista **Gestionar grupo de vehículos** contendrá los campos **Nombre** el cual representa el nombre, la marca y el modelo de ese grupo de vehículos, el segundo campo será el **Régimen** identificado por 1 o por 2. Además, en la primera vista se muestra un **Buscar** este elemento permitirá realizar la búsqueda de cualquier campo de los antes mencionados introduciendo el nombre, o la marca, o el modelo, o el tipo de vehículo, con sus campos correspondientes, además esta vista muestra las funcionalidades **Adicionar, Modificar, Eliminar e Imprimir**.

La segunda vista **Adicionar grupo de vehículos, pestaña Propiedades,** permitirá al usuario añadir una propiedad determinada. La misma contará con un listado con los campos **Seleccionar**, el segundo campo **Propiedad** y el tercer campo será **Valor**. Cuenta con tres botones para realizar la acción ya sea de **Cancelar, Aplicar** y **Aceptar** la nueva solicitud de gestionar propiedades que se esté creando, además esta vista muestra las funcionalidades **Adicionar** y **Eliminar**.

Observaciones:

Si no se activa uno de los componentes de selección, si no se rellenan los componentes de marca y modelo y no se selecciona un tipo de vehículo, no se puede completar la operación. En el caso de activar lectura y no seleccionar un tipo de unidad, entonces no se puede completar la operación.

Prototipo elemental de interfaz gráfica de usuario:

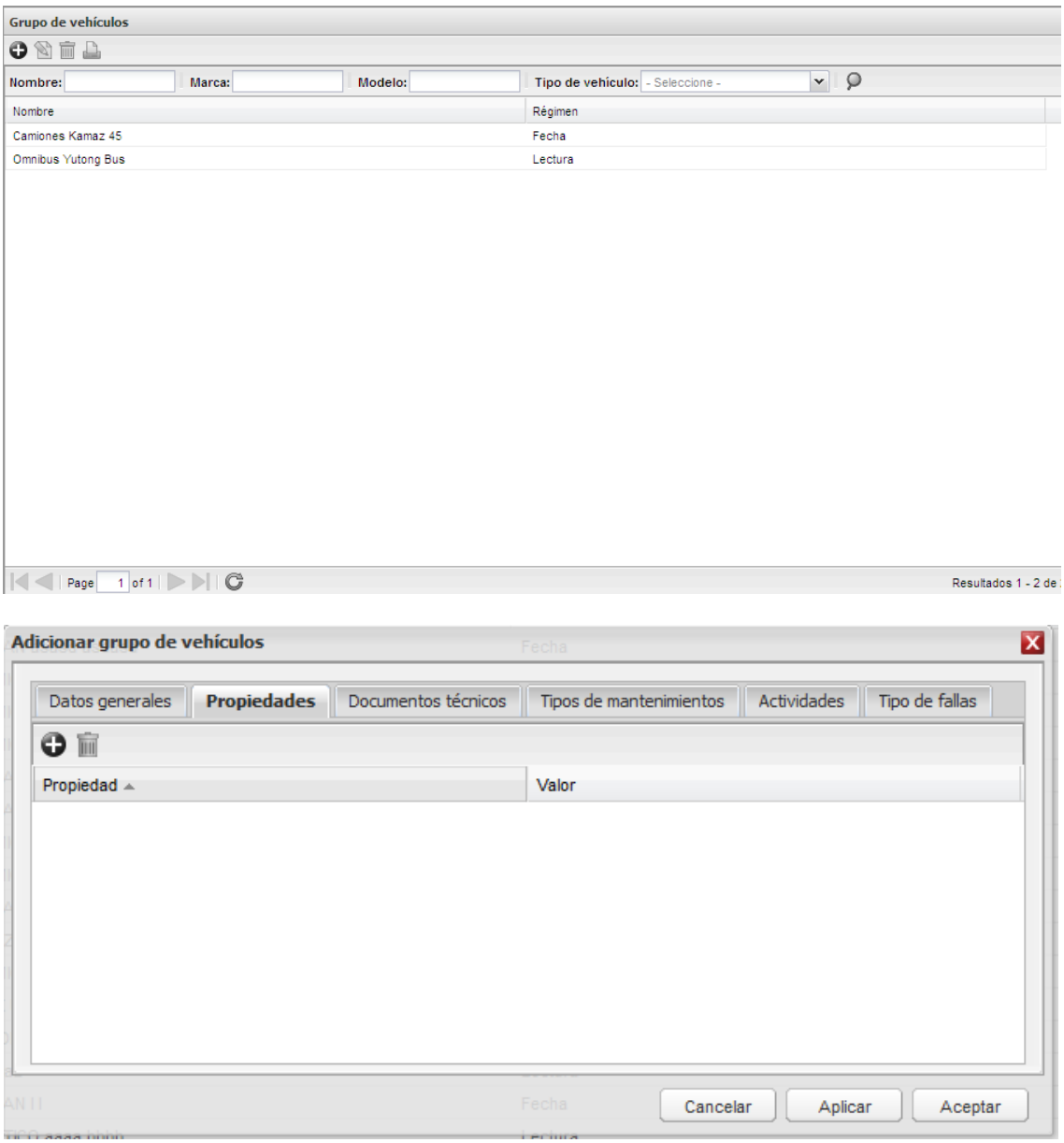


Tabla 16: HU Adicionar propiedades 2.

Número: 7	**Nombre del requisito:** Adicionar propiedades 2.
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas

Riesgo en Desarrollo: Bajo	**Tiempo Real:** 14 horas

Descripción:

El sistema debe brindar la posibilidad al usuario de gestionar grupos de vehículos determinados. La primera vista **Gestionar grupo de vehículos** contendrá los campos **Nombre** el cual representa el nombre, la marca y el modelo de ese grupo de vehículos, el segundo campo será el **Régimen** identificado por 1 o por 2. Además, en la primera vista se muestra un **Buscar** este elemento permitirá realizar la búsqueda de cualquier campo de los antes mencionados introduciendo el nombre, o la marca, o el modelo, o el tipo de vehículo, con sus campos correspondientes, además esta vista muestra las funcionalidades **Adicionar, Modificar, Eliminar e Imprimir**.

La segunda vista **Adicionar grupo de vehículos, pestaña Propiedades,** permitirá al usuario añadir una propiedad determinada. La misma contará con un listado con los campos **Seleccionar**, el segundo campo **Propiedad** y el tercer campo será **Valor**. Cuenta con tres botones para realizar la acción ya sea de **Cancelar, Aplicar** y **Aceptar** la nueva solicitud de gestionar propiedades que se esté creando, además esta vista muestra las funcionalidades **Adicionar** y **Eliminar**.

La tercera vista **Gestionar propiedades,** permitirá al usuario añadir una propiedad determinada. La misma contará con un listado con los campos **Seleccionar**, el segundo campo **Nombre**, el tercer campo será **Tipo de dato**, el cuarto campo **Longitud,** el quito campo será **¿Es un clasificador?** Cuenta con dos botones para realizar la acción ya sea de **Cancelar** y **Aceptar** la nueva solicitud de gestionar propiedades que se esté creando, además esta vista muestra las funcionalidades **Adicionar, Modificar** y **Eliminar**.

La cuarta vista **Adicionar propiedades,** permitirá al usuario añadir una propiedad determinada. La misma contará con los campos **Nombre**, el segundo campo **Longitud**, el tercer campo será **Tipo de dato**, el cuarto campo **Descripción,** el quito campo será **¿Es un clasificador?** Cuenta con un botón para realizar la acción **Adicionar** un valor de campo específico. Cuenta con dos botones para realizar la acción ya sea de **Cancelar, Aplicar** y **Aceptar** la nueva solicitud de adicionar propiedades.

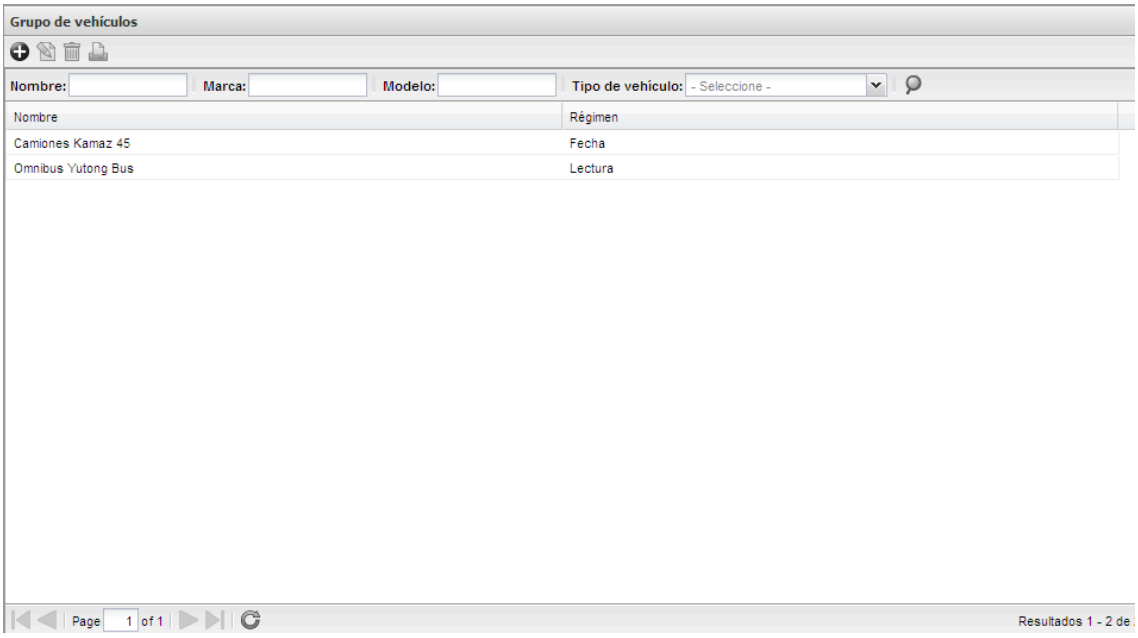
La quinta vista **Valores de campo,** permitirá al usuario añadir un valor de campo determinado. La misma contará con el campo **Valor**. Cuenta con un botón para realizar la acción **Cerrar** un valor de campo específico. Cuenta con dos botones para realizar la

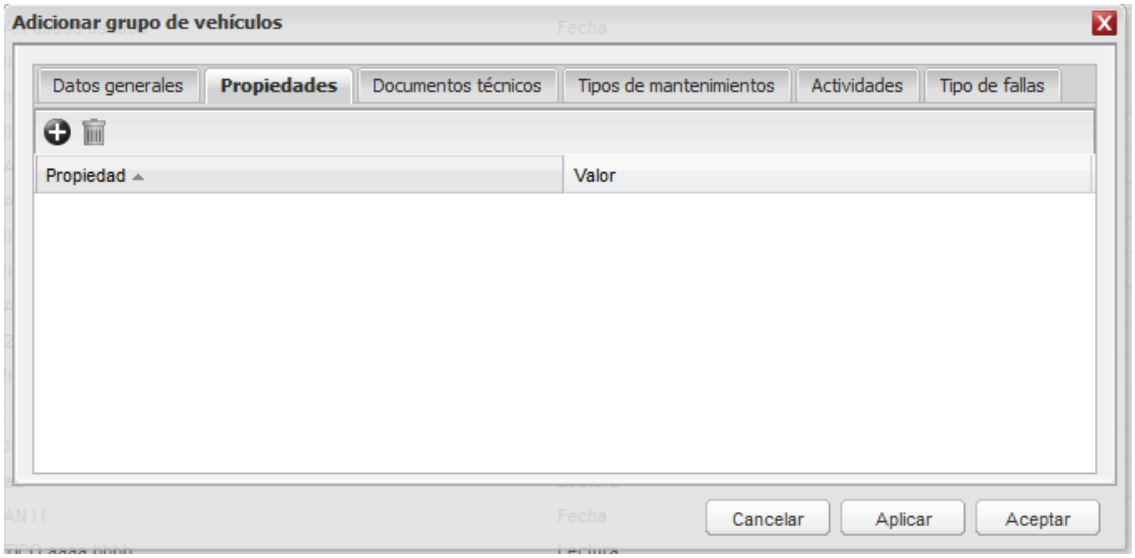
acción ya sea de **Cancelar,** la nueva solicitud de valores de campo, además esta vista muestra las funcionalidades **Adicionar, Modificar** y **Eliminar**.

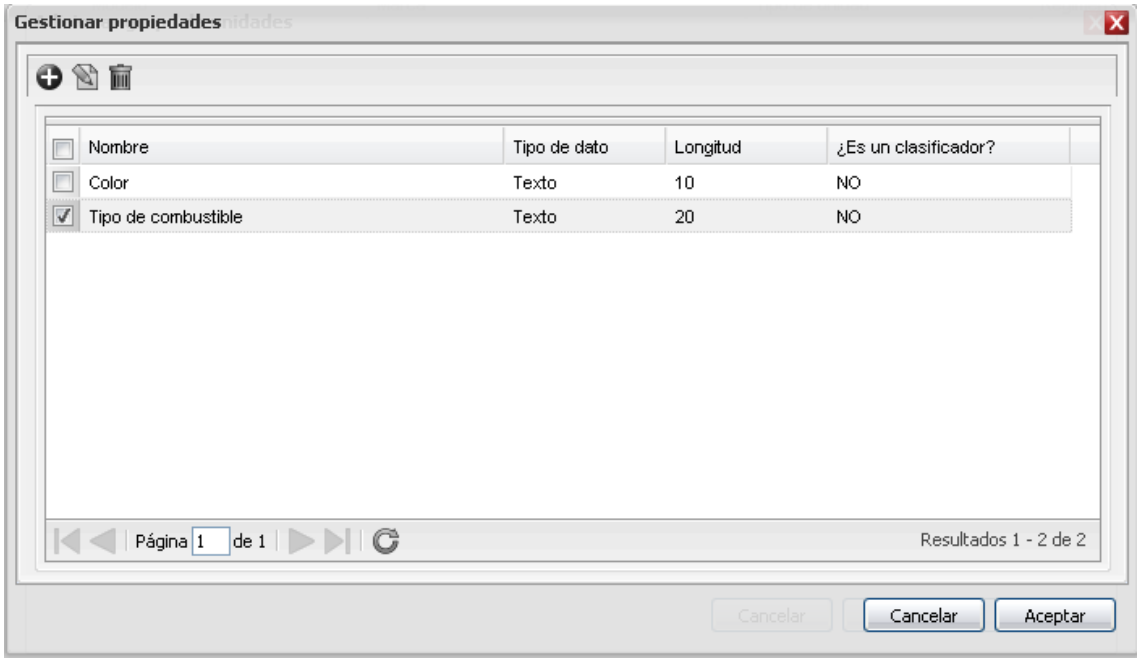
Observaciones:

Si no se activa uno de los componentes de selección, si no se rellenan los componentes de marca y modelo y no se selecciona un tipo de vehículo, no se puede completar la operación. En el caso de activar lectura y no seleccionar un tipo de unidad, entonces no se puede completar la operación.

Prototipo elemental de interfaz gráfica de usuario:







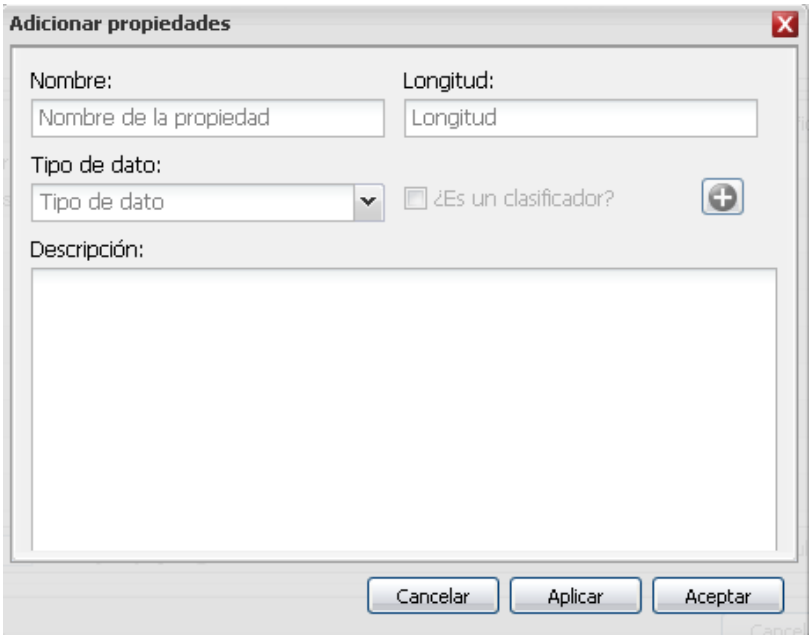


Tabla 17: HU Adicionar valor.

Número: 8	**Nombre del requisito:** Adicionar valor**.**
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1

Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 4 horas

Descripción:

El sistema debe brindar la posibilidad al usuario de gestionar grupos de vehículos determinados. La primera vista **Gestionar grupo de vehículos** contendrá los campos **Nombre** el cual representa el nombre, la marca y el modelo de ese grupo de vehículos, el segundo campo será el **Régimen** identificado por 1 o por 2. Además, en la primera vista se muestra un **Buscar** este elemento permitirá realizar la búsqueda de cualquier campo de los antes mencionados introduciendo el nombre, o la marca, o el modelo, o el tipo de vehículo, con sus campos correspondientes, además esta vista muestra las funcionalidades **Adicionar, Modificar, Eliminar e Imprimir**.

La segunda vista **Adicionar grupo de vehículos, pestaña Propiedades,** permitirá al usuario añadir una propiedad determinada. La misma contará con un listado con los campos **Seleccionar**, el segundo campo **Propiedad** y el tercer campo será **Valor**. Cuenta con tres botones para realizar la acción ya sea de **Cancelar, Aplicar** y **Aceptar** la nueva solicitud de gestionar propiedades que se esté creando, además esta vista muestra las funcionalidades **Adicionar** y **Eliminar**.

La tercera vista **Gestionar propiedades,** permitirá al usuario añadir una propiedad determinada. La misma contará con un listado con los campos **Seleccionar**, el segundo campo **Nombre**, el tercer campo será **Tipo de dato**, el cuarto campo **Longitud,** el quito campo será **¿Es un clasificador?** Cuenta con dos botones para realizar la acción ya sea de **Cancelar** y **Aceptar** la nueva solicitud de gestionar propiedades que se esté creando, además esta vista muestra las funcionalidades **Adicionar, Modificar** y **Eliminar**.

La cuarta vista **Adicionar propiedades,** permitirá al usuario añadir una propiedad determinada. La misma contará con los campos **Nombre**, el segundo campo **Longitud**, el tercer campo será **Tipo de dato**, el cuarto campo **Descripción,** el quito campo será **¿Es un clasificador?** Cuenta con un botón para realizar la acción **Adicionar** un valor de campo específico. Cuenta con dos botones para realizar la acción ya sea de **Cancelar, Aplicar** y **Aceptar** la nueva solicitud de adicionar propiedades.

La quinta vista **Valores de campo,** permitirá al usuario añadir un valor de campo determinado. La misma contará con el campo **Valor**. Cuenta con un botón para realizar

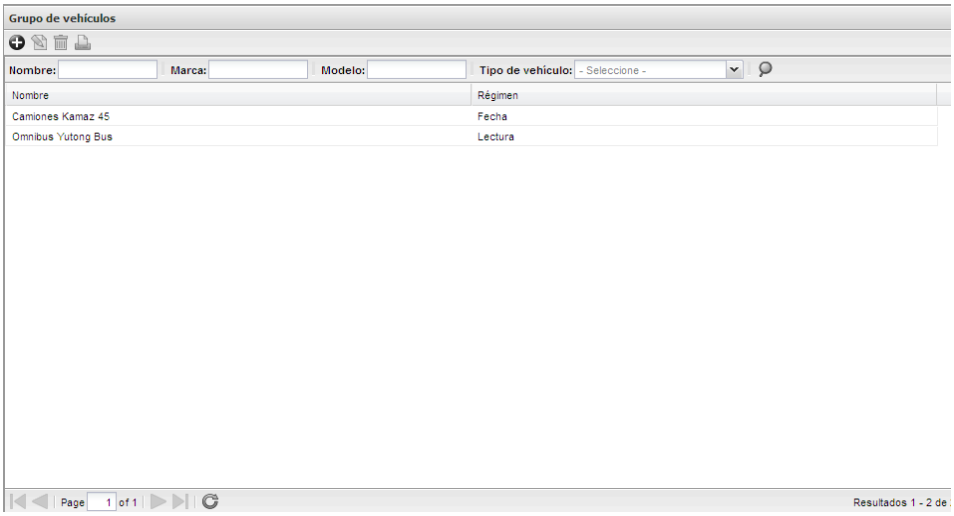
la acción **Cerrar** un valor de campo específico. Cuenta con dos botones para realizar la acción ya sea de **Cancelar,** la nueva solicitud de valores de campo, además esta vista muestra las funcionalidades **Adicionar, Modificar** y **Eliminar**.

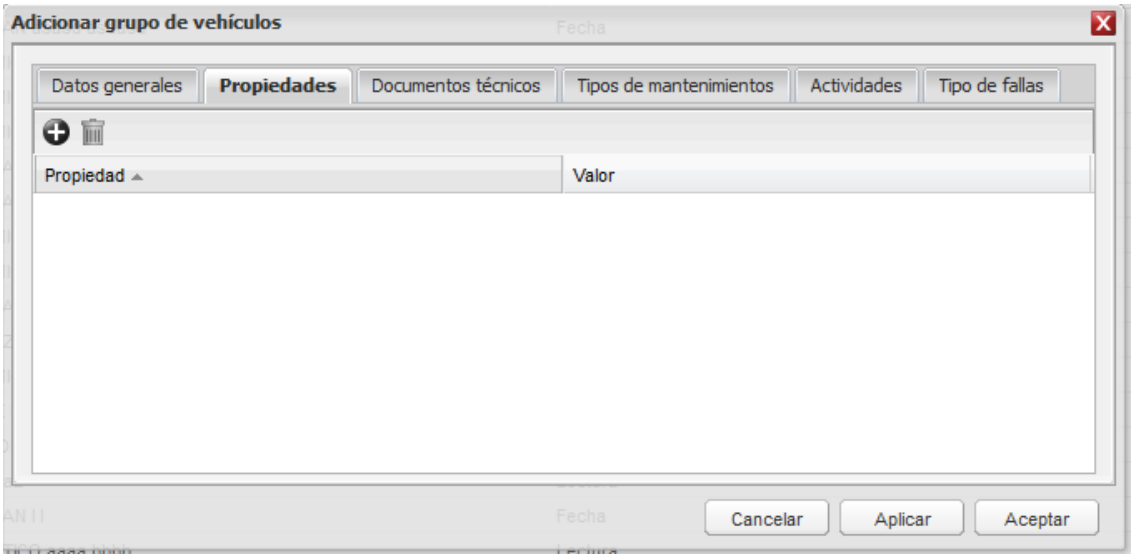
La sexta vista **Adicionar valor,** permitirá al usuario añadir un valor de campo determinado. La misma contará con un campo de texto. Cuenta con un botón para realizar la acción **Cerrar** un valor de campo específico. Cuenta con tres botones para realizar la acción ya sea de **Cancelar, Aplicar** y **Aceptar** la nueva solicitud de adicionar valor que se esté creando.

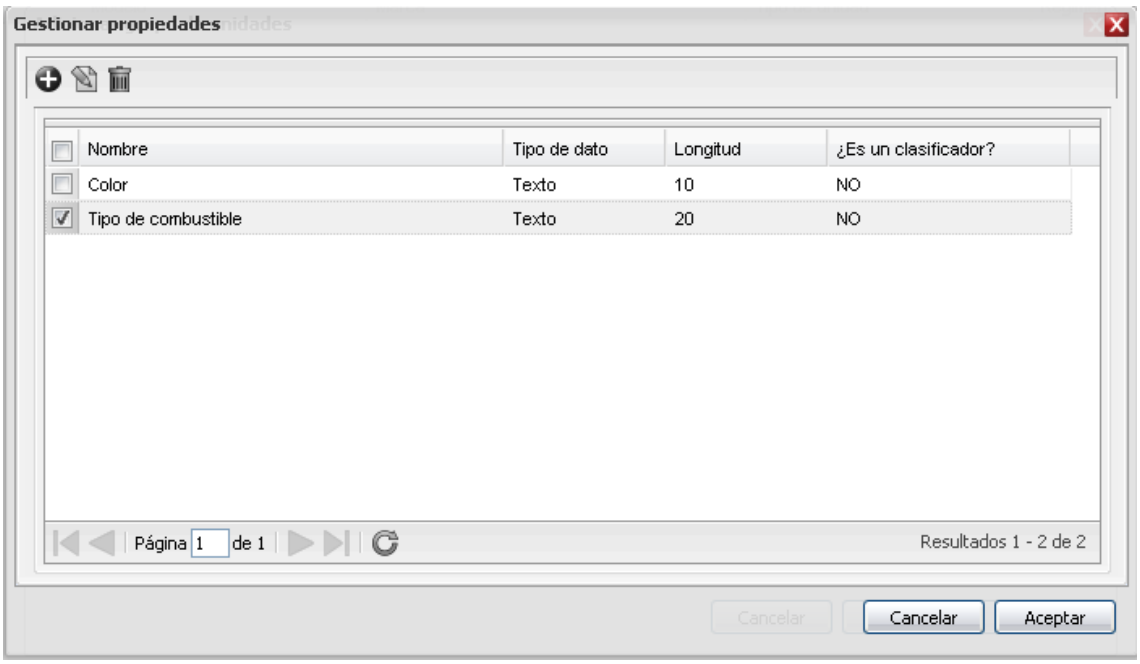
Observaciones:

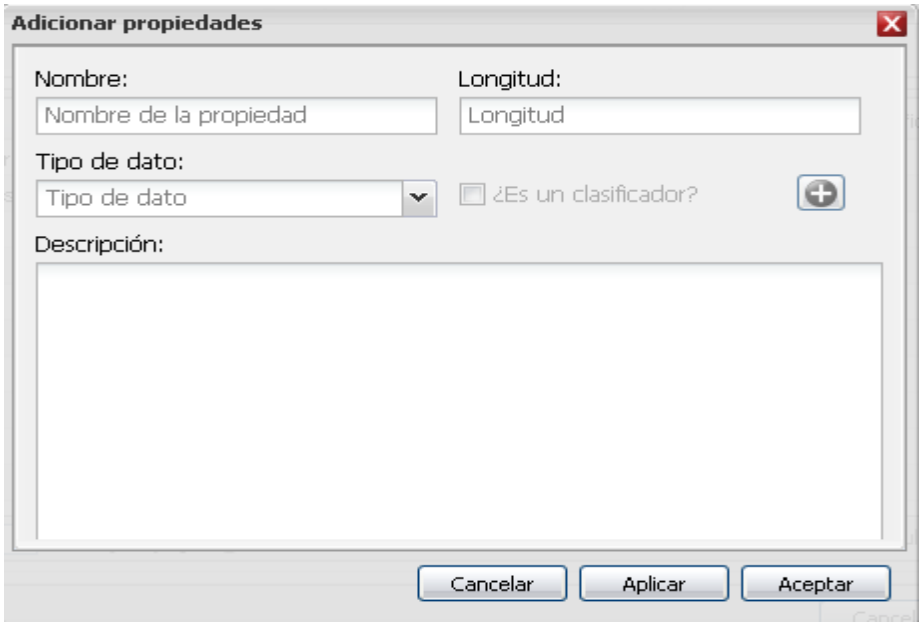
Si no se rellena el campo de texto **Tipo de falla**, no se puede completar la operación.

Prototipo elemental de interfaz gráfica de usuario:






Gestionar propiedades
Nombre
Tipo de dato
Longitud
¿Es un clasificador?
Color
Texto
10
NO
Tipo de combustible
Texto
20
NO
Página 1 de 1
Resultados 1 - 2 de 2
Cancelar
Aceptar


Adicionar propiedades
Nombre:
Nombre de la propiedad
Longitud:
Longitud
Tipo de dato:
Tipo de dato
¿Es un clasificador?
Descripción:
Cancelar
Aplicar
Aceptar

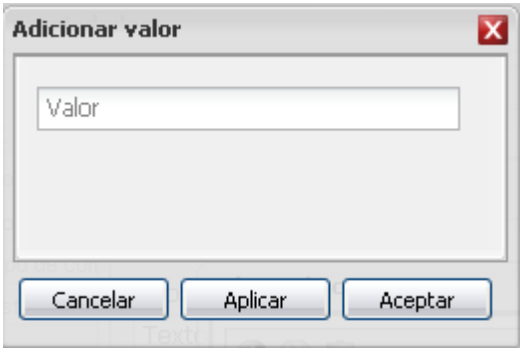
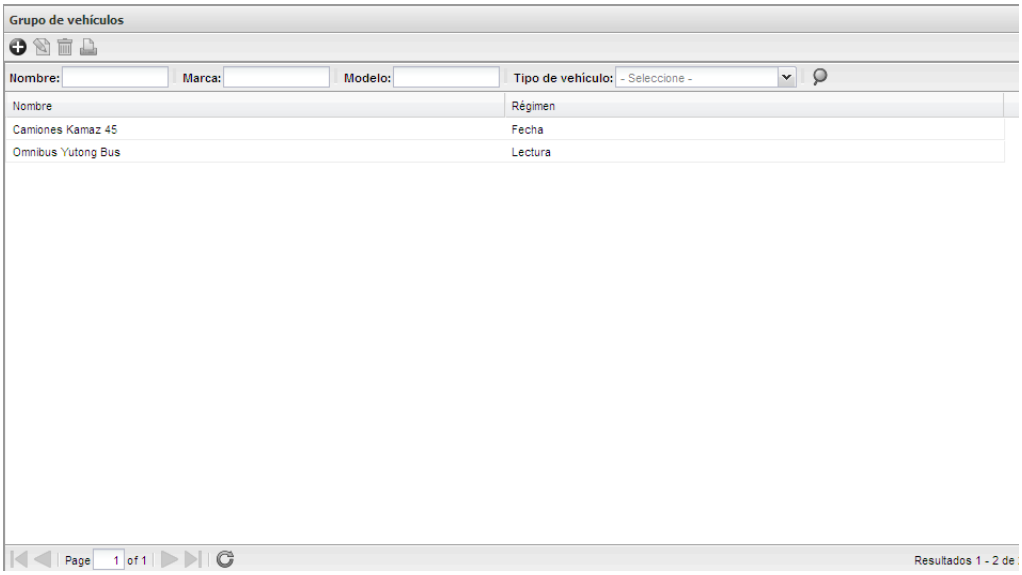
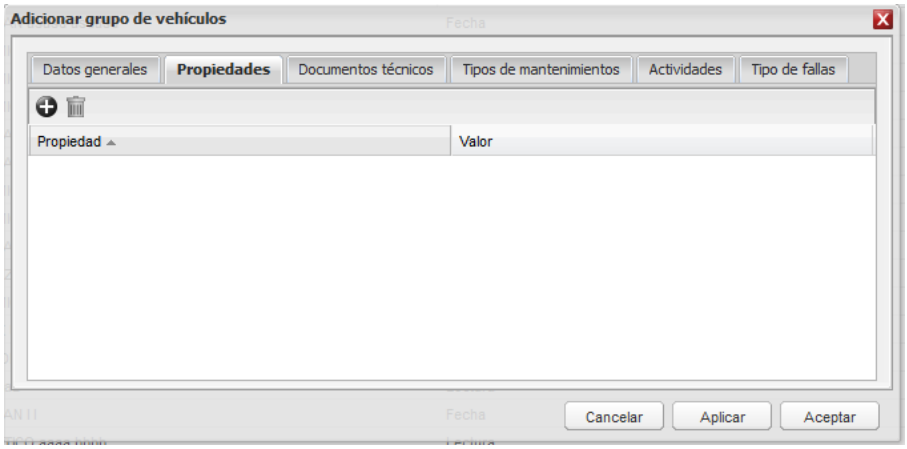
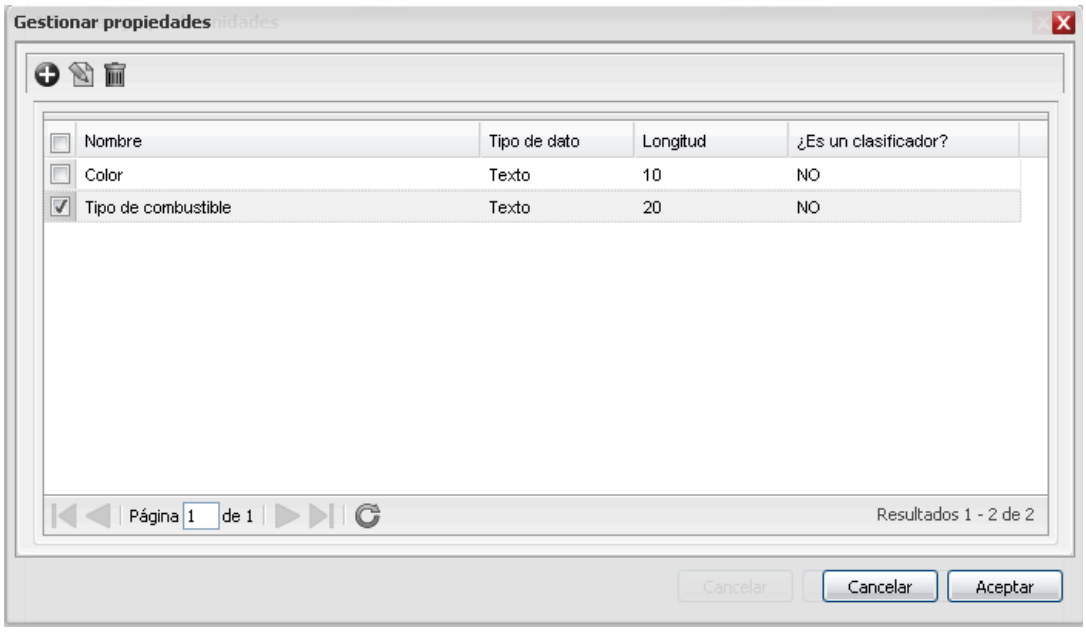

Adicionar valor
Valor
Cancelar
Aplicar
Aceptar

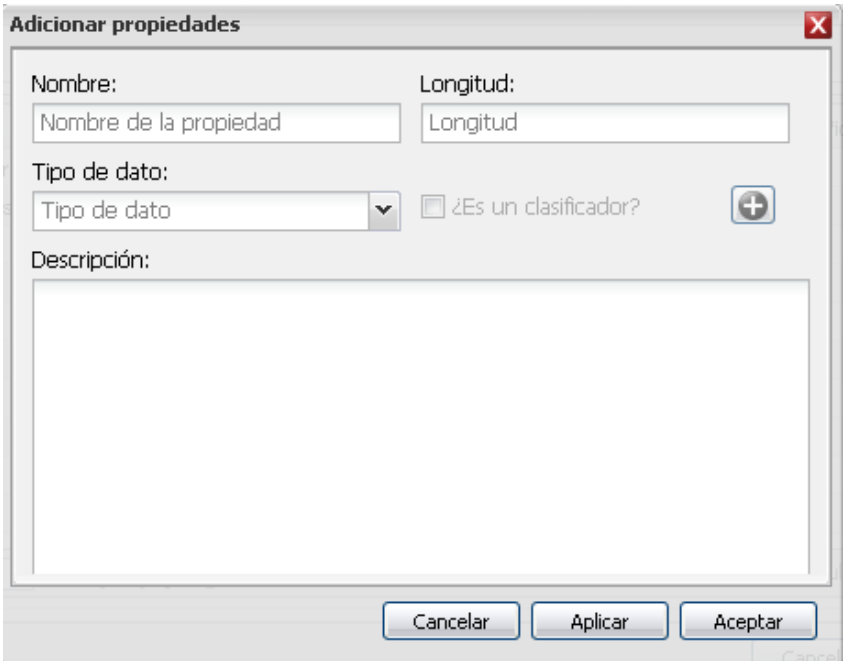
Tabla 18: HU Modificar valor.

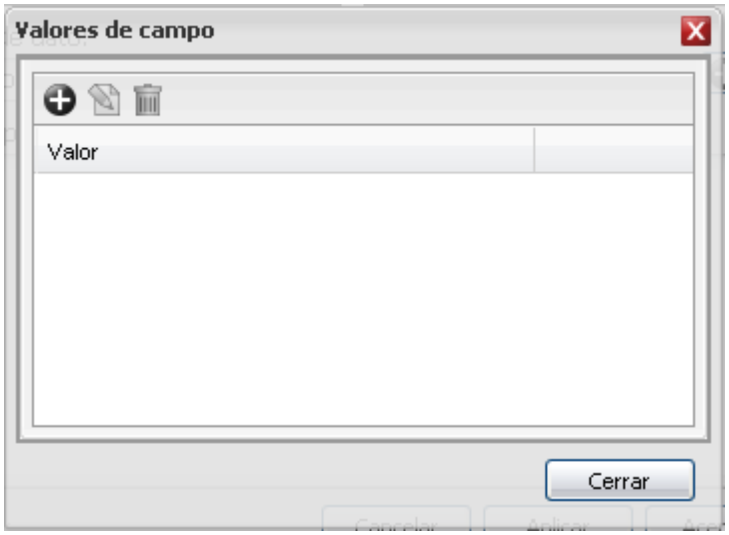
Número: 9	**Nombre del requisito:** Modificar valor.
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 4 horas
Descripción: Permite modificar el valor en específico mostrando los mismos campos de la historia de usuario 8 con los datos originales.	
Observaciones: Si deja alguno de los campos en blanco, el sistema se lo marcara en rojo.	
Prototipo elemental de interfaz gráfica de usuario:	




Adicionar grupo de vehículos
Datos generales
Propiedades
Documentos técnicos
Tipos de mantenimientos
Actividades
Tipo de fallas
Propiedad
Valor
Cancelar
Aplicar
Aceptar


Gestionar propiedades
Nombre
Tipo de dato
Longitud
¿Es un clasificador?
Color
Texto
10
NO
Tipo de combustible
Texto
20
NO
Página 1 de 1
Resultados 1 - 2 de 2
Cancelar
Cancelar
Aceptar


Adicionar propiedades
Nombre:
Nombre de la propiedad
Longitud:
Longitud
Tipo de dato:
Tipo de dato
¿Es un clasificador?
Descripción:
Cancelar
Aplicar
Aceptar


Valores de campo
Valor
Cerrar

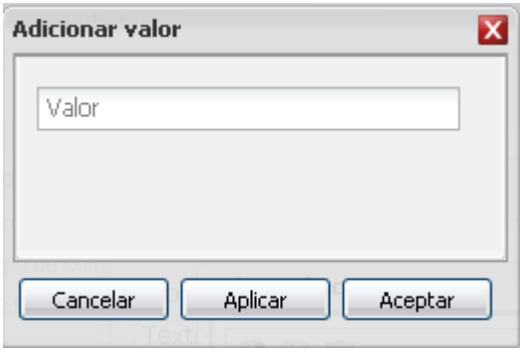
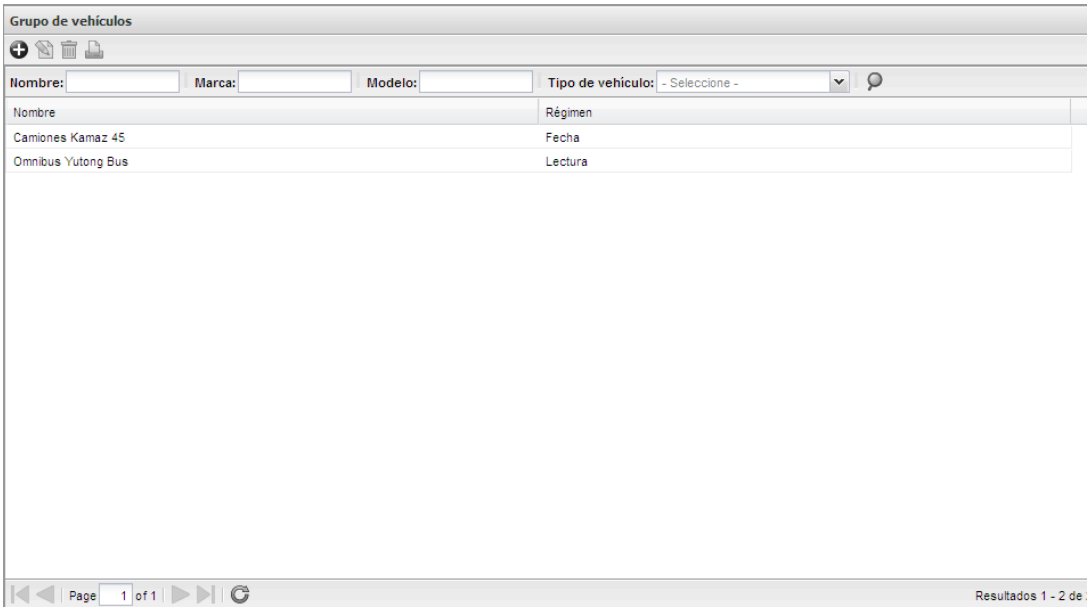
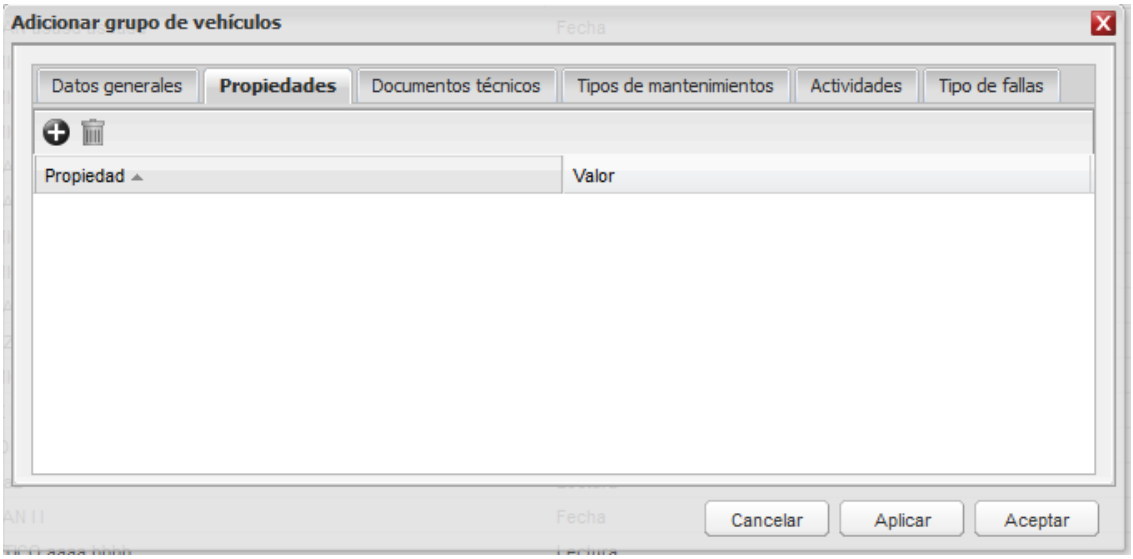
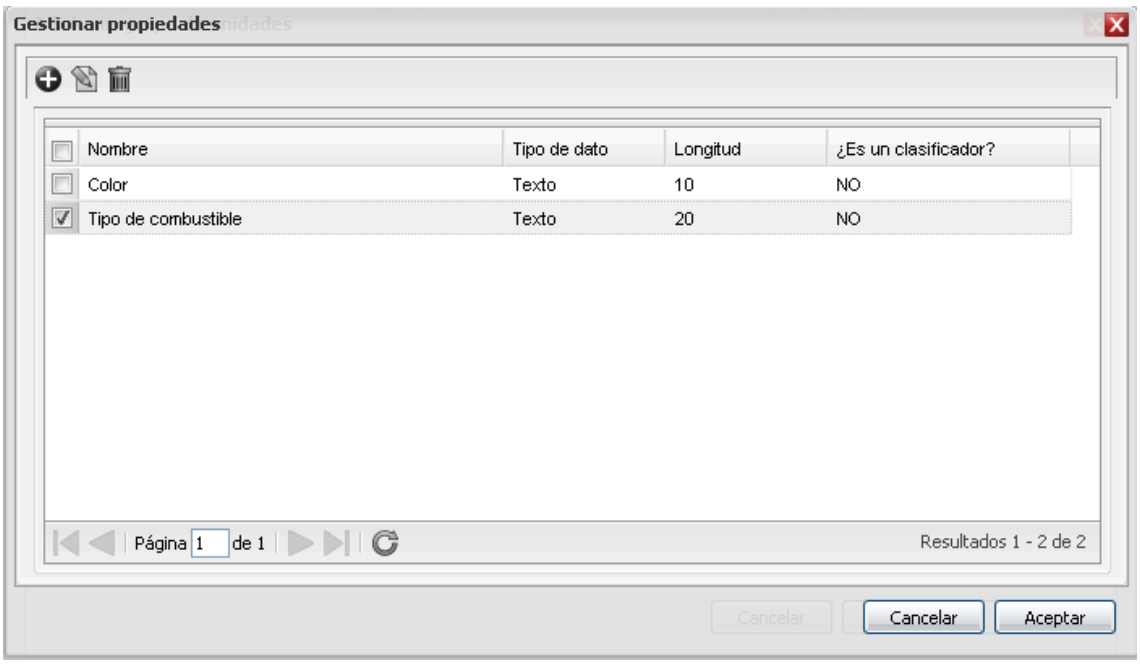

Adicionar valor
Valor
Cancelar
Aplicar
Aceptar

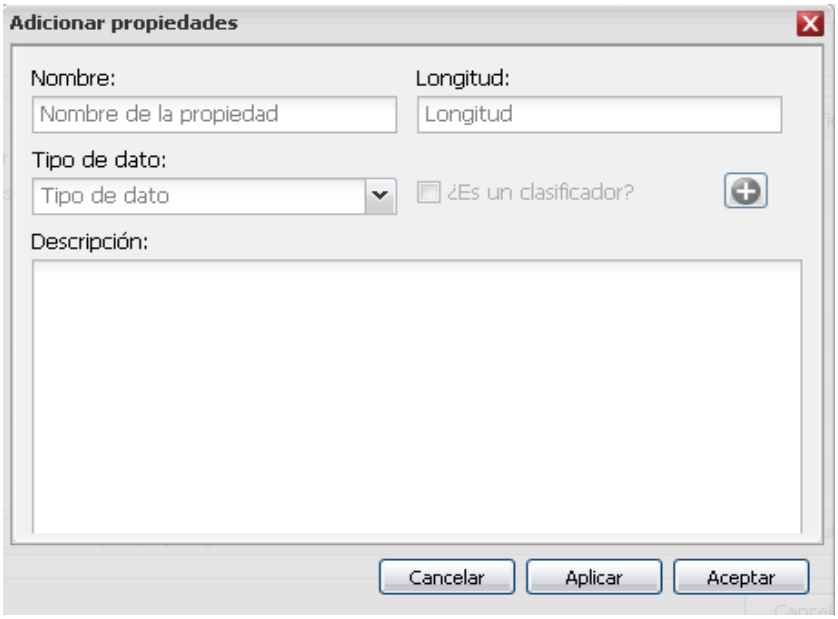
Tabla 19: HU Eliminar valor.

Número: 10	**Nombre del requisito:** Eliminar valor.
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 2 horas
Descripción: Permite eliminar un valor seleccionado.	
Observaciones: Si no se selecciona el valor, no se puede completar la operación.	
Prototipo elemental de interfaz gráfica de usuario:	




Adicionar grupo de vehículos
Datos generales
Propiedades
Documentos técnicos
Tipos de mantenimientos
Actividades
Tipo de fallas
Propiedad
Valor
Cancelar
Aplicar
Aceptar


Gestionar propiedades
Nombre
Tipo de dato
Longitud
¿Es un clasificador?
Color
Texto
10
NO
Tipo de combustible
Texto
20
NO
Página 1 de 1
Resultados 1 - 2 de 2
Cancelar
Aceptar



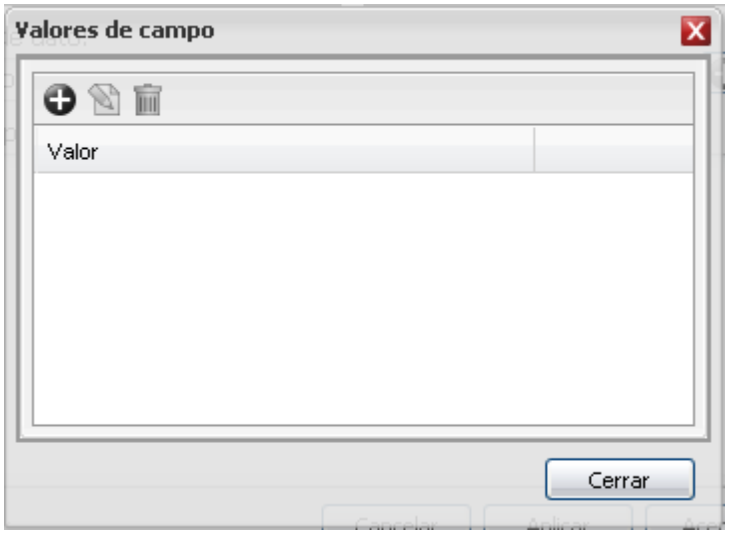


Tabla 20: HU Modificar propiedades.

Número: 11	**Nombre del requisito:** Modificar propiedades.
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 14 horas

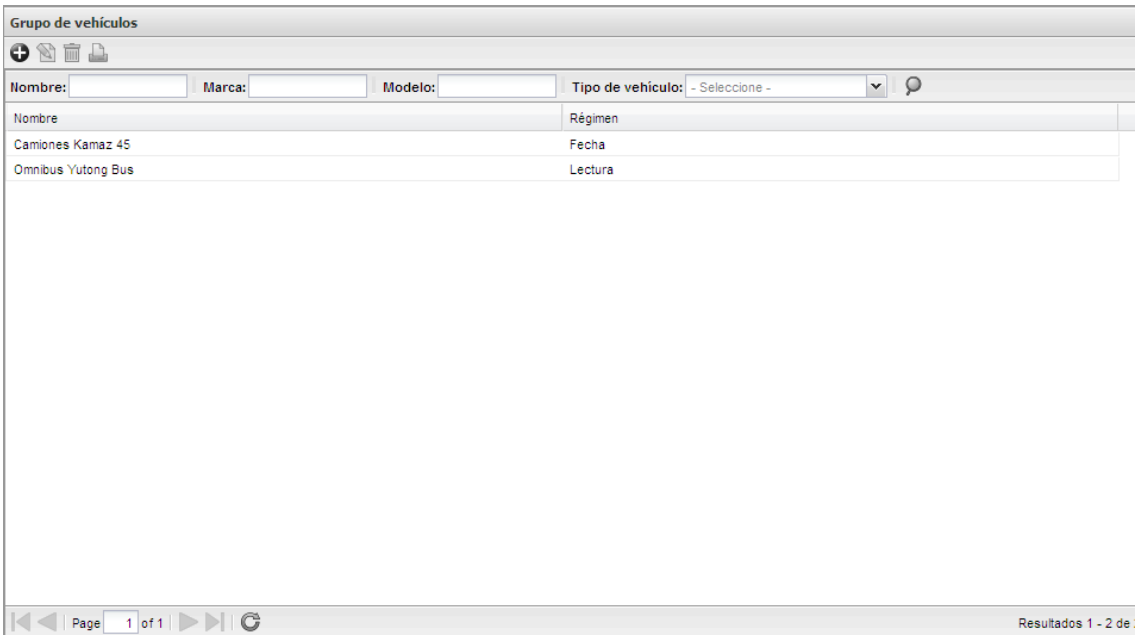
Descripción:

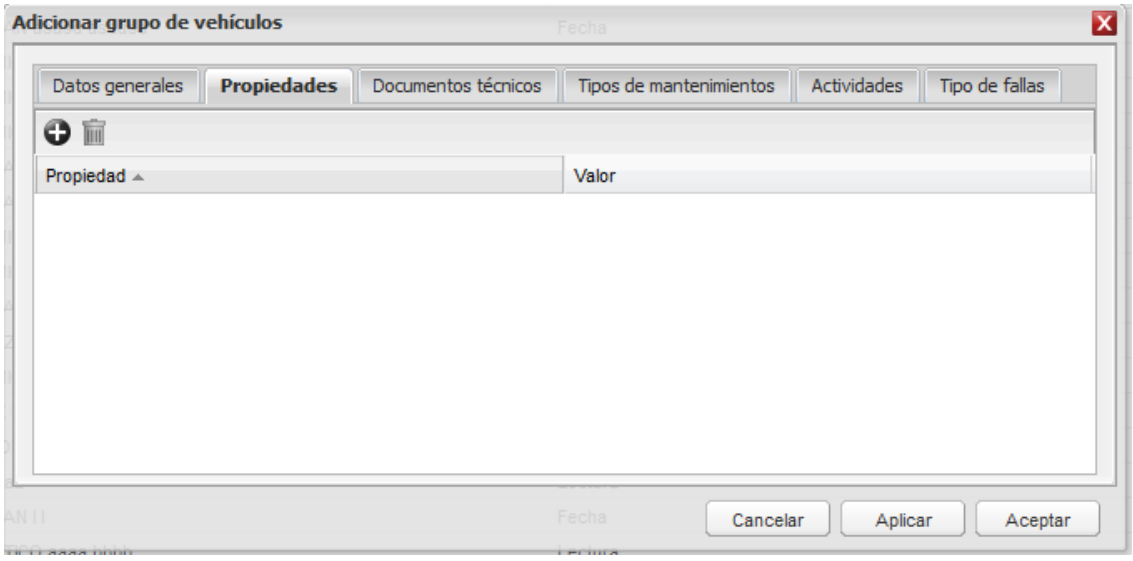
Permite modificar los datos de un grupo de vehículos en específico mostrando los mismos campos de la historia de usuario 7 con los datos originales.

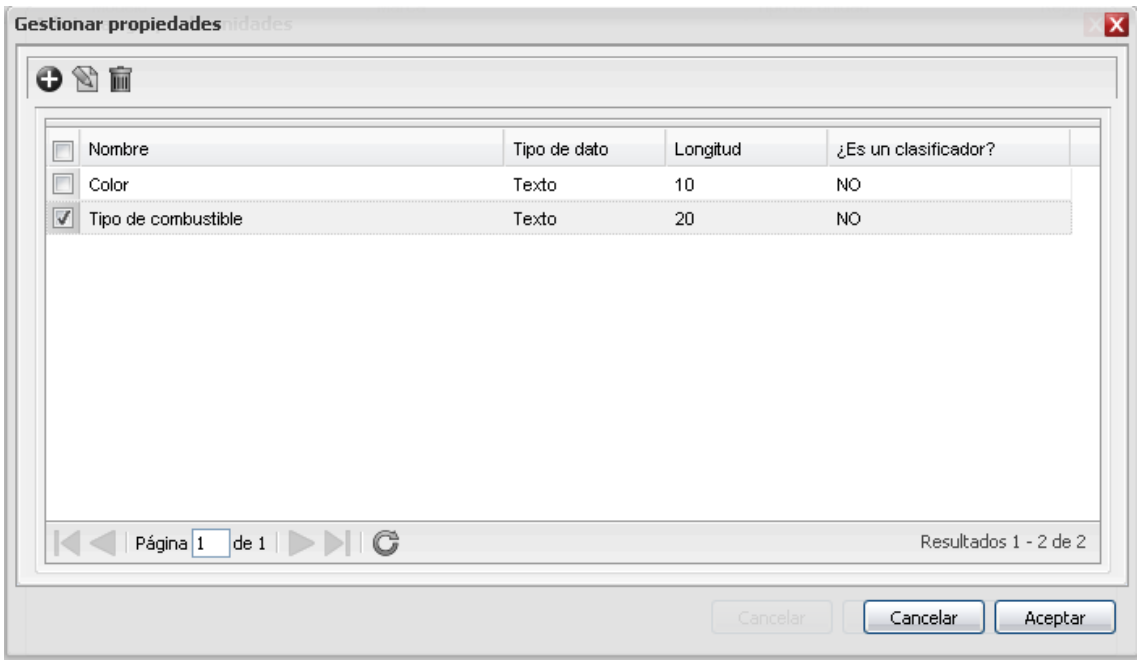
Observaciones:

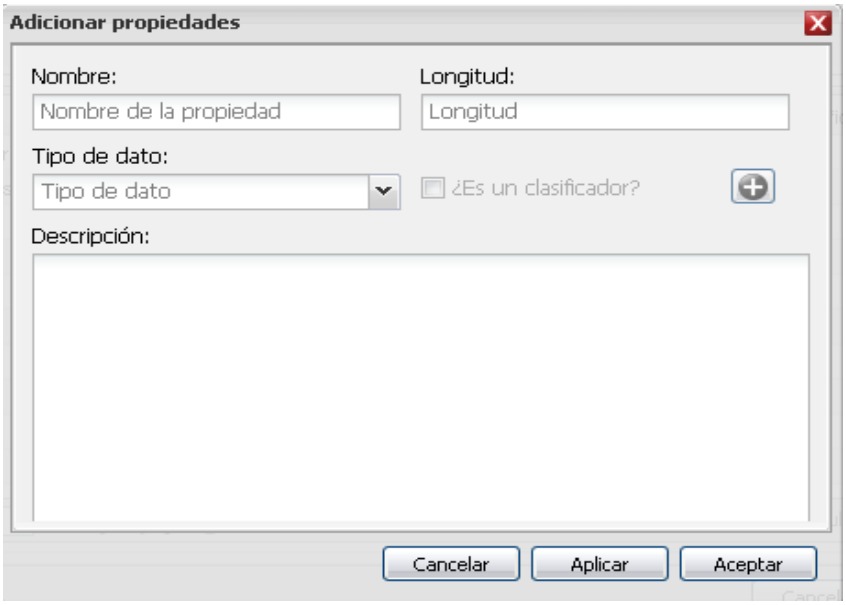
Si deja alguno de los campos en blanco, el sistema se lo marcara en rojo

Prototipo elemental de interfaz gráfica de usuario:






Gestionar propiedades
Nombre
Tipo de dato
Longitud
¿Es un clasificador?
Color
Texto
10
NO
Tipo de combustible
Texto
20
NO
Página 1 de 1
Resultados 1 - 2 de 2
Cancelar
Aceptar


Adicionar propiedades
Nombre:
Nombre de la propiedad
Longitud:
Longitud
Tipo de dato:
Tipo de dato
¿Es un clasificador?
Descripción:
Cancelar
Aplicar
Aceptar

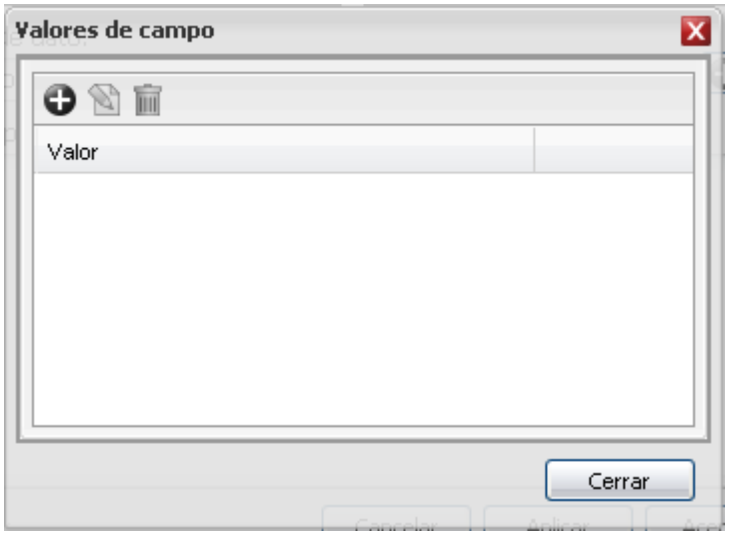
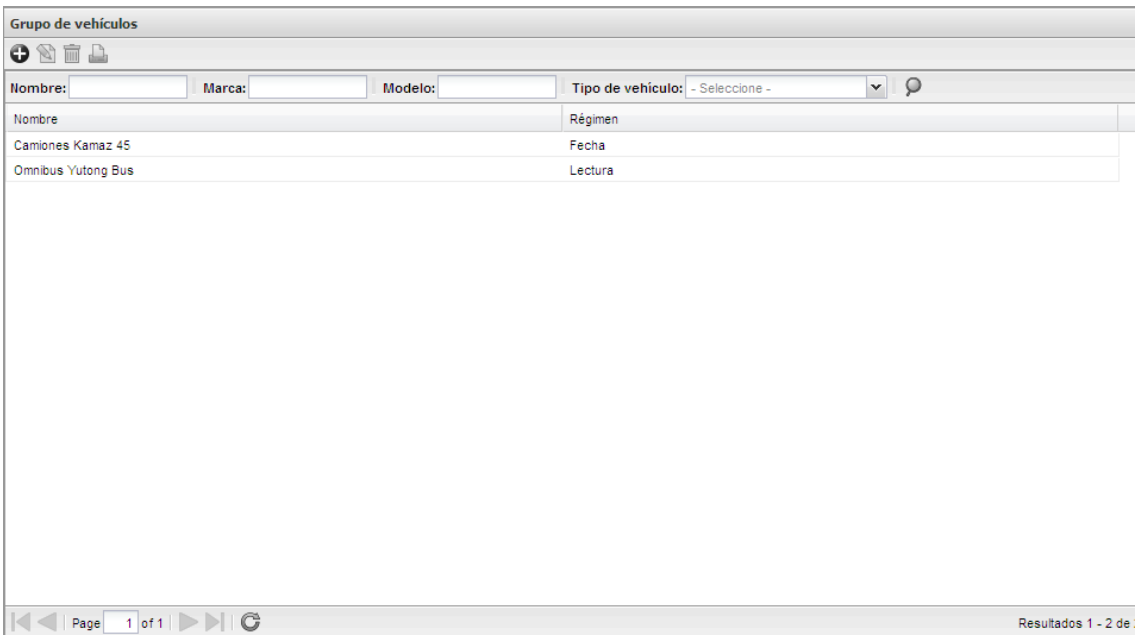
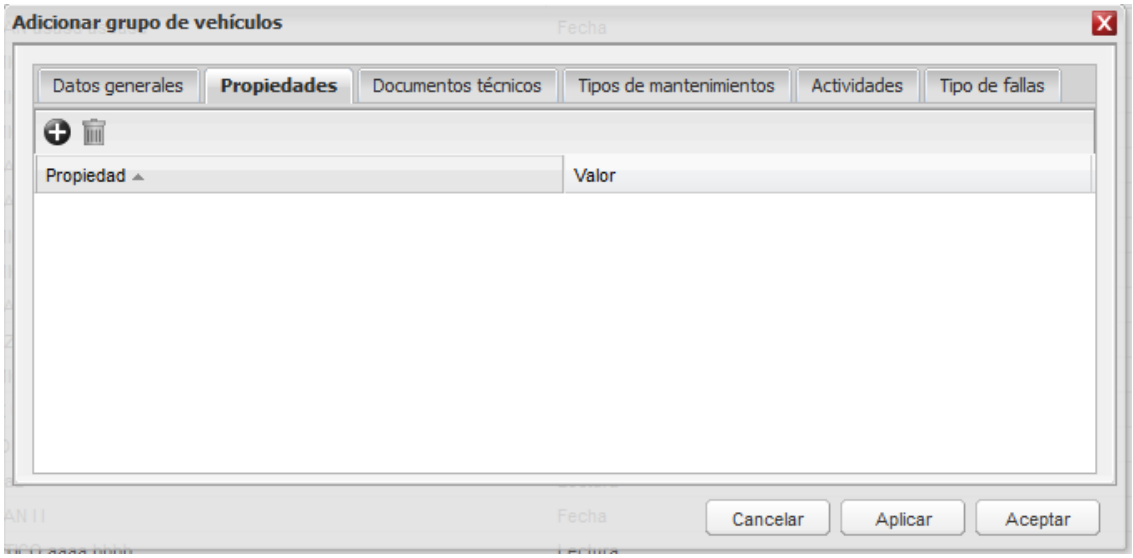


Tabla 21: HU Eliminar propiedades.

Número: 12	**Nombre del requisito:** Eliminar propiedades.
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 10 horas
Descripción: Permite eliminar las propiedades seleccionadas.	
Observaciones: Si no se selecciona una propiedad o varias propiedades, no se puede completar la operación.	
Prototipo elemental de interfaz gráfica de usuario:	


Grupo de vehículos
Nombre:
Marca:
Modelo:
Tipo de vehículo:
- Seleccione -
Nombre
Régimen
Camiones Kamaz 45
Fecha
Omnibus Yutong Bus
Lectura
Page 1 of 1
Resultados 1 - 2 de


Adicionar grupo de vehículos
Datos generales
Propiedades
Documentos técnicos
Tipos de mantenimientos
Actividades
Tipo de fallas
Propiedad
Valor
Cancelar
Aplicar
Aceptar

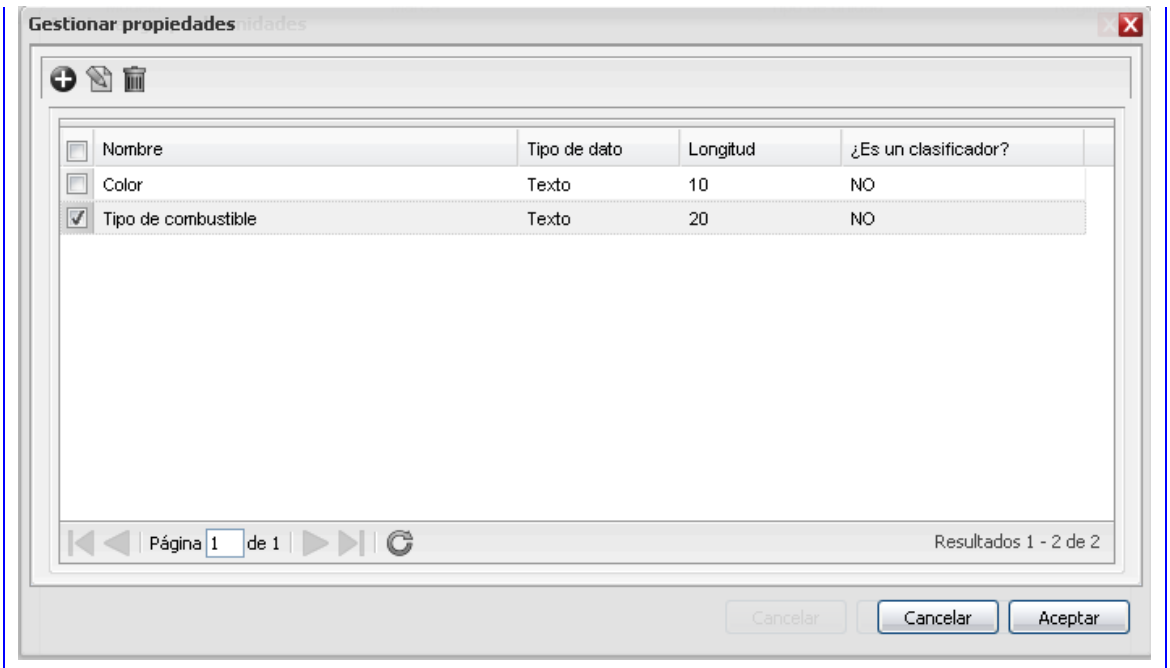


Tabla 22: HU Eliminar propiedades.

Número: 13	**Nombre del requisito:** Eliminar propiedades**.**
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 10 horas
Descripción: Permite eliminar las propiedades seleccionadas.	
Observaciones: Si no se selecciona una propiedad o varias propiedades, no se puede completar la operación.	
Prototipo elemental de interfaz gráfica de usuario:	

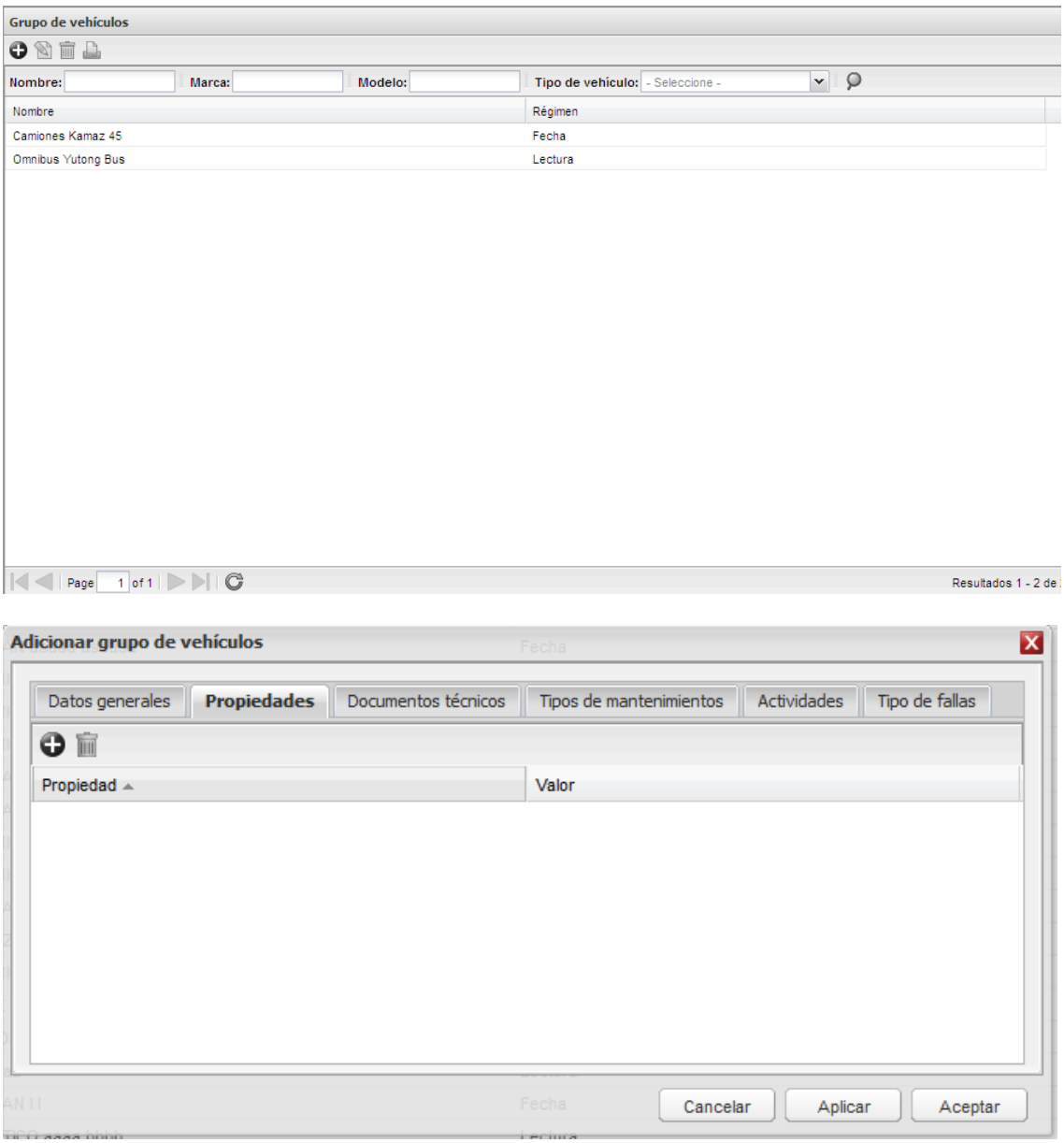


Tabla 23: HU Asociar documentos técnicos.

Número: 14	**Nombre del requisito:** Asociar documentos técnicos.
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas

Riesgo en Desarrollo: Bajo	**Tiempo Real:** 10 horas

Descripción:

El sistema debe brindar la posibilidad al usuario de gestionar grupos de vehículos determinados. La primera vista **Gestionar grupo de vehículos** contendrá los campos **Nombre** el cual representa el nombre, la marca y el modelo de ese grupo de vehículos, el segundo campo será el **Régimen** identificado por 1 o por 2. Además, en la primera vista se muestra un **Buscar** este elemento permitirá realizar la búsqueda de cualquier campo de los antes mencionados introduciendo el nombre, o la marca, o el modelo, o el tipo de vehículo, con sus campos correspondientes, además esta vista muestra las funcionalidades **Adicionar, Modificar, Eliminar e Imprimir**.

La segunda vista **Adicionar grupo de vehículos, pestaña Documentos técnicos** permitirá al usuario añadir un documento técnico determinado. La misma contará con los campos **Seleccionar**, el segundo campo será **Código1**, el tercer campo **Nombre** y el cuarto campo será **Descripción**. Cuenta con tres botones para realizar la acción ya sea de **Cancelar, Aplicar** o **Aceptar** la nueva solicitud de gestionar un grupo de vehículos que se esté creando, además cuenta con los botones de **Asociar, Quitar** y **Descargar**.

Observaciones:

Si no se asocia uno de los documentos, no se puede completar la operación. En el caso de no asociar algún documento se puede dejar la operación en blanco.

Prototipo elemental de interfaz gráfica de usuario:

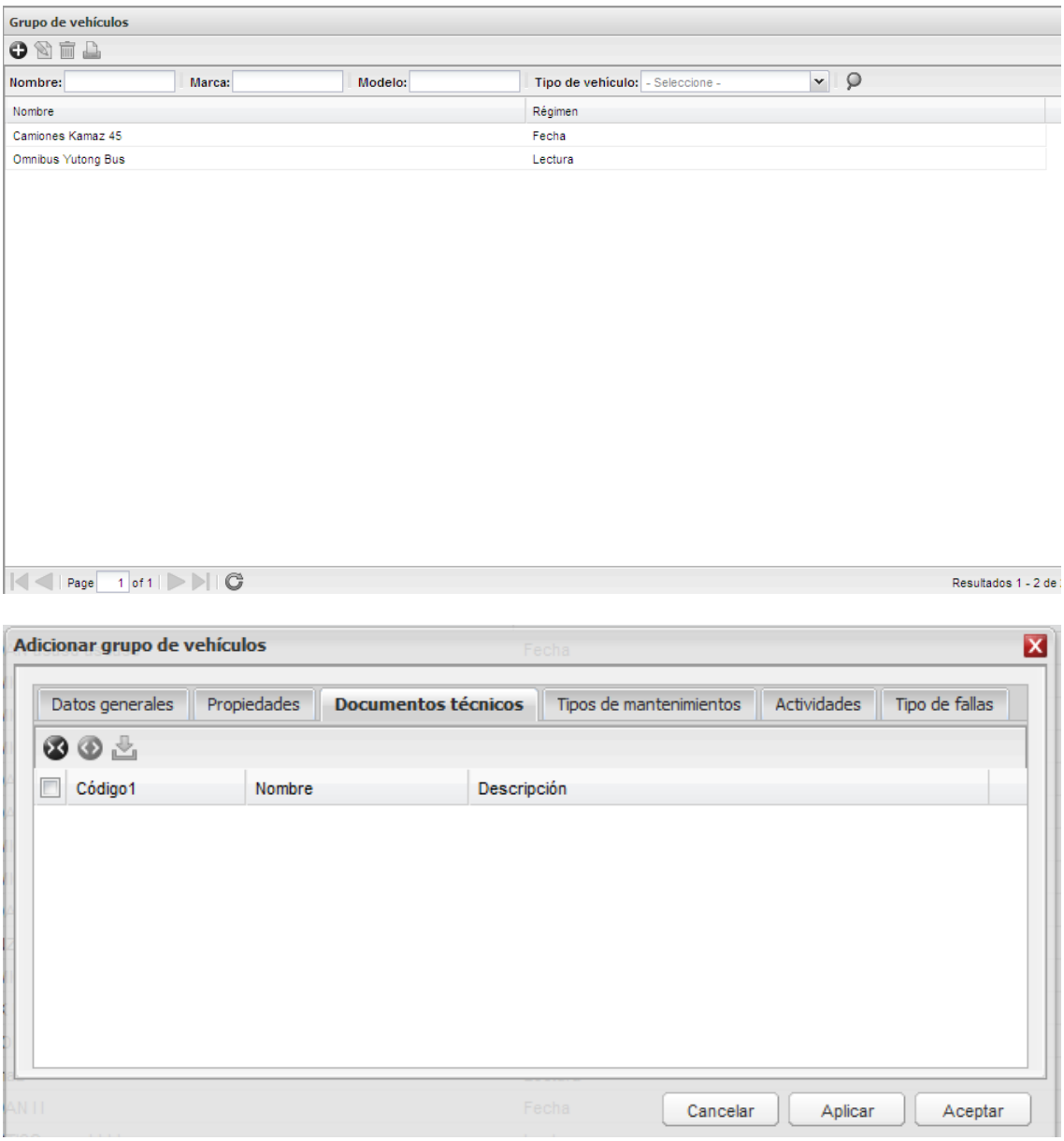


Tabla 24: HU Quitar documentos técnicos.

Número: 15	**Nombre del requisito:** Quitar documentos técnicos**.**
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 10 horas

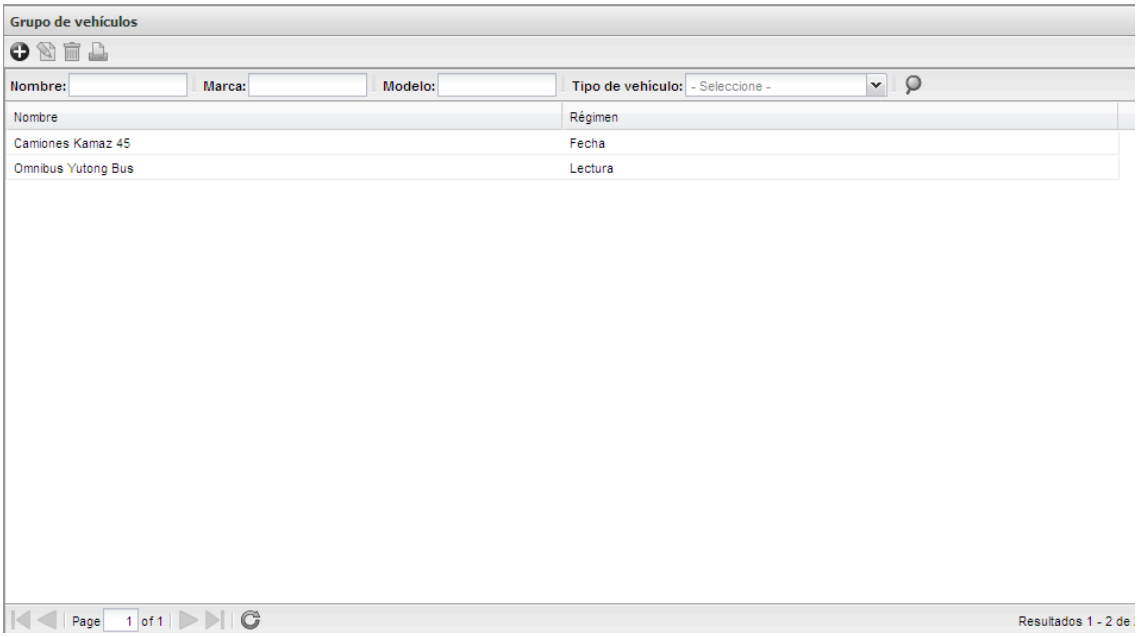
Descripción:

Se debe seleccionar un documento o varios documentos para poder quitarlos del grupo de vehículos.

Observaciones:

Si no se selecciona uno de los documentos, no se puede completar la operación. En el caso de no seleccionar algún documento se puede dejar la operación en blanco.

Prototipo elemental de interfaz gráfica de usuario:



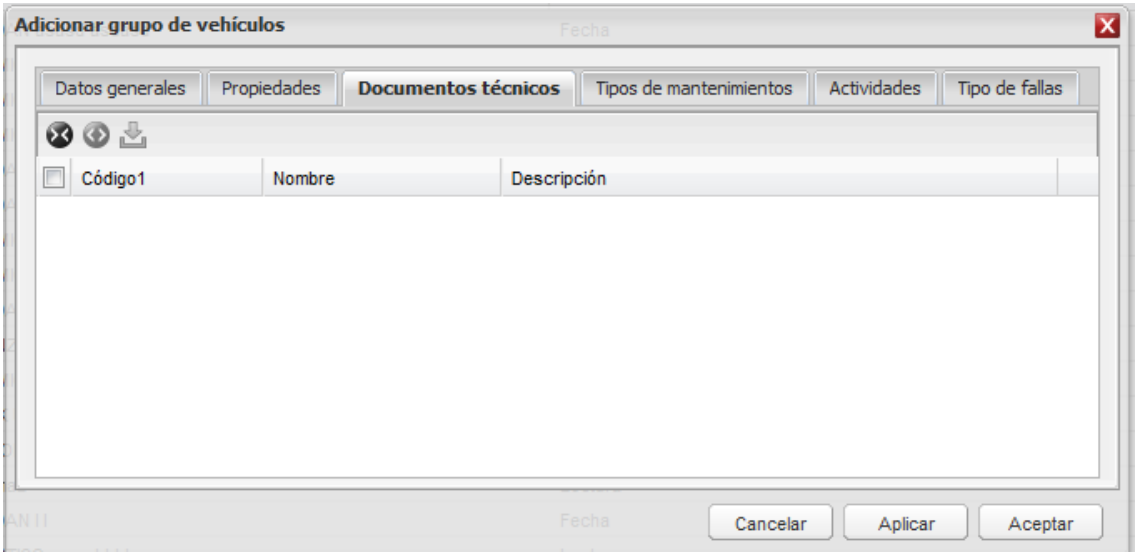
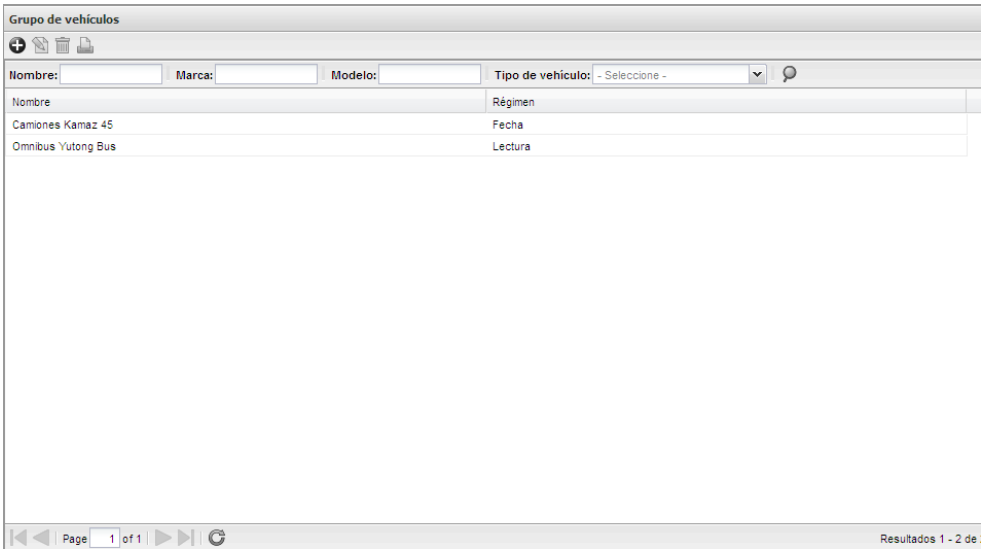


Tabla 25: HU Descargar documentos técnicos.

Número: 16	**Nombre del requisito:** Descargar documentos técnicos**.**
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 10 horas
Descripción: El sistema debe brindar la posibilidad al usuario de gestionar grupos de vehículos determinados. La primera vista **Gestionar grupo de vehículos** contendrá los campos **Nombre** el cual representa el nombre, la marca y el modelo de ese grupo de vehículos, el segundo campo será el **Régimen** identificado por 1 o por 2. Además, en la primera vista se muestra un **Buscar** este elemento permitirá realizar la búsqueda de cualquier campo de los antes mencionados introduciendo el nombre, o la marca, o el modelo, o el tipo de vehículo, con sus campos correspondientes, además esta vista muestra las funcionalidades **Adicionar, Modificar, Eliminar e Imprimir**. La segunda vista **Adicionar grupo de vehículos, pestaña Documentos técnicos** permitirá al usuario descargar un documento técnico determinado.	
Observaciones: Si no se selecciona uno de los documentos, no se puede completar la operación. En el caso de no seleccionar algún documento se puede dejar la operación en blanco.	
Prototipo elemental de interfaz gráfica de usuario:	



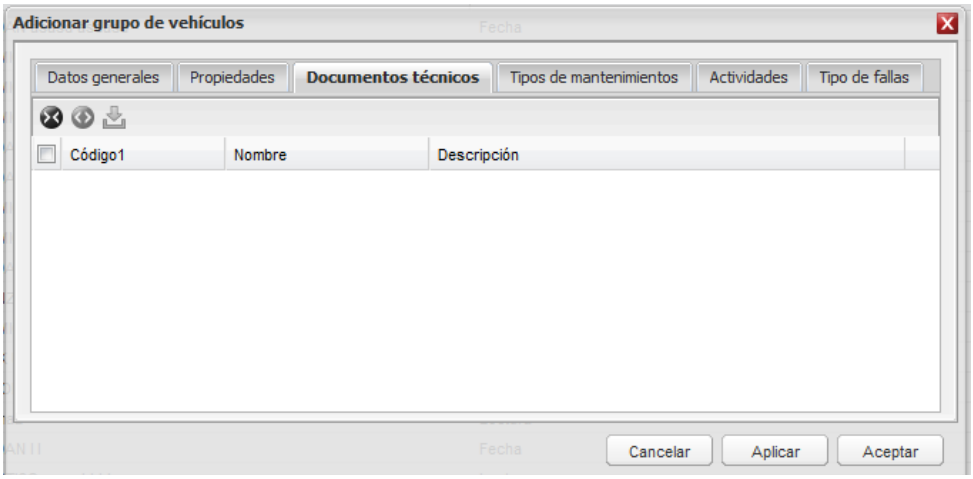


Tabla 26: HU Adicionar tipos de mantenimientos.

Número: 17	**Nombre del requisito:** Adicionar tipos de mantenimientos**.**
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 14 horas
Descripción:	

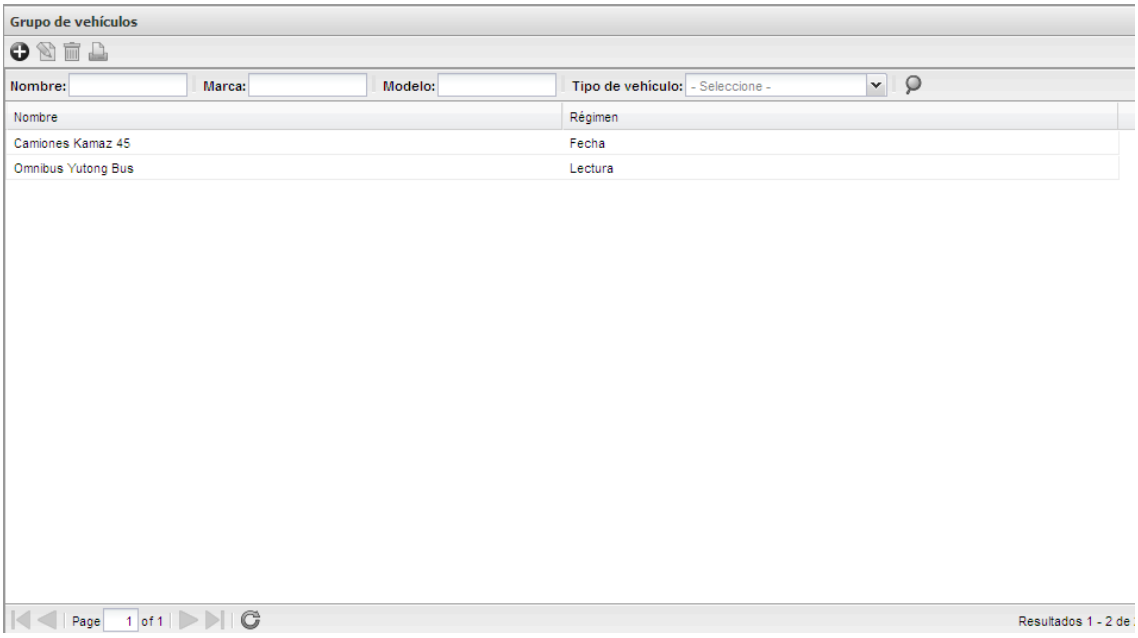
El sistema debe brindar la posibilidad al usuario de gestionar grupos de vehículos determinados. La primera vista **Gestionar grupo de vehículos** contendrá los campos **Nombre** el cual representa el nombre, la marca y el modelo de ese grupo de vehículos, el segundo campo será el **Régimen** identificado por 1 o por 2. Además, en la primera vista se muestra un **Buscar** este elemento permitirá realizar la búsqueda de cualquier campo de los antes mencionados introduciendo el nombre, o la marca, o el modelo, o el tipo de vehículo, con sus campos correspondientes, además esta vista muestra las funcionalidades **Adicionar, Modificar, Eliminar e Imprimir**.

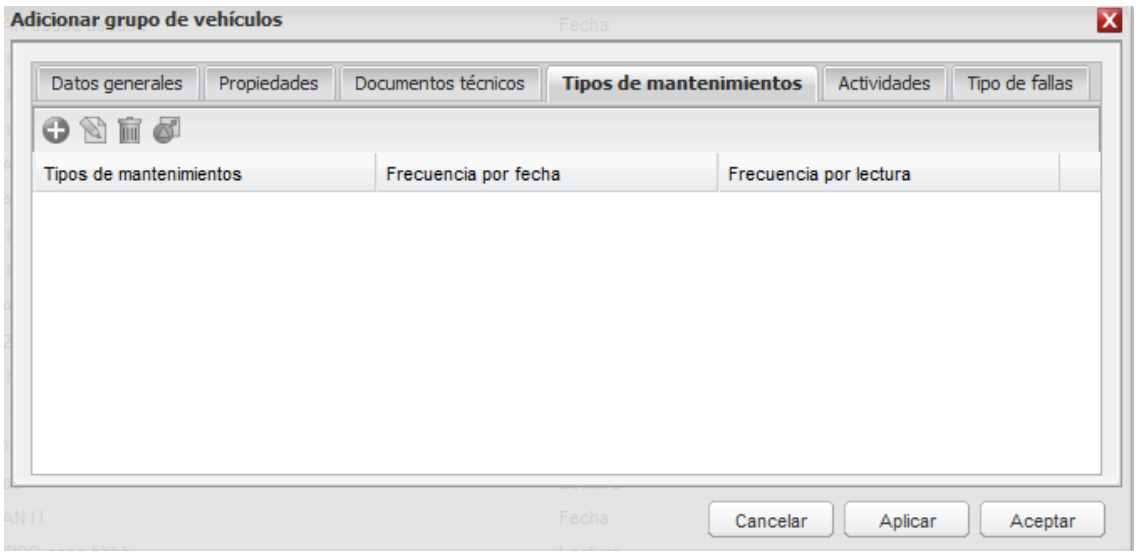
La segunda vista **Adicionar grupo de vehículos, pestaña Tipos de mantenimientos,** permitirá al usuario añadir una propiedad determinada. La misma contará con un campo de selección **Tipo de mantenimiento**, el segundo campo **Frecuencia por fecha,** el tercer campo de selección será **Unidad de fecha,** el cuarto campo es la **Frecuencia por lectura** y el quinto campo es **Unidad de medida**. Cuenta con tres botones para realizar la acción ya sea de **Cancelar, Aplicar** y **Aceptar** la nueva solicitud de adicionar tipo de mantenimiento que se esté.

Observaciones:

Si no se rellenan los componentes de frecuencia por fecha. Frecuencia por lectura, unidad de medida y no se selecciona un tipo de mantenimiento y una unidad de fecha, no se puede completar la operación.

Prototipo elemental de interfaz gráfica de usuario:





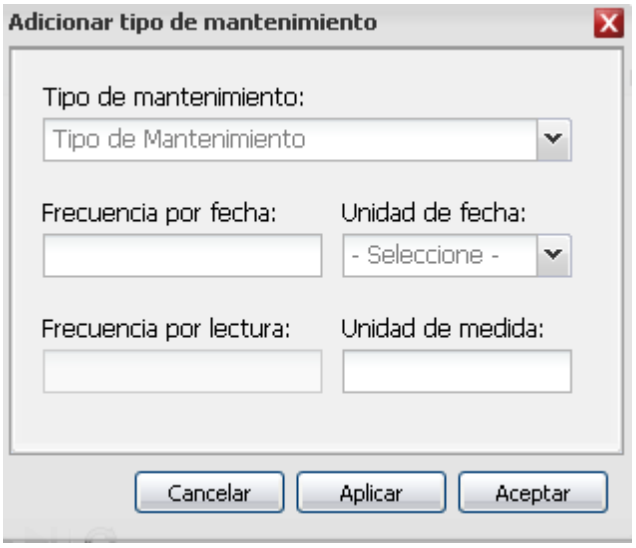


Tabla 27: HU Modificar tipos de mantenimientos.

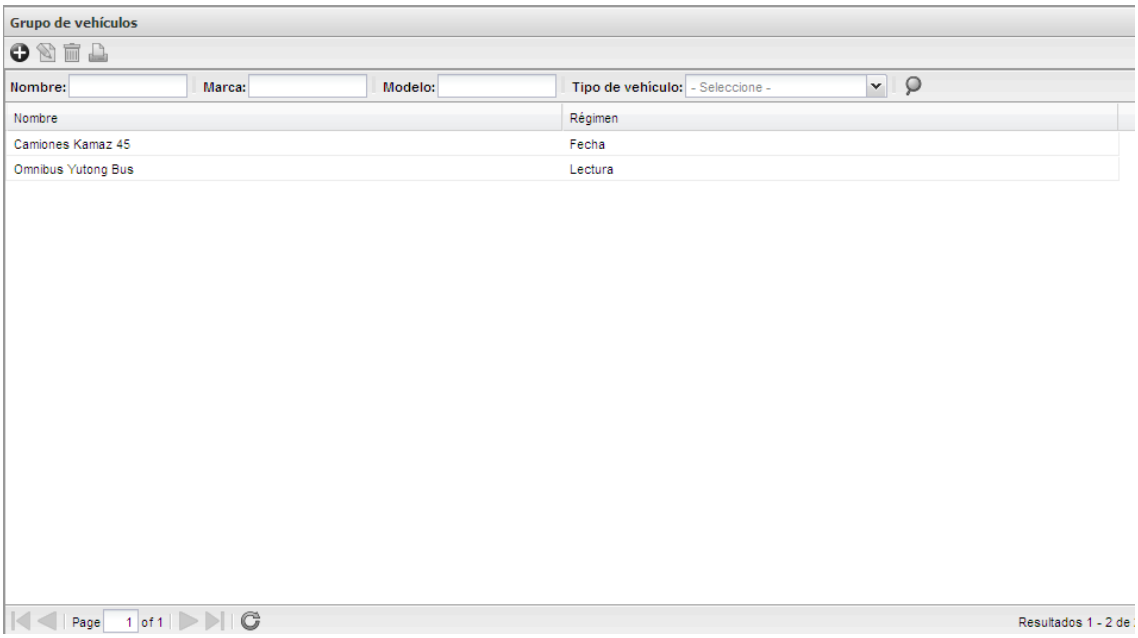
Número: 18	**Nombre del requisito:** Modificar tipos de mantenimientos**.**
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 6 horas
Descripción:	

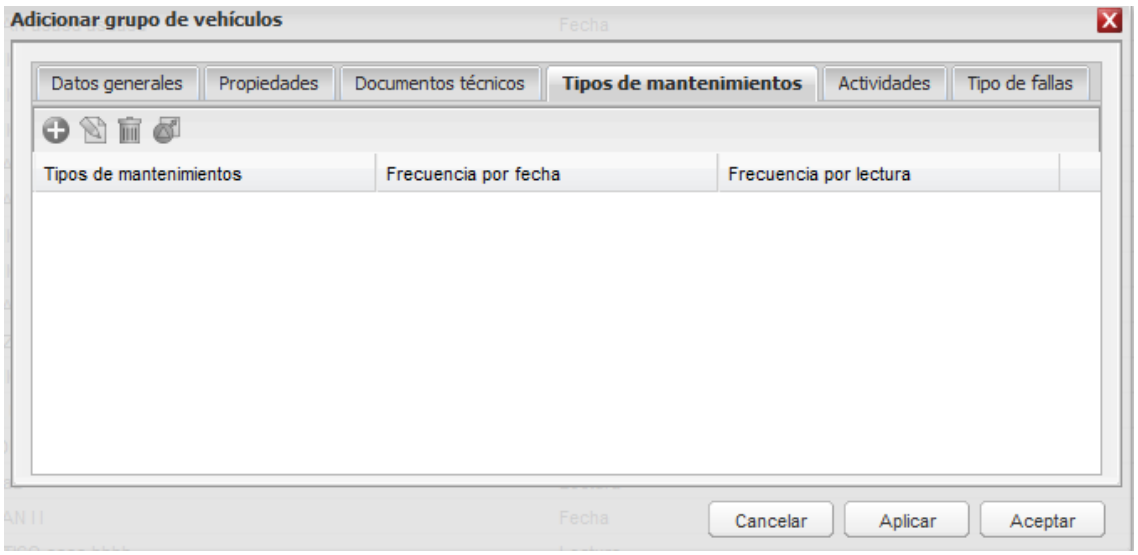
Permite modificar los datos de los tipos de mantenimientos en específico mostrando los mismos campos de la historia de usuario 17 con los datos originales.

Observaciones:

Si deja alguno de los campos en blanco, el sistema se lo marcara en rojo.

Prototipo elemental de interfaz gráfica de usuario:





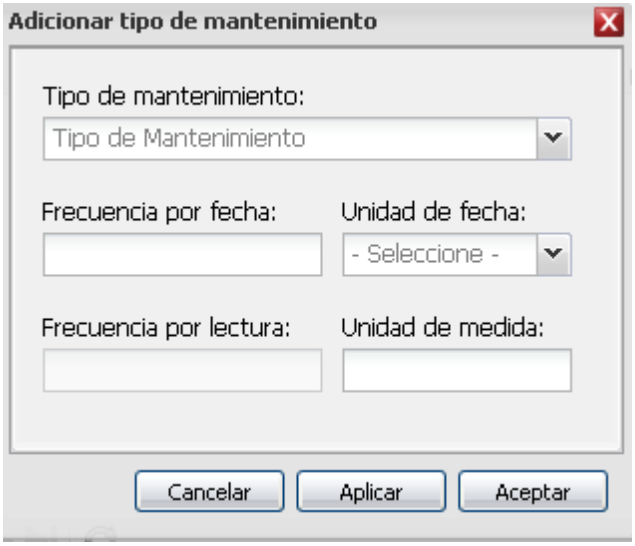


Tabla 28: HU Eliminar tipos de mantenimientos.

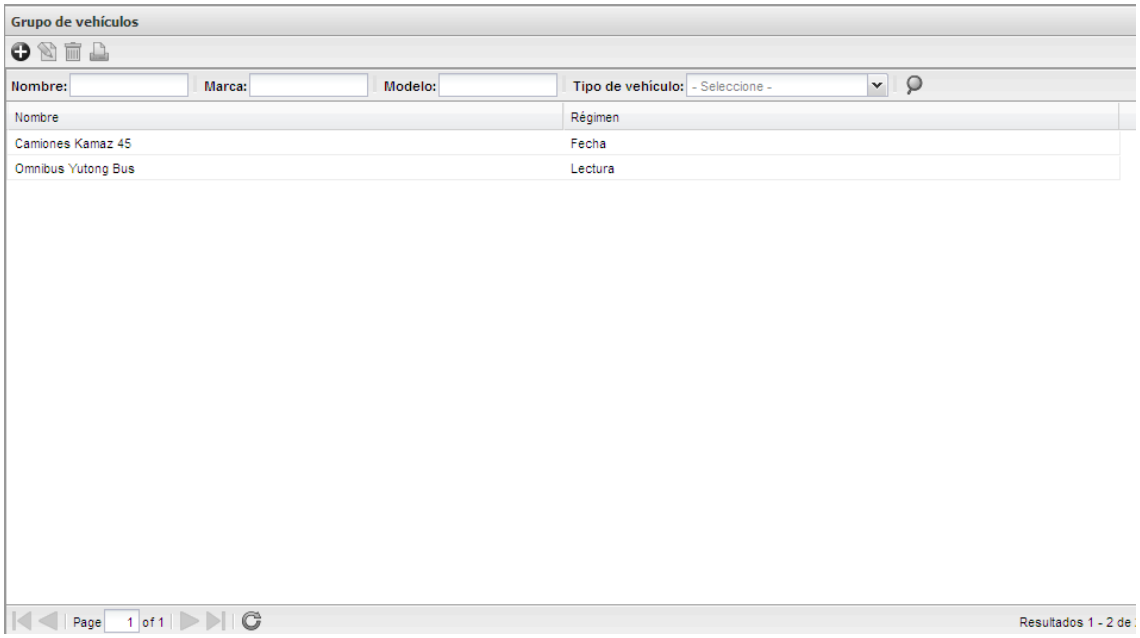
Número: 19	**Nombre del requisito:** Eliminar tipos de mantenimientos**.**
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 10 horas
Descripción:	

Permite eliminar un tipo de mantenimiento seleccionado.

Observaciones:

Si no se selecciona un tipo de mantenimiento, no se puede completar la operación.

Prototipo elemental de interfaz gráfica de usuario:



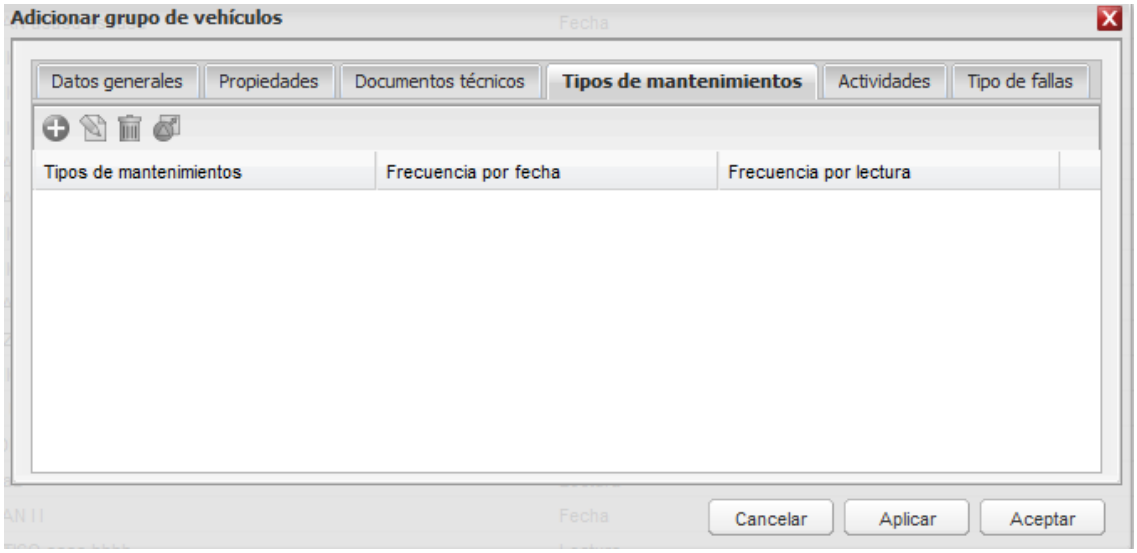


Tabla 29: HU Incluir tipos de mantenimientos.

Número: 20	**Nombre del requisito:** Incluir tipos de mantenimientos**.**
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 9 horas

Descripción:

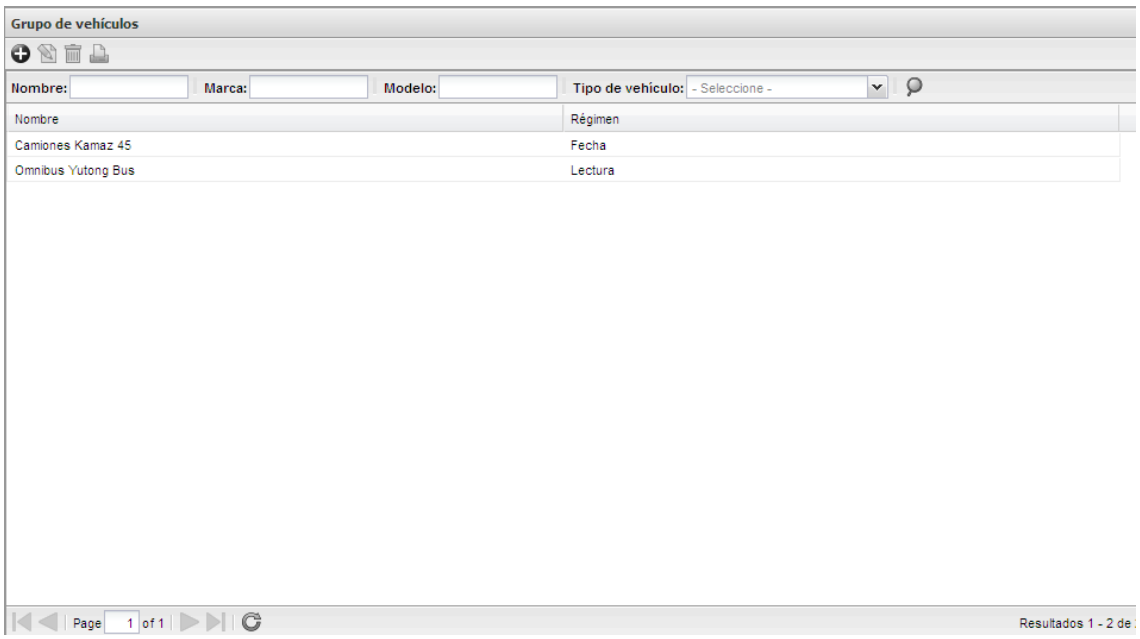
El sistema debe brindar la posibilidad al usuario de gestionar grupos de vehículos determinados. La primera vista **Gestionar grupo de vehículos** contendrá los campos **Nombre** el cual representa el nombre, la marca y el modelo de ese grupo de vehículos, el segundo campo será el **Régimen** identificado por 1 o por 2. Además, en la primera vista se muestra un **Buscar** este elemento permitirá realizar la búsqueda de cualquier campo de los antes mencionados introduciendo el nombre, o la marca, o el modelo, o el tipo de vehículo, con sus campos correspondientes, además esta vista muestra las funcionalidades **Adicionar, Modificar, Eliminar e Imprimir**.

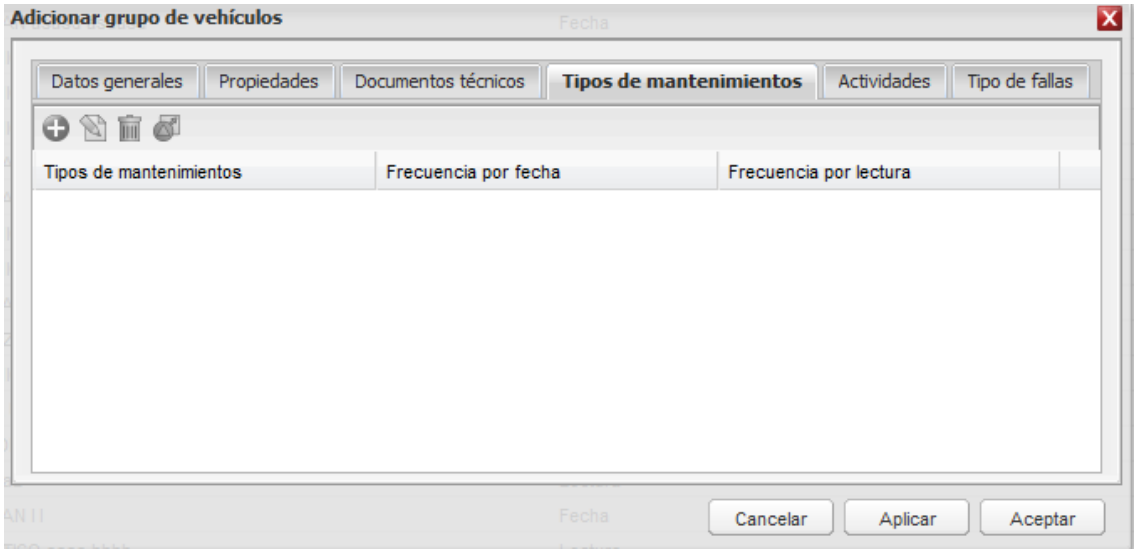
La segunda vista **Adicionar grupo de vehículos, pestaña Tipos de mantenimientos,** permitirá al usuario incluir un tipo de mantenimiento determinado. La misma contará con un campo de selección con el tipo de mantenimiento. Cuenta con dos botones para realizar la acción ya sea de **Cancelar** y **Aceptar** la nueva solicitud de inclusión a los tipos de mantenimientos que se esté.

Observaciones:

Si no se selecciona un tipo de mantenimiento, no se puede completar la operación.

Prototipo elemental de interfaz gráfica de usuario:


Grupo de vehículos
Nombre:
Marca:
Modelo:
Tipo de vehículo:
- Seleccione -
Nombre
Régimen
Camiones Kamaz 45
Fecha
Omnibus Yutong Bus
Lectura
Page 1 of 1
Resultados 1 - 2 de


Adicionar grupo de vehículos
Datos generales
Propiedades
Documentos técnicos
Tipos de mantenimientos
Actividades
Tipo de fallas
Tipos de mantenimientos
Frecuencia por fecha
Frecuencia por lectura
Cancelar
Aplicar
Aceptar

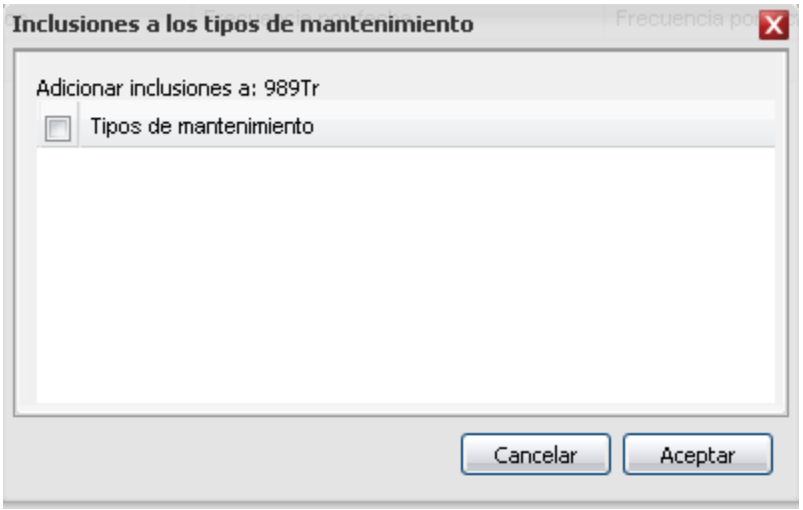

Inclusiones a los tipos de mantenimiento
Adicionar inclusiones a: 989Tr
Tipos de mantenimiento
Cancelar
Aceptar

Tabla 30: HU Adicionar actividades.

Número: 21	**Nombre del requisito:** Adicionar actividades**.**
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 14 horas

Descripción:

El sistema debe brindar la posibilidad al usuario de gestionar grupos de vehículos determinados. La primera vista **Gestionar grupo de vehículos** contendrá los campos **Nombre** el cual representa el nombre, la marca y el modelo de ese grupo de vehículos, el segundo campo será el **Régimen** identificado por 1 o por 2. Además, en la primera vista se muestra un **Buscar** este elemento permitirá realizar la búsqueda de cualquier campo de los antes mencionados introduciendo el nombre, o la marca, o el modelo, o el tipo de vehículo, con sus campos correspondientes, además esta vista muestra las funcionalidades **Adicionar, Modificar, Eliminar e Imprimir**.

La segunda vista **Adicionar grupo de vehículos, pestaña Actividades** permitirá al usuario añadir una actividad determinada. La misma contará con un listado con los campos **Selección**, el segundo campo será **Nombre** y el tercer campo **Duración**. Cuenta con tres botones para realizar la acción ya sea de **Cancelar, Aplicar** o **Aceptar** la nueva solicitud de gestionar una actividad que se esté creando, además esta vista muestra las funcionalidades **Adicionar, Modificar** y **Eliminar**.

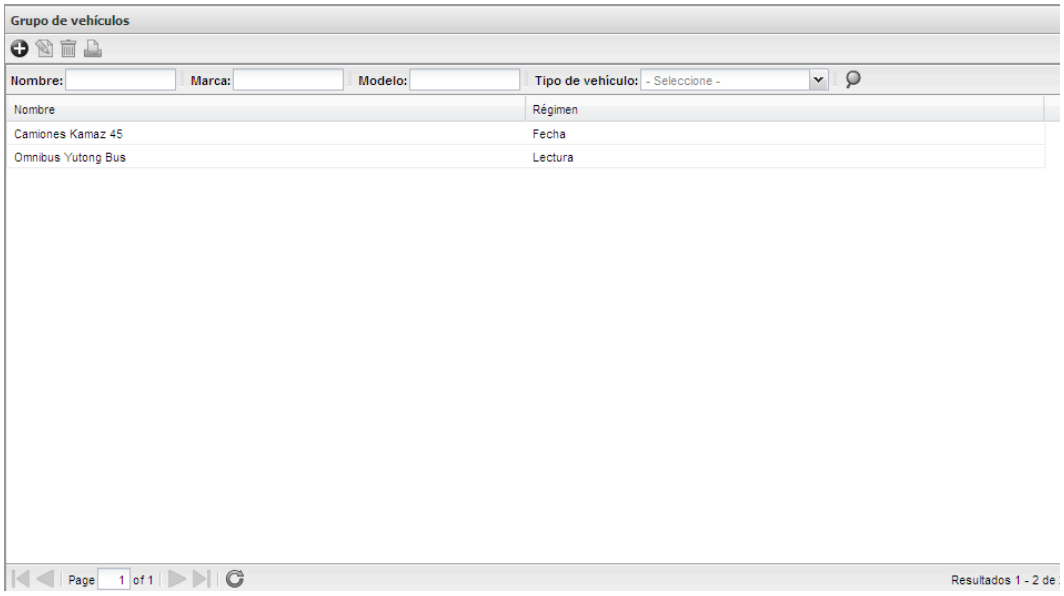
La tercera vista **Adicionar actividad** permitirá al usuario añadir una actividad determinada. La misma contará con el campo **Nombre**, el segundo campo será **Nombre**, el tercer campo será la **Duración,** permitiendo insertar en el campo de **Horas** y en el campo de **Minutos**. Cuenta con tres pestañas. La pestaña **Tipo de mantenimiento**, posee una lista con los campos **Tipos de mantenimiento, Frecuencia por fechan** y **Frecuencia por lectura**. La pestaña **Repuestos**, posee una lista con los campos **Código, Descripción, Unidad Medida** y **Cantidad,** además cuenta con dos botones uno de **Asociar repuestos** y otro de **Quitar repuestos**. La pestaña **Descripción**, posee un cuadro de texto. Cuenta con botones para realizar la acción ya

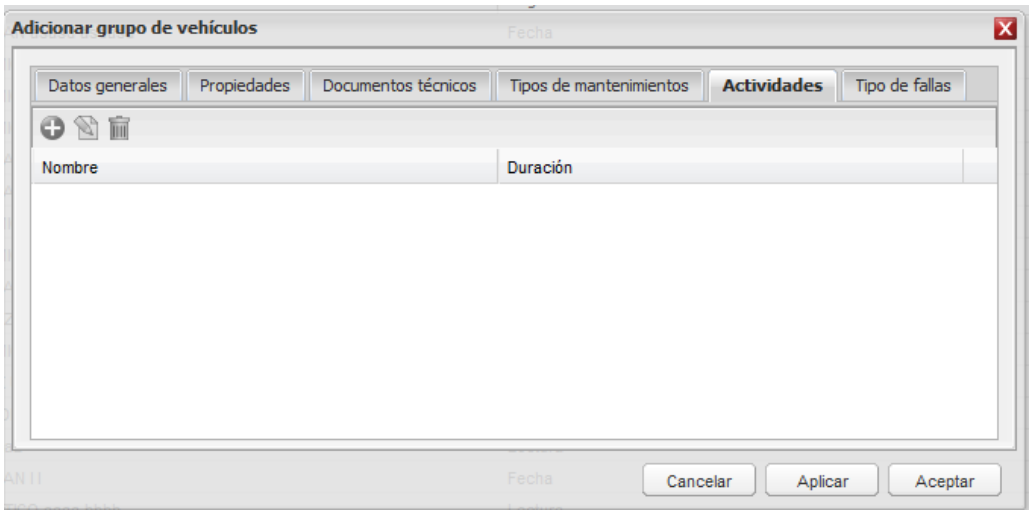
sea de **Cancelar, Aplicar** o **Aceptar** la nueva solicitud de gestionar una actividad que se esté creando.

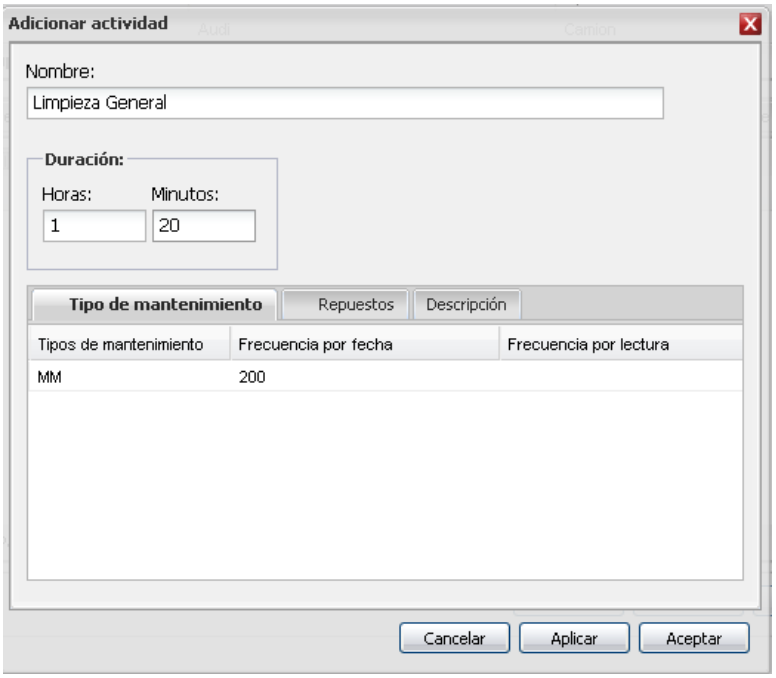
Observaciones:

Si no se activa uno de los componentes de selección, si no se rellenan los componentes de marca y modelo y no se selecciona un tipo de vehículo, no se puede completar la operación. En el caso de activar lectura y no seleccionar un tipo de unidad, entonces no se puede completar la operación.

Prototipo elemental de interfaz gráfica de usuario:







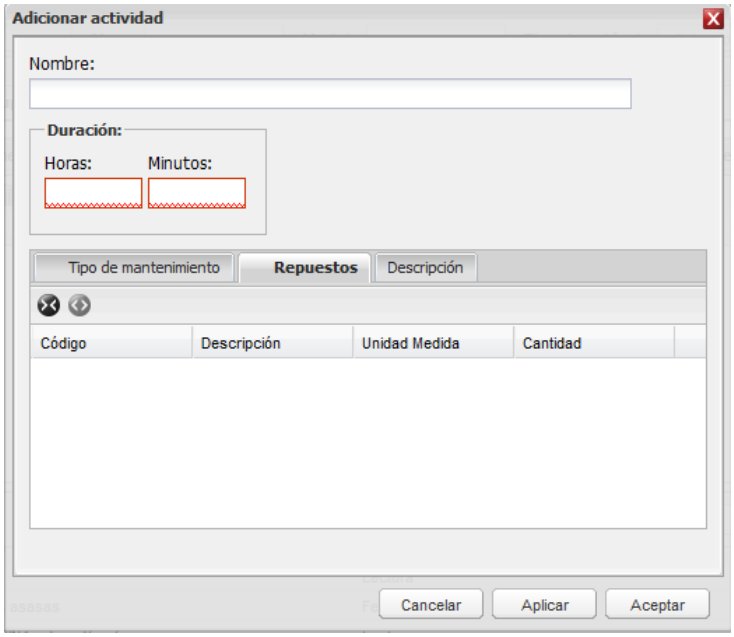


Tabla 31: HU Asociar repuestos.

Número: 22	**Nombre del requisito:** Asociar repuestos**.**
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1

Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 14 horas

Descripción:

El sistema debe brindar la posibilidad al usuario de gestionar grupos de vehículos determinados. La primera vista **Gestionar grupo de vehículos** contendrá los campos **Nombre** el cual representa el nombre, la marca y el modelo de ese grupo de vehículos, el segundo campo será el **Régimen** identificado por 1 o por 2. Además, en la primera vista se muestra un **Buscar** este elemento permitirá realizar la búsqueda de cualquier campo de los antes mencionados introduciendo el nombre, o la marca, o el modelo, o el tipo de vehículo, con sus campos correspondientes, además esta vista muestra las funcionalidades **Adicionar, Modificar, Eliminar e Imprimir**.

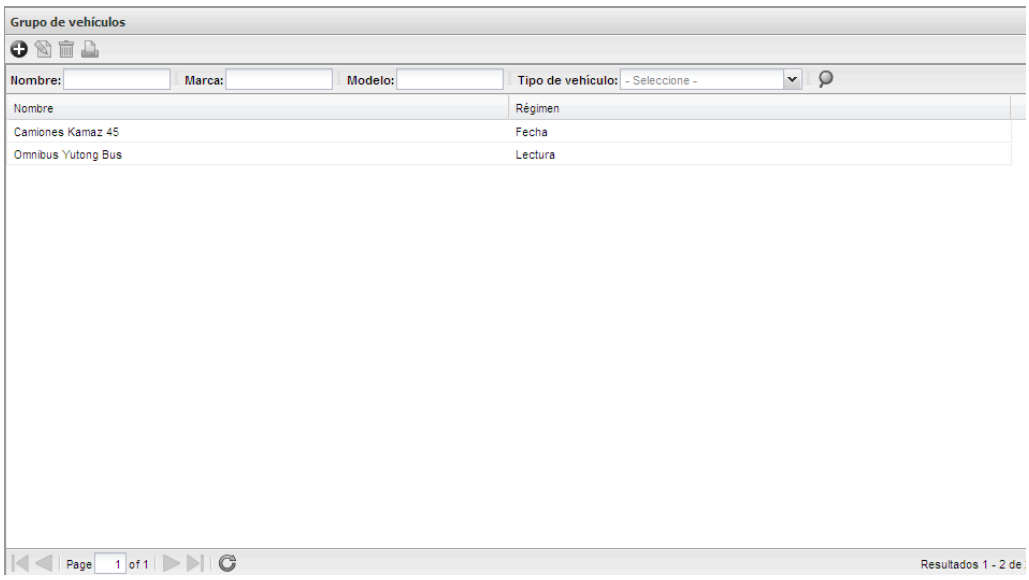
La segunda vista **Adicionar grupo de vehículos, pestaña Actividades** permitirá al usuario añadir una actividad determinada. La misma contará con un listado con los campos **Selección**, el segundo campo será **Nombre** y el tercer campo **Duración**. Cuenta con tres botones para realizar la acción ya sea de **Cancelar, Aplicar** o **Aceptar** la nueva solicitud de gestionar una actividad que se esté creando, además esta vista muestra las funcionalidades **Adicionar, Modificar** y **Eliminar**.

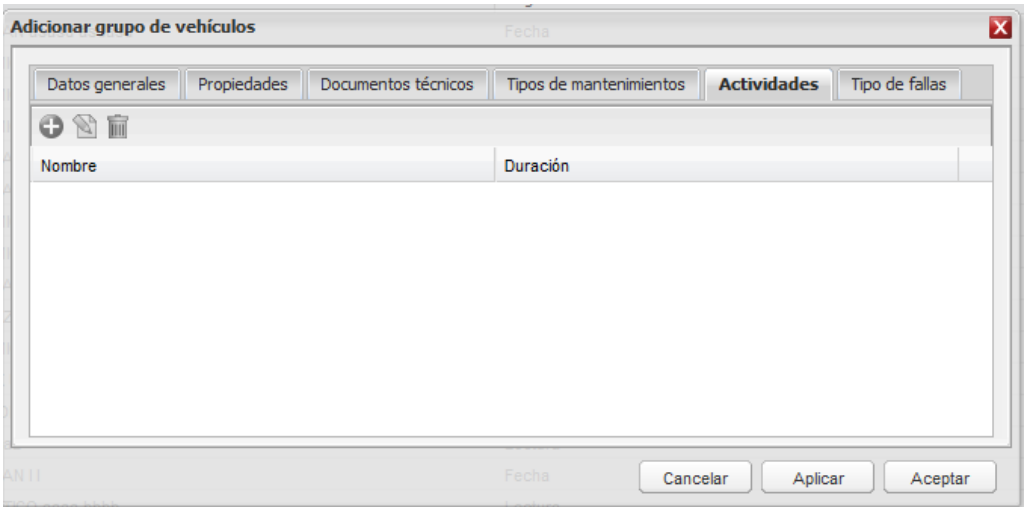
La tercera vista **Adicionar actividad** permitirá al usuario añadir una actividad determinada. La misma contará con el campo **Nombre**, el segundo campo será **Nombre**, el tercer campo será la **Duración,** permitiendo insertar en el campo de **Horas** y en el campo de **Minutos**. Se le asocia el tipo de repuesto que se quiera.

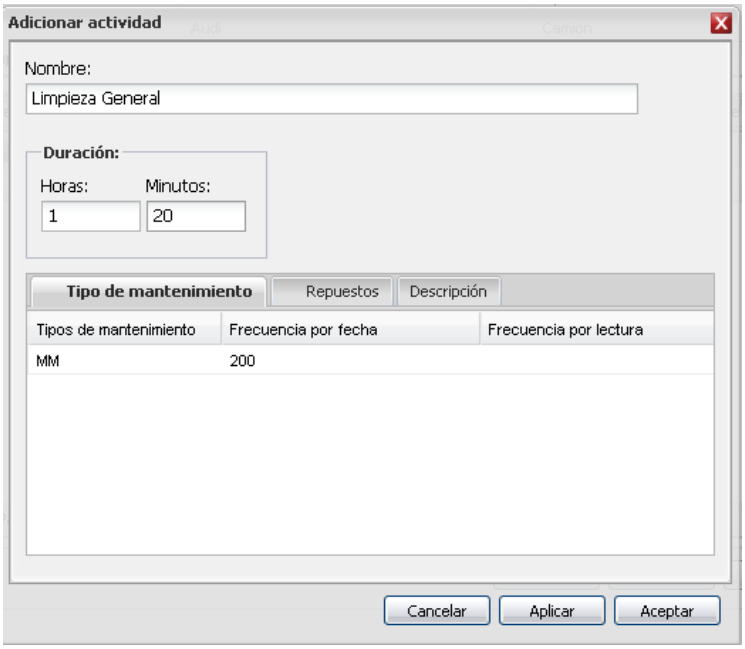
Observaciones:

Si no se activa uno de los componentes de selección, si no se rellenan los componentes de marca y modelo y no se selecciona un tipo de vehículo, no se puede completar la operación. En el caso de activar lectura y no seleccionar un tipo de unidad, entonces no se puede completar la operación.

Prototipo elemental de interfaz gráfica de usuario:


Grupo de vehículos
Nombre:
Marca:
Modelo:
Tipo de vehículo:
- Seleccione -
Nombre
Régimen
Camiones Kamaz 45
Fecha
Omnibus Yutong Bus
Lectura
Page 1 of 1
Resultados 1 - 2 de


Adicionar grupo de vehículos
Datos generales
Propiedades
Documentos técnicos
Tipos de mantenimientos
Actividades
Tipo de fallas
Nombre
Duración
Cancelar
Aplicar
Aceptar



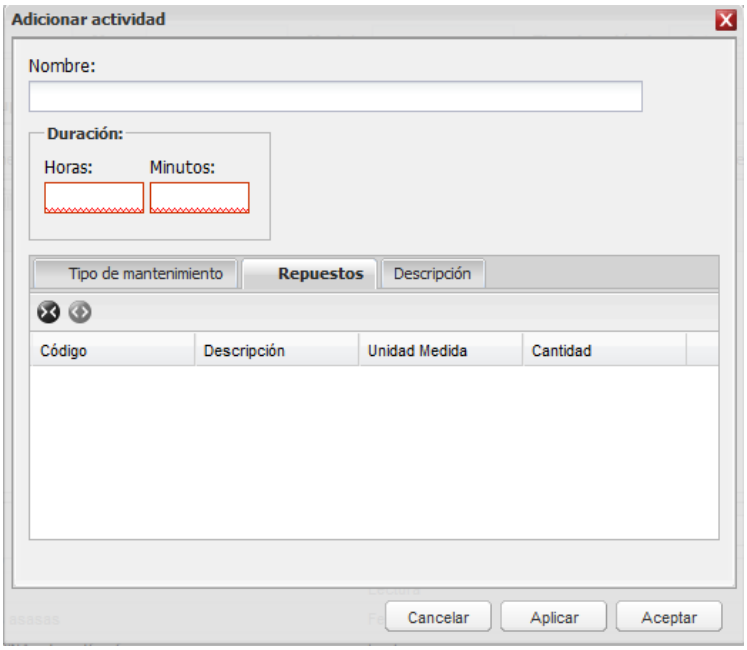


Tabla 32: HU Quitar repuestos.

Número: 23	**Nombre del requisito:** Quitar repuestos**.**
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1

Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 14 horas

Descripción:

El sistema debe brindar la posibilidad al usuario de gestionar grupos de vehículos determinados. La primera vista **Gestionar grupo de vehículos** contendrá los campos **Nombre** el cual representa el nombre, la marca y el modelo de ese grupo de vehículos, el segundo campo será el **Régimen** identificado por 1 o por 2. Además, en la primera vista se muestra un **Buscar** este elemento permitirá realizar la búsqueda de cualquier campo de los antes mencionados introduciendo el nombre, o la marca, o el modelo, o el tipo de vehículo, con sus campos correspondientes, además esta vista muestra las funcionalidades **Adicionar, Modificar, Eliminar e Imprimir**.

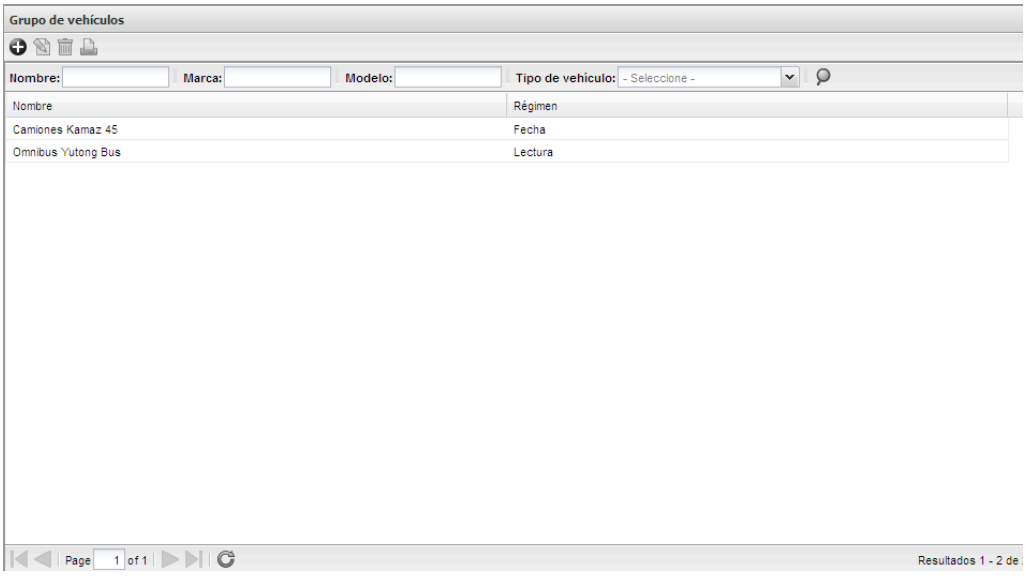
La segunda vista **Adicionar grupo de vehículos, pestaña Actividades** permitirá al usuario añadir una actividad determinada. La misma contará con un listado con los campos **Selección**, el segundo campo será **Nombre** y el tercer campo **Duración**. Cuenta con tres botones para realizar la acción ya sea de **Cancelar, Aplicar** o **Aceptar** la nueva solicitud de gestionar una actividad que se esté creando, además esta vista muestra las funcionalidades **Adicionar, Modificar** y **Eliminar**.

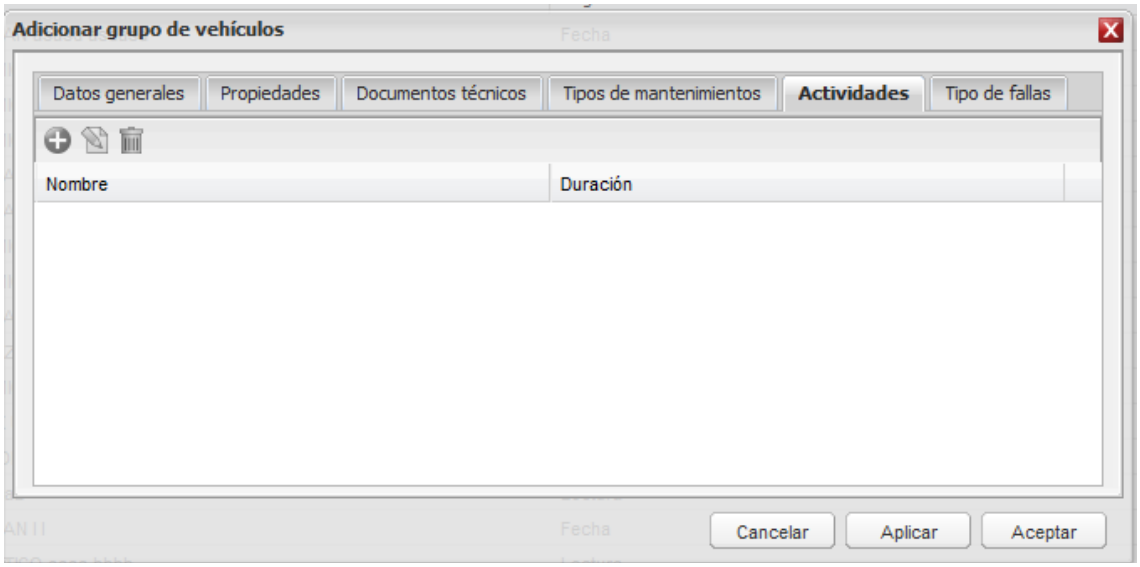
La tercera vista **Adicionar actividad** permitirá al usuario añadir una actividad determinada. La misma contará con el campo **Nombre**, el segundo campo será **Nombre**, el tercer campo será la **Duración,** permitiendo insertar en el campo de **Horas** y en el campo de **Minutos**. Se le quita el tipo de repuesto que se quiera.

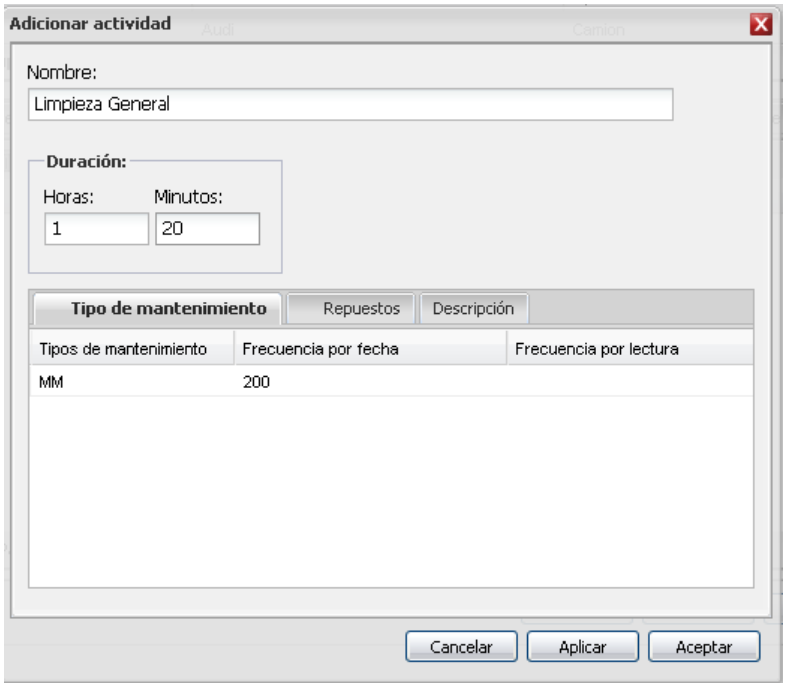
Observaciones:

Si no se activa uno de los componentes de selección, si no se rellenan los componentes de marca y modelo y no se selecciona un tipo de vehículo, no se puede completar la operación. En el caso de activar lectura y no seleccionar un tipo de unidad, entonces no se puede completar la operación.

Prototipo elemental de interfaz gráfica de usuario:


Grupo de vehículos
Nombre:
Marca:
Modelo:
Tipo de vehículo:
- Seleccione -
Nombre
Régimen
Camiones Kamaz 45
Fecha
Omnibus Yutong Bus
Lectura
Page 1 of 1
Resultados 1 - 2 de


Adicionar grupo de vehículos
Datos generales
Propiedades
Documentos técnicos
Tipos de mantenimientos
Actividades
Tipo de fallas
Nombre
Duración
Cancelar
Aplicar
Aceptar



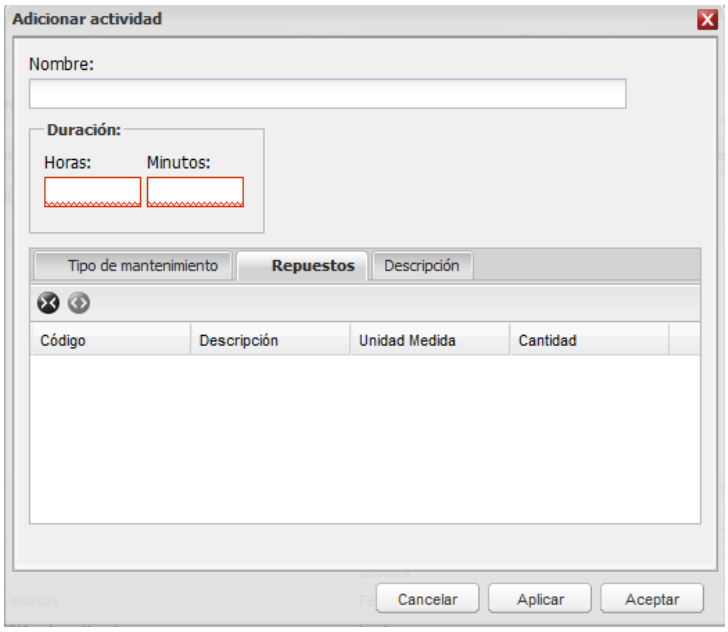


Tabla 33: HU Modificar actividades.

Número: 24	**Nombre del requisito:** Modificar actividades**.**
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1

Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 6 horas

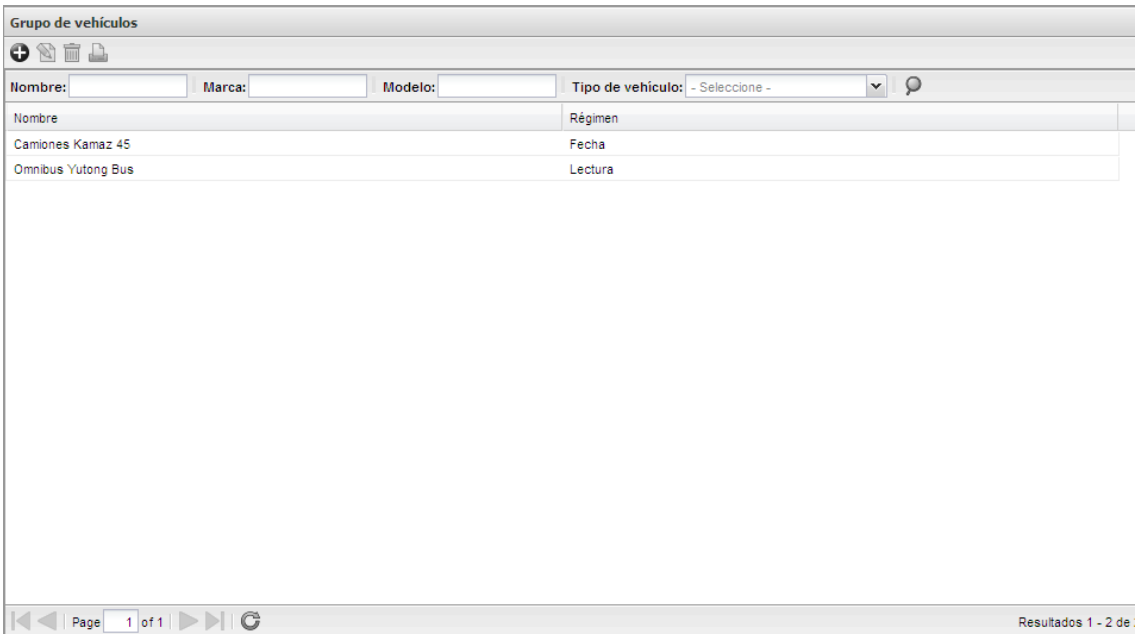
Descripción:

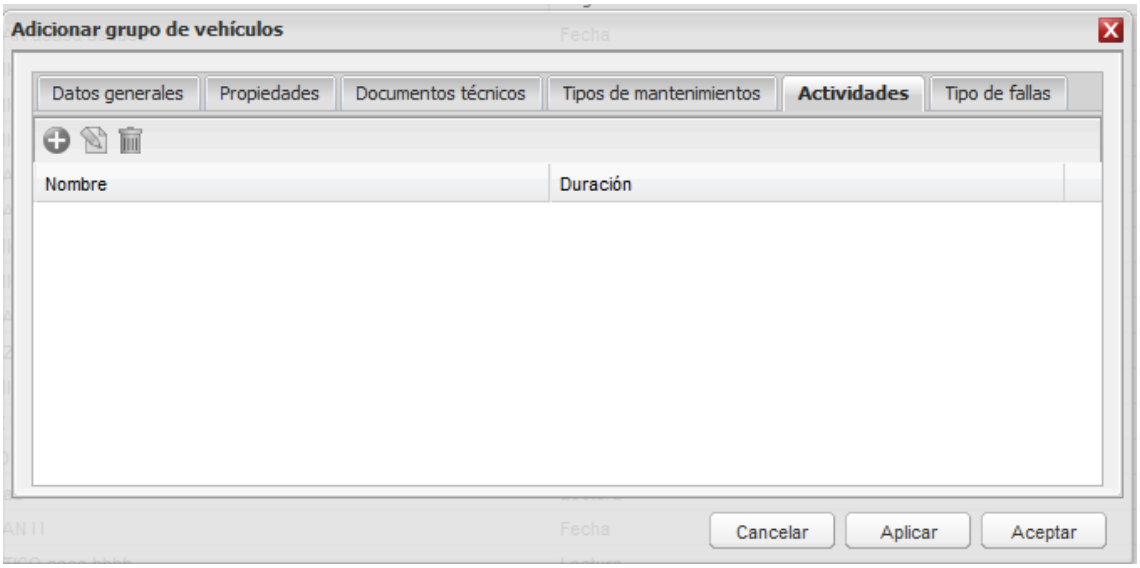
Permite modificar las actividades en específico mostrando los mismos campos de la historia de usuario 19 con los datos originales.

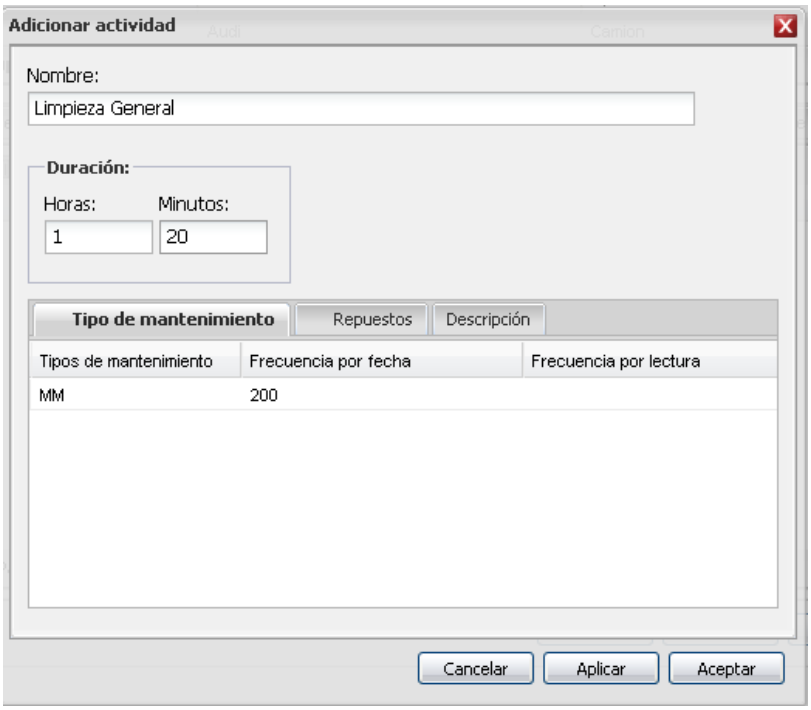
Observaciones:

Si deja alguno de los campos en blanco, el sistema se lo marcara en rojo.

Prototipo elemental de interfaz gráfica de usuario:




Adicionar grupo de vehículos
Datos generales
Propiedades
Documentos técnicos
Tipos de mantenimientos
Actividades
Tipo de fallas
Nombre
Duración
Cancelar
Aplicar
Aceptar


Adicionar actividad
Nombre:
Limpieza General
Duración:
Horas:
Minutos:
1
20
Tipo de mantenimiento
Repuestos
Descripción
Tipos de mantenimiento
Frecuencia por fecha
Frecuencia por lectura
MM
200
Cancelar
Aplicar
Aceptar

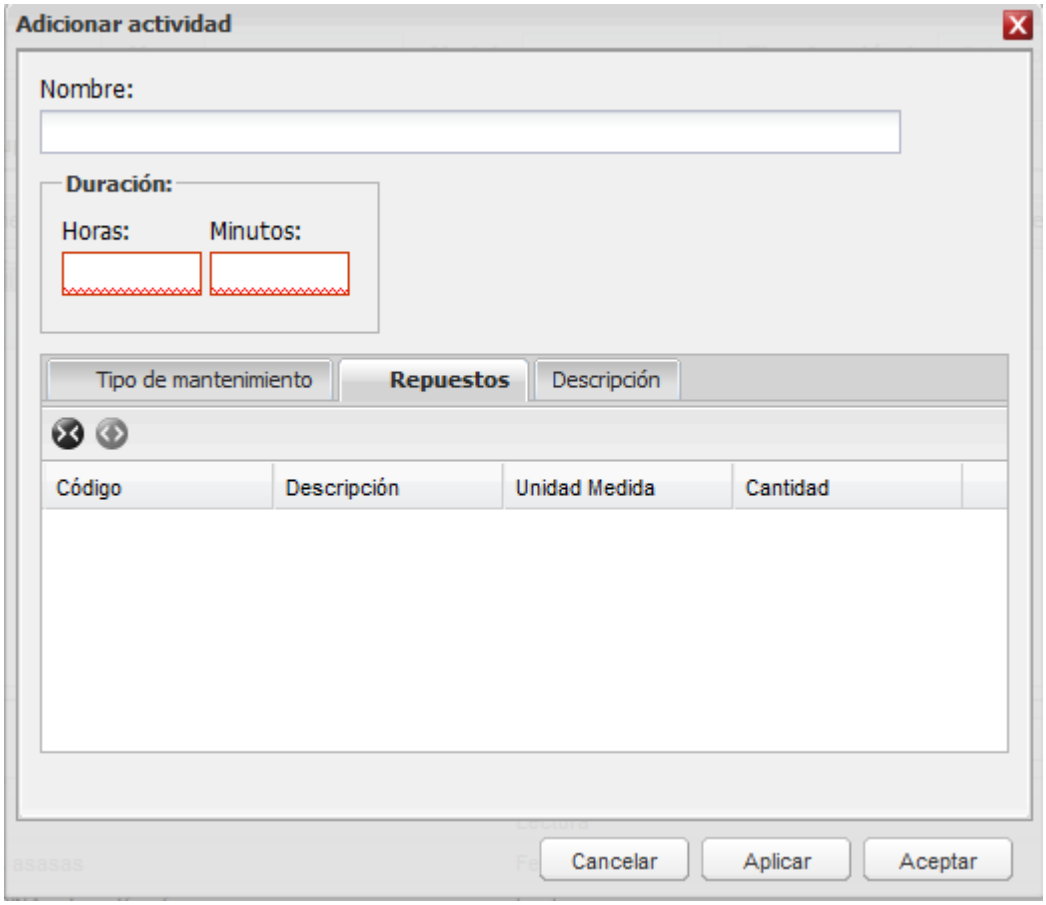
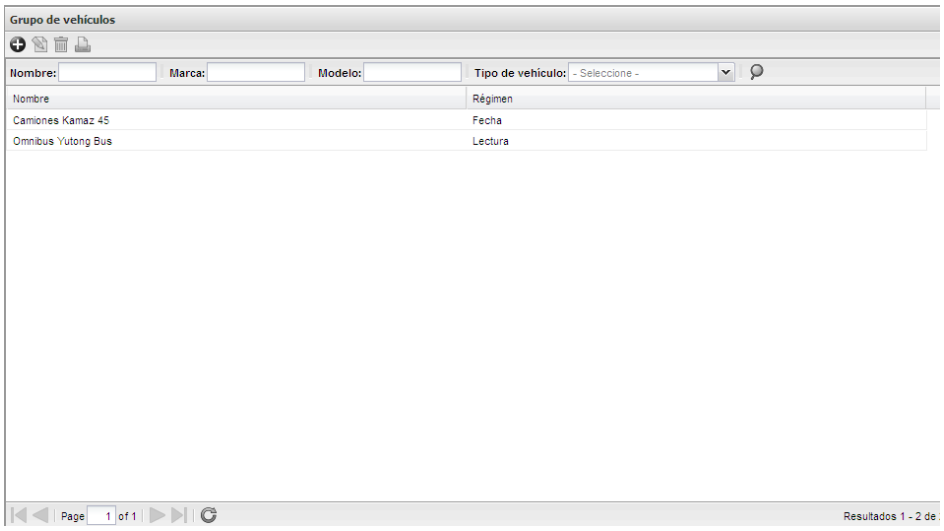


Tabla 34: HU Eliminar actividades.

Número: 25	**Nombre del requisito:** Eliminar actividades**.**
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 10 horas
Descripción: Permite eliminar una actividad seleccionada.	
Observaciones: Si no se selecciona una actividad, no se puede completar la operación.	

Prototipo elemental de interfaz gráfica de usuario:



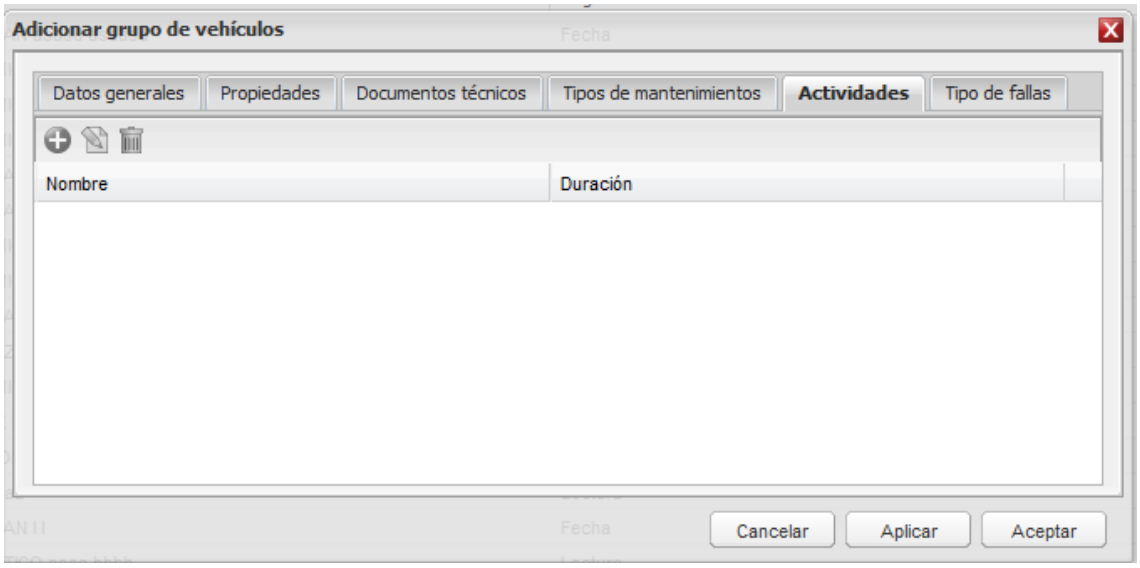


Tabla 35: HU Adicionar tipo de fallas

Número: 26	**Nombre del requisito:** Adicionar tipo de fallas**.**
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 4 horas

Descripción:

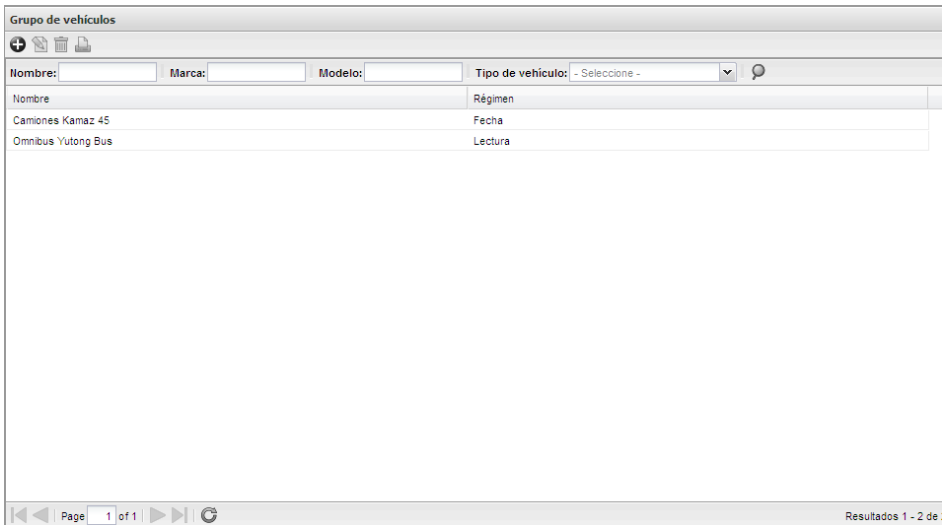
El sistema debe brindar la posibilidad al usuario de adicionar fallas determinadas. La primera vista **Adicionar grupo de vehículos, pestaña Tipos de fallas** contendrá las funcionalidades **Adicionar, Modificar y Eliminar**.

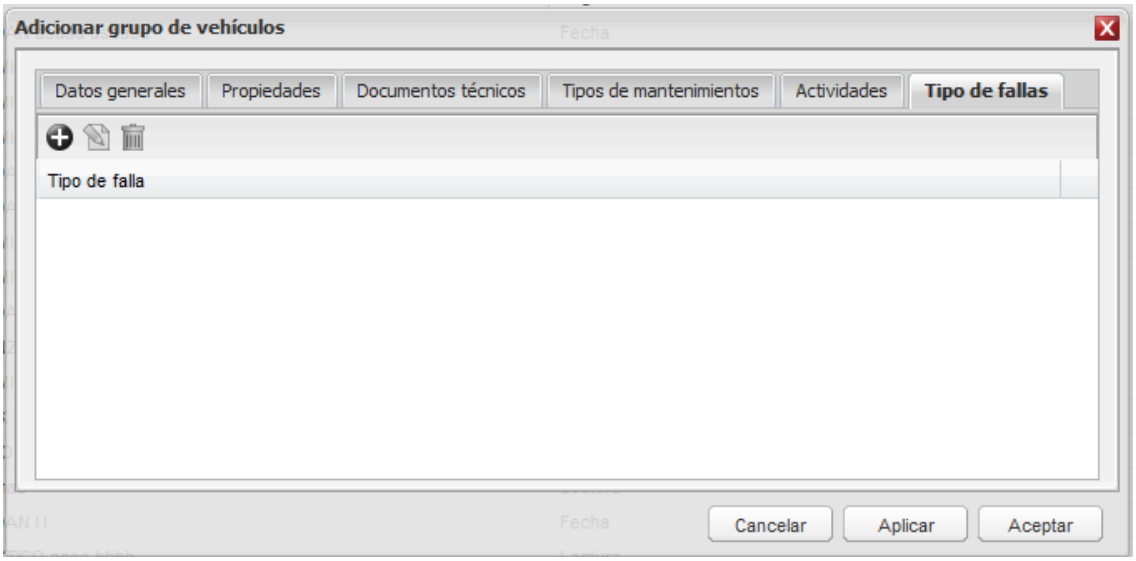
La segunda vista **Adicionar Tipo de falla** permitirá al usuario añadir un tipo de falla determinada. La misma contará con el campos **Tipo de falla**, además esta vista cuenta con tres botones para realizar la acción ya sea de **Cancelar, Aplicar** o **Aceptar** la nueva solicitud de adicionar un tipo de falla que se esté creando.

Observaciones:

Si no se rellena el campo de texto **Tipo de falla**, no se puede completar la operación.

Prototipo elemental de interfaz gráfica de usuario:





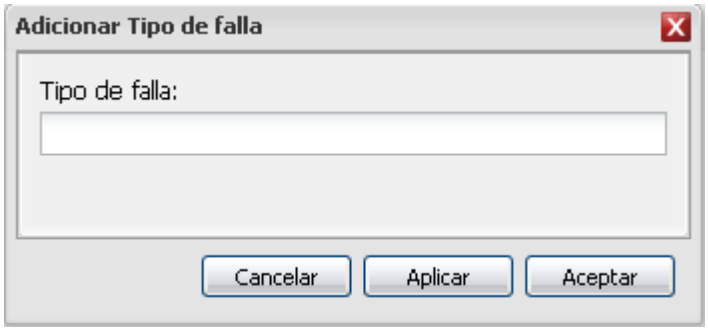
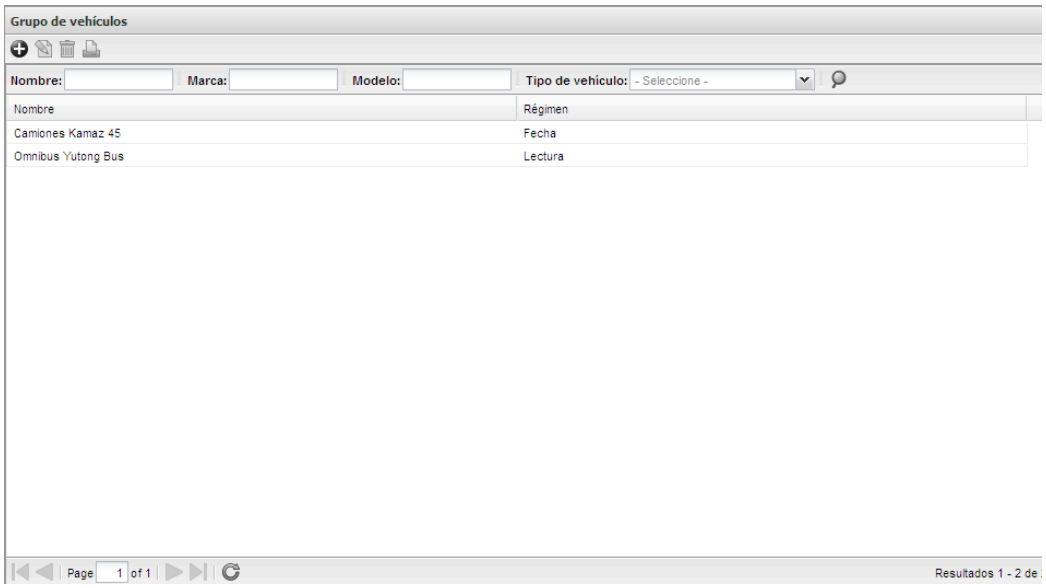


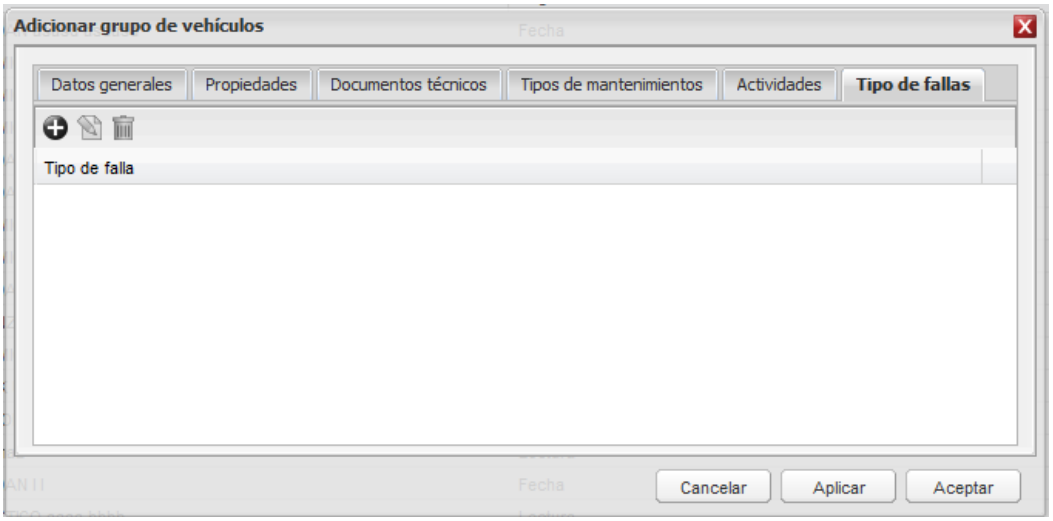
Tabla 36: HU Modificar tipo de fallas.

Número: 27	**Nombre del requisito:** Modificar tipo de fallas.
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 4 horas
Descripción: Permite modificar el tipo de falla en específico mostrando los mismos campos de la historia de usuario 26 con los datos originales.	
Observaciones:	

Si deja alguno de los campos en blanco, el sistema se lo marcara en rojo.

Prototipo elemental de interfaz gráfica de usuario:





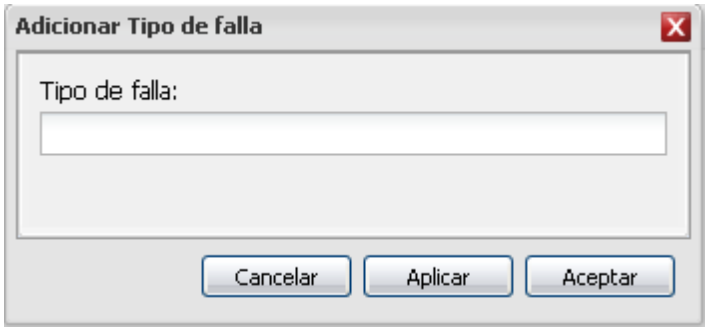
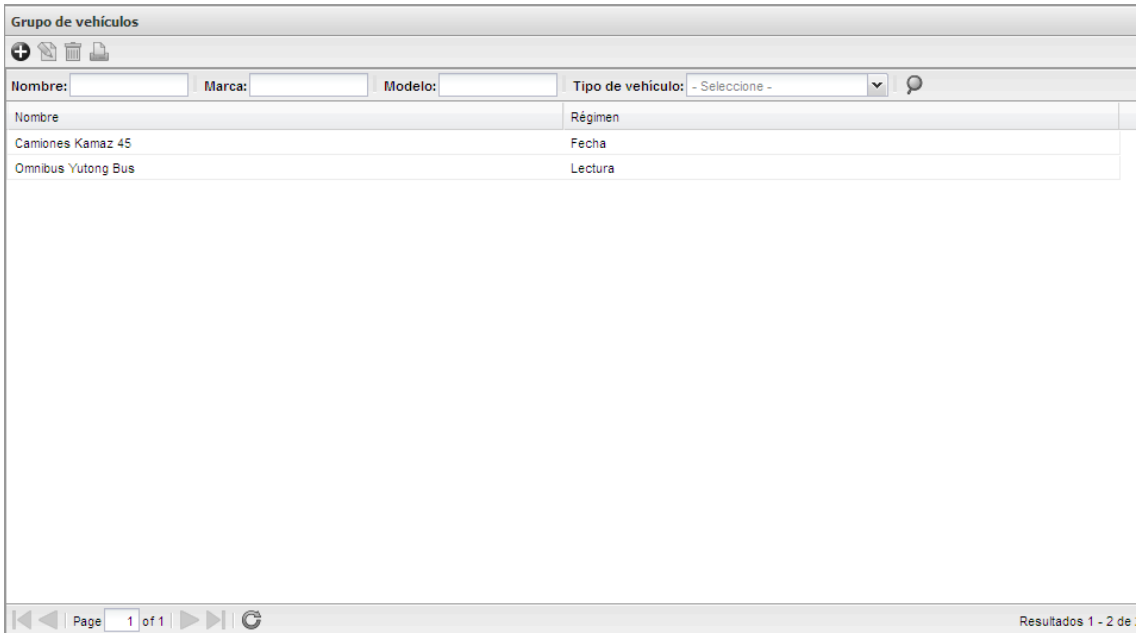
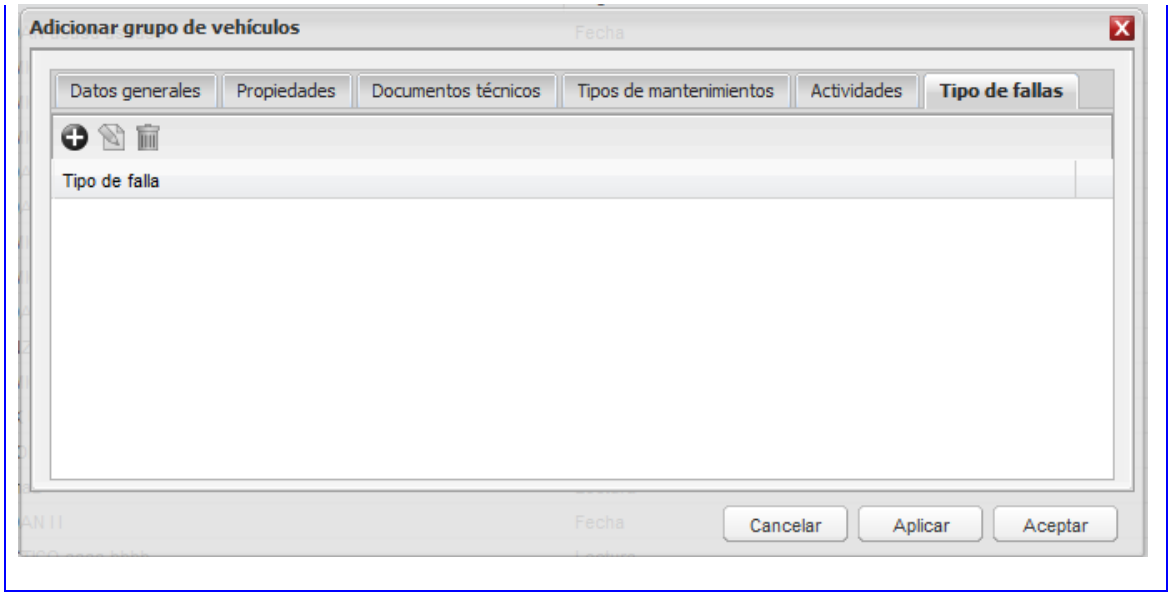


Tabla 37: HU Eliminar tipo de fallas.

Número: 28	**Nombre del requisito:** Eliminar tipo de fallas.
Programador: Yordan Bandera Rodríguez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 8 horas
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 2 horas
Descripción: Permite eliminar un tipo de falla seleccionada.	
Observaciones: Si no se selecciona un tipo de falla, no se puede completar la operación.	
Prototipo elemental de interfaz gráfica de usuario:	




Adicionar grupo de vehículos
Datos generales
Propiedades
Documentos técnicos
Tipos de mantenimientos
Actividades
Tipo de fallas
Tipo de falla
Cancelar
Aplicar
Aceptar