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

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

Autora:

Idelmis Téllez Peña

Tutor(es):

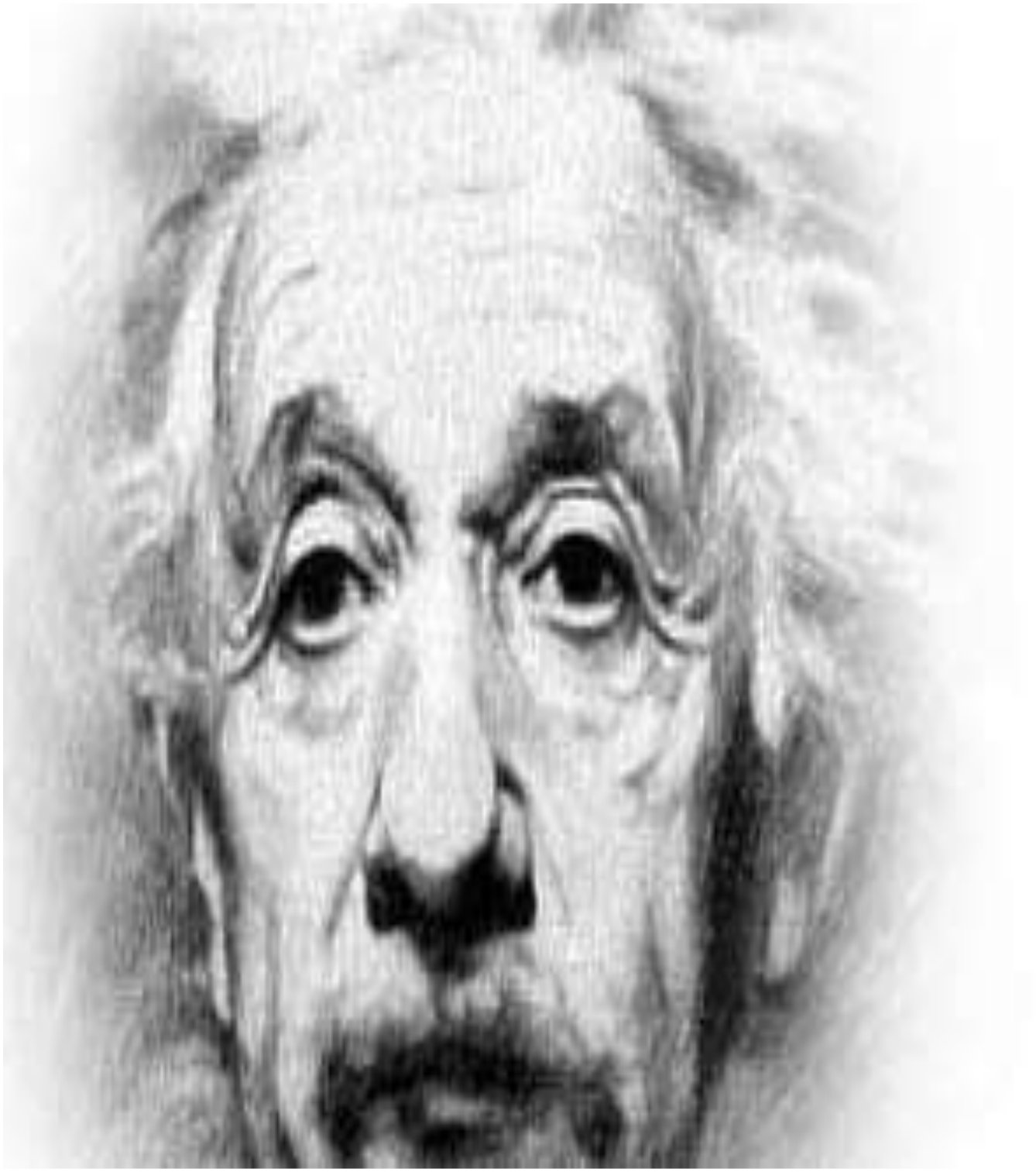
Ing**.** Ernesto Mató Roque

Ing. Yaniris Blanco Zamora

Co-Tutor(es):

Ing. Yorgüy Antonio Batista Desdin

Ing. Jorge Yosmiel Jiménez Quintana



“El aprendizaje es experiencia, todo lo demás es información”

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro que soy la autora de este trabajo titulado: Migración del módulo Vehículo del sistema Orbita a la arquitectura de referencia en PHP Bosón, y otorgo al Centro CEIGE de la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año_____.

Idelmis Téllez Peña

_____

Autora

Ing. Ernesto Mató Roque

_____

Tutor

Ing. Yaniris Blanco Zamora

_____

Tutor

Ing. Yorgüy Antonio Batista Desdín

_____

Co_Tutor

Ing. Jorge Yosmiel Jiménez Quintana

_____

Co_Tutor

DATOS DE CONTACTO

Autor: Idelmis Tellez Peña

Correo: **itellez@estudiantes.uci.cu**

Universidad de las Ciencia Informáticas, La Habana, Cuba

Tutor: Ing. Yaniris Blanco Zamora

Correo: **ybzamora@uci.cu**

Universidad de las Ciencia Informáticas, La Habana, Cuba

Tutor: Ing. Ernesto Mató Roque

Correo: **emato@uci.cu**

Universidad de las Ciencia Informáticas, La Habana, Cuba

DEDICATORIA

A mi abuela Celia, que Dios la llamó a su lado antes de que pudiese regalarle este título.

A mi madre Marina por ser la guía de mi vida y la fuerza de mi alma.

A mi padre Roberto por ser la visión objetiva de mis decisiones.

A mi Tato por ser mi compañero, mi amante y mi confidente.

A mi hermana Idalmis por ser mi ejemplo y guía.

AGRADECIMIENTOS

Gracias a mi madre por darme todo el amor del mundo, su apoyo en los momentos difíciles, sus consejos, sus palabras de aliento y sobre todo por traerme a la vida e inculcar en mí los valores que hoy me hacen la persona que soy.

Gracias a mi padre por ayudarme cada día a cruzar con firmeza el camino de la vida, porque con su apoyo y aliento hoy he logrado uno de mis más grandes anhelos.

Gracias a mi abuela Celia por ser el aire que respiro, por representar en mi vida la cúspide de la dedicación a la familia.

Un agradecimiento especial a mi novio Yorgüy (Tato) por acompañarme a lo largo de toda mi carrera, por estar a mi lado en todos los momentos buenos o malos, y sobre todo por darme su amor incondicional.

Gracias a mi hermana Idalmis por ser mi guía, por darme ánimos en los momentos más difíciles y por ser la mejor hermana del mundo.

Gracias a mi suegra Oralis y a mis suegros Luis y José por preocuparse por mí como si fuese su hija.

Gracias a mi piquete Yudelyn, Nairovis, Jany, Dayan y Eimee por ser mis amigos en la universidad, mis bastones en los momentos difíciles y sobre todo porque su amistad no tiene precio.

Gracias a mis amigos Mandy y Ángel por todo su apoyo incondicional y su amor a pesar de la distancia que nos separa.

Agradecer también aquellas personas que han demostrado su valía en los momentos difíciles: Jorge García, Roberts, Tito y José Luis.

Gracias a mis tutores Yaniris y Ernesto por guiarme en el transcurso de esta etapa tan importante.

Gracias al tribunal por su apoyo y sus recomendaciones que me ayudaron a preparar la tesis con mejor calidad.

Gracias a todos mis compañeros de brigada.

RESUMEN

El presente trabajo aborda la migración del módulo Vehículo del Sistema de Control Flota y Mantenimiento Orbita a la arquitectura de referencia PHP Bosón. Este sistema permite realizar la gestión de los vehículos, el registro de lectura y las hojas de rutas. 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 además se deja plasmado un estudio de los diferentes lenguajes y tecnologías a utilizar en el desarrollo del módulo Vehículo. Se realizan pruebas de rendimiento para comprobar que el nuevo módulo mejoró respecto al módulo anterior.

En el proceso de desarrollo del sistema se utiliza la metodología de desarrollo para la actividad productiva en la UCI permitiendo que el producto adquiera la mayor calidad posible. La implementación del módulo Vehículo para el nuevo sistema Orbita contribuirá a mejorar su desempeño debido a las actualizaciones que incluyen las tecnologías que conforman Bosón.

Palabras claves: Control de Flota, migración, módulo Vehículo, rendimiento.

Tabla de contenido

Índice de ilustraciones

Índice de tablas

Introducción

Las instituciones, empresas y gobiernos cada vez son más dependientes de la información, por ello se necesitan mecanismos de control que contribuyan a mejorar el funcionamiento interno y sistémico, la eficiencia y la competitividad con las demás instituciones. A esto pueden contribuir las llamadas Tecnologías de la Información y las Comunicaciones (TIC), las cuales han evolucionado y están cambiando el modo tradicional de realizar las actividades. En los últimos años, las TIC han tomado un papel relevante en nuestra sociedad y se utilizan en multitud de actividades formando parte de la mayoría de sectores: educación, robótica, administración pública, empleos y empresas (EDUCAMERICAS, 2008-2016).

Las empresas de transporte no están exentas de estos cambios, la utilización de sistemas de control de flotas le permiten a las mismas ahorrar dinero por concepto de reducción del uso de combustible, reducción de distancias y manejo de cargas. Estas empresas además buscan mantener un control sobre la información mecánica de los vehículos, su estado técnico y la gestión de su mantenimiento. La informatización del control de las flotas en Cuba fue una tarea que se comenzó en el año 2006. La misma utiliza tecnología GPS y posee una aplicación Web central combinada con aplicaciones locales. El proyecto ha alcanzado actualmente a más de 25 bases de transporte y más de 1 090 medios instalados reportando un ahorro de combustible promedio de más del 15%. Los proyectos que componen esta informatización del control de flotas según la Oficina Nacional de Estadísticas (ONE, 2007) son:

- “MóvilWeb”
- “CartoSIG”
- “Infraestructura Informática”

En el Centro de Informatización de Entidades (CEIGE) de la facultad 3 perteneciente a la UCI se desarrolló el sistema Orbita, un Sistema de Control de Flota y Mantenimiento, el cual gestiona los procesos relacionados con los mantenimientos en una base de transporte. Dentro del sistema Orbita se encuentra el módulo Vehículo que permite llevar el control y la administración de los vehículos, así como la información de cada uno.

La tecnología empleada para el desarrollo del sistema Orbita fue el marco de trabajo Sauxe en su versión 2.0. Se utilizó en la capa de presentación ExtJs en su versión 3.4, a petición del cliente. ExtJs carga todas las bibliotecas javascript al inicio atentando con el rendimiento del sistema y su licencia no es completamente libre, para ser

comercializado debe de pagar su uso (Crysfel Villa, 2016).En la capa de negocio hace uso de Zend Framework 1.2, un marco de trabajo que no permite la generación automática de ¿Creación de Crear?, Leer, Actualizar y Borrar (CRUD), la generación de código php mediante líneas de comandos para la creación y mantenimiento de aplicaciones y el almacenamiento en caché de las vistas (Zend framework, 2016).

En la capa de acceso a datos 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 en el momento del cambio hacia otra tecnología. Otro inconveniente que presenta el uso de esta versión de Doctrine es que a partir del 1 de junio del 2011 se dejó de dar soporte, trayendo consigo que no se corrijan posibles errores y vulnerabilidades que atentan contra su estabilidad y rendimiento (Doctrine, 2016).

Para un estudio sobre los problemas que presenta el sistema Orbita se realizó una encuesta (ver Anexo 3) a 4 especialistas que intervinieron en el desarrollo del sistema Orbita, se identificaron varios factores que influyó negativamente en el proceso de desarrollo de dicho sistema, arrojando como resultado las siguientes dificultades:

El 100 % de los encuestados coinciden que el sistema Orbita presenta problemas en la documentación relacionada con la arquitectura y funcionamiento de Sauxe puesto que la misma no se encuentra bien organizada.

- El 75% de los encuestados coinciden que el tiempo de respuesta de las interfaces relacionadas con las acciones adicionar vehículo, registro de lectura y adicionar hoja de ruta oscila entre 8 y 12 segundos.
- El 100% de los encuestados coinciden que el tiempo de respuesta de la página de acceso al sistema oscila entre 0 a 12 segundos, esto influye negativamente en el trabajo con el sistema por parte de los usuarios.

Teniendo en cuenta lo que plantea el autor Jakob Nielsen[1], en el libro Usability Engineering existen tres límites importantes en el tiempo de respuesta (Nielsen, 2000-2016) :

- 0,1 segundo: es el límite en el cual el usuario siente que está manipulando los objetos desde la interfaz de usuario.
- 1 segundo: es el límite en el cual el usuario siente que está navegando libremente sin esperar demasiado una respuesta del servidor.

[1]Jakob Nielsen es un ingeniero de interfaces de la empresa Sun Microsystems y una de las autoridades más respetadas en el ámbito mundial sobre usabilidad en la web.

- 10 segundos: es el límite en el cual se pierde la atención del usuario, si la respuesta tarda más de 10 segundos se deberá indicar algún mecanismo por el cual el usuario pueda interrumpir la operación.

Además de la encuesta realizada donde se reflejan un conjunto de problemas del sistema, se evaluó el rendimiento utilizando la herramienta JMeter en las funcionalidades referentes al módulo Vehículo. Para la ejecución de la prueba se simulan las peticiones para 1 usuario. El sistema se encuentra en un servidor virtualizado con las siguientes prestaciones: un microprocesador Intel(R) Xeon (R) 4 núcleos 2.4 GHz, 8GB de RAM y de almacenamiento 40 GB de disco duro. Con estas prestaciones se obtuvieron los siguientes resultados:

- El tiempo máximo que puede tardar el módulo Vehículo para ejecutar determinadas funcionalidades con un usuario realizando peticiones de manera simultánea fue de 12008 milisegundos.
- La velocidad del sistema Orbita fue 4.3 segundos en responder.
- El tiempo de ejecución promedio de una petición con un usuario fue de 225 ms.

Los resultados obtenidos mediante la encuesta y las pruebas de rendimiento confirmaron que los tiempos de respuesta del sistema Orbita entorpecen el trabajo de los usuarios que lo utilizan. Se identificó además que las tecnologías utilizadas para desarrollar el sistema se encuentran obsoletas y sin soporte impidiendo realizar mejoras sobre las mismas. La poca documentación del marco de trabajo empleado entorpece la corrección de algunas de estas dificultades.

A partir de la problemática antes planteada se define como **Problema a resolver:** ¿Cómo contribuir a mejorar el rendimiento del módulo Vehículo del sistema Orbita?

Para la solución de este problema se plantea como Objetivo General: Migrar el módulo Vehículo del sistema Orbita a la arquitectura de referencia en PHP Bosón, contribuyendo a mejorar el rendimiento del sistema.

Se plantea como Objeto de estudio: Migración de tecnologías de desarrollo en sistemas de gestión.

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

Para dar cumplimiento al objetivo general planteado se han definido los siguientes **Objetivos Específicos:**

1. Realizar un estudio del estado del arte para la elaboración del marco teórico alrededor del objeto de estudio.

2. Analizar la ingeniería de requisitos del módulo Vehículo para sistema Orbita

3. Realizar el diseño de componentes y clases para el sistema Orbita aplicando los patrones de diseños, algoritmos y técnicas consideradas como necesarias en el estudio.

4. Implementar el módulo Vehículo para el sistema Orbita, siguiendo las técnicas de programación estudiadas en la tecnología de desarrollo seleccionada para la migración.

5. Validar funcionalmente la solución propuesta para el sistema Orbita.

Se define como **Idea a defender**: si se realiza la migración del módulo Vehículo del sistema Orbita a la arquitectura de referencia en PHP Bosón, se contribuye a mejorar su rendimiento.

Posibles resultados: módulo Vehículo del sistema Orbita sobre la arquitectura de referencia en PHP Bosón.

A lo largo de la historia la humanidad siempre ha puesto en práctica la investigación científica que no es más que una serie de pasos que conllevan a una búsqueda de conocimientos, de información sobre algo nuevo e inesperado mediante la aplicación de métodos. Para la realización de este trabajo de diploma se utilizaron los siguientes métodos científicos:

Métodos teóricos:

- **Histórico-Lógico**: se utiliza para estudiar la teoría conocida hasta el momento, así como para conocer los antecedentes de la gestión de vehículos y las soluciones informáticas existentes.
- **Analítico-Sintético**: permite buscar la esencia del problema, los rasgos y características que distinguen al problema que se investiga.
- **Modelación**: la modelación es el proceso mediante el cual se crea una representación o modelo para investigar la realidad. Se pone en práctica en el momento de realizar los diagramas y modelos que permitirán la posterior implementación de las funcionalidades.

Métodos empíricos:

- **La entrevista**: es una técnica de recopilación de información mediante una conversación profesional, con la que además de obtener información acerca de lo que se investiga, tiene importancia desde el punto de vista educativo; los resultados a lograr en la misión dependen en gran medida del nivel de comunicación entre el investigador y los participantes en la misma.
- **Medición**: se utiliza en las métricas aplicadas para validar los requisitos, el código y en las pruebas realizadas al sistema para verificar y validar su correcto funcionamiento.

El trabajo de diploma está constituido por tres capítulos estructurados de la siguiente forma:

Capítulo 1: **Fundamentación teórica**. Se definen elementos teóricos que sustentan la investigación y se realiza un análisis profundo sobre el proceso de gestión de vehículo del sistema Orbita. También se definen las herramientas y lenguajes a utilizar en la implementación, así como la metodología y los patrones de diseño.

Capítulo 2: **Análisis y diseño**. Se analiza y especifica el diseño de la propuesta de solución. Se realiza las historias de usuarios a los requisitos funcionales del sistema y los patrones de diseño para obtener un mejor resultado en la implementación.

Capítulo 3: **Implementación y prueba**. Se implementa el módulo para darle solución a la situación problemática planteada. Se realizan las pruebas para validar funcionalmente la propuesta de solución y se valida que la propuesta de solución cumpla con el objetivo planteado.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En el presente capítulo se analizará la migración de los procesos dentro del Sistema de Control de Flota y Mantenimiento Orbita y un estudio del arte en cuanto al objeto de estudio y campo de acción definidos para el trabajo de diploma. Se analiza detalladamente el proceso de migración, así como la descripción de la arquitectura, las tecnologías y las herramientas que pueden darle solución a la problemática existente.

1.2 Conceptos asociados a la investigación

La sustitución de viejas tecnologías y herramientas por elementos más modernos se realiza fundamentalmente para mejorar uno o varios aspectos del software. La modificación de un software para hacer que algún aspecto del mismo funcione de manera más eficiente y/o utilice menos recursos (mayor rendimiento) es el principal objetivo de esta investigación. Los recursos informáticos no son infinitos, por lo tanto la utilización de nuevas tecnologías permitirían disminuir el consumo de los mismos. En la investigación se hará uso de los dos conceptos, ya que abarcan y expresan una definición con todo lo relacionado con el rendimiento de un sistema informático.

1.2.1 Migración

La migración es el “Traslado de una aplicación de un ordenador a otro en condiciones de compatibilidad. Migrar es también elevar una versión de un producto software a otra de más alto nivel, o bien el movimiento de una arquitectura a otra” (mastermagazine, 2015).

“Paso de los programas, archivos y datos de un sistema desde una determinada plataforma tecnológica a otra diferente” (Pacheco, Juan Manuel Fernández, 2010).

“En tecnología de la información, la migración es el proceso de pasar de la utilización de un único entorno operativo a otro entorno operativo, es decir, en la mayoría de los casos, se piensa que es mejor. La migración puede implicar la mudanza hacia el nuevo hardware, hacia el nuevo software, o ambos. La migración puede ser a pequeña escala, tales como la migración de un sistema único, o en gran escala, con la participación de muchos sistemas, nuevas aplicaciones, o una red rediseñado” (TechTarget, 2016).

Teniendo en cuenta las definiciones aportadas por estos autores se constata que coinciden en que migración es el traslado de una aplicación informática tanto en

hardware como en software o en ambos persiguiendo una mejora en su desempeño. En el presente trabajo se adopta el concepto aportado por (mastermagazine, 2015), el mismo se ajusta a las necesidades de la investigación como se describe en las secciones siguientes.

1.2.2 Rendimiento

Rendimiento: “La eficacia total de un sistema informático, incluyendo el tiempo de respuesta individual y la disponibilidad” (TechTarget, 2016).

El rendimiento define la capacidad de respuesta del sistema, es decir, el tiempo necesario para responder a estímulos (eventos) o el número de eventos procesados en un cierto intervalo de tiempo… Las cualidades de rendimiento se expresan a menudo mediante el número de transacciones por unidad de tiempo que procesa el sistema o por la cantidad de tiempo que se tarda en completar una transacción. (Jacobson, et al., 1999).

El estudio de bibliografías de referencia en esta área del conocimiento, permite identificar que no existe una clasificación única que defina rendimiento; por lo que el autor se inclina por el propuesto por (Jacobson, et al., 1999) . El autor de la presente investigación establece que el rendimiento será medido como el tiempo de respuesta de las funcionalidades del sistema.

1.2.3 Necesidad de migración

El sistema Orbita que se encuentra en funcionamiento fue desarrollado en el marco de trabajo Sauxe en su versión 2.0. El mismo utiliza en la capa de negocio el marco de trabajo Zend Framework en su versión 1.2. Esta versión que pertenece al paquete ZF1 de la compañía Zend Technologies se encuentra sin soporte desde el año 2014 según el sitio oficial (Zend Technologies Ltd, 2016). Sauxe además propone para la capa de acceso a datos el uso del ORM[2] Doctrine en su versión 1.2.2 liberada en el año 2009 y ya no cuenta con soporte por parte de Doctrine Project, la última versión estable es la 2.5.4 liberada en el presente año (Doctrine Project, 2016). En la capa de presentación se utilizó el marco de trabajo ExtJs en su versión 3.4, en la versión 4.0 se introducen cambios en la estructura de clases que modifican la estructura de programación en referencia a versiones anteriores. La última versión estable es la 6.0 la cual incluye una filosofía presentada en la versión 5.0 que revolucionaba la comunicación de ExtJS con los marcos de trabajo utilizados en el servidor (Symfony, Ruby on Rails, Django, .NET) (Sencha Inc, 2016).

[2] Del inglés Object-Relational mapping (Mapeo de objeto-relacional)

La evolución de las tecnologías usadas en el desarrollo del sistema Orbita han permitido optimizar sus procesos, convirtiendo a sus antecesores en obsoletas. La adopción de versiones mejoradas de tecnologías como Doctrine en su rama 2.0 introduciría los siguientes cambios:

- El primer cambio importante introducido por Doctrine ORM 2.5 es la obligación de usar PHP 5.4 o superior**.**
- La nueva caché de segundo nivel que introduce Doctrine 2.5 en vez de guardar las entidades en memoria, se guardan en un sistema de caché como Memcache, Redis, Riak o MongoDB. Esta caché guarda el resultado de búsquedas más complejas, no solo aquellas que buscan entidades a partir de su clave primaria.
- Permite el soporte para asociaciones ManyToMany en criterios de búsquedas.

Estos cambios facilitarían el trabajo de los programadores aprovechando las bondades de las nuevas versiones de PHP y mejorando los tiempos necesarios para realizar consultas a las bases de datos. La migración del sistema Orbita hacia nuevas tecnologías no solo persigue un producto que provea un mayor rendimiento, sino además la corrección de deficiencias en las tecnologías. La actualización tecnológica le permitirá al proyecto de Mantenimiento Vehicular de la Dirección de Transporte de la UCI contar con un módulo de control de vehículos adaptado al auge de las nuevas tecnologías.

1.3 Sistema de gestión de flota y mantenimiento

La gestión de flotas es la gestión del conjunto de vehículos comerciales de una empresa u organización, como camiones, furgonetas, coches y otros vehículos. Puede incluir una variedad de objetivos y funciones como el mantenimiento, el seguimiento y control de vehículos, la detención remota, el diagnóstico mecánico, la administración de conductores, la gestión de combustible, la gestión de la seguridad y en general, todo lo referido al análisis de los datos e información disponible y a la toma de decisiones vinculadas a la flota de vehículos (Ful-Mar, 2016).

La gestión de una flota de vehículos se basa en dos aspectos diferenciados y coordinados: el mantenimiento de los vehículos y el tráfico de la flota. La gestión del mantenimiento de grandes flotas de vehículos es una tarea compleja, que debe tener en cuenta diversos factores. El tipo de vehículos que la forman (turismos, furgonetas, vehículos industriales, etc.), su uso (urbano, carretera, intensivo, etc.), los planes de

mantenimiento establecidos por los fabricantes de los vehículos y la realización, por medios propios o ajenos, de las operaciones en ellos incluidas, son aspectos claves para gestionar con eficiencia una flota de vehículos (Gestión de Flotas, 2010).

1.3.1 Proceso de vehículo en los sistemas de gestión de flota y mantenimiento

Los problemas previos que existían cuando no había ningún sistema de gestión de flota y mantenimiento vehicular según el análisis realizado por (García, 2016) son:

- Lenta generación de informes gerenciales.
- Información sesgada o no fiable al 100 % (aproximación de números por la indisponibilidad automática de los datos).
- Información disponible solo en papeles y no compartida.
- Información sobre los vehículos dispersa en distintas Unidades Administrativas.

Por lo tanto, fue indispensable automatizar el registro y control de los procesos y que involucran a la flota de vehículos para lograr los siguientes objetivos:

- Gestión del costo que implica mantener la flota de vehículos.
- Información entrelazada de los diferentes procesos que involucran a los vehículos.
- Mecanismos de control automatizados para los diferentes procesos.
- Información fiable y en línea, sin recurrir a papeles o a otras Unidades Administrativas.

Además se identificó que era necesario (García, et al., 2016):

- Integrar y consolidar la información en un único sistema informático corporativo.
- Revisar e implementar los procesos especificados para cada área involucrada.
- Controlar los gastos por mantenimiento y otros relacionados a los vehículos.
- Controlar y gestionar los montos mínimos y máximos de las licitaciones.

Un sistema de control de vehículos es una aplicación informatizada que permite llevar el control y administración de vehículos en todo lo que respecta a mantenimientos, control de combustible y choferes. Este tipo de aplicaciones ponen

a disposición las herramientas necesarias para guardar registro del kilometraje, consumos de combustible propio y externo, materiales, costes de piezas, mano de obra y horas de funcionamiento. El software también elabora avisos periódicos del mantenimiento de la flota de vehículos. Este control permite prevenir posibles fraudes relacionados con los vehículos.

Dentro del sistema Orbita se encuentra un módulo llamado Vehículo que no es más que una aplicación automatizada que debe permitir llevar el control y administración de vehículos en todo lo que respecta a sus características, a su responsable, el área de la entidad a la que pertenece y a los choferes asignados.

Debe permitir:

- Llevar un control sobre las hojas de rutas de cada vehículo.
- La agrupación de los vehículos por su marca y modelo.
- La gestión de las pólizas de seguro de cada vehículo.

1.4 Gestión del control de vehículos en otros sistemas de control de flotas

Los sistemas para la gestión de flota de vehículos son tan heterogéneos en sus procedimientos y procesos como las necesidades existentes en las bases de transporte donde son utilizados. Con el fin de analizar características similares entre diferentes sistemas de este tipo a continuación se analizan soluciones del ámbito internacional y nacional.

1.4.1 Sistemas internacionales

GIM (Gestión Integrada de Mantenimiento) es un sistema de gestión dedicado al mantenimiento de flotas. Realiza el seguimiento de los distintos tipos de mantenimiento como el mantenimiento correctivo y preventivo. Permite además, generar libros Microsoft Excel totalmente personalizados, permitiendo confeccionar rápidamente y sin límite alguno cualquier tipo de informe deseado (listados, cuadros de análisis, estadísticas y gráficos) (GIM, 2009).

Ventajas:

- Posee un sistema de control de vehículos y de gestión de mantenimiento que tiene implementada una amplia gama de funcionalidades.

- Los requerimientos para su implantación son modestos y adaptables a compañías tanto de grande, como de mediano formato (GIM, 2009).

Desventajas:

- No incluye la gestión de las inspecciones técnicas con el objetivo de controlar el estado de las partes de los vehículos, lo que impide controlar con mayor eficiencia el estado de los mismos y prever en qué momento es más óptimo la realización de los mantenimientos preventivos planificados.

SoftFlot ofrece dentro de su portafolio de productos de aplicaciones empresariales, una herramienta para la administración de su parque vehicular denominado Autocontrol. Autocontrol es un sistema que controla los vehículos de la organización, automatizando las principales funciones de esta actividad. Cuenta con los módulos de solicitud, asignación, registro de las condiciones físicas de entrada y salida del vehículo, seguimiento y trámites administrativos, hasta el servicio de mantenimiento que cada vehículo requiere incluyendo la generación de órdenes de servicio (Ingeniería DEC México, 2014).

Este sistema está difundido por países de América Latina como Chile, México y Venezuela. Entre las funcionalidades que contiene este software se encuentran:

- Control de vehículo.
- Control de recursos humanos.
- Asignación de choferes.
- Almacén de recursos materiales.
- Costos y presupuestos.
- Control de combustibles
- Control de accidentes.
- Asignación, registro y gestión del calendario de tareas de mantenimiento a los vehículos.
- Reporte de fallas.

Ventajas:

- Es de sencilla instalación y gestión de los procesos con interfaces amigables para el usuario.
- Está compuesto por módulos de Control de vehículo, Seguimiento Técnico de los vehículos y Sistemas de Seguridad Automotor.
- Posee compatibilidad con sistemas ERP desarrollados por la misma empresa que fabrica este software.

- Permite el control online y en tiempo real de la gestión ilimitada de vehículos (Ingeniería DEC México, 2014).

Desventajas:

- Está diseñado para trabajar sobre plataformas de software privativas pertenecientes a Microsoft y Oracle.
- Estas plataformas privativas incluyen licencias de uso que deben ser pagadas (Ingeniería DEC México, 2014).

1.4.2 Sistemas nacionales

Siscompa.Net garantiza el control y gestión de cualquier flota automotor de transporte de cargas, contribuyendo al ahorro de recursos materiales, combustible y tiempo.

El sistema brinda información confiable sobre todo el parque automotor que poseen las bases con respecto a su estado técnico, historia y explotación, planificando y controlando toda la operación en taller, mantenimientos, la actividad de tráfico, la de seguridad automotor y el control de los portadores energéticos. Esta herramienta permite potenciar la actividad de ahorros en combustibles, lubricantes, gomas, baterías, etc. Ofrece a los directivos la información necesaria para la toma de decisiones, siendo una poderosa herramienta de dirección (transoft, 2012).

Ventajas:

- El hecho de ser creado en nuestro país, por lo que está programado en función de satisfacer necesidades puntuales comunes en nuestro país, por lo que ha sido desplegado en empresas cubanas.

Desventajas:

Está orientado a Sistemas Operativos de Microsoft a partir de Windows 2000, por lo que no cumple con las políticas de migración a plataformas de software libre del país.

Orbus es un sistema que permite planificar los servicios de cada ruta en una base de transporte por parte de la entidad rectora de forma automatizada. Además de programar el régimen de paradas de ida y regreso, registra la distancia en kilómetros, el tiempo en minutos entre cada parada y el tiempo en la parada. Actualiza la programación de los servicios de las rutas de las bases de transporte en los sistemas MovilWeb y Corcel. Permite agilizar el proceso de planificación del servicio de ómnibus y pasaje y la organización superior puede, desde su ubicación, modificar y reordenar los servicios de todas las bases subordinadas; contribuyendo todo esto a aumentar la calidad y agilidad del servicio en la transportación de pasajeros y el bienestar de la población (transoft, 2012).

Valoración de los sistemas de control de vehículos estudiados.

Tabla 1: Comparación entre los sistemas estudiados.

Parámetros	Sistemas de control de Flota y Mantenimiento.			
	Nacionales		Internacionales	
	Siscompa.Net	Orbus	GIM	SoftFlot
Origen	Cuba	Cuba	Chile	México
Software Libre	No	No	No	No
Licencia	Si	Si	Si	Si

Luego de realizar un estudio a los sistemas de control de Flota y Mantenimiento antes mencionados, donde se tuvo en cuenta los aspectos fundamentales del software, se evidencia la no existencia de un sistema que responda cabalmente a las necesidades para la migración del sistema Orbita. La adopción de estos sistemas le provocaría dificultades al personal de transporte en cuanto a pasar todos los datos manualmente para uno de los sistemas. Además como los sistemas no son de software libre no se puede modificar el código existente.

1.5 Metodología, lenguajes y herramientas de desarrollo

1.5.1 Metodología de desarrollo

En la actualidad las metodologías representan un papel importante en el desarrollo del software, ya que de ellas dependen que el producto sea terminado con la mayor calidad posible. Actualmente el desarrollo de software dentro de la actividad productiva de la UCI se caracteriza por el uso de diferentes metodologías de desarrollo entre robustas y ágiles. A pesar de la variedad de metodologías usadas, se ha comprobado que muy pocos proyectos la aplican en su totalidad. Para lograr una estandarización en la utilización de estas metodologías de desarrollo y adecuar las mismas a las características particulares de la UCI se decidió realizar una

variación de la metodología AUP. A continuación se explican las fases descritas por (Universidad de las Ciencias Informáticas, 2012) para la variación AUP-UCI.

Fases variación AUP-UCI (Universidad de las Ciencias Informáticas, 2012) **:**

- **Inicio**: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
- **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

La realización de la presente investigación se centró en la fase de ejecución que comprende 5 disciplinas como los requisitos, análisis y diseño, implementación, pruebas internas y pruebas de aceptación. Buscando una estandarización en el desarrollo de los módulos del nuevo sistema el cliente propone la utilización de una metodología única. Para lograr esta estandarización siguiendo los esfuerzos que realiza la UCI se decide usar la variación UCI de AUP.

1.5.2 Marco de trabajo Symfony 2

Symfony 2 es un marco de trabajo rápido, flexible y fácil de aprender, que permite a los desarrolladores construir aplicaciones webs. A continuación se muestran algunas características de este marco de trabajo.

Características (SensioLabs, 2016):

- Su código, y el de todos los componentes y librerías que incluye, se publican bajo la licencia MIT de software libre.
- La documentación del proyecto también es libre e incluye varios libros y decenas de tutoriales específicos.
- Los componentes de Symfony son tan útiles y están tan probados, que proyectos tan gigantescos como Drupal 8 están construidos con ellos.
- Utilizar Symfony es gratuito.
- Independiente del sistema gestor de bases de datos.

- Cuenta con una potente línea de comandos que facilitan generación de código, lo cual contribuye a ahorrar tiempo de trabajo.

1.5.3 Arquitectura de referencia en PHP Bosón

Debido al amplio espectro de aplicaciones que se desarrollan en la universidad sobre el lenguaje PHP, la Dirección General de Proyectos (DGP) de la UCI le asignó la tarea de crear una arquitectura al Departamento de Desarrollo de Componentes del Centro CEIGE. Se decidió crear una arquitectura de referencia donde se especifican las buenas prácticas para el desarrollo por la definición de patrones y tecnologías a utilizar. De este modo sería posible llegar a un punto común entre las diversas soluciones. Las aplicaciones que utilizan este lenguaje al no contar con la misma arquitectura es prácticamente imposible integrarla o reutilizarla.

Bosón es una implementación de esta arquitectura de referencia. La arquitectura Bosón en su última liberación utiliza el marco de trabajo Symfony 2 en su versión 2.7. Constituye un conjunto de componentes desarrollados sobre Symfony 2 que responden a la mayoría de los requisitos tecnológicos de un amplio espectro de aplicaciones. Materializa los requisitos comunes de las arquitecturas base para sistemas de gestión web a partir del desarrollo de los siguientes componentes (Universidad de las Ciencias Informáticas, 2016):

Componente Caché: Componente que permite a las aplicaciones que se están desarrollando la gestión de la caché, así como el almacenamiento y recuperación de la información almacenada en la misma de forma rápida y sencilla. Permite guardar información en cualquiera de los siguientes formatos, texto plano, arreglos o tablas hash, objetos o instancias de clases, estructuras complejas como listas, pilas, colas, árboles y otras estructuras de mayor complejidad.

Componente Estructuras de Datos: Componente capaz de homogeneizar el uso de las estructuras de datos (nodos, árboles, pilas, colas, listas y grafos) en PHP y de esta forma se minimiza los tiempos de acceso y se logran formas más efectivas de inserción y eliminación de datos en estructuras de almacenamiento.

Componente Excepciones: Componente que permite controlar de forma centralizada los diferentes tipos de excepciones y tratarlas según el tipo especificado. Garantiza además la internacionalización de los mensajes de las excepciones, el manejo declarativo del comportamiento, el mecanismo extensible de creación de nuevos tipos y la codificación.

Componente Estructuras Dinámicas: Componente que permite modelar y gestionar estructuras y nomencladores en toda la amalgama de probabilidades que componen un negocio. Utiliza el patrón Entidad Atributo Valor (EAV) para gestionar las estructuras lo cual disminuye la creación de grandes volúmenes de tablas destinadas a estos temas.

Componente Trazas: Componente que detecta y registra las trazas generadas por un sistema a partir de la captura de eventos, por lo cual resulta útil para las auditorías. Permite registrar trazas de acción, rendimiento, acceso a datos y excepciones ocurridas.

Componente Aspectos: Componente que permite las bondades de la Arquitectura Orientada a Aspectos (AOP) mediante la definición de reglas de negocio que pueden ser ejecutadas antes o después de las acciones de un controlador en un orden definido.

Componente Integración: Componente que brinda un nuevo mecanismo de integración en Symnfony2 haciendo uso de servicios REST. Expone recursos como servicios de forma sencilla a partir de anotaciones en las entidades. De esta forma se hace transparente para el usuario la implementación de la lógica, la creación de rutas y el manejo de código de errores.

Componente Seguridad: Componente que provee un mecanismo de autenticación haciendo uso del estándar abierto SAML[3] así como un mecanismo de autorización mediante la extensión del patrón RBAC[4]. Permite además la autenticación a partir de varias fuentes como base de datos, servidores LDAP, Facebook, Google entre otras.

Componente IUX: Componente que permite la creación de interfaces utilizando el proyecto de Interfaz Única (IUX), el cual define estándares de diseño para cada una de las líneas temáticas de la UCI. Integra el IUX con el motor de plantillas TWIG utilizado por Symfony2 para el desarrollo de interfaces. Proporciona a los desarrolladores la posibilidad de elegir qué línea desea utilizar, para así generar automáticamente elementos gráficos ajustados a la línea temática.

Componente Portal: Componente que brinda un área de trabajo única, que se integre con los mecanismos de autenticación y autorización de la plataforma y con el componente de interfaz única. Es una plataforma que soporta las tecnologías de

[3] Security Assertion Markup Language (Lenguaje de Marcado para Confirmaciones de Seguridad)
[4] Role-Based Access Control (Control de Acceso basado en roles)

presentación HTML5, CCS3, JS, y mecanismos de comunicación asíncrona con el servidor como Ajax.

Teniendo en cuenta que la arquitectura de Bosón está desarrollada sobre el marco de trabajo Symfony2 hereda del mismo sus ventajas (Fabien Potencier, 2013).

Ventajas

- Permite automatizar tareas.
- Facilita el desarrollo de código legible.
- Permite la creación de funcionalidades de Crear, Leer, Actualizar y Borrar (del original en inglés: Create, Read, Update and Delete)
- Sesiones de Usuario.
- Permite la flexibilidad.
- Seguridad.
- Documentado y buen soporte.

Además de estas ventajas, la principal ventaja que ofrece Bosón es la estandarización del desarrollo web de aplicaciones basadas en PHP. De manera tal que se facilite la reutilización de código y se tengan como base las funcionalidades básicas que se utilizan en todos los proyectos. También, la arquitectura Bosón facilita el desarrollo al contar con componentes ya implementados y configuraciones básicas que le permiten al equipo de trabajo centrarse en los problemas del negocio que se está informatizando (Fabien Potencier, 2013).

El cliente del nuevo sistema Orbita necesita que su mecanismo de autenticación se base en el servicio de usuarios y contraseñas de la UCI. Las interfaces de la nueva solución deben seguir la misma línea temática de su predecesor para lograr una integración con la interfaz única propuesta para los productos realizados en la UCI. Además es necesario que se cuente con una interfaz principal genérica para todos los módulos así como la convergencia de las funcionalidades principales de los mismos. Teniendo en cuenta que la arquitectura de referencia Bosón suple estas necesidades y que el mismo fue elegido por los clientes para desarrollar el nueva sistema Orbita se decide utilizarlo para el desarrollo del módulo Vehículo.

1.5.4 Lenguajes de desarrollo

PHP

Symfony 2 es el marco de trabajo base para la arquitectura de Bosón. Este marco de trabajo utiliza PHP como lenguaje de programación. PHP es un lenguaje de programación de uso general, de código del lado del servidor originalmente diseñado

para el desarrollo web de contenido dinámico. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante.

Según (The PHP Group, 2016) **las principales características de PHP son:**

- Es un lenguaje multiplataforma.
- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- El código fuente escrito en PHP es invisible al navegador web y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables, aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones.

HTML5

Es una colección de estándares para el diseño y desarrollo de páginas web. Esta colección representa la manera en que se presenta la información en el explorador de internet y la manera de interactuar con ella. Permite una mayor interacción entre nuestras páginas web y contenido media (video, audio, entre otros), así como una mayor facilidad a la hora de codificar un diseño básico. HTML5 radica fuertemente en las capacidades del explorador, por lo que en estos momentos no todos los exploradores lo soportan (actualmente sólo Chrome, Safari, Firefox y Opera soportan la mayoría de las características) (TRAZOSWEB, 2010). Realizar páginas web utilizando HTML 5 brinda beneficios como (HTML5, 2011):

- Al ser el código más sencillo y simplificado, cargan más rápido las páginas en el navegador.
- HTML 5 incluye etiquetas orientadas principalmente a los buscadores, para facilitarles comprender el contenido de las páginas, lo que nos beneficia, por ejemplo: header y footer.

CSS3

Lenguaje conocido por el nombre hojas de estilo en cascada que viene del inglés Cascading Style Sheets, del que toma sus siglas. Es usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). Además sirve para definir la estética de un sitio web en un documento externo y eso mismo permite que modificando ese documento (la hoja CSS) se pueda cambiar la estética entera de un sitio web. Entre sus ventajas están que pueden mostrarse distintas hojas de estilo según el dispositivo utilizado y se obtiene un mayor control de la presentación del sitio al poder tener todo el código CSS reunido en uno, lo que facilita su modificación (CSS3, 2012).

Ventajas (CSS3, 2012)**:**

- Se obtiene un mayor control de la presentación del sitio al poder tener todo el código CSS reunido en uno, lo que facilita su modificación.
- Se consigue hacer mucho más legible el código HTML al tener los estilos CSS aparte.
- Permite ahorrar tiempo y trabajo al poder seguir varias técnicas (bordes redondeados, sombra en el texto, sombra en las cajas, etc.) sin necesidad de usar un editor gráfico.

JavaScript

Es un lenguaje de scripting basado en el navegador que ejecuta el código del lado del cliente. Esto significa que cualquier código que se escribe en JavaScript se entrega desde el servidor junto a las páginas web, todo el código se ejecuta desde el navegador del usuario (en el dispositivo del usuario) en lugar de hacerlo directamente en el servidor donde se encuentra la página web. JavaScript es una excelente solución para poner en práctica la validación de datos de un formulario en el lado del cliente. Si un usuario omite escribir su nombre en un formulario, una función de validación en JavaScript puede desplegar en pantalla un mensaje para hacerle saber al usuario acerca de la omisión. Este tipo de funcionalidades son más ventajosas que tener una rutina de validación del lado del servidor para controlar el

error, dado que el servidor en éste caso no tiene que hacer ningún tipo procesamiento de información adicional. Una de las áreas en la que sobresale radicalmente JavaScript es en la creación de efectos dinámicos tales como imágenes dinámicas y presentaciones de diapositivas, donde su uso se ha convertido en algo común hoy en día (JavaScript, 2016).

Las principales características de este lenguaje son (Marta E. Zorrilla Pantaleón, 2011):

- Es simple, no hace falta tener conocimientos de programación para poder hacer un programa en JavaScript.
- Se maneja objetos dentro de la página Web y sobre ese objeto se puede definir diferentes eventos. Dichos objetos facilitan la programación de páginas interactivas, a la vez que se evita la posibilidad de ejecutar comandos que puedan ser peligrosos para la máquina del usuario, tales como formateo de unidades y modificación de archivos.
- Es dinámico, responde a eventos en tiempo real.

1.5.5 Herramientas de desarrollo

Visual Paradigm 8.0

Es una herramienta de Ingeniería de Software Asistida por Computadora (CASE) que propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Brinda numerosas ventajas tales como: cuenta con una licencia gratuita y comercial, es multiplataforma, contiene soporte para varios idiomas y es fácil de instalar y actualizar. También el diseño es centrado en casos de uso y enfocado al negocio, presenta un uso de un lenguaje estándar común que facilita la comunicación para todo el equipo de desarrollo y posee la capacidad de realizar ingeniería directa e inversa (Pressman, 2007).

Facilita el Lenguaje Unificado de Modelado (UML) y permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado. La corrección sintáctica controla que el modelado con UML sea correcto. La coherencia entre diagramas al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades. Permite integrarse con otras aplicaciones, lo cual aumenta la productividad y el trabajo multiusuario que facilita el trabajo en grupo. También, permite generar código de forma automática, reduciendo

los tiempos de desarrollo y evitando errores en la codificación del software (Pressman, 2007).

Servidor de Base de Datos

PostgreSQL 9.1 es un potente motor de base de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de base de datos comerciales. Entre sus ventajas se encuentran que utiliza un modelo cliente/servidor, su código está disponible libremente para todas aquellas personas que deseen utilizarlo. Además, realiza copias de seguridad y cuenta con una amplia y completa documentación. Contiene numerosos tipos de datos y permite definir nuevos. También funciona correctamente con grandes cantidades de datos y una alta concurrencia de usuarios accediendo al mismo tiempo. Ofrece incorporar conceptos adicionales como son clases, herencia, tipos y funciones. Se evidencia flexibilidad en las restricciones (Constraints), los disparadores (Triggers) y las reglas (Rules) (PostgreSQL-es, 2009-2012).

Características (PostgreSQL-es, 2009-2012)**:**

- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP…), cadenas de bits, etc. También permite la creación de tipos propios.
- Incluye herencia entre tablas, por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Copias de seguridad en caliente (Online).
- Juegos de caracteres internacionales.
- Acceso cifrado vía SSL.
- Completa documentación.
- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 7 y 8 de 32/64bit.

Desventajas (PostgreSQL-es, 2009-2012)**:**

- Sin experiencia en la herramienta la configuración puede resultar difícil.
- Consume más recursos que MYSQL por lo que se necesitan mayores prestaciones de hardware para ejecutarlo.

pgAdmin

Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL sencillas hasta el desarrollo de bases de datos complejas. La interfaz gráfica es compatible con todas las características de PostgreSQL y facilita la

administración de este servidor. La aplicación también incluye un editor de resaltado de sintaxis SQL, un editor de código del lado del servidor, un agente de planificación de tareas de SQL, entre otras características. Posee un entorno de escritorio visual. Es instalable en las plataformas Linux, Solaris y Windows. Permite conectarse a bases de datos PostgreSQL que estén ejecutándose en cualquier plataforma (pdAdmin, 2016).

Firefox 45.0

Firefox es un navegador de Internet con interfaz gráfica de usuario desarrollado por la Corporación Mozilla y un gran número de voluntarios. (MozillaES, 2009). Este navegador posee numerosas ventajas entre las que se encuentran que es de código libre y multiplataforma, que guarda la contraseña para entrar a un sitio determinado y establece conexiones seguras a sitios web. (Mozilla.org, 1998-2012).

Características de Mozilla Firefox:

- Una de las principales características de este navegador es la navegación por pestañas.
- Trabaja de forma excelente en computadoras sin hardware muy potente, el programa está diseñado para realizar un bajo consumo de recursos (Mozila, 2012).

Ventajas (Plusesmas, 2015)**:**

- Las contraseñas son más seguras con Firefox. Antes de ir a un sitio web a través de Google, se pueden apreciar los riesgos asociados con este sitio.
- Verifica las actualizaciones de los antivirus con regularidad.
- Presenta muchas extensiones útiles que se pueden descargar y ofrecen funcionalidades adicionales.
- El uso de "favoritos" es rápido y fácil gracias al ícono correspondiente. Se puede asignar palabras claves a tus marcadores y favoritos.
- Es rápido.
- Se actualiza de forma automática.

Desventajas (Plusesmas, 2015)**:**

- Firefox no utiliza procesos separados para cada pestaña, lo que puede provocar un bloqueo. Pero si Firefox "se cuelga", casi siempre se puede recuperar la sesión.
- No todos los sitios web tienen un diseño optimizado para Firefox. Ocasionalmente, se puede tener problemas de visualización.

PhpStorm 7

Es un poderoso editor de PHP centrado en la productividad del desarrollador, de navegación rápida y comprobación de errores sobre la marcha. Entre sus características principales están que es un editor de código inteligente ya que entiende su código y su estructura con soporte para PHP 5.3, 5.4, 5.5 y 5.6. También ofrece numerosas opciones para depurar el código PHP pudiendo inspeccionar las variables locales relevantes al contexto y definidas por el usuario, incluyendo arreglos y objetos complejos. Además se podrá configurar la depuración desde un servidor remoto y evaluar una expresión en tiempo de ejecución. Es un editor de HTML / CSS / JavaScript donde se dispondrá de una función de autocompletado de código. (Pikuru, 2012-2014).

Características:

- Permite la gestión de proyectos fácilmente.
- Proporciona un fácil autocompletado de código.
- Soporta el trabajo con PHP 5.5
- Sintaxis abreviada.

GitLab 8.5.1

Es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario. Sin embargo, se ha convertido desde entonces en un sistema de control de versiones con funcionalidad plena. (GIT, 2016)

Modela sus datos más como un conjunto de instantáneas de un mini sistema de archivos. Cada vez que se confirmas un cambio, o se guardas el estado del proyecto en Git, él básicamente hace una foto del aspecto de todos sus archivos en ese momento, y se guarda una referencia. Para ser eficiente, si los archivos no se han modificado, no almacena el archivo de nuevo, sólo un enlace al archivo anterior idéntico que ya tiene almacenado. (GIT, 2016)

Características (GIT, 2016)**:**

- GIT es distribuido, una de las diferenciaciones más importante con otros repositorios como Subversión (SVN). No es necesario un servidor central, cada programador tiene una copia completa del repositorio en su equipo y al hacer un cambio se propaga a todos los nodos.

- Esto permite un gran rendimiento en grandes desarrollos, las búsquedas son mucho más eficaces lo que supone una gran rapidez para detectar diferencias entre archivos.
- Gratuito.
- Código Abierto.
- Multiplataforma: Linux, Windows.

JMeter 2.10

La aplicación JMeter es un software de código abierto, una aplicación desarrollada en Java y diseñada para cargar una conducta de funcionamiento de prueba y la medida de rendimiento. Fue diseñada originalmente para aplicaciones Web, pero desde entonces se ha expandido a otras funciones de prueba (apache.org, 2015).

JMeter es una herramienta de testing cuyas funcionalidades se pueden resumir en tres (apache.org, 2015):

- Diseñar un plan de pruebas esto es, generar un fichero .jmx.
- Ejecutar un plan de pruebas.
- Ver de distintas formas los resultados de la ejecución de un plan de pruebas.

JMeter puede ser utilizado para probar el rendimiento tanto en los recursos estáticos y dinámicos (Web Services (SOAP / REST), Web dinámicas idiomas - PHP, Java, ASP.NET, archivos, etc. -, objetos Java, bases de datos y consultas, servidores FTP y más). Se puede utilizar para simular una carga pesada en un servidor, grupo de servidores, la red para analizar el rendimiento general bajo diferentes tipos de carga. Se puede utilizar para realizar un análisis gráfico de rendimiento o para probar su comportamiento bajo carga pesada concurrente (apache.org, 2015).

JMeter implementa las siguientes funcionalidades (Junta de Andalucía, 2014):

- Simulación con el nivel de concurrencia que se desee, las interacciones de una o varias comunidades de usuarios virtuales (perfiles o roles) con la aplicación.
- Interfaz gráfica de usuario para diseñar casos de prueba.
- Posibilidad de grabar las interacciones de un usuario, vía navegador, con la interfaz HTTP de una aplicación, y convertir la grabación en un plan de pruebas.
- Ejecución de un testplan desde la interfaz GUI o desde línea de comando, permitiendo así la automatización de la ejecución de pruebas.
- Construcción de un plan de pruebas configurables.

- Visualización de los resultados de la ejecución de un plan de pruebas de múltiples formas, obteniendo diferentes métricas.

Ventajas (Junta de Andalucía, 2014)**:**

- Permite generar un informe de pruebas de resultados de manera automática sin necesidad de tratar los datos en hojas de cálculo.
- Permite estudiar de manera gráfica la relación de los diferentes indicadores de las pruebas, generando automáticamente gráficas de rendimiento.
- Desenmascara los posibles errores de la aplicación por la aplicación de concurrencia.
- Permite recuperar de manera íntegra los resultados de pruebas anteriores, o almacenarlos para posibles comparaciones de resultados

1.6 Patrones de diseño

Los patrones de diseño expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software (Patrones, 2010). Teniendo en cuenta que Bosón está desarrollado sobre el marco de trabajo Symfony2 el mismo hereda los patrones de diseño.

1.6.1 Patrones GRASP

Los patrones GRASP representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es el acrónimo para General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades) (Ingenería de Software, 2011).

Experto: Es un patrón que se usa para asignar responsabilidades; es un principio básico que suele ser útil en el diseño orientado a objetos. Tiene como beneficio que se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Soporta un bajo acoplamiento, lo cual favorece tener sistemas más robustos y de fácil mantenimiento. Además, el comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más cohesivas que son más fáciles de comprender y mantener (Ingenería de Software, 2011).

Creador: Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos (Ingenería de Software, 2011).

Alta Cohesión: Asigna una responsabilidad de modo que la cohesión siga siendo alta. En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme (Ingenería de Software, 2011).

Bajo Acoplamiento: Asignar una responsabilidad para mantener bajo acoplamiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Una clase con bajo (o débil) acoplamiento no depende de muchas otras (Ingenería de Software, 2011).

Controlador: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones: el sistema global (controlador de fachada), la entidad u organización global (controlador de fachada), algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas) (Ingenería de Software, 2011).

1.6.2 Patrones GoF

Los patrones GoF (Gang of Four, en español Pandilla de los Cuatro, formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides) se clasifican en 3 categorías basadas en su propósito: creacionales, estructurales y de comportamiento (Juan Pavón Mestras, 2004).

- **Creacionales**: Abstraen el proceso de creación de instancias y ocultan los detalles de cómo los objetos son creados o inicializados.
- **Estructurales**: Se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades.
- **Comportamiento**: Están relacionados con los algoritmos y la asignación de responsabilidades entre los objetos. Son utilizados para organizar, manejar y combinar comportamientos.

1.7 Patrones de arquitectura

La arquitectura de software pertenece a una rama del mundo de la ingeniería del software donde es la organización fundamental de un sistema. Las relaciones entre sus componentes permiten una interacción entre ellos.

Teniendo en cuenta que Bosón está desarrollado sobre el marco de trabajo Symfony2, el mismo hereda los patrones de su arquitectura base. El patrón de arquitectura Modelo Vista Controlador (MVC), conocido por sus siglas en inglés

Model View Controller, permite realizar la programación multicapa, separando en tres componentes distintos los datos de una aplicación, la interfaz del usuario y la lógica de control (Desarrolo Web, 2014).

Modelo: Representa la información con la que trabaja la aplicación, o sea, su lógica de negocio.

Vista: Convierte el modelo en una página web que facilita al usuario interactuar con ella.

Controlador: Es el encargado de procesar las interacciones del usuario y ejecuta los cambios adecuados en el modelo o en la vista. La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista), lo que permite un mantenimiento más sencillo de las aplicaciones. El controlador es el encargado de aislar al modelo y a la vista de los detalles del protocolo usado para las peticiones. El modelo se encarga de la abstracción de la lógica referida a los datos, lo que permite que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos que la aplicación utiliza.

Ilustración 1: Diagrama MVC (Desarrolo Web, 2014).

1.8 Prueba de validación

La validación es un conjunto de procesos de comprobación y análisis que aseguran que el software que se desarrolla está acorde a su especificación y cumple las necesidades de los clientes. También es una evaluación del sistema o de componentes, solo que es en el transcurso o al final del proceso del desarrollo, donde se determina si cumple con lo especificado. Además implica la valoración de los productos de trabajo para determinar si corresponde con las especificaciones. Incluye las especificaciones de requisitos, la documentación del diseño, diversos principios generales de estilo, estándares del lenguaje de instrumentación, estándares del proyecto, estándares organizacionales y expectativas del usuario, al igual que las meta especificaciones para los formatos y notaciones utilizadas en la especificación de productos diversos (Yarelis González, 2012).

Tipos de pruebas:

- **Pruebas de rendimiento:** La herramienta JMeter 2.10 es desarrollada en la plataforma Java dentro del proyecto Jakarta: JMeter permite realizar pruebas de rendimiento y pruebas funcionales sobre aplicaciones web. También permite la ejecución de pruebas entre distintos ordenadores para realizar pruebas de rendimiento (The Apache Software Foundation, 2016).
- **Pruebas de aceptación:** "La validación del software se logra mediante una serie de pruebas que demuestren que se cumple con los requisitos… Al construir software personalizado para un cliente se aplica una serie de pruebas de aceptación que permiten al cliente validar todos los requisitos" (Pressman, 2007). "La mayoría de los constructores de productos de software emplean procesos llamados prueba alfa y prueba beta para descubrir errores que sólo el usuario final podría detectar (Pressman, 2007). Las pruebas alfa se realizan en el lugar de trabajo del desarrollador, de esta manera se crea un ambiente controlado donde los usuario finales pueden trabajar con el sistema y registrar los errores y problemas. La **prueba alfa** se lleva a cabo en el lugar de desarrollo pero por un cliente. Se usa el software de forma natural con el desarrollador como observador del usuario y registrando los errores y los problemas de uso. Las pruebas alfa se llevan a cabo en un entorno controlado (Pressman, 2007).La **prueba beta** se lleva a cabo por los usuarios finales del software en los lugares de trabajo de los clientes. A diferencia de la prueba alfa, normalmente el desarrollador no está presente. Así, la prueba beta es una aplicación en vivo del software en un entorno que no puede ser controlado por el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador. Como resultado de los problemas informados durante la prueba, el desarrollador del software lleva a cabo modificaciones y así prepara una versión del producto de software para toda la clase de clientes (Pressman, 2007).

- **Pruebas de caja blanca:** La prueba de caja blanca, en ocasiones llamada prueba de caja de cristal, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Al emplear los métodos de prueba de caja blanca, el ingeniero de software podrá derivar casos de prueba que 1) garanticen que todas las rutas independientes dentro del módulo se han ejercitado por lo

menos una vez, 2) ejerciten los lados verdadero y falso de todas las decisiones lógicas, 3) ejecuten todos los bucles en sus límites y dentro de sus límites operacionales, 4) ejerciten estructuras de datos internos para asegurar su validez (Pressman, 2007).

- **Pruebas de caja negra:** Las pruebas de caja negra, también denominadas pruebas de comportamiento, se concentran en los requisitos funcionales del software. Es decir, permiten al ingeniero del software derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa. La prueba de caja negra no es una opción frente a las técnicas de caja blanca. Es, en cambio, un enfoque complementario que tiene probabilidades de descubrir una clase diferente errores de los que se descubrirán con los métodos de caja blanca. Las pruebas de caja negra tratan de encontrar errores en las siguientes categorías: 1) funciones incorrectas o faltantes, 2) errores de interfaz, 3) errores en estructuras de datos o en acceso a bases de datos externas, 4) errores de comportamiento o desempeño, 5) errores de inicialización y término (Pressman, 2007).

1.9 Conclusiones parciales

Durante el desarrollo del capítulo se realizaron una serie de análisis que permitieron identificar:

- La obsolescencia en la tecnologías y los problemas detectados en el rendimiento permitió identificar una necesidad de migración del módulo Vehículo del sistema Orbita para mejorar el rendimiento del sistema completo.
- El estudio de las herramientas y tecnologías permitió que la propuesta de solución contara con las especificaciones de la migración del módulo de Vehículo, ya que las herramientas utilizadas en el sistema Orbita se encuentran obsoletas y sin soporte.
- Se definió la arquitectura de Bosón como base para el desarrollo, adoptando el marco de trabajo Symfony2 y consecuentemente el lenguaje de programación PHP para el servidor.
- La metodología de desarrollo a emplear es la variante AUP de la Universidad de la Ciencias Informáticas (UCI) teniendo en cuenta las características del sistema a desarrollar.

Capítulo 2: Análisis y diseño.

2.1 Introducción

En el presente capítulo se realiza el análisis y diseño del módulo Vehículo del sistema Orbita presentando los diagramas de clases del diseño realizados, utilizando los estereotipos web. Además se analiza la migración del sistema teniendo en cuenta las tecnologías utilizadas en la arquitectura MVC. También se describen los requisitos funcionales y no funcionales.

2.2 Migración del sistema Orbita

En el mundo de las tecnologías y las comunicaciones existen diversas aplicaciones que dentro de unos años de creadas se vuelven obsoletas en cuanto a su lenguaje o marco de trabajo. A partir de las deficiencias identificadas en las tecnologías que se utilizaron en el desarrollo del sistema Orbita se decidió migrar el sistema para la arquitectura Bosón que tiene como base el marco de trabajo Symfony 2. Las tecnologías que componen al sistema Orbita están obsoletas y sin soporte. Se identificaron problemas de rendimiento en el sistema debido al manejo incorrecto de la caché ralentizando los tiempos de respuesta. Estos problemas impiden la creación de un cacheado correctamente almacenado y fácil de recuperar de peticiones previas. La utilización de Symfony 2 permite contar con una tecnología que contiene versiones mejoradas de Doctrine y un sistema de caché. Según (SensioLabs, 2016) Symfony tiene más de dos mil contribuyentes a sus proyectos, y cuenta con una cifra superior a los trescientos mil desarrolladores y cerca de cinco millones de descargas al mes.

Symfony 2 es un marco de trabajo basado en el patrón de diseño MVC lo que permite una mejor organización del código. El **modelo** incluye tanto la estructura de la base de datos como todo el código de consulta. La **vista** es el modo en que se van a representar los datos. El **controlador** se refiere al procesado de datos (borrowbits, 2013).

2.2.1 Modelo

La utilización de Doctrine como ORM[5] le permite a Symfony 2 separar la lógica de datos en una estructura bien definida a partir de clases. Doctrine es el encargado de gestionar todo lo referente al modelo de datos. Proporciona persistencia transparente de los objetos desde el lenguaje PHP. Utiliza el patrón Data Mapper en el núcleo del

[5] Mapeador de Objetos Relacional del inglés Object Relational Mapping

proyecto, logrando separar la lógica de dominio / negocio de la persistencia en un sistema de bases de datos relacional. Una de las principales características de Doctrine es su lenguaje SQL llamado Lenguaje de Consulta de Doctrine (DQL - Doctrine Query Language). Soporta las operaciones de Crear, Obtener, Actualizar y Borrar (CRUD - Create, Retrieve, Update and Delete) habituales, desde la creación de nuevos registros a la actualización de los antiguos. Crea manualmente y automáticamente el modelo de base de datos a implementar. Soporta varios motores de bases de datos (como MySQL, PostgreSQL, Microsoft SQL) (borrowbits, 2013).

Características (borrowbits, 2013):

- Sus objetos persistentes (llamadas entidades en Doctrine 2) no están obligados a extender más de una clase base abstracta. Doctrine 2 permite el uso de simples objetos de PHP.
- Presenta una arquitectura y potentes algoritmos que logran realizar un funcionamiento más rápido.
- La herencia no es un problema en la actualidad porque hay diferentes tipos para elegir.

2.2.2 Vista

La utilización de Bosón permite el uso de HTML5, CSS3 y JS, mezclados en el marco de trabajo Bootstrap en su versión 2 y resumidos en el Componente Portal. Además, el componente IUX establece plantillas base con la estrategia marcaria de la UCI. La estrategia marcaria es conjunto de pautas a tener en cuenta para la realización futura de los productos que serán desarrollados en los diferentes centros de la UCI, con el objetivo de estandarizar el diseño para garantizar una coherencia entre estos y lograr establecer una política de identidad corporativa dentro de la universidad en cuanto a sus aplicaciones. La misma tiene un conjunto de líneas temáticas como Xedro, Xilema, Xauce, Xavia y Xabal. Para ser consecuentes con el desarrollo se escoge la misma línea temática Xedro utilizada en el sistema Orbita original. Para hacer uso de estas plantillas Symfony cuenta para el manejo de las vistas con Twig, un motor de plantillas para PHP desarrollado por la empresa que creó Symfony. Los motores de plantillas proporcionan un lenguaje simplificado para las vistas y permiten un código más elegante. Además, facilitan la manipulación por parte de diseñadores y maquetadores sin conocimientos específicos del lenguaje. Twig nace con el objetivo de facilitar a los desarrolladores de aplicaciones web que utilizan la arquitectura MVC el trabajo con la parte de las vistas, gracias a que se

trata de un sistema que resulta muy sencillo de aprender y capaz de generar plantillas con un código preciso y fácil de leer.
Twig reúne las siguientes características (GitHub, 2013):

- Permite el uso de variables.
- Permite uso de funciones y métodos.
- Permite inclusión de vistas parciales.
- Facilita el uso de condicionales, bucles y asignaciones.
- Manejo de errores y excepciones.
- Herencia.

2.2.3 Controlador

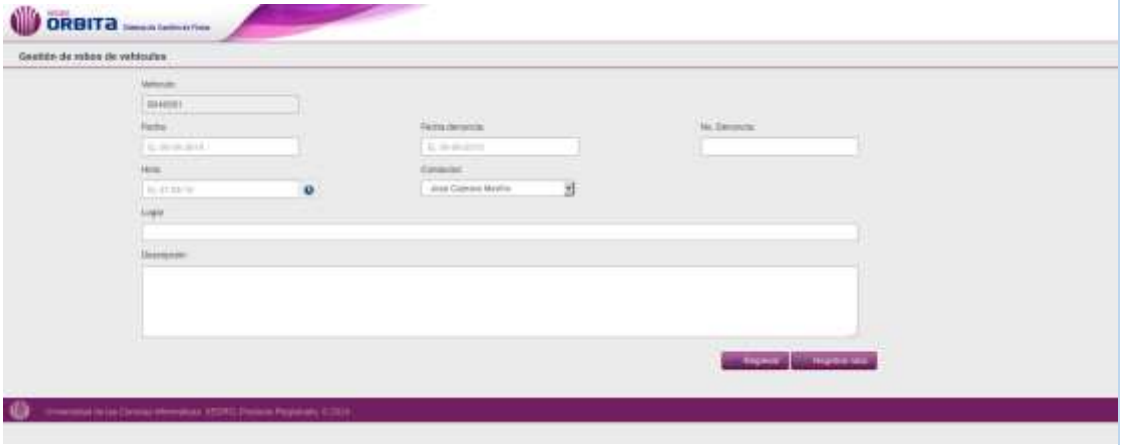
El controlador es la parte de la aplicación que contiene lo que se llama la lógica de negocio, que es una forma elegante de decir que cada controlador se encarga de una funcionalidad completa de la aplicación. En Symfony 2 todos los controladores extienden de una clase base Controller, la cual cuenta con funcionalidades como procesamiento de plantillas, creación de rutas, manejo de formulario y acceso a funciones de Doctrine para recuperación de datos. El controlador es el encargado de procesar las peticiones y a partir de la lógica implementada responder al usuario.

2.3 Historia de Usuarios para los requisitos funcionales de la aplicación.

Las Historias de Usuario (HU) son la técnica del escenario 4 utilizada en la variación AUP de la UCI para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer. El tratamiento de las historias de usuario es muy dinámico y flexible. En cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en pocas semanas.

A continuación, se muestra la HU Registrar incidencias, el resto de las HU podrán ser encontradas en el documento “Historias de Usuario Vehículo Orbita Bosón.docx” anexado al presente trabajo:

Historia de Usuario	
Número:HU24	**Nombre Historia de Usuario:** Registrar incidencia por robo.
Modificación de Historia de Usuario Número: 1	

Programador: Idelmis Tellez	**Iteración Asignada:** 1
Prioridad en Negocio: Media	**Tiempo Estimado:** 15 horas
Riesgo en Desarrollo: Poca experiencia de los estudiantes en las tecnologías de desarrollo.	**Tiempo Real:** 14 horas
Descripción: Se realiza el registrar incidencia por robo a aquellos vehículos que están en estado operativo.	
Observaciones: El vehículo debe tener asignado un conductor.	
Prototipo de interfaz: 	

2.4 Requisitos del negocio del sistema Orbita

La migración del módulo Vehículo del sistema Orbita se realiza con el objetivo de mejorar su rendimiento y que se integre con los demás módulos que se están desarrollando, para que los usuarios puedan interactuar con dicho sistema y utilizar los servicios que brindan estos módulos.

Los requisitos de un sistema describen los servicios que ha de ofrecer el sistema y las restricciones asociadas a su funcionamiento (DECSAI, 2010).Suelen especificar en lenguaje natural, se expresan de forma individual, se organizan de forma jerárquica (a distintos niveles de detalle) ya menudo se numeran (para facilitar su gestión) (DECSAI, 2010).

Los requisitos deben ser:

- **Claros y concretos** (evitando imprecisiones y ambigüedades).
- **Concisos** (sin rodeos ni figuras retóricas).
- **Completos y consistentes**.

Los requisitos deben indicar:

- Lo que se espera que haga el sistema (¿qué?)
- Su justificación (¿por qué ha de ser así? ¿quién lo propuso?)
- En su caso, los criterios de aceptación que sean aplicables (¿cómo se verifica su cumplimiento?)

2.4.1 Requisitos funcionales

Los requisitos funcionales de un sistema describen lo que el sistema debe hacer. Estos requisitos dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque general tomado por la organización al redactar requerimientos. Cuando se expresan como requisitos del usuario, habitualmente se describen de una forma bastante abstracta. Sin embargo, los requisitos funcionales del sistema describen con detalle la función de éste, sus entradas y salidas y excepciones. También constituyen la definición de los servicios que el sistema debe proporcionar, cómo debe reaccionar a una entrada particular y cómo debe reaccionar a una entrada particular y cómo se debe comportar ante situaciones particulares (Olivera Sosa, 2010).

Los requisitos funcionales:

- Describen el funcionamiento del sistema.
- Suelen expresar como objetivos del sistema.
- Deben describir los servicios que hay que proporcionar con todo detalle: los casos de uso.

Las técnicas utilizadas para la obtención de los requisitos funcionales (César Arturo Guerra, 2007):

Entrevista:

Es de gran utilidad para obtener información cualitativa como opiniones, o descripciones subjetivas de actividades. Es una técnica muy utilizada, y requiere una mayor preparación y experiencia por parte del analista. La entrevista se puede definir como un “intento sistemático de recoger información de otra persona” a través de una comunicación interpersonal que se lleva a cabo por medio de una conversación estructurada.

Observación:

Por medio de esta técnica el analista obtiene información sobre la forma en que se efectúan las actividades. Este método permite observar la forma en que se llevan a cabo los procesos y, por otro, verificar que realmente se sigan todos los pasos especificados. En muchos casos los procesos son una cosa en papel y otra muy diferente en la práctica.

Estudio de documentación:

Varios tipos de documentación, como manuales y reportes, pueden proporcionar al analista información valiosa con respecto a las organizaciones y a sus operaciones. La documentación difícilmente refleja la forma en que realmente se desarrollan las actividades, o donde se encuentra el poder de la toma de decisiones. Sin embargo, puede ser de gran importancia para introducir al analista al dominio de operación y el vocabulario que utiliza.

Requisitos funcionales (RF)

Tabla 2: Requisitos funcionales del sistema

No.	Requisito	No.	Requisito
RF1	Gestionar vehículo.	RF2.3	Modificar accidente.
RF1.1	Adicionar Vehículo.	RF2.4	Cancelar accidente.
RF1.1.1	Inicializar valores.	RF2.5	Imprimir accidente.
RF1.1.2	Asegurar vehículo.	RF2.6	Buscar accidente.
RF1.1.3	Quitar seguro del vehículo.	RF3	Registrar incidencia.
RF1.1.4	Listar accesorio vehículo.	RF3.1	Registrar incidencia por robo.
RF1.1.5	Adicionar accesorio vehículo.	RF3.2	Registrar incidencia por hurto.
RF1.1.6	Eliminar accesorio vehículo.	RF4	Gestionar registro de lectura.
RF 1.2	Modificar vehículo.	RF4.1	Listar registro de lectura.
RF1.2.1	Inicializar valores.	RF4.2	Registrar registro de lectura.
RF1.2.2	Asegurar vehículo.	RF4.3	Eliminar última lectura.
RF1.2.3	Quitar seguro vehículo.	RF4.4	Reiniciar medidor.
RF1.2.4	Listar accesorio vehículo.	RF4.5	Historial de lectura.
RF1.2.5	Adicionar accesorio vehículo.	RF4.6	Buscar lectura.
RF1.2.6	Eliminar accesorio vehículo.	RF5	Gestionar hoja de ruta.
R F1.3	Asignar vehículo.	RF5.1	Listar hoja de ruta.
RF 1.4	Listar vehículo.	RF5.2	Emitir hoja de ruta.
R F1.5	Buscar vehículo.	RF5.3	Modificar hoja de ruta.
RF1.6	Imprimir expediente vehículo.	RF5.4	Eliminar hoja de ruta.
RF2	Gestionar accidente.	RF5.5	Consultar hoja de ruta.

RF2.1	Listar accidente.	RF5.6	Buscar hoja de ruta.
RF 2.2	Adicionar accidente.		

2.4.2 Requisitos no funcionales

"Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Normalmente apenas se aplican a características o servicios individuales del sistema" (Sommerville, 2005).

2.4.2.1 Usabilidad:

RNU 1 El módulo debe presentar un acceso fácil y rápido, para facilitar su uso por usuarios con pocos conocimientos en el campo de la informática.

2.4.2.2 Diseño e implementación:

RNDI 1 Se utilizará PostgreSQL en su versión 9.1 como Sistema Gestor de Bases de Datos.

RNDI 2 Se utilizará PHP como lenguaje de programación y Symfony2 como marco de trabajo.

RNDI 3 Se utilizará Modelo-Vista-Controlador como arquitectura del sistema.

RNDI 4 Se utilizará PhpStorm como IDE para el desarrollo del sistema.

RNDI 5 Se utilizará Visual Paradigm como herramienta de modelado.

RNDI 6 Para el diseño de las páginas se utilizarán las siguientes tecnologías: CSS3, HTML5, JavaScript y JQuery.

2.4.2.3 Funcionamiento

Software:

RNFS 1 Para el funcionamiento en la PC cliente será necesario Firefox 28 o superior.

Hardware:

RNFH 1 El servidor para aplicaciones web deberá tener las siguientes características mínimas:

- Procesador Intel Pentium Dual Core a 2.0 Ghz, equivalente o superior.
- Tarjeta de red o capacidad de conectividad.
- 1 GB de memoria RAM.
- Capacidad de 40 GB de disco duro.

2.4.2.4 Seguridad

Confidencialidad:

RNFC 1 El acceso al módulo, así como a la información, se encontrarán protegidos contra accesos no autorizados utilizando para la autenticación el servicio de LDAP[6] que brinda la UCI.

RNFC 2 La autenticación del módulo será la primera acción del usuario, el cual deberá proporcionar un usuario único y una contraseña, los cuales serán de uso exclusivo del usuario.

RNFC 3 Las diferentes áreas del módulo se encontrarán protegidas contra acceso no autorizado utilizando roles.

Integridad:

RNFI 1 La información podrá ser modificada solo por el personal autorizado.

Fiabilidad:

RNFF 1 Ante una inserción de datos, vista de detalles, modificación o eliminación, el módulo debe responder en no más de 3 segundos.

RNFF 2 El módulo debe responder en un promedio de 4 segundos la petición de un reporte.

Interfaz de usuario:

RNFIU 1 Las interfaces del módulo contendrán los datos de forma estructurada, permitiendo la correcta interpretación de la información.

RNFIU 2 La entrada incorrecta de datos será mostrada al usuario claramente, detallando los campos donde se encuentra el error.

RNFIU 3 El diseño de la interfaz del sistema responderá con la ejecución de acciones de forma rápida, minimizando los pasos a dar en cada proceso.

2.5 Validación de los requisitos

El proceso de validación de requisitos comprende actividades que generalmente se realizan una vez obtenida una primera versión de la documentación de requisitos. Los requisitos una vez definidos necesitan ser validados. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos. Pocas son las propuestas existentes que ofrecen técnicas para la

[6] Protocolo compacto de acceso a directorios del inglés (Lightweight Directory Access Protocol)

realización de la validación y muchas de ellas consisten en revisar los modelos obtenidos en la definición de requisitos con el usuario para detectar errores o inconsistencias. La validación de los requisitos tiene como objetivo comprobar que estos son correctos. Esta fase debe realizarse o de lo contrario se corre el riesgo de implementar una mala especificación, con el costo que eso conlleva. Los parámetros a validar en los requisitos son (Dra.María del Carmen Gómez Fuentes, 2011):

- **Validez:** No basta con preguntar a un usuario, todos los potenciales usuarios pueden tener puntos de vista distintos y necesitar otros requisitos.
- **Consistencia:** No debe haber contradicciones entre unos requisitos y otros.
- **Completitud:** Deben estar todos los requisitos. Esto es imposible en un desarrollo iterativo, pero, al menos, deben estar disponibles todos los requisitos de la iteración en curso.
- **Realismo:** Se pueden implementar con la tecnología actual.
- **Verificabilidad:** Tiene que existir alguna forma de comprobar que cada requisito se cumple.

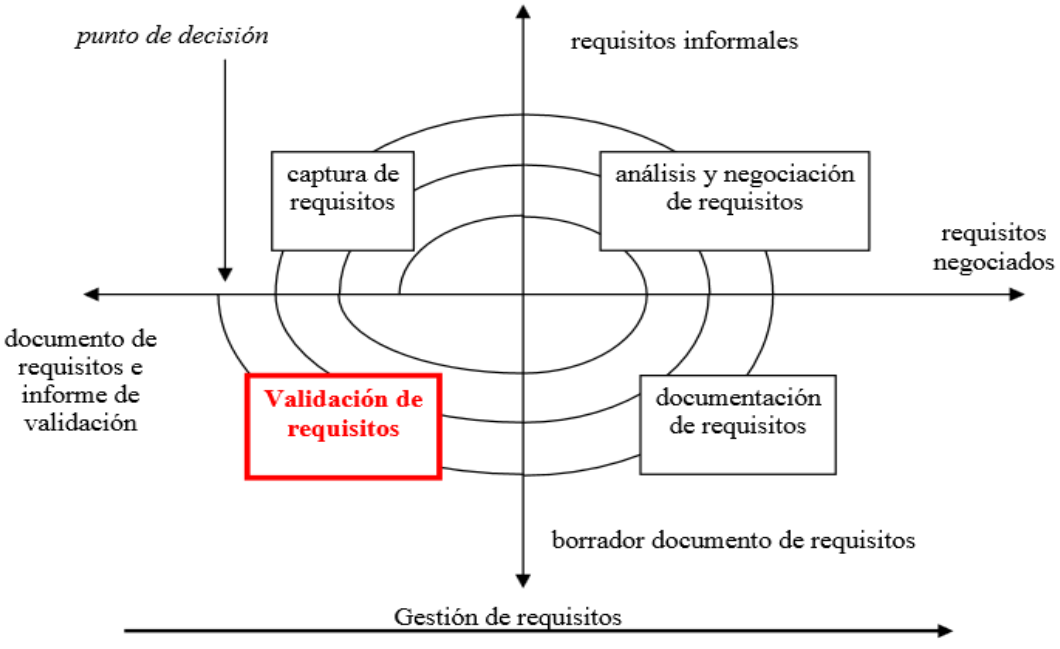


Ilustración 2: Validación en el proceso de requisitos (Paloma Cáceres, 2010-2011).

2.5.1 Técnicas para la validación de requisitos.

Las técnicas de validación de requisitos son: revisiones de requisitos, construcción de prototipos y generación de casos de prueba (Dra.María del Carmen Gómez Fuentes, 2011).

Una revisión de requisitos es un proceso manual en el que intervienen tanto el cliente como personal involucrado en el desarrollo del sistema. Puede ser formal o informal, y tiene el fin de verificar que el documento de requisitos no presente anomalías ni omisiones. En una revisión formal, los revisores deben tomar en cuenta:

- Que el requerimiento se pueda verificar de modo realista.
- Que las personas que adquieren el sistema o los usuarios finales comprendan correctamente el requerimiento.
- ¿Qué tan adaptable es el requerimiento? Es decir, ¿puede cambiarse el requisito sin causar efectos de gran escala en los otros requerimientos del sistema?

Las revisiones de requisitos consisten en una o varias reuniones planificadas, donde se intenta confirmar que los requisitos poseen los atributos de calidad deseados. Estas reuniones son llevadas a cabo por el analista encargado del proyecto y un conjunto de compañeros que, preferiblemente, no están relacionados con el proyecto y, además, son competentes en la actividad de requisitos. El resultado final de las reuniones de revisión es un documento que contiene la lista de defectos localizados y una lista de acciones recomendadas.

La construcción de prototipos consiste en mostrar un modelo ejecutable del sistema a los usuarios finales y a los clientes, así estos pueden experimentar con el modelo para ver si cumple con sus necesidades reales. Se utilizan, fundamentalmente, para comprobar la corrección y completitud de la especificación de requisitos. Existen varios tipos de prototipos, cada uno de los cuales permite la realización de un tipo determinado de pruebas y con un determinado nivel de realismo. Los más comunes son (Dra.María del Carmen Gómez Fuentes, 2011):

Mock-ups. Se trata de pantallas, típicamente dibujadas a mano en papel, que representan un aspecto concreto del sistema. El soporte que proporcionan a la validación es muy limitado, con la excepción, quizás, de aclarar el interfaz gráfico deseado en casos complejos.

Storyboards. Son una evolución de los mock-ups, ya que además del interfaz, se muestra la secuencia de acciones, o escenarios, que se deben realizar con el programa. Por ejemplo: es habitual que, antes de cerrar un programa, se pregunte si se desea cerrar el fichero de trabajo. Un storyboard permite la tarea de “cerrar programa”.

Los requisitos deben poder probarse, es por esto que debe hacerse una generación de casos de prueba. Si una prueba es difícil o imposible de diseñar, normalmente significa que los requisitos serán difíciles de establecer y deberían ser considerados nuevamente (Dra.María del Carmen Gómez Fuentes, 2011).

Los casos de prueba son artefactos bien definidos en el contexto de la prueba del software. En dicho contexto, un caso de prueba es la descripción de una acción bien definida que se debe realizar con el software. Por acción bien definida, debe entenderse que están perfectamente descritos tanto los datos de entrada como las tareas a realizar y los resultados esperados. Durante la validación de requisitos, los casos de prueba se utilizan del mismo modo que durante la prueba del software: es necesario poder describir, como se ha indicado anteriormente, tanto los datos de entrada como las tareas a realizar y los resultados esperados, para lo cual es necesario que estén perfectamente descritos los requisitos.

2.6 Diagrama de clases del diseño.

La creación de estos diagramas se lleva a cabo en la fase de ejecución. Describe gráficamente las especificaciones de las clases de software y de las interfaces.

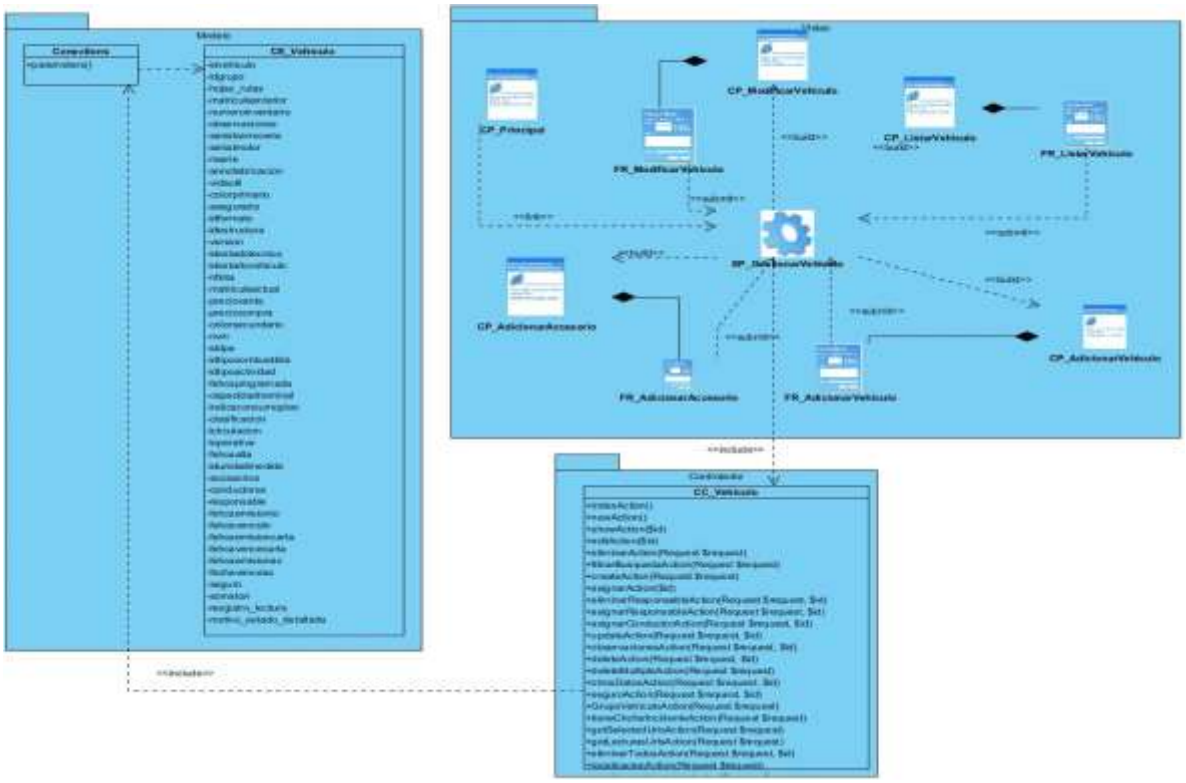


Ilustración 3: Diagrama de clases del diseño con estereotipos web.

En la ilustración se puede observar el diagrama de clase del diseño con estereotipos web Hoja de Ruta. Se realiza la petición a la Server Page (SP) desde la Client Page (CP). La SP realiza la búsqueda de los datos necesarios, procesa estos datos y los muestra a través de la vista.

2.7 Patrones de diseño

En la implementación del sistema se utilizaron varios patrones de diseño. A continuación se mencionan y se ponen ejemplo de estos patrones.

Patrones generales de software para asignación de responsabilidades (GRASP)

Experto:

Un ejemplo de este patrón se encuentra en la clase ¨Vehículo¨ que es la encargada de tener el control de los datos de los vehículos. Esta clase es la única que cuenta con todo la información referente al vehículo así como el manejo de las clases que tienen relación con el mismo.

Creador:

La creación de objetos es una actividad muy frecuente en los sistemas orientados a objetos, por lo tanto es conveniente asignar esta responsabilidad de manera que potencie un bajo acoplamiento, una mayor claridad y una alta reutilización. La nueva instancia debe ser creada por la clase que tenga la información mínima necesaria para hacerlo. Este patrón se ve evidenciado en la clase “VehiculoController”. Esta clase tiene la responsabilidad de crear instancias de la clase “Vehiculo”. El controlador está relacionado con la clase del dominio, recibiendo de la vista los datos necesarios para crear la instancia del objeto. “VehiculoController” es el creador de objetos de “Vehiculo”, como se evidencia en la Ilustración.

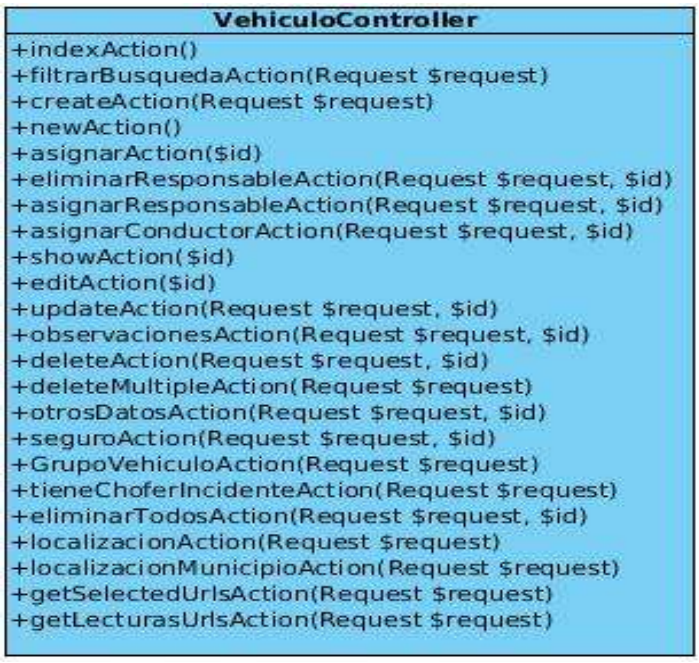


Ilustración 4: VehiculoController, patrón creador.

Alta cohesión:

Permite asignar responsabilidades con una alta cohesión ya que los controladores definen las acciones y colaboran con otras clases para realizar diferentes operaciones. Las clases controladoras contienen diferentes funcionalidades que se

encuentran estrechamente relacionadas posibilitando que el software sea flexible frente a grandes cambios.

Controlador:

Un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Un evento del sistema es un evento de alto nivel generado por un actor externo; un evento de entrada externa. El controlador es el encargado de recibir el evento, interpretarlo y ejecutar una determinada operación de respuesta. La utilización del marco de trabajo Symfony2 permite reconocer fácilmente este patrón ya que todos los controladores terminan con la palabra Controller.

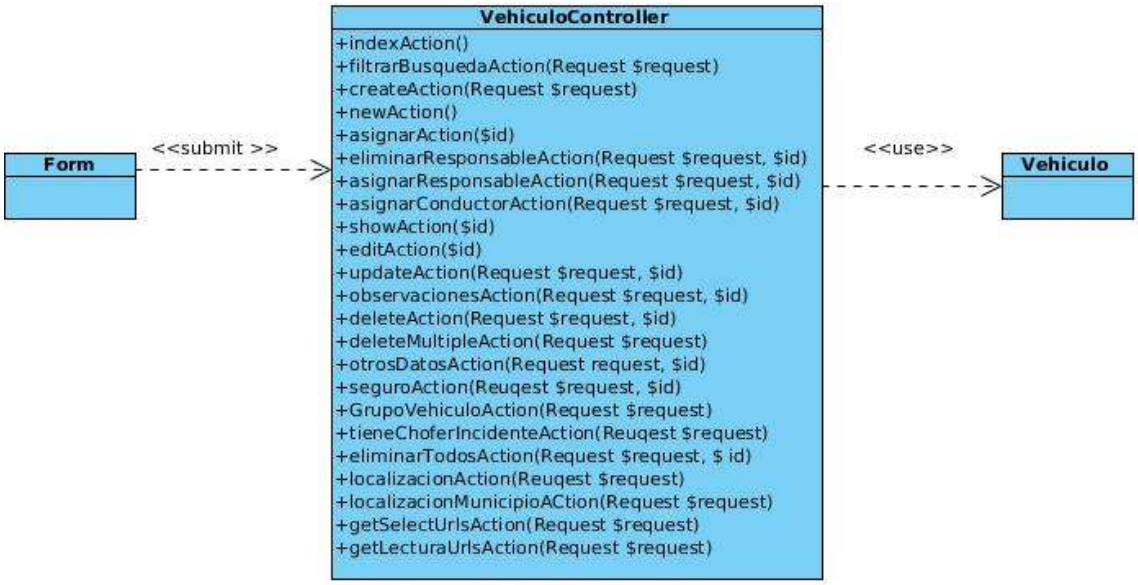


Ilustración 5: Patrón controlador.

Bajo Acoplamiento: La clase AccesorioController hereda únicamente de Controller para alcanzar un bajo acoplamiento de clases. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja.

Tabla 3: Patrón Bajo Acoplamiento

Controlador	Clase del dominio
Class AccesorioController extends Controller(){}	class Accesorio{ private $vehiculo; private $tipoaccesorio; private $codigo; private $estado; private $ninventario; public function getId(){} public function setCodigo($codigo){}

	public function getCodigo() public function setNinventario($ninventario) public function getNinventario() public function setVehiculo(Vehiculo $vehiculo) public function getVehiculo() public function setEstado(EstadoAccesorio $estado = null) public function getEstado() public function setTipoaccesorio(TipoAccesorio $tipoaccesorio) public function getTipoaccesorio() }

2.8 Mapeo de la base de datos

En el proceso de migración del sistema de Orbita es necesario conservar todos los datos que han sido creados con el uso del sistema actual. Para lograr este objetivo se utilizará la misma base de datos del sistema anterior para el desarrollo de los módulos. Esta base de datos se encuentra estructurada en esquemas y los nombres de las tablas se construyen “nombre_esquema.nombre_tabla”. Atendiendo a que la presente investigación se centra en el módulo Vehículo solo fueron creadas las clases de las tablas referentes al mismo. Para el funcionamiento del módulo es necesario manejar datos que se encuentran en 30 tablas del esquema “mod_mantenimiento”.

“Una de las tareas más comunes y difíciles para cualquier aplicación implica la persistencia y la lectura de información hacia y desde una base de datos. Symfony 2 viene integrado con Doctrine, una biblioteca cuyo principal objetivo es proveer una herramientas poderosas para hacer este proceso más fácil” (SensioLabs, 2016). Dentro de Symfony 2 estas tareas se realizan por las clases entidades y las clases repositorios. La clase entidad, es una clase básica que posee datos y ayuda a cumplir con el requerimiento de negocio. Esta clase no realiza persistencia a la base de datos, es una clase PHP simple. La clase repositorio es la encargada de realizar las consultas y manejar la persistencia de los datos. Symfony 2 propone la creación de una clase repositorio por cada clase entidad, y en esta clase manejar las consultas personalizadas.

Con el fin de crear una clase que responda a una tabla de la base de datos se utiliza las anotaciones. A continuación se puede observar la declaración de la clase entidad Vehículo, donde la anotación @ORM\Table especifica el nombre de la tabla a la cual

hace referencia a “mod_manteniemiento.dat_vehiculo” y la anotación @ORM\Entity especifica la clase repositorio donde se encuentran las consultas personalizadas para esta clase.

```
/**
 * Vehiculo
 *
 * @ORM\Table(name="mod_mantenimiento.dat_vehiculo")
 * @ORM\Entity(repositoryClass="Vehiculo\VehiculoBundle\Repository\VehiculoRepository")
 *
 */
class Vehiculo {
```

Ilustración 6: Mapeo de la clase Vehículo

La creación de entidades en Symfony 2 parten del principio de que estas clases tienen un atributo “id” que sea único y que en la mayoría de los casos es utilizado como llave primaria de la tabla a la que hace referencia la clase. A continuación se muestra la declaración de este atributo en la clase Vehículo, la anotación @ORM\Column permite definir el tipo de dato de este atributo en la base de datos así como el nombre de la columna a la cual hace referencia.

```
/**
 * @var integer
 *
 * @ORM\Column(name="idvehiculo", type="bigint")
 * @ORM\Id
 * @ORM\GeneratedValue(strategy="SEQUENCE")
 * @ORM\SequenceGenerator(sequenceName="mod_mantenimiento.sec_vehiculo_seq")
 */
private $id;
```

Ilustración 7: Atributo "id" de la clase Vehículo

Doctrine soporte varios tipos de relaciones entre tablas, a continuación se muestra la relación entre la clase Vehículo y la clase Estado Técnico.

```
/**
 * @ORM\ManyToOne(targetEntity="EstadoTecnico", fetch="EAGER")
 * @ORM\JoinColumn(name="idestadotecnico", referencedColumnName="idestadotecnico")
 */
private $estadotecnico;
```

Ilustración 8: Relación Uno a Muchos de Vehículo con Estado Técnico

Esta estructura propuesta por Symfony 2 le permite al desarrollador abstraerse de la base de datos en la implementación de la lógica de negocio y concentrarse en las funcionalidades que a su vez utilizan los objetos de las entidades creadas.

2.9 Conclusiones parciales

En este capítulo se abarcaron los aspectos más significativos del análisis y diseño como:

- La definición de la arquitectura para la migración permitió adecuar la estructura del módulo para lograr un mejor proceso de desarrollo.
- La utilización de patrones de diseño permitió adoptar buenas prácticas que permitan obtener un producto de mayor calidad.
- La realización de las historias de usuarios permitió definir la especificación de requisitos logrando un mayor entendimiento de las necesidades específicas del cliente.

Capítulo 3: Implementación y Pruebas

3.1 Introducción

La calidad del producto de software es de vital importancia. Este es un aspecto importante para el prestigio de las empresas desarrolladoras en el mundo tan competitivo que existe hoy. En el proceso de desarrollo de software las posibilidades de cometer errores son muy amplias, por lo cual se realizan pruebas y validaciones de los resultados para así reducir al máximo cualquier error encontrado. Las pruebas y validaciones se llevan a cabo en cada una de las etapas de desarrollo para detectar a tiempo las imperfecciones e irregularidades y proporcionar una visión objetiva de la madurez y calidad de los procesos asociados.

3.2 Diagrama de componente

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes o bibliotecas cargadas dinámicamente. Un diagrama de componentes muestra un conjunto de componentes y sus relaciones (Altanova, 2012).

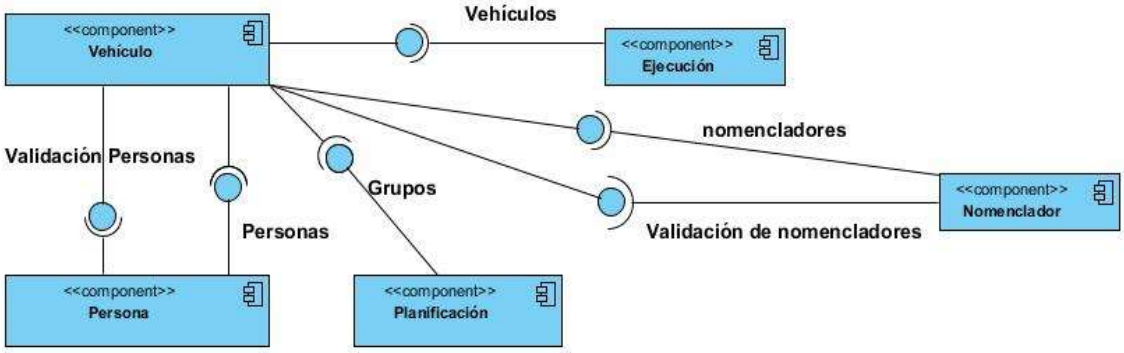


Ilustración 9: Diagrama de componente.

El módulo Vehículo consume servicio del módulo Recursos Humanos para obtener los datos de las personas para poder asignar un responsable de vehículo, así como su conductor. En las siguientes imágenes se presencia el consumo de servicio.

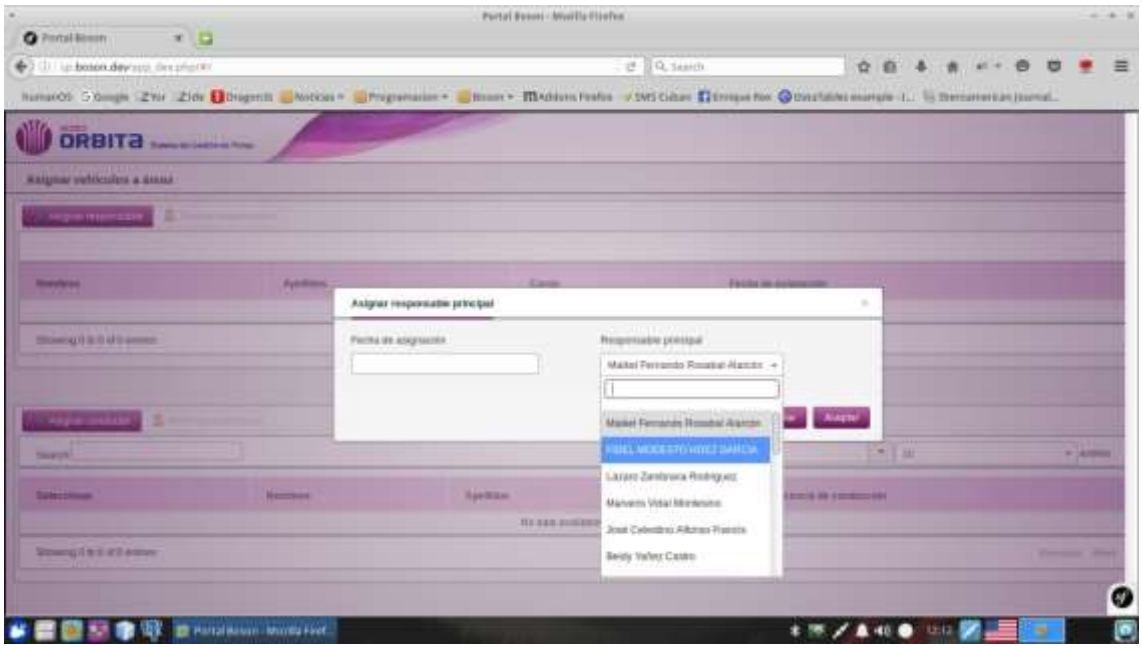


Ilustración 10: Funcionalidad asignar responsable.

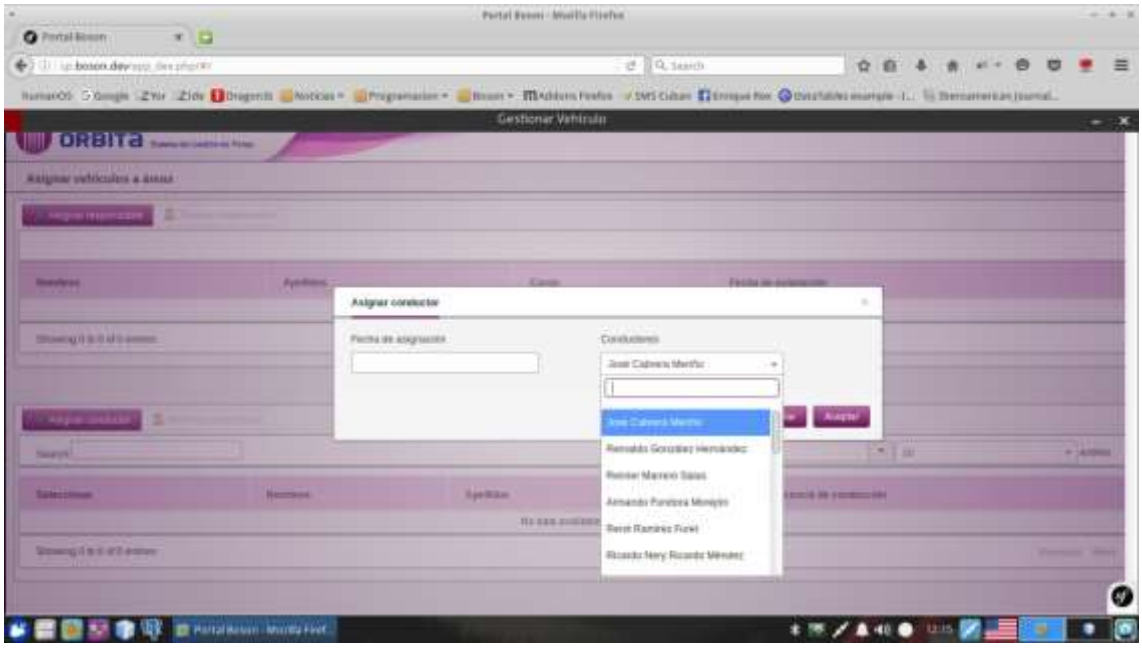


Ilustración 11: Funcionalidad asignar conductor.

El módulo Vehículo consume servicio del módulo Planificación al obtener los grupos de vehículos. Este consumo de servicio se evidencia en la funcionalidad adicionar vehículo. En la siguiente imagen se observar esta funcionalidad.

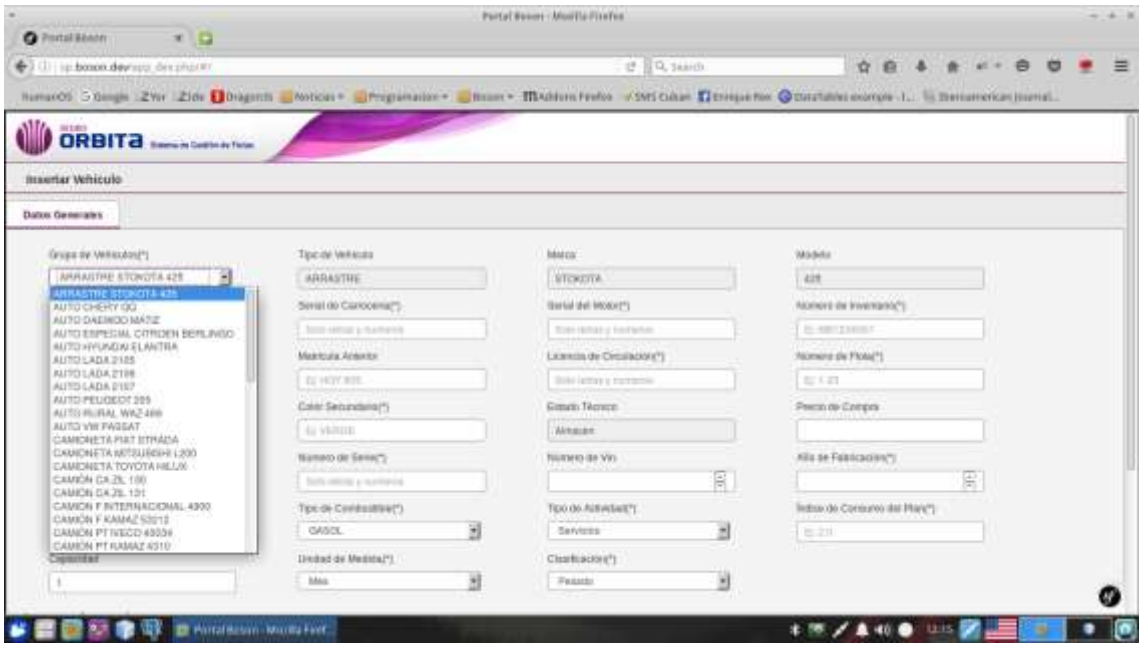


Ilustración 12: Funcionalidad adicionar vehículo.

3.3 Pruebas de software

La prueba de software ocupa el mayor porcentaje del esfuerzo técnico en el proceso del software. Su objetivo es descubrir errores; se cumple planeando y ejecutando una serie de pasos (pruebas de unidad, integración, validación y sistema). Las pruebas de unidad e integración se concentran en la verificación funcional de cada componente y en la incorporación de componentes en la arquitectura del software. La prueba de validación demuestra el cumplimiento con los requisitos del software, y la prueba del sistema valida el software una vez que se ha incorporado a un sistema mayor (Pressman, 2007).

Para probar y corregir el nuevo sistema de software fueron utilizadas las pruebas de caja blanca y caja negra llamadas pruebas funcionales o estructurales.

3.3.1 Prueba de caja negra: validación de la interfaz

El método utilizado para realizar las pruebas de caja negra a la aplicación se encuentra:

Partición equivalente

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre 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, 2007).

Tabla 4: Diseño de caso de prueba de Caja Negra

Escenario	Descripción	Matrícula anterior	Matrícula actual	Tipo de dato	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Adicionar vehículo introduciendo datos válidos.	El sistema debe permitir adicionar vehículo en el sistema.	*V* HTR 587	*V* *B034567*	*V* *varchar*	*V* Matrícula con que se identifica el vehículo.	Se activa otras opciones para trabajar en el vehículo.	Se introducen los datos del vehículo correctamente. -Se presiona el botón Aceptar.
EC 1.2 Adicionar vehículo introduciendo datos inválidos.	El sistema debe permitir adicionar vehículo en el sistema.	HTR34	*B0345*	*varchar*		Se muestra los campos en rojo.	Se introducen los datos Inválidos del vehículo. Se presiona el botón Aceptar. Se muestra un mensaje de error. Se presiona el botón Aceptar.
EC 1.3 Adicionar vehículo dejando campos vacíos.	Se dejan campos vacíos al adicionar un vehículo.					Se muestra los campos en rojo.	Se introducen los datos dejando algún campo en blanco. Se presiona el botón Aceptar. Se muestra

							campos en rojo. Se presiona el botón Aceptar.
EC 1.4 Cancelar.	Se cancela la operación de adicionar un vehículo.						Se introducen o no los datos del vehículo. Se presiona el botón Cancelar.

En la Tabla 5: Iteraciones de pruebas se puede observar el número de iteraciones y las no conformidades encontradas.

Tabla 5: Iteraciones de pruebas

Número de Iteraciones	Número de no conformidades	Asociadas a
1ra	30	Errores de interfaz, de validación y ortografía.
2da	10	Errores de interfaz, de validación y ortografía.
3ra	0	

3.3.2 Prueba de caja blanca: validación de la implementación

Dentro de las técnicas de prueba de caja blanca se encuentra la prueba del camino básico. Esta técnica le permite a los desarrolladores de casos de pruebas obtener una medida de la complejidad lógica de un diseño procedimental y provee una medida para definir un conjunto básico de rutas de ejecución. Para realizar esta técnica se debe construir una estructura que represente en notación simple el flujo del control, a esta representación se le llama gráfica de flujo. Una de las dificultades fundamentales de esta técnica es la determinación del número de rutas, para esto se utiliza la métrica de software complejidad ciclomática. Esta métrica consiste en determinar una medida cualitativa de la complejidad lógica del programa. “Cuando se emplea en el contexto del método de prueba de la ruta básica, el valor calculado mediante la complejidad ciclomática define el número de rutas independientes en el conjunto básico de un programa, y proporciona un límite superior para el número de

pruebas que deben aplicarse para asegurar que todas las instrucciones se hayan aplicado por lo menos una vez" (Pressman, 2007).

```php
public function eliminarConductoresAction(Request $request, $id) {
    $em = $this->getDoctrine()->getManager();
    $vehiculo = $em->getRepository('VehiculoBundle:Vehiculo')->find($id);
    $conductores_ids = $request->request->get('choose');
    $cantidad = 0;
    if ($conductores_ids) {
        foreach ($conductores_ids as $id_conductor) {
            $conductor = $em->getRepository('VehiculoBundle:Conductor')->findOneBy(array('idtrabajador' => $id_conductor, 'vehiculo' => $vehiculo));
            if (!is_null($conductor)) {
                $cantidad++;
                $em->remove($conductor);
                $em->flush();
            }
        }
        if ($cantidad == 0) {
            $mensaje = 'No fue eliminado ningún conductor';
        } else if (count($conductores_ids) == $cantidad) {
            $mensaje = 'Fueron eliminadas todos los conductores';
        } else {
            $mensaje = 'No fueron eliminados todos los conductores';
        }
    } else {
        $mensaje = 'No fue eliminado ningún conductor';
    }

    return new JsonResponse(array('mensaje' => $mensaje));
}
```

Ilustración 13: Camino básico, método eliminarConductores

Para el cálculo de la complejidad ciclomática se tiene en cuenta que esta corresponde al número de regiones. Estas regiones son calculadas utilizando la fórmula V (G) = (E-N)+2, donde E representa el número de aristas, N el número de nodos y V (G) es la complejidad ciclomática.

Tabla 6: Prueba de Caja Blanca, Camino básico

Prueba estructural de Caja Blanca	Probador: Idelmis Tellez Peña
El número de regiones corresponde a la complejidad ciclomática. E=17 N=13 V (G) = (E – N) + 2=6 Rutas linealmente independientes: 1) 1-3-13	Representación de la gráfica de flujo.

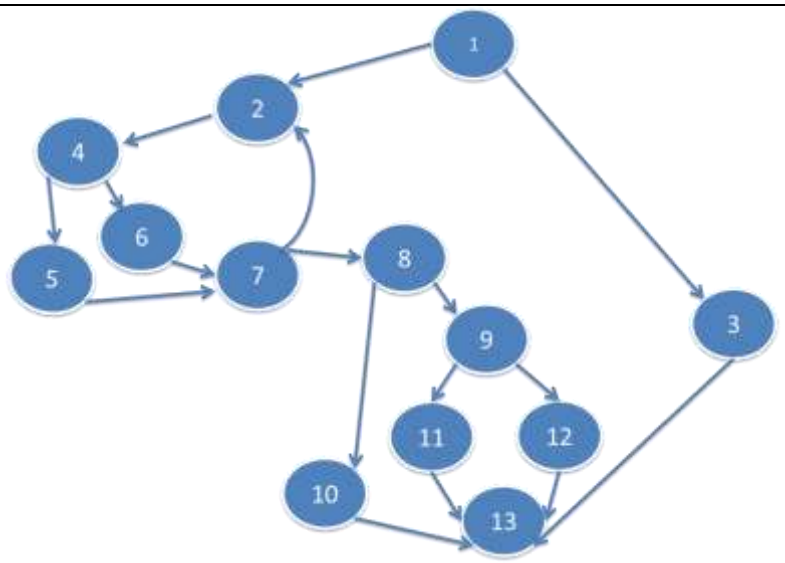
2) 1-2-4-5-7-2-4-5-7-8-9-11-13 3) 1-2-4-5-7-2-4-6-7-8-9-12-13 4) 1-2-4-6-7-2-4-6-7-8-10-13	

Tabla 7: Caso de prueba, Camino básico 1

Caso de prueba para el camino básico 1	
Descripción: Se realiza la petición sin ningún id.	
Condición de ejecución: En el objeto request no aparezca ningún id para eliminar.	
Procesamiento de la prueba	
Dato de entrada:	En el objeto request no aparece el campo choose.
Tipo de dato esperado:	Objeto request, identificador del vehículo.
Resultados esperados:	`{"mensaje":"No fue eliminado ningún conductor"}`

Tabla 8: Caso de prueba, Camino básico 2

Caso de prueba para el camino básico 2	
Descripción: Se realiza la petición con 2 ids válidos para los conductores.	
Condición de ejecución: Todos los ids son válidos.	
Procesamiento de la prueba	
Dato de entrada:	Choose=[90000000289, 90000000299] arreglo con los id de los conductores.
Tipo de dato esperado:	Objeto request, identificador del vehículo.
Resultados esperados:	`{"mensaje":"Fueron eliminadas todos los conductores"}`

Tabla 9: Caso de prueba, Camino básico 3

Caso de prueba para el camino básico 3	
Descripción: Se realiza la petición con un id válido y otro no válido para conductores.	
Condición de ejecución: Al menos un id no es válido.	
Procesamiento de la prueba	
Dato de entrada:	Choose=[90000000289, 25] arreglo con los id de los conductores.
Tipo de dato esperado:	Objeto request, identificador del vehículo.
Resultados esperados:	`{"mensaje":"No fueron eliminados todos los conductores"}`

Tabla 10: Caso de prueba, Camino básico 4

Caso de prueba para el camino básico 4	
Descripción: Se realiza la petición con ids no válidos para conductores.	
Condición de ejecución: Ninguno de los ids es válido.	
Procesamiento de la prueba	
Dato de entrada:	Choose=[30, 25] arreglo con los id de los conductores.
Tipo de dato esperado:	Objeto request, identificador del vehículo.
Resultados esperados:	`{"mensaje":"No fue eliminado ningún conductor"}`

Luego de aplicar la prueba del camino básico se constató que el método eliminar conductores se comportó de la manera esperada mostrando los resultados de una manera correcta.

3.4 Validación del rendimiento del sistema

A continuación se muestra la prueba realizada al módulo Vehículo del sistema Orbita anterior y al nuevo módulo, persiguiendo como objetivo verificar si existe una mejora en el rendimiento.

3.4.1 Pruebas del sistema

Las pruebas del sistema ejercitan profundamente el sistema con el objetivo de comprobar que se hayan integrado adecuadamente todos los elementos del software y del hardware, y que se realizan las funciones adecuadas. Las pruebas ya aplicadas

se centran en el código, de ahí surge la necesidad de comprobar el rendimiento y la respuesta del sistema ante peticiones. Para la realización de estas pruebas se utilizó como servidor una máquina con las siguientes prestaciones: microprocesador Intel(R) Core(TM) i3 a 1.60GHz, 4Gb de memoria RAM y 40Gb de disco duro dedicados al sistema.

Resultados de las pruebas de rendimiento

A continuación se muestran, en la Tabla 9, los resultados obtenidos de la aplicación antes y después de la migración.

Indicadores

- **Media:** tiempo de ejecución promedio de una petición con un usuario.
- **Columna Máximo:** tiempo máximo de ejecución invertido para una petición con un usuario realizando peticiones de manera simultánea.
- **Rendimiento:** velocidad del sistema.

Tabla 11: Comparación de los dos sistemas Orbita

	Sistema Orbita viejo	Sistema Orbita nuevo
Media	225 ms	447 ms
Máximo	12008 ms	2770 ms
Rendimiento	4.3 s	2.2 s

En la tabla se reflejan los tiempos de respuestas del sistema Orbita viejo y nuevo. A pesar de que el tiempo de ejecución promedio de una petición con un usuario en el sistema viejo fue de 225 milisegundos y en el sistema nuevo fue de 447, el autor considera poco significativo este aumento teniendo en cuenta que este parámetro se encuentra por debajo del medio segundo. Se puede constatar además que existe una diferencia de casi 10 segundos en el tiempo máximo de respuesta. Por último, se constata que la velocidad de respuesta del sistema anterior según la herramienta es de 4.3 segundos y en el nuevo sistema 2.2 segundos. De esta manera se observa una reducción de 2.1 segundos (2100 milisegundos).

Los ambientes de prueba del rendimiento de las dos aplicaciones distan en cuanto a las prestaciones de las computadoras servidoras, favoreciendo al sistema antiguo en este aspecto. A pesar de esta diferencia se observa una mejora en dos de los tres parámetros medidos en cuanto al tiempo de respuesta, por lo que se concluye que el rendimiento del módulo Vehículo mejoró. Aunque se observa una mejora en el rendimiento del módulo, es necesario esperar a la integración de todos los módulos del sistema Orbita, para comprobar si existe una mejora en el rendimiento.

3.5 Prueba de aceptación

El objetivo fundamental de estas pruebas es validar si las HU fueron implementadas según las especificaciones del módulo Vehículo.

Tabla 12: Iteraciones de los casos de pruebas basados en las HU

Iteración	No conformidades
1ra	18
2da	6
3ra	0

3.6 Conclusiones parciales

El análisis de la estructura del módulo permitió:

- Identificar necesidades de información que provienen de otros módulos del sistema y que son necesarios para el funcionamiento de la solución propuesta.
- El diagrama de componentes permitió esclarecer las fuentes de esta información y su manejo a través de servicios.
- Se realizaron prueba de Caja Blanca y de Caja Negra con el fin de encontrar y corregir errores para validar funcionalmente el sistema.
- El uso de varias técnicas de prueba permitió corregir deficiencias y obtener un producto final.
- A partir de las pruebas de aceptación se determinó que funcionalmente el módulo satisface las necesidades de los clientes y se obtuvo la carta de aceptación por parte de los mismos.
- Las pruebas de rendimiento realizadas tanto al módulo Vehículo del sistema anterior, como a la propuesta de solución permitieron realizar una comparación y determinar que si mejoran los parámetros analizados.

Conclusiones generales

El análisis del sistema Orbita implementado anteriormente arrojó que existen dificultades en su funcionamiento debido a que las tecnologías utilizadas en el desarrollo ya se encuentran obsoletas y sin soporte. Atendiendo a estas dificultades se decidió migrar el sistema a una nueva tecnología y por consiguiente es necesario migrar las funcionalidades del Módulo Vehículo.

El análisis de las historias de usuarios generadas en la implementación del sistema anterior, así como de los requisitos derivados de las mismas, permitieron redefinir las nuevas historias de usuarios y los nuevos requisitos.

La adopción de la arquitectura Bosón conllevó al rediseño de la estructura de los componentes y clases. Las diferencias entre ZendFramework y Symfony 2.7 se introducen cambios sustanciales en la estructura de los componentes.

El proceso de implementación permitió desarrollar el producto a partir del diseño elaborado, cumpliendo con todos los requisitos redefinidos del sistema anterior y la arquitectura Bosón.

Se comprobó que el nuevo módulo Vehículo cumplía con todas las especificaciones del cliente y se obtuvo el certificado de aceptación.

Se comprobó que el nuevo módulo Vehículo mejoró su rendimiento respecto al módulo del sistema anterior a través de las pruebas realizadas.

Recomendaciones

Realizar la integración del módulo Vehículo con el resto de los módulos del sistema Orbita implementados sobre la arquitectura de referencia en PHP Bosón, de manera tal que funcione como un todo y supla las necesidades del Control de Flotas y Mantenimiento.

Realizar las pruebas de rendimiento al sistema Orbita integrando los módulos que lo componen con el fin de comprobar si mejoró el rendimiento del mismo.

Bibliografía

Ingeniería DEC México. 2014. http://www.ingenieriadec.com. *http://www.ingenieriadec.com.* [En línea] 2014.

Junta de Andalucía. 2014. Marco de Desarrollo de la Junta de Andalucía. [En línea] 2014. [Citado el: 28 de febrero de 2016.] http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/388.

Altanova. 2012. altanova. *altanova.* [En línea] 2012. http://www.altova.com.

apache.org. 2015. http://jmeter.apache.org/. *http://jmeter.apache.org/.* [En línea] 12 de 1 de 2015. [Citado el: 20 de febrero de 2016.]

Autoridad Nacional de los Servicios Públicos Panamá. 2011. PROCEDIMIENTO PARA LA ADMINISTRACIÓN DE LA FLOTA VEHICULAR. [En línea] 2011. http://www.asep.gob.pa.

2012. http://navegasfa.blogspot.com/. *http://navegasfa.blogspot.com/.* [En línea] 26 de enero de 2012. [Citado el: 22 de febrero de 2016.]

borrowbits. 2013. borrowbits. *borrowbits.* [En línea] 2013. [Citado el: 26 de marzo de 2016.] http://borrowbits.com.

César Arturo Guerra. 2007. SG Buzz. *SG Buzz.* [En línea] octubre de 2007. [Citado el: 20 de mayo de 2016.] https://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia.

Cluster de Investigación Aplicada. 2016. Cluster de Investigación Aplicada. [En línea] Cluster de Investigación Aplicada, 2016. [Citado el: 10 de Febrero de 2016.] http://www.pol.una.py.

Crysfel Villa. 2016. https://quizzpot.com. *https://quizzpot.com.* [En línea] 12 de 4 de 2016.

CSS3. 2012. CSS3.info. [En línea] WEBFLUX, 2012. http://www.css3.info/.

DECSAI. 2010. http://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf. *http://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf.* [En línea] 2010.

Desarrollo Web. 2014. Desarrollo Web. [En línea] 2014. [Citado el: 15 de mayo de 2016.] http://www.desarrolloweb.com/articulos/que-es-mvc.html.

Desarrolo Web. 2014. http://www.desarrolloweb.com/articulos/que-es-mvc.html. *http://www.desarrolloweb.com/articulos/que-es-mvc.html.* [En línea] 2014.

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

Doctrine Project. 2016. Doctrine Project. [En línea] 2016. http://www.doctrine-project.org/.

Dra.María del Carmen Gómez Fuentes. 2011. 2011.

EDUCAMERICAS. 2008-2016. EDUCAMERICAS. *EDUCAMERICAS.* [En línea] 2008-2016. [Citado el: 12 de febrero de 2016.] http://www.educamericas.com/articulos/reportajes/la-importancia-de-las-tic-en-el-mundo-empresaria.

ETECSA. 2014. Empresa de Telecomunicaciones de Cuba S.A. [En línea] ETECSA, 2014. http://www.etecsa.cu.

Fabien Potencier. 2013. LibrosWeb. *LibrosWeb.* [En línea] 2013. [Citado el: 15 de marzo de 2016.] http://librosweb.es/libro/symfony_1_2/capitulo_1/symfony_en_pocas_palabras.html.

ferruiz.com. 2008. http://www.ferruiz.com. *http://www.ferruiz.com.* [En línea] 26 de abril de 2008. [Citado el: 20 de febrero de 2016.]

Ful-Mar. 2016. Ful-Mar. *Ful-Mar.* [En línea] 2016. [Citado el: 15 de 1 de 2016.] http://www.ful-mar.com.ar.

Gamma, Erich, y otros. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 1995.

García, MSc. María Elena. 2016. Cluster de Investigación Aplicada. *Sistema de Gestión de Flota de Vehículos - ANDE.* [En línea] 2016. http://www.pol.una.py/cia/?q=node/44.

García, MSc. María Elena y González Rodas, Lic. Guillermo. 2016. Cluster de Investigación Aplicada. *Sistema de Gestión de Flota de Vehículos - ANDE.* [En línea] 2016. http://www.pol.una.py/cia/?q=node/44.

Gestión de Flotas. 2010. Gestión de Flotas. *Planes de mantenimiento de vehículos y organización del tráfico.* [En línea] 2010. [Citado el: 20 de 1 de 2016.] https://www.fundacionmapfre.org.

Gilbarco Inc. 2016. Gilbarco. [En línea] 2016. http://www.gilbarco.com.

GIM. 2009. http://www.sociopartner.cl. *http://www.sociopartner.cl.* [En línea] 2009. [Citado el: 12 de abril de 2016.]

Git. 2016. https://git-scm.com. *https://git-scm.com.* [En línea] 2016.

GIT. 2016. https://git-scm.com. *https://git-scm.com.* [En línea] 10 de 1 de 2016. [Citado el: 1 de Febrero de 2016.]

GitHub. 2013. https://github.com. *https://github.com.* [En línea] 27 de 11 de 2013. [Citado el: 26 de marzo de 2016.] https://github.com.

HTML5. 2011. [En línea] 2011. [Citado el: 20 de marzo de 2016.] http://www.html5tutorial.info/.

Informáticas, Universidad de la Ciencias. Eva.uci.cu. *Eva.uci.cu.* [En línea] [Citado el: 12 de 10 de 2015.] http://eva.uci.cu/.

Ingenería de Software. 2011. http://www.cartagena99.com/recursos/alumnos/apuntes/Patrones%20de%20Diseno.pdf. *http://www.cartagena99.com/recursos/alumnos/apuntes/Patrones%20de%20Diseno.pdf.* [En línea] 2011.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 1999. *The Unified Software Development Process.* s.l. : Addison Wesley Longman, 1999.

JavaScript. 2016. JavaScript.com. [En línea] JavaScript.com, 2016. https://www.javascript.com/.

Juan Pavón Mestras. 2004. http://www.fdi.ucm.es/profesor/jpavon/poo/2.14pdoo.pdf. *http://www.fdi.ucm.es/profesor/jpavon/poo/2.14pdoo.pdf.* [En línea] 2004. [Citado el: 12 de abril de 2016.]

Lieberman, Gerald J. 2005. *Introducción a la Investigación de Operaciones.* Habana : Félix Varela, 2005.

Marta E. Zorrilla Pantaleón. 2011. http://personales.unican.es. *http://personales.unican.es.* [En línea] 2011. [Citado el: 22 de marzo de 2016.]

mastermagazine. 2015. mastermagazine. *mastermagazine.* [En línea] 2015. [Citado el: 15 de 1 de 2016.] http://www.mastermagazine.info/termino/5893.php.

Microsoft. 2015. Microsoft. [En línea] Microsoft, 2015. [Citado el: 9 de Septiembre de 2015.] http://www.microsoft.com.

Mozila. 2012. Mozila. [En línea] 2012. [Citado el: 15 de mayo de 2016.] https://www.mozilla.org/es-MX/firefox/desktop/.

1998-2012. mozila.org. *mozila.org.* [En línea] 1998-2012. [Citado el: 12 de 11 de 2015.]

Mozilla.org. 1998-2012. Mozilla.org. *Mozilla.org.* [En línea] 1998-2012. [Citado el: 20 de enero de 2016.]

MozillaES. 2009. MozillaES. *MozilaES.* [En línea] 2009. [Citado el: 20 de enero de 2016.]

Nielsen, Jakob. 2000-2016. *Designing Web Usability: The Practice of Simplicity.* s.l. : Prentice-Hall, 2000-2016. pág. 432.

Olivera Sosa. 2010. Requerimientos-funcionales-y-no-funcionales. *Requerimientos-funcionales-y-no-funcionales.* [En línea] 2010. https://es.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales.

ONE. 2007. *Tecnologías de la Informacion y las Comunicaciones(TIC).Uso y acceso en Cuba.* 2007.

Pacheco, Juan Manuel Fernández. 2010. Real Academia Española. [En línea] Madrid,España, 2010. [Citado el: 3 de 2 de 2016.]

Paloma Cáceres. 2010-2011. Análisis e Ingeniería de Requisitos. [En línea] 2010-2011. [Citado el: 24 de mayo de 2016.] http://www.kybele.etsii.urjc.es/docencia/air_gis_m/2010-2011/Material/%5BAIR-1011%5D%20TEMA%205-6-7.pdf.

Patrones. 2010. Patrones de diseño. [En línea] 2010. [Citado el: 24 de marzo de 2016.] patronesdediseno.net16.net.

pdAdmin. 2016. pdAdmin. *pdAdmin.* [En línea] 2016. [Citado el: 16 de marzo de 2016.] https://www.pgadmin.org/.

Pikuru. 2012-2014. Pikuru. *Pikuru.* [En línea] 2012-2014. [Citado el: 21 de enero de 2016.] http://www.pikuru.com.

Plusesmas. 2015. Plusesmas. *Plusesmas.* [En línea] 2015. [Citado el: 7 de mayo de 2016.] http://www.plusesmas.com.

PostgreSQL-es. 2009-2012. PostgreSQL-es. *PostgreSQL-es.* [En línea] 2009-2012. [Citado el: 15 de 11 de 2015.]

PostgresSQL-es. 2009-2012. PostgresSQL-es. *PostgresSQL-es.* [En línea] 2009-2012. [Citado el: 19 de enero de 2016.]

Potencier, Fabien. 2012. Libros.es. *Symfony la guía definitiva.* [En línea] 2012. [Citado el: 15 de Octubre de 2015.] http://es.slideshare.net/nereosa/symfony-guia-definitiva.

Pressman. 2005. *Ingenería de software.* 2005.

Pressman, Roger. 2007. *Ingeniería del Software, Un enfoque práctico.* s.l. : Mc Graw Hill, 2007.

Sencha Inc. 2016. Sencha. [En línea] Sencha Inc , 2016. https://www.sencha.com/.

SensioLabs. 2016. SensioLabs. *SensioLabs.* [En línea] SensioLabs, 2016. [Citado el: 1 de Mayo de 2016.] https://symfony.com/what-is-symfony.

Sommerville, Ian. 2005. *Ingeniería del software.* Madrid : Pearson Educación S.A., 2005. 84-7829-074-5.

TechTarget. 2016. TechTarget. [En línea] 2016. http://whatis.techtarget.com.

—. 2016. TechTarget. *TechTarget.* [En línea] 2016. [Citado el: 15 de 1 de 2016.] http://searchcio.techtarget.com.

The Apache Software Foundation. 2016. The Apache Software Foundation. *The Apache Software Foundation.* [En línea] 2016. [Citado el: 15 de mayo de 2016.] http://jmeter.apache.org/.

The PHP Group. 2016. PHP. [En línea] The PHP Group, 2016. http://php.net/.

transoft. 2012. http://www.transoft.transnet.cu. *http://www.transoft.transnet.cu.* [En línea] 2012.

TRAZOSWEB. 2010. TRAZOSWEB. *TRAZOSWEB.* [En línea] 2010. [Citado el: 21 de enero de 2016.] http://www.trazos-web.com.

Universidad de las Ciencias Informáticas. 2016. Comunidad cubana de PHP. [En línea] 2016. [Citado el: 18 de Enero de 2016.]

—. 2012. *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : Universidad de las Ciencias Informáticas, 2012.

Yarelis González. 2012. yarelisgonzalez.blogcindario.com. [En línea] 2012. http://yarelisgonzalez.blogcindario.com/2012/01/00003-tipos-de-pruebas-y-validacion-del-software.html.

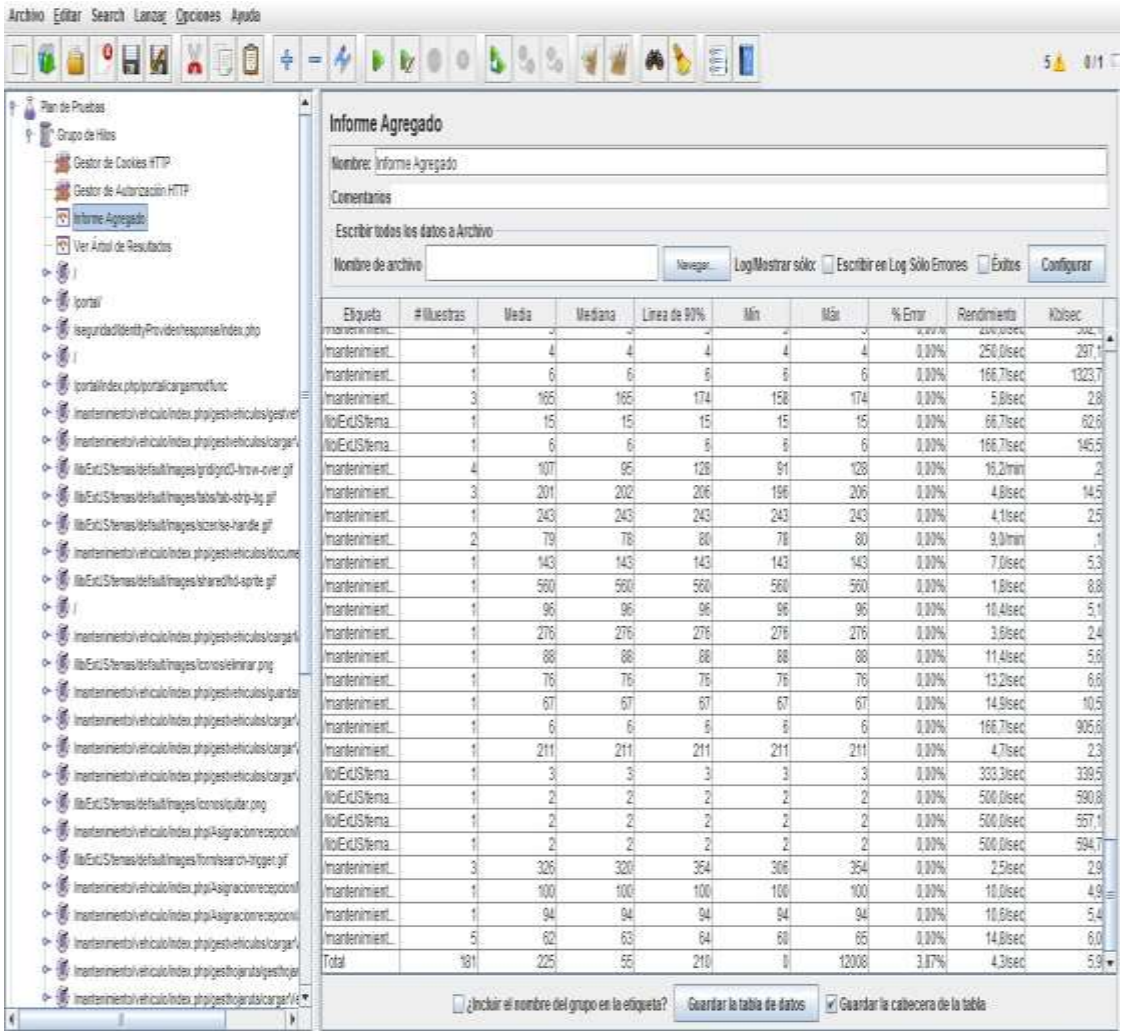
Zend framework. 2016. Zend framework. [En línea] 2016. [Citado el: 12 de febrero de 2016.] http://framework.zend.com/downloads/archives.

Zend Technologies Ltd. 2016. Zend Framework 2. [En línea] Zend Technologies Ltd, 2016. [Citado el: 25 de Enero de 2016.] http://framework.zend.com.

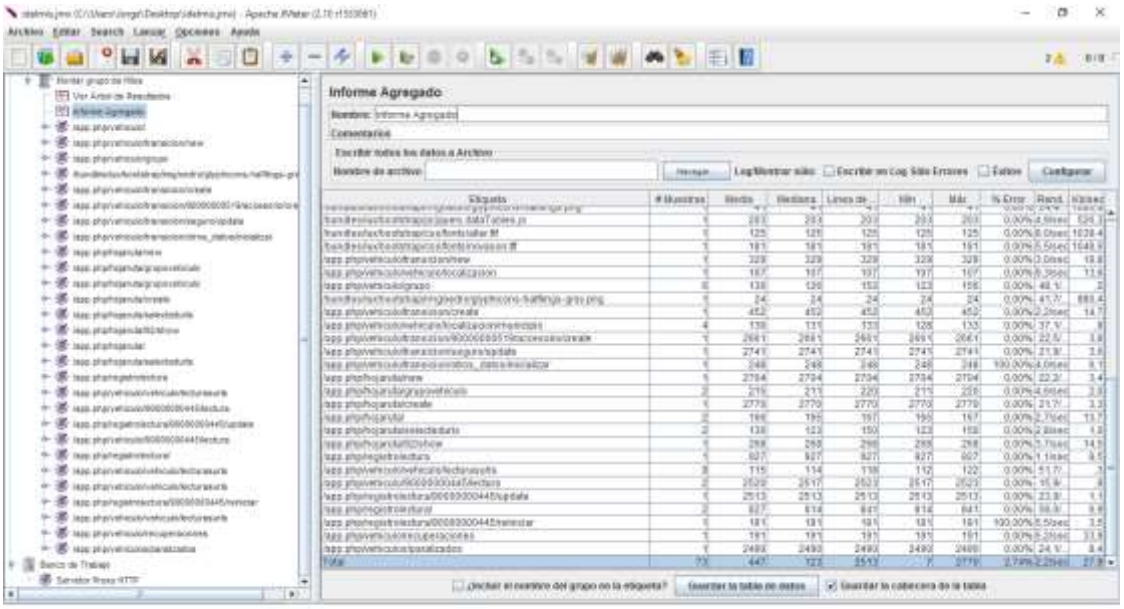
Zendos. 2010. Zendos tecnología . *Zendos tecnología .* [En línea] 2010. http://www.zendos.es.

Anexos

Anexo 1: Prueba de rendimiento con la herramienta JMeter al sistema Orbita implementado en Sauxe.



Anexo 2: Prueba de rendimiento con la herramienta JMeter al sistema Orbita implementado en Bosón.



Anexo 3: Encuesta

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. Para ello es necesario 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.

1. ¿Los módulos asociados al Sistema de Control de Flota y Mantenimiento Orbita se encuentran integrados? Sí___No____
2. Tiempo de respuesta de la página principal

   _de 0 a 12segs __de 13 a 27segs más de 27segs

3. Tiempo de respuesta de las demás páginas.

   _de 0 a 7 segs de 8 a 12 segs más de 13 segs

4. ¿Se cargan correctamente todos los componentes visuales del sistema?

   Sí___No____

   ¿En caso de NO, cuáles no se cargan? _____

   _____

5. ¿Se cargan correctamente los elementos de las listas? Sí___No____

   ¿En caso de NO, cuáles no se cargan? _____

   _____

6. ¿El sitio evita que los usuarios se registren de manera innecesaria?

   _____

   _____

7. ¿En qué criterio se basó para elegir EXTJs en su versión 3.4 y Sauxe en su versión 2.0 para el desarrollo del sistema?

   _____

8. ¿Todos los estilos se han creado en hojas CSS? Sí____No____

9. ¿El sistema cuenta con un mapa o buscador que facilite el acceso directo a los contenidos? Sí_____No____

10. ¿Existe una manera lógica de acceder a páginas relacionadas o a otras secciones? _____

11. ¿Cada pantalla empieza con un título que describe su contenido? Sí____No____

12. ¿El sitio provee una clara retroalimentación cuando una tarea ha sido completada exitosamente? Sí____No____

13. ¿Los campos en los formularios contienen ayudas, ejemplos o modelos de respuestas para demostrar el dato que se debe introducir? Sí_No____

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

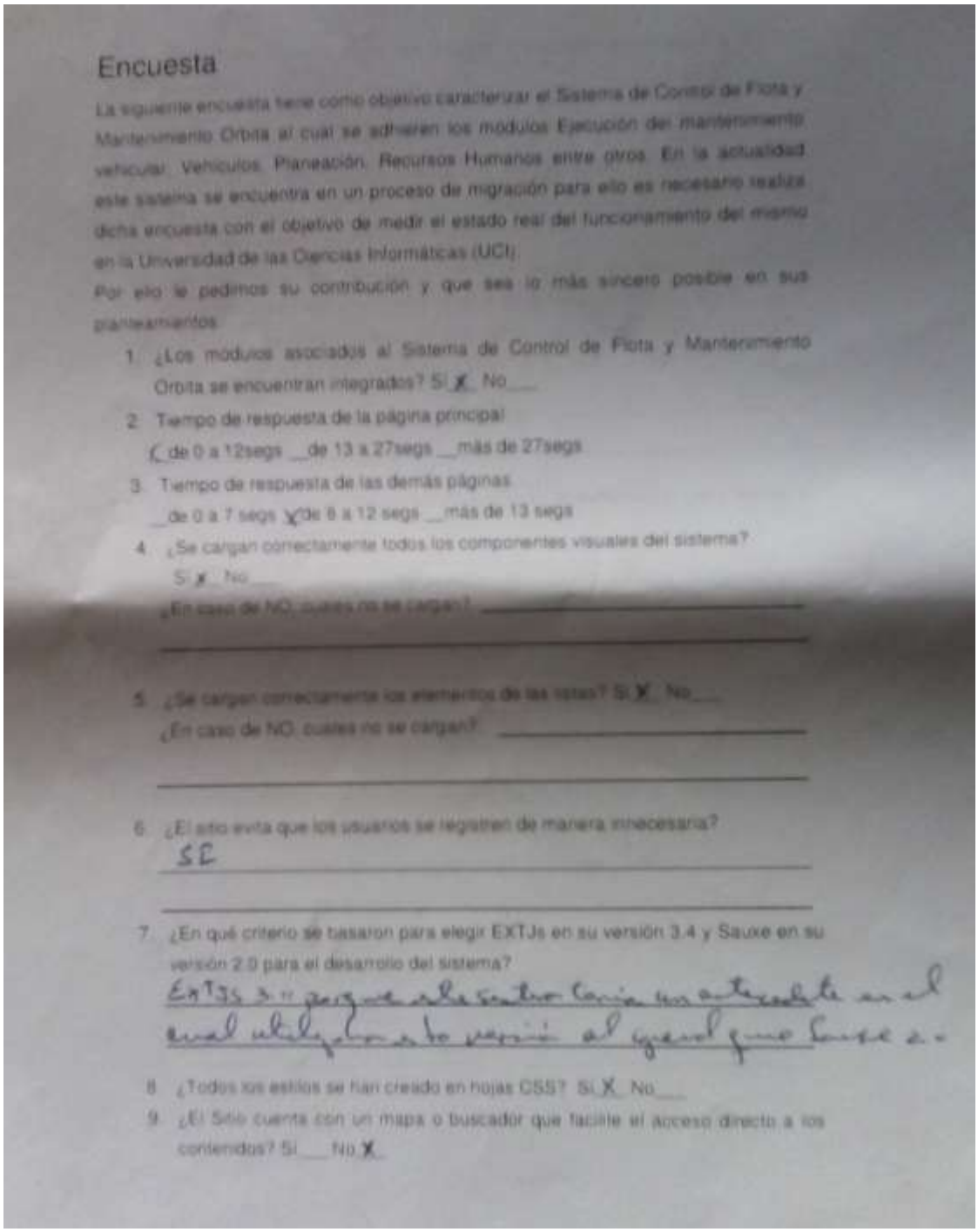
¿En caso de NO, dónde se encuentra ubicada? _____

15. El marco de trabaja Sauxe brindó suficiente documentación para realizar el sistema Orbita. Argumente.

16. Considera usted que a la hora de la implementación del sistema Orbita se realizaron malas prácticas de programación. ¿Cuáles?

17. Al sistema Orbita se le implementó un sistema de auditorías al código fuente o al cumplimiento de las definiciones arquitectónicas. Argumente.

18. Para el desarrollo del sistema Orbita se tuvo en cuenta el control y seguimiento a la implementación de las soluciones.



Encuesta

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 para ello es necesario realizar 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.

1. ¿Los módulos asociados al Sistema de Control de Flota y Mantenimiento Orbita se encuentran integrados? Si X No___
2. Tiempo de respuesta de la página principal
   X de 0 a 12segs __de 13 a 27segs __más de 27segs
3. Tiempo de respuesta de las demás páginas
   __de 0 a 7 segs X de 8 a 12 segs __más de 13 segs
4. ¿Se cargan correctamente todos los componentes visuales del sistema?
   Si X No___
   ¿En caso de NO, cuales no se cargan? _____
   _____
5. ¿Se cargan correctamente los elementos de las vistas? Si X No___
   ¿En caso de NO, cuales no se cargan? _____
   _____
6. ¿El sitio evita que los usuarios se registren de manera innecesaria?
   SE
   _____
7. ¿En qué criterio se basaron para elegir EXTJs en su versión 3.4 y Sauxe en su versión 2.0 para el desarrollo del sistema?
   EXTJS 3.4 porque es la última versión estable en el cual utilizamos esta versión al grado que sauxe 2.-
8. ¿Todos los estilos se han creado en hojas CSS? Si X No___
9. ¿El Sitio cuenta con un mapa o buscador que facilite el acceso directo a los contenidos? Si___ No X

10. ¿Existe una manera lógica de acceder a páginas relacionadas o a otras secciones? SI

11. ¿Cada pantalla empieza con un título que describe su contenido? Si X No___

12. ¿El sitio provee una clara retroalimentación cuando una tarea ha sido completada exitosamente? Si X No___

13. ¿Los campos en los formularios contienen ayudas, ejemplos o modelos de respuestas para demostrar el dato que se debe introducir? Si___ No X

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

¿En caso de NO, donde se encuentra ubicada?

15. El marco de trabajado Sauxe brindó suficiente documentación para realizar el sistema Orbita. Argumente

Si

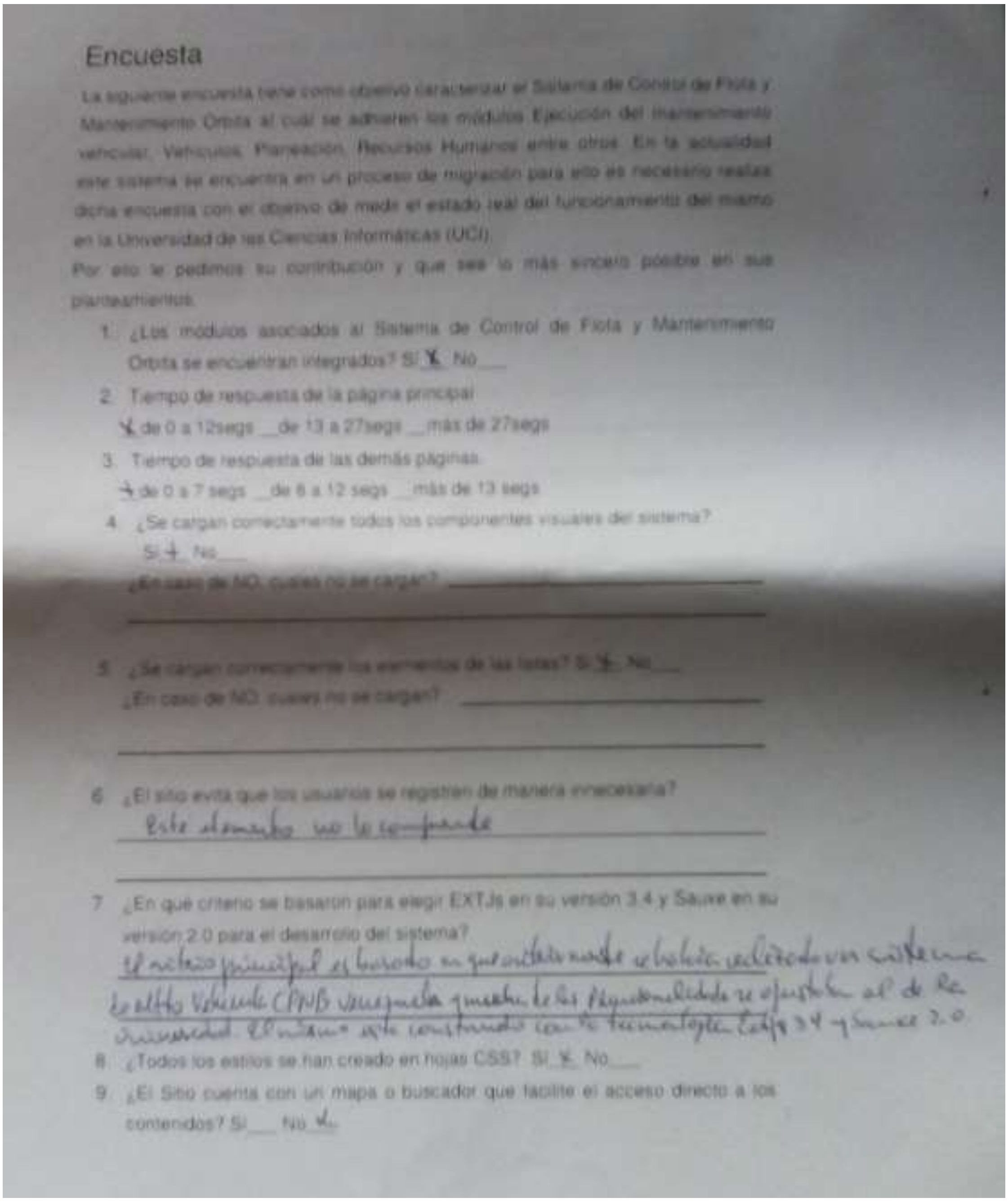
16. Considera usted que a la hora de la implementación del sistema Orbita se realizaron malas prácticas de programación. ¿Cuáles?

No se muy bien identificarlas, pero algunas

17. Al sistema Orbita se le implementó un sistemas de auditorías al código fuente o al cumplimiento de las definiciones arquitectónicas. Argumente

NO

18. Para el desarrollo del sistema Orbita se tuvo en cuenta el control y seguimiento a la implementación de las soluciones.



Encuesta

La siguiente encuesta tiene como objetivo caracterizar el Sistema de Control de Flota y Mantenimiento Orbita al cuál 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 para ello es necesario realizar 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.

1. ¿Los módulos asociados al Sistema de Control de Flota y Mantenimiento Orbita se encuentran integrados? Sí X No ___
2. Tiempo de respuesta de la página principal
   X de 0 a 12segs __de 13 a 27segs __más de 27segs
3. Tiempo de respuesta de las demás páginas
   X de 0 a 7 segs __de 8 a 12 segs __más de 13 segs
4. ¿Se cargan correctamente todos los componentes visuales del sistema?
   Sí X No ___
   ¿En caso de NO, cuales no se cargan? ______
5. ¿Se cargan correctamente los elementos de las listas? Sí X No ___
   ¿En caso de NO, cuales no se cargan? ______
6. ¿El sitio evita que los usuarios se registren de manera innecesaria?
   Este elemento no lo comprende
7. ¿En qué criterio se basaron para elegir EXTJs en su versión 3.4 y Sauxe en su versión 2.0 para el desarrollo del sistema?
   [illegible] ExtJs 3.4 y Sauxe 2.0
8. ¿Todos los estilos se han creado en hojas CSS? Sí X No ___
9. ¿El Sitio cuenta con un mapa o buscador que facilite el acceso directo a los contenidos? Sí ___ No X

10. ¿Existe una manera lógica de acceder a páginas relacionadas o a otras secciones? [illegible]

11. ¿Cada pantalla empieza con un título que describe su contenido? Si X No___

12. ¿El sitio provee una clara retroalimentación cuando una tarea ha sido completada exitosamente? Si X No___

13. ¿Los campos en los formularios contienen ayudas, ejemplos o modelos de respuestas para demostrar el dato que se debe introducir? Si___ No X

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

¿En caso de NO, donde se encuentra ubicada? ______________________

15. El marco de trabajo Sauxe brindó suficiente documentación para realizar el sistema Orbita. Argumente.

[illegible] DC.

16. Considera usted que a la hora de la implementación del sistema Orbita se realizaron malas prácticas de programación. ¿Cuáles?

Sí, [illegible] el acceso a BD [illegible] inyección SQL.

17. Al sistema Orbita se le implementó un sistemas de auditorías al código fuente o al cumplimiento de las definiciones arquitectónicas. Argumente.

[illegible]

18. Para el desarrollo del sistema Orbita se tuvo en cuenta el control y seguimiento a la implementación de las soluciones.

[illegible] seguimiento a la implementación [illegible]