FACULTAD 3

Migración del módulo Ejecución 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: Yudelyn Reyes Navarro

Tutor(es):
Ing.Yaniris Blanco Zamora
Ing.Ernesto Mató Roque

Co-Tutor(es):
Ing. Yorguy Antonio Batista Desdín
Ing. Jorge Yosmiel Jiménez Quintana

LA HABANA, JUNE DE 2016
"AÑO 58 DE LA REVOLUCIÓN"

Seamos realistas y hagamos lo imposible.

Che

DECLARACIÓN DE AUTORÍA

Declaro ser la autora de la presente tesis y reconozco a 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 _____.

_____

Yudelyn Reyes Navarro

Autora

_____ _____

Ing.Yaniris Blanco Zamora

Tutor

Ing.Ernesto Mató Roque

Tutor

Datos del Autor:

Nombre: Yudelyn Reyes Navarro

Correo electrónico: yrnavarro@estudiantes.uci.cu

Universidad de las Ciencia Informáticas, La Habana, Cuba

Datos de tutor:

Nombre: Ing. Yaniris Blanco Zamora

Correo electrónico: ybzamora@uci.cu

Universidad de las Ciencia Informáticas, La Habana, Cuba

Datos de tutor:

Nombre: Ing. Ernesto Mató Roque

Correo electrónico: emato@uci.cu

Universidad de las Ciencia Informáticas, La Habana, Cuba.

Datos de Co-Tutor:

Nombre: Ing. Yorguy Antonio Batista Desdín

Correo electrónico: yabatista@uci.cu

Universidad de las Ciencia Informáticas, La Habana, Cuba.

Datos de Co-Tutor:

Nombre: Ing. Jorgue Yosmiel Jiménez Quintana

Correo electrónico: jyquintana@uci.cu

Universidad de las Ciencia Informáticas, La Habana, Cuba

DEDICATORIA

Dedico el presente logro A:

A mi mamá Eliada Navarro Navarrro, por ser mi sostén y mi guía, por su amor y apoyo incondicional, por estar en todos los momentos de mi vida los buenos, los malos y los insoportables, por nunca tener un reproche para mí, por quererme de la forma que lo hace, por ser mía amiga, mi mejor amiga.

A mi papá Jose Luis Reyes Aguirres por aceptarme tal y como soy, por educarme y enseñarme aquellos valores que trascienden más allá de la escuela, por complacerme cada capricho, y por dejarme ser tu niña, por nunca quitarme de ese lugar aun cuando ya están los nietos. A mi papá por ser el amor de mi vida.

AGRADECIMIENTOS

Le agradezco a mis hermanos Alexey y Yunior, gracias por hacer mi vida más lindas y por los hermosos sobrinos que me dieron. Los amos.

A Lixey, Sebastián, Luis Angel gracias por permitirme ser su ejemplo a seguir y por el amor incondicional. A mis abuelas Digna y Esperanza, y a mis tías y tios,mis primos gracias por todo, los quiero.

A Arletys, Xiomara, Ada, Doriagne, Alexander, Noeldisy, Yoendris, Frank a todas aquellas personas que me quieren y siempre me han apoyado desde lejos.

Le agradezco a la persona que ha hecho posible la culminación de este trabajo, a ti por estar en cada paso que di desde el inicio, por enseñarme que la amistad es como el amor que cuando te corresponden llegas a volar, te agradezco porque has sido más que un amigo, has compartido las lágrimas conmigo, la comida en los tiempos de abundancia y de escases, has librados mis batallas, y sé que nunca podré compensarte por todo lo que has hecho por mi porque en el mundo existen muchos amigos pero muy pocos son TITO, te quiero.

Le Agradezco a Idelmis, Nairovis, Eime por hacer que cada día valiera la pena, que en cada derrota tuviera un bastón, por las criticas algunas buenas, casi todas malas, por las peleas, pero sobre todo le agradezco por ser amigas de las que llegan a tu vida arman un revuelo, se van, pero sabes que con ellas siempre puedes contar.

Le agradezco a mi amigo de alma mi Yorguy gracias por todo, por las comidas, los chiste y por demostrarme que siempre puedo contar contigo, el mejor regalo que te puedo dejar que me hagas es el de Willy. Le agradezco a Juan, a Aliuska, Dianelis y por favor si se me queda alguien no se pongan triste en la fiesta le pago con honores.

Le Agradezco a mis compañeros de clase, en especial a Carlos Andrés y a Yoel, por enseñarme la buena vida del estudiante, a Asprón, y al colectivo completo por cada momento vivido, y aun me mantengo ese es y será la mejor brigada que hay hasta ahora.

Le Agradezco a Dayan, José Luis, Richard, Yoslenys, José Carlos, por cada momento compartido y por sobre todas las cosas por ser ustedes cuando andan conmigo, gracias por dejarme entrar a su mundo y sentirme parte de él.

Le Agradezco a Roberto, Cueva, Camaguey, Edward, Lachi, Ever y a Yaser por colmar mi paciencia y por tener mucha conmigo, gracias por el apodo que me pusieron que ya saben que me encanta.

Le agradezco a todas las personas presente por estar aquí, aquellos que aportaron su granito de arena y a aquellos que aportaron una piedra, gracias por compartir este momento culminante de mi vida.

Le agradezco a Ailyn, Abilio, mis amores gracias por todo a ti cavillita por demostrarme que, aunque la distancia nos separa siempre habrá un cielo que nos una.

Le agradezco a Dariel por permitirme vivir el mejor momento de mi carrera universitaria al lado de él, gracias por hacerme reír y por permitir que esta etapa haya sido única y transcendente, por darme la nueva y hermosa familia que tengo y por sobre todas las cosas por quererme. Te quiero.

Por último a Mi padres por permitirme vivir.

RESUMEN

El presente trabajo describe la migración del módulo Ejecución a la arquitectura de referencia en php Bosón, para la dirección de Transporte de la UCI. Este módulo permite realizar las actividades requeridas en la dirección de transporte, dígase crear solicitudes de mantenimiento, emitir órdenes de trabajo, registrar recursos utilizados, entre otras. La migración del módulo en cuestión contribuye a corregir deficiencias presentadas por la solución que lo antecede, dentro de la que se destaca la actualización de las tecnologías de desarrollo. A partir del estudio del arte se deja plasmado un análisis de varias herramientas que se utilizan para realizar las pruebas de rendimiento. El desarrollo de software estuvo guiado por la fase de ejecución de la metodología de Desarrollo para la actividad productiva en la UCI que combina las buenas prácticas del nivel 2 del Modelo de Madurez de la Capacidad Integrado. Además se especifican la utilización de las herramientas, tecnologías y lenguajes de programación que se van a utilizar en la implementación del módulo Ejecución del Sistema de Control de Flota y Mantenimiento Orbita, teniendo en cuenta la arquitectura de referencia Bosón, donde se especifican las buenas prácticas para el desarrollo y se definen un conjunto de patrones y tecnologías a utilizar.

Palabras clave: Ejecución, mantenimiento, migración, rendimiento.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURA

ÍNDICE DE TABLA

INTRODUCCIÓN

Las nuevas tecnologías empleadas en la computación y el desarrollo de Internet, no sólo como instrumentos de difusión y entretenimiento sino como herramienta de negocio, han cambiado la fisonomía de la informática en la última década. Con el auge en paralelo de los entornos cliente/servidor y de las comunicaciones, se ha aproximado la tecnología telemática a los usuarios finales, de tal modo que casi no se puede imaginar un ambiente de negocio que no considere la informática como una herramienta básica para su desarrollo. Sin embargo, tras una fase de descubrimiento de las funcionalidades y del potencial de desarrollo de estos sistemas, se ha pasado a la lógica preocupación por aspectos cuantitativos y no funcionales que ya se planteaban en arquitecturas centralizadas. Así, por ejemplo, la seguridad, la disponibilidad, la eficiencia o el rendimiento son algunas de las características que normalmente se evalúan a la hora de explotar un sistema informático (Xavier Molero, 2015).

Desde hace algunos años el desarrollo de la informática y las nuevas tecnologías está en constante crecimiento, la utilización de las mismas proporciona facilidades en el control y desarrollo de sistemas, por lo cual en la actualidad muchas empresas y organismos se han motivado en el uso de estas en aras de optimizar los procesos que realizan en tiempo y costo de ejecución (DESOFT, 2015). Dentro de estos procesos se encuentran los relacionados con el mantenimiento de equipos e instalaciones, pues la informatización de estos proporciona un adecuado control en la ejecución del mantenimiento; permitiendo la prevención a tiempo de problemas funcionales. Aunque a los sistemas informáticos se les puede dar mantenimiento eso no evita que en algún momento los mismos tengan que migrar a otros más modernos, el término migración en el mundo de la informática, se basa en hacer que los datos y las aplicaciones existentes funcionen en una computadora, software o sistema operativo distinto (Ciolli, 2008).

Actualmente, el software libre se ha convertido en una premisa de la independencia tecnológica en Cuba, el mismo brinda un conjunto de libertades a los usuarios que permiten el estudio del funcionamiento del programa, además que este sea adaptable a sus necesidades (Stallman, 2015). Muchas instituciones cubanas, entre ellas la Universidad de las Ciencias Informáticas (UCI) han optado por el uso del software libre contribuyendo al desarrollo de sistemas informáticos de gran importancia para el país. Entre ellos se encuentran el Sistema para la Informatización de la Gestión de la Fiscalía (SIGEF), Sistema para Informatización de los Tribunales populares cubanos (SITPC), Sistema de Planificación de actividades (SIPAC), entre otros.

En la UCI en el Centro de Informatización de Entidades (CEIGE) se desarrolló el Sistema de Control de Flota y Mantenimiento Orbita, el cual gestiona los procesos de la ejecución del mantenimiento de los vehículos en una base de transporte, enfocándose en la aplicación del mantenimiento preventivo y correctivo, partiendo de la generación de una orden de trabajo[1] y registrando los recursos tanto humanos como materiales utilizados en la realización de estos mantenimientos, ya sean preventivos o correctivos.

La tecnología empleada para el desarrollo del sistema fue Sauxe en su versión 2.0, este marco de trabajo en la capa presentación utiliza la actualización de ExtJs 3.4 lo cual es una salvedad desarrollada en el proyecto, por necesidades particulares del cliente de dicho sistema. Aunque se empleó la actualización de ExtJs 3.4 el mismo se encuentra atado a una licencia de pago para su uso comercial (Sencha, 2016).

En la capa de negocio hace uso del marco de trabajo Zend Framework 1.2 que no permite la generación automática de CRUD[2] mediante líneas de comandos para la creación, mantenimiento de aplicaciones y el almacenamiento en la caché de las vistas. Dichas características deben tenerse en cuenta para un desarrollo ágil y un mayor rendimiento del sistema (Zend Technologies Ltd, 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 la independencia y es más costoso a la hora del cambio hacia otra tecnología. Otro inconveniente que presenta el uso de esta versión de Doctrine es que a partir del 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, 2009).

En la encuesta (Anexos 1) realizada a varios especialistas del sistema Orbita, se identificaron varios factores que influyen negativamente en el proceso de desarrollo del mismo, dando los siguientes resultados:

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.

[1] Una orden de trabajo es un documento que recoge los trabajos de mantenimiento que se le van a realizar a una o varias unidades policiales, los materiales que se consumieron en cada uno de los trabajos realizados y los recursos humanos.
[2] CRUD es el acrónimo a Create, Read, Update and Delete, lo mismo que Crear, Listar, Actualizar y Eliminar.

El 75% de los encuestados coinciden que el tiempo de respuesta de las interfaces relacionadas con las acciones adicionar solicitud, generar órdenes de trabajo, emitir órdenes de trabajo y asociar baterías 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 , en el libro Usability Engineering existen tres límites importantes en el tiempo de respuesta (Nielsen, 2010) :

- ✓ 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.
- ✓ 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 Ejecución. Para realizar 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 Ejecución para ejecutar determinadas funcionalidades con un usuario realizando peticiones de manera simultánea fue de 11997 milisegundos.
- ✓ La velocidad del sistema Orbita fue de 5,1 segundos en responder.
- ✓ El tiempo de ejecución promedio de una petición fue de 193 milisegundos.

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. Por lo cual el autor de la presente investigación define que el rendimiento será medido como el tiempo de respuesta de las funcionalidades del sistema.

A partir de la problemática antes planteada se define como **Problema a resolver:** ¿Cómo mejorar el rendimiento del módulo Ejecución del Sistema de Control de Flota y Mantenimiento Orbita?

Con el fin de resolver el problema identificado se propone como **objetivo general:** Migrar el módulo Ejecución del Sistema de Control de Flota y Mantenimiento Orbita a la arquitectura de referencia en PHP Bosón, para mejorar el rendimiento del sistema.

Se define como **objeto de estudio:** migración de tecnologías de desarrollo en sistemas de gestión y como **campo de acción:** migración de tecnologías de desarrollo en sistemas de mantenimiento vehicular.

Para darle cumplimiento al objetivo general se desglosó el mismo en los siguientes **objetivos específicos:**

OE1. Realizar un estudio del estado del arte para la elaboración del marco teórico alrededor del objeto de estudio.

OE2. Analizar la ingeniería de requisitos del módulo Ejecución para el sistema Orbita

OE3. Rediseñar la estructura de componentes y clases para el sistema Orbita aplicando los patrones de diseños, algoritmos y técnicas consideradas como necesarias en el estudio.

OE4. Implementar el módulo Ejecución para el sistema Orbita, siguiendo las técnicas de programación estudiadas en la tecnología de desarrollo seleccionada para la migración.

OE5. Validar funcionalmente la solución propuesta para el sistema Orbita.

La investigación parte de la siguiente **idea a defender:** Si se realiza la migración del módulo de ejecución del sistema Orbita a la arquitectura de referencia en PHP Bosón, se contribuye a mejorar su rendimiento.

Métodos de investigación:

Métodos Teóricos:

- ✓ **Analítico-sintético:** se utilizó durante el estudio y análisis de las bibliografías consultadas para fundamentar, asumir posiciones y determinar los elementos en relación con el objeto investigado, además de las características generales y las relaciones esenciales del proceso que se está investigando (Chagoya, 2008).
- ✓ **Histórico-lógico:** se utilizó para estudiar la teoría conocida hasta el momento y el análisis de la trayectoria completa del módulo Ejecución del sistema Orbita obteniendo las características que hacen necesaria la migración de dicho módulo (Chagoya, 2008).

Métodos empíricos:

- ✓ **Entrevistas:** se utilizó con el cliente mediante una conversación planificada con el fin de obtener información sobre las funcionalidades que se estarán migrando del módulo Ejecución del sistema Orbita (Chagoya, 2008).

- ✓ **Encuesta:** se utilizó con el objetivo de recolectar información relacionada con el sistema Orbita, y así obtener la opinión de los especialistas que intervinieron en su implementación referente a los problemas que incidieron en el desarrollo de dicho sistema y que afectaron al mismo (Chagoya, 2008).

El presente trabajo se encuentra estructurado en tres capítulos, resumidos de la siguiente forma:

- ✓ En el **Capítulo 1: Fundamentación Teórica** se realiza una investigación para esclarecer los conceptos más importantes. Se analizan varias herramientas que se utilizan para evaluar el rendimiento. Se especifica sobre la utilización de las herramientas, tecnologías y lenguajes de programación que se van a utilizar para la migración e implementación del módulo Ejecución del sistema Orbita sobre la arquitectura de referencia en PHP Bosón. Se determinan los patrones de diseños y arquitectónicos, además se plantea la metodología de desarrollo por la cual estará transitado el módulo de Ejecución del sistema Orbita.
- ✓ En el **Capítulo 2: Diseño e implementación** se realiza una descripción detallada de la migración propuesta reflejando los cambios introducidos en la arquitectura del proyecto haciendo uso de MVC[3], además se expondrán los requisitos funcionales y no funcionales que garantizarán el desarrollo de la solución al problema, también se dejan reflejados los productos de trabajo con los cuales cuenta la metodología que se está utilizando, ejemplo de estos son: los diagramas de diseño de clase web, el modelo de datos, las historias de usuario entre otros.
- ✓ En el **Capítulo 3: Validación y pruebas** se presentan los productos de trabajo asociados a las disciplinas de implementación de la solución, así como las pruebas utilizadas para validar la solución propuesta donde se puntualizan las no conformidades resultantes en cada iteración.

[3] El modelo–vista–controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

El presente capítulo aborda los principales conceptos asociados al dominio del problema y que están relacionados con el proceso ejecución del mantenimiento vehicular de la dirección de transporte de la UCI. Se realiza un estudio del estado del arte donde queda reflejado un análisis crítico de sistemas de gestión de flotas de vehículos existentes en Cuba y el mundo. Además, se describen las características de los lenguajes de programación, herramientas, tecnologías y metodología para el desarrollo de la solución propuesta.

1.2 Conceptos asociados a la investigación

1.2.1 Rendimiento

La RAE (Real Academia Española) define rendimiento como:

2. m. Proporción entre el producto o el resultado obtenido y los medios utilizados (Real Academia Española, 2016).

Len Bass, Paul Clements y Rick Kazman afirman que 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. Además, exponen que 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 (Bass, 1998).

Ian Gorton[4] define el rendimiento como la medida que indica la cantidad de trabajo que una aplicación debe realizar en un tiempo determinado, y/o el plazo de tiempo que debe cumplirse para ejecutar una operación correcta (Ian Gorton, 2003).

Partiendo de los conceptos antes planteados el rendimiento no es más la medida del t Ciempo utilizado para realizar una tarea. Para la presente investigación se propone como concepto a seguir el definido por el autor Ian Gorton

1.2.2 Migración

La RAE (Real Academia Española) define migración como:

3. f. Inform. Paso de los programas, archivos y datos de un sistema desde una determinada plataforma tecnológica a otra diferente (Real Academia Española, 2016).

[4] Es el Director de Informática en el campus de Seattle donde se graduó, tiene 25 años de experiencia trabajando en la industria del software, laboratorios académicos y gubernamentales, tanto en los Estados Unidos y Australia.

La migración de aplicaciones es el proceso de mover un programa de aplicación de un ambiente a otro. Los ejemplos incluyen la migración desde un servidor de empresa en las instalaciones al entorno de un proveedor de nube o de un entorno a otro (Rouse, 2014).

El concepto de migración de un sistema no está taxativamente definido. La migración evita el redesarrollo completo del sistema al usar todos los antecedentes disponibles (requerimientos, diseños, etc.) y siempre implica un cambio en el ambiente de operación. Por lo tanto, al hablarse de migración se está haciendo referencia a la necesidad de trasladar un sistema a una nueva plataforma manteniendo sus funcionalidades y provocando mínimo impacto en su operación (Ciolli, 2008).

La migración de un sistema de información tiene por finalidad su traslado a un nuevo ambiente operativo, conservando su funcionalidad y datos originales. En todos los casos se persigue posibilitar el mantenimiento y posterior adecuación a nuevos requerimientos (Ciolli, 2008).

Siempre que lleva a cabo un proyecto de modernización, una consolidación de proveedores o actividades de fusión y adquisición, la oferta de aplicaciones cambia sin cesar.Tanto si las aplicaciones son a medida como si se encuentran en un sistema mainframe heredado, en el entorno local o en el cloud, se implementa, actualiza, migra, retira o consolida la gama de aplicaciones (Informática, 2016).

Los proveedores de software les dan diferentes nombres y calificativos a las actualizaciones según las distintas versiones, los errores que son solucionados o las nuevas funcionalidades que presentan. Estos nombres pueden ser: actualizaciones importantes, de seguridad, de alta prioridad o recomendadas (Muñoz Tamayo, 2009).

A continuación, se listan las ventajas y desventajas de la migración de versiones.

Ventajas (Muñoz Tamayo, 2009):

- ✓ Es posible que ocurra la incorporación de nuevas y últimas tecnologías.
- ✓ Eliminación de errores presentes en las versiones anteriores.
- ✓ Desarrollo de nuevas funcionalidades que enriquecen el manejo de las herramientas.
- ✓ El soporte técnico actualizado sobre la versión actualizada es brindado por los proveedores de la herramienta.

Desventajas (Muñoz Tamayo, 2009):

- ✓ Algo muy importante que se tiene que tener en cuenta a la hora de actualizar es el precio que se tiene que pagar para mantener actualizado un sistema.
- ✓ No todas las versiones nuevas que aparecen son compatibles al 100% con sus versiones anteriores. En muchos casos se estrecha más el tiempo entre una versión y otra.

Teniendo en cuenta los conceptos antes planteados la migración se basa en el cambio de las tecnologías hacia otras con el objetivo de proveer un producto con mayor calidad, debido a que las mismas utilizan un conjunto de tecnologías más avanzadas. Por lo cual para la presente investigación se decide como cocepto a seguir el propuesto por la autora María Elena Ciolli.

1.2.3 Ejecución

La RAE (Real Academia Española) define ejecución como:
1.f. Acción y efecto de ejecutar (Real Academia Española, 2016).

En informática, ejecutar es la acción de iniciar la carga de un programa o de cualquier archivo ejecutable. En otras palabras, la ejecución es el proceso mediante el cual una computadora lleva a cabo las instrucciones de un programa informático. Se pueden ejecutar programas compilados (por ejemplo, en Windows, los .EXE) o programas interpretados (por ejemplo, los scripts).Ejecutar un programa implica que éste estará en estado de ejecución y, por ende, en memoria, hasta que se finalice (ALEGSA, 2010).

Tomando como punto de partida de los conceptos antes expuestos se puede afirmar que la ejecución no es más que la operación de acometer el inicio o carga de un programa, archivos o instrucciones. Por lo cual para la presente investigación se decide el concepto dado por ALEGSA: Diccionario de informática y tecnología.

1.2.4 Mantenimiento

La RAE (Real Academia Española) define mantenimiento como:
2. m. Conjunto de operaciones y cuidados necesarios para que instalaciones, edificios, industrias, etc., puedan seguir funcionando adecuadamente (Real Academia Española, 2016).

El mantenimiento es el encargado de manejar la evaluación de políticas relacionadas con frecuencias o ciclos, niveles de inventario mediante los conceptos de gerencia de riesgo e incertidumbre, planes de corto y mediano plazo, costeo de actividades, estrategias de contratación, planes de procura y recursos humanos, para asegurar los costos óptimos y la integridad de las instalaciones, máquinas y equipos (Cáceres, 2011).

La Asociación Francesa de Normalización AFNOR, define el mantenimiento como un conjunto de actividades destinadas a mantener o a restablecer un bien a un estado o a unas condiciones dadas de seguridad en el funcionamiento, para cumplir con una función requerida. Estas actividades suponen una combinación de prácticas técnicas, administrativas y de gestión (César Leonidas Padilla Valdez, 2012).

Mantenimiento Correctivo: Es el conjunto de actividades de reparación y sustitución de elementos deteriorados por repuestos que se realiza cuando aparece el fallo (Abella, 2008).

Mantenimiento Preventivo: Es el conjunto de actividades programadas de antemano, tales como inspecciones regulares, pruebas, reparaciones, etc., encaminadas a reducir la frecuencia y el impacto de los fallos de un sistema (Abella, 2008).

Partiendo de los conceptos antes estudiados se puede plantear que el mantenimiento no es más que la acción llevada a cabo para detectar las fallas y averías de las máquina e instalaciones previniendo el deterioro de los mismos. Por lo cual para la presente investigación se decide como cocepto a seguir el propuesto por la AFNOR.

1.3 Necesidad de la migración

La UCI es un pilar importante de la industria cubana de software, donde se desarrollan un número significativo de proyectos nacionales y de exportación, para ello la universidad tiene la intención de estandarizar el desarrollo en php por lo cual se implementó una arquitectura de referencia denominada Bosón, en la cual se empleó Symfony2 como marco de trabajo base debido a la posibilidad de ser extensible y a los numerosos esfuerzos de su comunidad por impulsar el uso de buenas prácticas de programación (PIÑERO, 2010).

En la arquitectura de referencia Bosón el patrón de empaquetado cuenta con tres niveles, dos de ellos responsabilidad de la arquitectura de software y un tercer nivel relacionado con las abstracciones propias del proceso de diseño (Abraham Calás Torres, 2016). A continuación se listas los mismos (Abraham Calás Torres, 2016):

- ✓ **Nivel Sistema**: nivel más alto del encapsulamiento. Agrupa a todos los componentes y al marco de trabajo que implemente la arquitectura.
- ✓ **Nivel Componente:** corresponde a la abstracción de los procesos concretos que responden a una solución.
- ✓ **Nivel Diseño**: corresponde a las abstracciones de diseño (clases, librerías y otros recursos). La estructura de cada componente está conformada básicamente por cinco capas: presentación, servicios, negocio, acceso a datos y datos.

La arquitectura utiliza en la capa de presentación el marco de trabajo JavaScript: AngularJS para agregar interactividad a HTML, JQuery para el manejo de objetos del navegador y Bootstrap para estilizar las aplicaciones. En la capa de servicios el único componente es la Interfaz de Servicios RESTful, que tiene como objetivo comunicarse bidireccionalmente con los recursos definidos en la capa de negocio. También es papel de este componente proveer

un formato estandarizado (estilo REST) para brindar recursos a través del protocolo HTTP. Por otra parte, en la capa de acceso a datos incluye Doctrine 2, el mapeador de objetos-relacional seleccionado por Symfony2, su objetivo es persistir información en bases de datos relacionales, o extraer los datos en forma de objetos (Abraham Calás Torres, 2016).

Dentro de los requerimientos planteados por el cliente el sistema debe contar con el servicio de autenticación haciendo uso del protocolo LDAP[5], el mismo se utiliza en la UCI permitiendo autenticarse a través del usuario y contraseña. Otros de los requerimientos planteados es que el sistema a migrar debe poseer una interfaz única que estandarice visualmente la aplicación y que siga la línea temática de los productos que se realizan en la UCI, además que debe mantener la base del diseño haciendo uso de XEDRU. Con el objetivo de lograr homogeneidad en el código el cliente decidió desarrollar el Sistema Orbita utilizando la arquitectura de referencia Bosón, debido a la gama de beneficios que brinda la misma y para realizar una mejor integración entre los módulos se decidió utilizar dicha arquitectura, donde se especifican las buenas prácticas para el desarrollo, así como la definición de patrones y tecnologías a utilizar para la implementación del módulo Ejecución.

La migración del sistema Orbita hacia una nueva tecnología no solo plantea la necesidad de la creación de un producto que provea un mayor rendimiento, también se espera que el mismo sea capaz de proveer mejoras en las funcionalidades con las que cuenta, además que la modernización de este brindará una mejor interfaz visual a los usuarios. En la capa de presentación actualmente el sistema cuenta con ExtJs en su versión 3.4, este marco de trabajo se encuentra atado a una licencia lo que significa que para comercializar un software hay que pagar su uso. Teniendo en cuenta que el software libre se ha convertido en una premisa de la independencia tecnológica por la cual aboga Cuba, la UCI se suma en el uso de estas por lo que se decide emplear tecnologías que no presenten privatización en su licencia, por lo cual se opta por HTML5, CCS3, Bootstrap, jQuery. En la capa de negocio se empleó en el desarrollo del antiguo sistema el marco de trabajo Zend Framework 1.2 en la actualidad el mismo cuenta con versiones superiores (Zend Technologies Ltd, 2016). Partiendo de la deficiencia que se presentan en el proceso de desarrollo, por problemas de organización en la documentación existente en Sauxe se aboga por el uso de Symfony2 que fue diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web además que proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. También automatiza las tareas más comunes,

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

permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación (LibrosWeb, 2015).

Symfony2 trae incluido la versión más actualizada del ORM[6] Doctrine. Doctrine 2 implementa el diseño Mapa de Datos proponiendo que los objetos de negocio sean POCO[7], o sea que no tengan nada de código que los acople a una tecnología de persistencia, ya que no es su función manejarla. Los objetos de negocio solo deben preocuparse de cumplir con el modelo del dominio del diagrama de clases y con las asociaciones pertinentes. La principal ventaja de este ORM es el rendimiento en ejecución y en la forma tan concisa en la que se pueden escribir consultas muy complejas. También cuenta con una amplia comunidad y colaboradores alrededor de todo el mundo, es uno de los marcos de persistencias más usados y con mayor prestigio a escala mundial (Doctrine Team, 2010).

1.4 Módulo ejecución del mantenimiento

El módulo Ejecución del mantenimiento vehicular del Sistema Orbita de la dirección de transporte de la UCI comienza con la creación de la solicitud de servicio. La misma se realiza para asignarle a un vehículo la fecha determinada para darle mantenimiento, después que esta solicitud se conforma comienza el proceso de generación de órdenes de trabajo producto de un mantenimiento preventivo planificado o correctivo. A través de este proceso es donde se emiten las órdenes de trabajo, se establece el umbral de mantenimiento y se programa el mantenimiento que se le realizará a un vehículo determinado. Antes de generarse una orden de trabajo debe realizarse una inspección técnica con el objetivo de determinar trabajos de mantenimiento preventivo que puedan ser detectados (Hernández, 2012).

Luego de la inspección técnica se realizan las órdenes de trabajo donde se registran los recursos utilizados en la realización de un mantenimiento determinado, dígase repuestos, herramientas y recursos humanos, también en las órdenes de trabajo se les puede realizar el cambio de batería a un vehículo determinado. En caso de que las órdenes de trabajo se generen para mantenimientos correctivos se registran las causas y tipos de fallas asociados, en aras de realizar análisis que permitan tomar decisiones respecto al mantenimiento preventivo en la dirección de transporte de la UCI (Hernández, 2012).

1.5 Rendimiento: Herramientas para las pruebas de rendimiento.

El rendimiento se mide por la velocidad de procesamiento, el tiempo de respuesta, consumo de recursos, rendimiento efectivo total y eficacia, por lo cual es importante ir actualizando los sistemas a la última versión, para obtener un mejor rendimiento y no tener que lamentar

[6] Del inglés Object-Relational mapping (Mapeo de objeto-relacional)
[7] Componentes Portables (en inglés Portable Components).

problemas indeseados, además es necesario tener en cuenta los tipos, métodos y herramientas de pruebas debido a que la pruebas requieren que se descarten ideas preconcebidas sobre la “calidad o corrección” del software desarrollado (ISTQB, 2014).

Para mejorar la eficiencia de los productos liberados en la UCI se utilizan 3 tipos de pruebas (Jiménez Quintana, y otros, 2015):

- ✓ Rendimiento
- ✓ Carga
- ✓ Estrés

Estas fueron definidas a sus ves por el Comité Internacional de Calificación en Pruebas de Software[8] (ISQTB) que es por donde se rige la universidad para realizar dichas pruebas.

Las pruebas de rendimiento están enfocadas a monitorear el tiempo en flujo de ejecución, acceso a datos, en llamada a funciones y sistema para identificar y direccionar los cuellos de botellas y los procesos ineficientes. Un elemento que no puede faltar en las pruebas de rendimiento es una herramienta que multiplique la actividad de los usuarios, es decir, que sea capaz de aumentar la carga de trabajo ejecutando una funcionalidad simultáneamente desde muchas instancias (ISTQB, 2014).

Para realizar la prueba de rendimiento se realiza un análisis de un subconjunto del total de herramientas existentes, teniendo en cuenta que las mismas cumplan con un foco específico, y se completan los siguientes datos para cada una de ellas: nombre, descripción y Localizador de Recursos Uniforme[9]. Las herramientas seleccionadas para el estudio cumplen con la característica de ser libres o de código abierto:

Pylot: Es una herramienta de código abierto para realizar pruebas de rendimiento y escalabilidad de servicios web. Ejecuta pruebas de carga mediante el protocolo HTTP [10], lo cual es muy útil para planear la capacidad, realizar una evaluación comparativa y análisis del sistema. Genera concurrencia de solicitudes, verifica las respuestas del servidor y produce reportes teniendo en cuenta métricas. Las pruebas pueden ser ejecutadas y monitoreadas desde una interfaz gráfica o por consola**Fuente especificada no válida.**.

JMeter: Es una aplicación de escritorio desarrollada en Java, diseñada para medir el rendimiento y el comportamiento de los sistemas ante las pruebas de sobrecarga. Se puede utilizar para probar el rendimiento tanto de los recursos estáticos como dinámicos (archivos, Servlets, scripts de Perl, Java Objects, bases de datos y consultas, servidores FTP, entre otros). También, para simular una sobrecarga en un servidor, una red o un objeto, para poner

[8]Es una organización dedicada a la calificación y certificación de profesionales y empresas en el área de Pruebas e Software.

[9] Uniform Resource Locator, en inglés, abreviado URL.

[10] Protocolo de Transferencia de Hipertexto común del inglés Hypertext Transfer Protocol.

a prueba su resistencia o para analizar el rendimiento global para diferentes tipos de carga. Además, permite hacer un análisis gráfico de rendimiento o probar el servidor/script/comportamiento del objeto con sobrecargas concurrentes**Fuente especificada no válida.**.

OpenSTA: Es un conjunto de herramientas para ejecutar pruebas de sobrecarga y medir el rendimiento de aplicaciones sobre plataformas Windows mediante los protocolos HTTP y el HTTPS [11]. Los resultados y estadísticas generados durante las pruebas se pueden recopilar mediante una variedad de mecanismos automatizados o controlados por el usuario. Una vez completadas las pruebas, las trazas pueden ser revisadas, graficadas, filtradas y exportadas para que puedan ser utilizadas por un sistema de generación de reportes más sofisticado **Fuente especificada no válida.**.

Silk Performer: Permite crear pruebas de carga y estrés potentes y realistas para los usuarios en una variedad de entornos de aplicaciones sobre plataformas Windows mediante los protocolos HTTP. Esta solución líder de pruebas de carga simula picos de carga de cualquier tamaño procedentes desde distintas ubicaciones geográficas sin que tenga que invertir en instalación y hardware de pruebas de estrés y carga. La detección de problemas al principio del ciclo de lanzamiento permite solucionar los problemas antes de que lleguen a los usuarios finales, lo que reduce el tiempo de ciclo del proyecto y los costes de desarrollo (Corporation, Borland Software, 2015). Ver tabla 1

[11] Protocolo Seguro de Transferencia de Hipertexto común del inglés Hypertext Transfer Protocol Secure.

Tabla 1: Datos generales. Fuente (elaboración propia, 2016)

Herramienta	Inicio del Proyecto	Licencia	Plataforma	Guía de instalación	Manual de usuario
Pylot	Octubre 2007	-	Windows, GNU-Linux/Unix	Sí	Sí
JMeter	Diciembre 1998	Apache License (Versión 2.0) es compatible con GPL	Independiente	No requiere	Sí
OpenSTA	Febrero 2001	GNU GPL	Windows	Sí	Sí
Skill Performer	-	Licencia propietaria	Windows	Sí	Sí

Se completa la siguiente tabla de acuerdo con los criterios que se consideran relevantes para realizar la selección, teniendo en cuenta la información proporcionada por el sitio oficial de la herramienta u otros sitios reconocidos.

Del conjunto de herramientas propuestas en el estudio se descarta Pylot pues es un proyecto relativamente joven comparado con los otros relacionados en el estudio. OpenSTA, a diferencia de Pylot, es un proyecto de varios años de experiencia; no obstante, solo es posible utilizarlo en estaciones con Windows, lo cual no es compatible con el ambiente donde se ubicará la aplicación a probar, por otra parte, Skill Performer se encuentara atado a una licencia propietaria por lo cual se descarta. A partir de los elementos planteados se decide utilizar JMeter en su versión 2.10. La herramienta se ejecuta independiente a la plataforma de hardware utilizada, ya que es un proyecto 100% Java. Cuenta con un manual de usuario disponible que detalla cada una de sus funcionalidades, con frecuencia se publican nuevas versiones y tiene un buen grado de retroalimentación pues los errores reportados son resueltos en cortos períodos de tiempo.

1.6 Metodología, lenguajes y herramientas para el desarrollo.

1.6.1 Metodología de desarrollo para la Actividad Productiva en la UCI (AUP[12]-UCI).

A raíz del crecimiento tecnológico se hace necesario el desarrollo de software dentro de la actividad productiva de la UCI. La elección de la metodología es un proceso minucioso ya que guía el proceso de desarrollo para alcanzar la satisfacción del cliente y del equipo de desarrollo, además que de ello depende la calidad del desarrollo y del producto final. La UCI se caracteriza por el uso de diferentes metodologías de desarrollo entre robustas y ágiles entre las que se encuentran XP, OPEN UP, BPM, SCRUMP, entre otras. A pesar de la variedad de metodologías usadas, se ha comprobado que muy pocos proyectos la aplican en su totalidad (Sánchez, 2014).

Al no existir una metodología de software universal, ya que todas metodologías deben ser adaptadas a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Por lo cual se aplica en los proyectos productivos de la universidad la metodología de desarrollo para la Actividad Productiva en la UCI y de la misma parte el desarrollo del módulo Ejecución del Sistema Orbita (Sánchez, 2014).

La metodología de desarrollo para la Actividad Productiva en la UCI tiene 3 fases (Sánchez, 2014):

- ✓ **Inicio:** 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**: Se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. La misma cuenta con 7 disciplinas de las cuales solo se estarán desarrollando las siguientes: Requisitos, Análisis y Diseño, Implementación, Pruebas Internas y Pruebas de Aceptación.
- ✓ **Cierre:** Se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Por las características del trabajo de diploma la fase de inicio no se realiza porque como explica el objetivo de la fase en esta etapa se lleva a cabo la planeación del proyecto, puesto que lo que se realiza es un componente para un proyecto que ya se abrió o en este caso migrar un producto que ya se hizo, entonces no hay que recurrir a la misma, además tampoco

[12] Proceso Unificado Ágil del común en inglés Agile Unified Process.

se estará realizando la fase cierre puesto que del sistema Orbita sólo se está desarrollando un módulo y no el sistema completo como tal, por lo cual no se puede realizar el cierre del mismo.

1.6.2 Arquitectura de referencia Bosón

La arquitectura de referencia es un diagrama que permite separar los distintos componentes de una solución del datawarehousing[13] e integrar en una clasificación común los distintos tipos de información necesarios para construir un datawarehousing (Donadello, 2013).

El desarrollo de una arquitectura de referencia haciendo uso de las tecnologías libres, garantiza que todas las soluciones que se desarrollen sobre PHP cumplan con la versatilidad necesaria para la creación de sistemas de diferentes dominios. Al utilizar como marco base a Symfony2 se garantiza la presencia de una comunidad mundial brindando soporte y actualizaciones a diario, permitiendo que los equipos a desarrollar se centren en la implementación lógica de su negocio abstrayéndose de elementos con la gestión de la caché, el manejo de estructuras de datos, el registro de trazas, la seguridad de los sistemas, la gestión de excepciones, entre otros elementos (Amat, 2015). El marco de trabajo Bosón materializa los requisitos comunes de las arquitecturas base para sistemas de gestión web a partir del desarrollo de los siguientes componentes (Torres, 2015):

Componente Caché: Permite a las aplicaciones que se están desarrollando la gestión de la caché, permitiendo 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 (Torres, 2015).

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 (Torres, 2015).

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

[13] Es una colección de datos orientada a un determinado ámbito, integrado, no volátil y variable en el tiempo, que ayuda a la toma de decisiones en la entidad en la que se utiliza.

comportamiento, el mecanismo extensible de creación de nuevos tipos y la codificación (Torres, 2015).

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 EAV para gestionar las estructuras lo cual disminuye la creación de grandes volúmenes de tablas destinadas a estos temas (Torres, 2015).

Componente Trazas: Detecta y registra las trazas generadas por un sistema a partir de la captura de eventos por lo cual resulta útil para las auditorias. Permite registrar trazas de acción, rendimiento, acceso a datos y excepciones ocurridas (Torres, 2015).

Componente Aspectos: 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 (Torres, 2015).

Componente Integración: 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 manejo de códigos de errores (Torres, 2015).

Componente Seguridad: Provee un mecanismo de autenticación haciendo uso del estándar abierto SAML[14] así como un mecanismo de autorización mediante la extensión del patrón RBAC[15]. Permite además la autenticación a partir de varias fuentes como base de datos, servidores LDAP, Facebook, Google, Linkedln, entre otras (Torres, 2015).

Componente IUX: 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 (Torres, 2015).

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

[14] Aserción de seguridad Lenguaje de marcado común del inglés Security Assertion Markup Language.

[15] Control de Acceso con base en roles.

HTML5, CS3, DHTML, JS, y mecanismos de comunicación asíncrona con el servidor como Ajax y Websockets (Torres, 2015).

1.6.3 Lenguajes de desarrollo

PHP5

Para la selección del lenguaje de programación del módulo se tuvo en cuenta el framework Symfony2, utilizado en el desarrollo del módulo Ejecución, popular en le realización de aplicaciones PHP[16]. Se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta el día de hoy. A continuación, se muestran algunas ventajas por la cual fue seleccionado este lenguaje para la realización del módulo de ejecución del Sistema Orbita a la arquitectura de referencia en PHP Bosón.

Ventajas (Andi Gutmans, 2004) :

- ✓ Es un lenguaje multiplataforma.
- ✓ Completamente 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 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 del módulo Ejecución 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 página oficial (Sitio 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.
- ✓ Biblioteca nativa de funciones sumamente amplia e incluida.

Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar (muchos otros lenguajes tampoco lo hacen), aun estando dirigido a alguna en particular, el programador puede aplicar en su trabajo cualquier técnica de programación y/o desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño MVC, que permiten separar el tratamiento y acceso a los Datos, la Lógica de control y la Interfaz de usuario en tres componentes independientes (Andi Gutmans, 2004).

JavaScript 1.8

[16] Procesador de hipertexto

Es un lenguaje de programación ligero y orientado a objetos. El intérprete del lenguaje se encuentra en los navegadores, para interpretar los códigos (scripts) escritos en las páginas. Estos códigos pueden hacer que exista interactividad dinámica entre el usuario y la página, controlar el navegador o crear páginas completas HTML sin que se tenga que ir nuevamente al servidor. JavaScript brinda facilidades a los usuarios tales como (Álvarez, 2012):

- ✓ Permite la verificación de los datos introducidos por el usuario antes de enviar el formulario al servidor.
- ✓ Permite el manejo de pequeñas cantidades de información al igual que en una base de datos.
- ✓ Permite el preprocesado de información antes de enviarla al servidor.

Este lenguaje por sus características y posibilidades permite un desarrollo completo de cualquier aplicación web por compleja que sea, facilita el manejo de información en el navegador sin necesidad de realizar llamadas redundantes a una base de datos, por lo cual se va a utilizar para el desarrollo del módulo Ejecución del Sistema Orbita posibilitando una mayor rapidez del mismo (Álvarez, 2012).

HTML5

HTML5 es un lenguaje markup (de hecho, las siglas de HTML significan Hyper Text Markup Language) usado para estructurar y presentar el contenido para la web. Es uno de los aspectos fundamentales para el funcionamiento de los sitios, pero no es el primero. Es de hecho la quinta revisión del estándar que fue creado en 1990. A fines del año pasado (2015), la W3C[17] la recomendó para transformarse en el estándar a ser usado en el desarrollo de proyectos venideros. Por así decirlo, HTML5 está relacionado también con la entrada en decadencia del viejo estándar HTML 4, que se combinaba con otros lenguajes para producir los sitios que se pueden ver hoy en día (BarbaraPVN, 2013).

Con el uso de HTML5, se puede reducir la dependencia de los plug-ins que se deben instalar para poder ver una determinada web. Caso emblemático, es el de Adobe Flash, que se ve claramente perjudicado por la instauración de este estándar, con HTML5 se amplía el horizonte del desarrollo de aplicaciones que pueden ser usadas en una multiplicidad de dispositivos, además los usuarios pueden acceder a sitios web de manera offline, sin estar conectados a internet. Se suma también la funcionalidad de drag and drop[18], y también la

[17] Consorcio Mundial de la red común del inglés World Wide Web Consortium, es un consorcio internacional que genera recomendaciones y estándares que aseguran el crecimiento de la World Wide Web a largo plazo.

[18] "Desplazar y liberar" o "arrastrar y soltar" (*drag and drop*) es una expresión informática que se refiere a la acción de mover, con el cursor del ratón, los objetos de una ventana a otra o entre partes de una misma ventana.

edición online de documentos ampliamente popularizada por Google Docs (BarbaraPVN, 2013).

CSS3

CSS son las siglas del inglés Cascading Style Sheets (Hojas de Estilo en Cascada) básicamente consiste en la información que define como va a ser la presentación de una web. Cuando se refiere a presentación, se refiere colores, efectos, tipos de letra que escogemos, de tal manera que se independiza del HTML que es el lenguaje donde se estructura toda la información que se manda al ordenador para que el navegador presente la bonita página web. CSS3 es la versión 3, donde se definen las características de este lenguaje, en esta versión han añadido muchas cosas y mejorado otra. Ejemplo de esto es la tecnología desarrollada para separar la presentación de la estructura HTML. Esta tecnología aplica reglas de estilo a los elementos HTML, quedando de esa manera separada de la estructura HTML. Poco a poco este lenguaje se ha ido haciendo más importante entre los diseñadores gracias a toda la facilidad de uso, y los resultados que son muy flexibles (Delgado, 2015).

1.6.4 Herramientas de desarrollo

1.6.4.1 Entorno de desarrollo integrado

Un entorno de desarrollo integrado, llamado también IDE (sigla en inglés de integrated development environment), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios (Sánchez, 2014).

NetBeans v8.0

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso, además es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal del proyecto. Algunas de las características de la aplicación son (NetBeans, 2016):

- ✓ Gestión de la interfaz de usuario (menús y barras de herramientas).
- ✓ Gestión de configuración de usuario.
- ✓ Gestión de almacenamiento (guardar o cargar algún tipo de dato).
- ✓ Gestión de ventana.
- ✓ Librería visual de Netbeans.

Apache 2.2

Es una tecnología de código de fuente abierta, puede ser usado en varios sistemas operativos, lo que lo hace prácticamente universal. Es un servidor altamente configurable de diseño modular, trabaja con gran cantidad de lenguajes ejemplos de estos: Perl, PHP y otros lenguajes de script. Apache es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa (Ciberaula, 2010).

Características (Ciberaula, 2010):

- ✓ Multiplataforma.
- ✓ Apache es una tecnología gratuita de código de fuente abierta.
- ✓ Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que se instalen cuando lo necesiten.
- ✓ Apache trabaja con Java y páginas JSP[19] . Teniendo todo el soporte que se necesita para tener páginas dinámicas.
- ✓ Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor.
- ✓ Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.

1.6.4.2 Servidor de base de datos

Un SGBD o Sistema Gestor de Base de Datos[20] es una colección de datos relacionados entre sí, estructurados y organizados, un conjunto de programas que acceden y gestionan esos datos. La colección de esos datos se denomina Base de Datos (BD). Los sistemas gestores de base de datos están diseñados para gestionar grandes bloques de información, que implica tanto la definición de estructuras para el almacenamiento como de mecanismos para la gestión de la información. El SGBD es una aplicación que permite a los usuarios definir, crear y mantener la DB y proporciona un acceso controlado a la misma (Alegsa, 2010).

PostgreSQL 9.3

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad

[19] Páginas de Servidor Java del inglés Java Server Programming.

[20] DBMS, siglas del inglés Data Base Management System.

transaccional. A pesar de esto, PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos. PostgreSQL está considerado como una de las bases de datos de código abierto más avanzadas del mundo. Proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle. PostgreSQL es un servidor de base de datos relacional libre, que se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y uniones de gran tamaño. Fue diseñado para ambientes de alto volumen y tiene mejor soporte para vistas, procedimientos almacenados en el servidor, transacciones y almacenamiento de objetos de gran tamaño (PostgreSQL, 2011).

1.6.4.3 Navegador

Mozilla Firefox 40+

Firefox es un navegador de Internet con interfaz gráfica de usuario desarrollado por la Corporación Mozilla y un gran número de voluntarios. Se empleará por sus ventajas (Mozilla, 2016):

- ✓ Es de código libre y multiplataforma.
- ✓ Permite insertar complementos desarrollados por terceros.
- ✓ Posee función de autocompletamiento en la barra de navegación, lo cual facilita la navegación por internet.
- ✓ Guarda la contraseña para entrar a un sitio determinado.
- ✓ La restauración de las ventanas y pestañas que se estuvieran usando antes de cerrar el navegador.
- ✓ Seguridad avanzada en la navegación pues permite la identificación de algún sitio web de forma instantánea, posibilita navegar de forma privada y sin registro de lo accedido en internet.
- ✓ Establece conexiones seguras a sitios web.
- ✓ La integración al antivirus que posea la computadora, soporte para varios idiomas y alta configurabilidad.
- ✓ Protección en cuanto a malware se refiera, puesto que, en caso de acceso a un sitio sospechoso, el navegador le notificará y le dará las razones.

1.6.4.4 Frameworks

Un framework simplifica el desarrollo de las aplicaciones, ya que automatiza muchos de los patrones utilizados para resolver las tareas comunes. Además, un framework proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas (Gerner, 2006).

Symfony 2.7

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Symfony separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web (LibrosWeb, 2015).

Symfony está desarrollado completamente con PHP y es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows. A continuación, se muestran algunas de sus características (LibrosWeb, 2015):

Symfony se diseñó para que se ajustara a los siguientes requisitos:

- ✓ Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- ✓ Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo**.**
- ✓ Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo**.**
- ✓ Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

jQuery

jQuery es un framework para el lenguaje JavaScript, que permiten programar sin preocuparse por los diferentes navegadores, ya que funciona igual en todas las plataformas más habituales. Este framework, ofrece una infraestructura con la que se tendrá mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Por ejemplo, con jQuery se obtiene ayuda en la creación de interfaces de usuario, efectos dinámicos, aplicaciones que hacen uso de Ajax, etc. Además, todas estas ventajas con jQuery se obtienen de manera gratuita, ya que el framework tiene licencia para uso en cualquier tipo de plataforma, personal o comercial. Para la realización de un proyecto jQuery es rico en sus funciones, ya que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol Document Object Model (DOM) y manejar eventos (Álvarez, 2012).

Bootstrap

Bootstrap, es un framework originalmente creado por Twitter, permite crear interfaces web con CSS y JavaScript, cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. Es decir, el sitio web se adapta automáticamente al tamaño de una PC, una Tablet u otro dispositivo. Esta técnica de diseño y desarrollo se conoce como Responsive Design o Diseño Adaptativo. Este Framework permite a los usuarios abstraerse de tener que preocuparse por las media queries[21] y los porcentajes en los CSS para hacer una web Responsive, facilitando la programación del sitio, y el mismo se basa en la simplicidad de sus interfaces (Mark Otto, 2015).

AngularJS

AngularJS es un framework de JavaScript de lado cliente, sirve para agregar interactividad a HTML, contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo**.** El mismo fue desarrollado por la empresa Google. Angular promueve y usa patrones de diseño de software. En concreto implementa lo que se llama MVC. Básicamente estos patrones marcan la separación del código de los programas dependiendo de su responsabilidad. Eso permite repartir la lógica de la aplicación por capas, lo que resulta muy adecuado para aplicaciones de negocio y para las aplicaciones SPA[22] (Single Page Aplication) (Álvarez, 2012).

1.6.4.5 Lenguaje de Modelado UML[23]

Es un lenguaje para la especificación, visualización, construcción y documentación de los productos de trabajo de un proceso de sistema intensivo. Es utilizado para la modelación de sistemas con características orientadas a objetos. Facilita a los integrantes de un equipo multidisciplinario participar e intercomunicarse fácilmente, promueve la reutilización e incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos (Cornejo).

Visual Paradimg 8.0

Visual Paradigm es una herramienta UML profesional, soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una construcción más rápida de

[21] Módulo CSS3 que permite adaptar la representación del contenido a características del dispositivo como la resolución de pantalla o la presencia de características de accesibilidad como el braille.

[22] Aplicación de página única es una aplicación web o es un sitio web que cabe en una sola página con el propósito de dar una experiencia más fluida a los usuarios como una aplicación de escritorio.

[23] Lenguaje Unificado de Modelado

aplicaciones de calidad, mejores y aún menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML (Cornejo).

Ventajas (Cornejo) :

- ✓ Tiene disponible distintas versiones: Enterprise, Professional, Standard, Modeler, Personal y Community (que es gratuita).
- ✓ Se pueden dibujar todos los tipos de diagramas de clase, generar el código de diagramas y generar la documentación.
- ✓ Alta interoperabilidad: Los usuarios y proveedores de tecnología pueden integrar con Visual Paradigm modelos en sus soluciones con un mínimo esfuerzo.

1.6.4.6 Sistema Control de Versiones

Un sistema control de versiones registra todos los cambios hechos en uno o más proyectos**,** guardando así versiones del producto en todas sus fases del desarrollo. Las versiones son como fotografías que registran su estado en ese momento del tiempo y se van guardando a medida que se hacen modificaciones al código fuente (Git , 2016).

GitLab 8.5.1

Es un sistema control de versiones distribuido. No depende de acceso a la red o un repositorio central. Enfocado a la velocidad, uso práctico y manejo de proyectos grandes, el mismo fue creado por Linus Torvalds, el creador del núcleo Linux. GitLab da el poder para hacer commit cualquier momento que se desee, incluso cuando no se está conectado a la red. También permite utilizar flujos de trabajos más flexibles que ayudan a trabajar mejor y con menos estrés en general. Posibilita el gran rendimiento en grandes aplicaciones, gracias a ser un sistema distribuido, cualquier búsqueda que se ejecute será muchísimo más eficaz, lo que supone una mejora en tiempos de detección de diferencias entre archivos. GitLab permite manejar el código y mejorar el flujo de trabajo de manera que hará más óptimo y productivo el tiempo de trabajo (GitLab, 2016).

1.7 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Los patrones ayudan a capturar conocimiento y a crear un vocabulario técnico, hacen el diseño orientado a objetos más flexible, elegante y en algunos casos reusable (E. Gamma, 1995). Los componentes de Symfony2 utilizan patrones de diseño los mismos son soluciones genéricas a problemas

comunes en el diseño y desarrollo de aplicaciones de software, debido a que la arquitectura de referencia Bosón utiliza como marco base a Symfony2, la misma se beneficia con los patrones de diseños que utiliza (Fabien Potencier, 2011).

Los patrones GRASP[24] son guías o principios que sirven para asignar responsabilidades a las clases, dentro de los patrones utilizados están (Larman, 1999):

Experto: Es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele ser útil en el diseño orientado a objetos. La aplicación del patrón Experto consiste en asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Este 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. Esto soporta un bajo acoplamiento, lo que 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 fáciles de comprender y mantener (Larman, 1999).

Creador: Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Permite crear instancias de otras clases según la responsabilidad de la misma. Esto posibilita un bajo acoplamiento, mejores oportunidades de reutilización (Larman, 1999).

Controlador: La aplicación del patrón Controlador consiste en asignar la responsabilidad de administrar un mensaje de eventos del sistema a una clase que represente: el negocio, la organización global, o el sistema global (Larman, 1999).

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. Este patrón soluciona el problema de lograr una dependencia escasa y un aumento de la reutilización, asignando responsabilidades a las clases de tal forma que la dependencia entre las mismas sea la menor posible (Larman, 1999).

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. La utilización de este patrón permite que las clases contengan las responsabilidades estrechamente relacionadas, sin tener que realizar un trabajo excesivo, posibilitando la claridad y facilidad con que se entiende el diseño,

[24] General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades)

simplificando el mantenimiento y las mejoras en funcionalidad, logrando en ocasiones un bajo acoplamiento (Larman, 1999).

Patrones GOF**,** Patrones publicados por Gamma, Helm, Johnson y Vlossodes en 1995. Esta serie de patrones permiten ampliar el lenguaje, aprender nuevos estilos de diseño y además se introduce más notación UML, son una descripción de clases y objetos que se comunican entre sí, adaptada para resolver un problema general de diseño en un contexto particular, dentro de los patrones utilizados están (Larman, 1999):

Creacional: Se encargan de la creación de instancias de los objetos. Abstraen la forma en que se crean los objetos, permitiendo tratar las clases a crear de forma genérica, dejando para después la decisión de que clase crear o cómo crearla. Según donde se tome dicha decisión se pueden clasificar los patrones de creación en: patrones de creación de clases (la decisión se toma en los constructores de las clases y usan la herencia para determinar la creación de las instancias) (Larman, 1999).

Estructural: Son los que plantean las relaciones entre clases, las combinan y forman estructuras mayores. Tratan de conseguir que los cambios en los requisitos de la aplicación no ocasionen cambios en las relaciones entre los objetos. Lo fundamental son las relaciones de uso entre los objetos, y éstas están determinadas por las interfaces que soportan los objetos. Estudian cómo se relacionan los objetos en tiempo de ejecución. Sirven para diseñar las interconexiones entre los objetos (Larman, 1999).

Comportamiento: Plantea la interacción y cooperación entre las clases. Los patrones de comportamiento estudian las relaciones de llamadas entre los diferentes objetos, normalmente ligados con la dimensión temporal (Larman, 1999).

1.8 Patrones de arquitectura

Los patrones de arquitectura están orientados a representar los diferentes elementos que componen una solución de software y las relaciones entre ellos. A diferencia de los patrones de diseño de software que están orientados a objetos y clases (patrones creacionales, estructurales, de comportamiento, de interacción, etc.), los patrones de arquitectura están a un mayor nivel de abstracción. Los patrones de arquitectura forman parte de la llamada Arquitectura de Software (arquitectura lógica de un sistema), que de forma resumida comprende (E. Gamma, 1995):

- ✓ El diseño de más alto nivel de la estructura del sistema
- ✓ Los patrones y abstracciones necesarios para guiar la construcción del software de un sistema.

- ✓ Los fundamentos para que analistas, diseñadores, programadores, trabajen en una línea común que permita cubrir restricciones y alcanzar los objetivos del sistema. Los objetivos del sistema, no solamente funcionales, sino de mantenimiento, auditoría, flexibilidad e interacción con otros sistemas.
- ✓ Las restricciones que limitan la construcción del sistema acorde a las tecnologías disponibles para su implementación.

Christopher Alexander[25], un reconocido arquitecto inglés dice que "cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, así como la solución a ese problema, de tal modo que se pueda aplicar esta solución un millón de veces, sin hacer lo mismo dos veces". Y aunque se refiere al mundo de la arquitectura, sus palabras describen a la perfección la esencia de los patrones de diseño de software (García, 2012).

El patrón MVC consiste en tres tipos de objetos: el modelo que es el objeto de la aplicación, la vista que es su representación y el controlador que define el modo en que la interfaz reacciona a la entrada del usuario. De una manera poco ortodoxa se puede decir que el controlador controla el flujo de la aplicación, pide al modelo aquello que el usuario solicita, y le devuelve (al usuario) una representación del modelo a través de la vista. El siguiente gráfico ilustra la cooperación entre estos tres objetos (García, 2012). (Ver Figura 1):

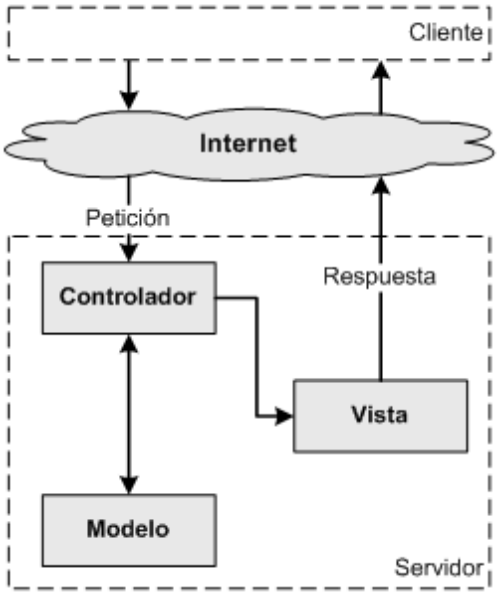


Figura 1: El patrón MVC. Fuente (Fabien Potencier, 2016)

[25] Christopher Alexander es un arquitecto, reconocido por sus diseños destacados de edificios en California, Japón y México. Estudio en Trinity College, Universidad de Cambridge, Universidad de Harvard.

Esta separación en capas con responsabilidades bien definidas permite, además de mejorar la organización del código, la posibilidad de representar de diferentes formas la misma información. En Symfony cada parte del patrón MVC constituye un sistema de varios componentes (García, 2012).(Ver Tabla 1):

Tabla 2: Estructura de la arquitectura MVC. Fuente (elaboración propia, 2016)

Patrón MVC	Componentes
Controlador	El Controlador frontal, los filtros, las acciones y los objetos request, response y sesión.
Vista	El Layout de la aplicación, las plantillas de los módulos, los slots, los partials, los componentes y los helpers.
Modelo	El ORM, los formularios, las extensiones propias que el programador realice de las clases del ORM y las clases y funciones propias que el programador construya para implementar la lógica de negocio.

1.9 Pruebas de validación

La verificación y validación (V&V) es el nombre que se da a los 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. La V&V es un proceso de ciclo de vida completo. Inicia con las revisiones de los requerimientos y continúa con las revisiones del diseño y las inspecciones del código hasta la prueba del producto. Existen actividades de V&V en cada etapa del proceso de desarrollo del software (Sommerville, 2005). La verificación y la validación no son la misma cosa, aunque es muy fácil confundirlas, Boehm (1979) expresó la diferencia entre ellas de forma sucinta:

- ✓ **Verificación**: ¿Estamos construyendo el producto correctamente?
- ✓ **Validación**: ¿Estamos construyendo el producto concreto?

Dentro de las técnicas V&V existen aproximaciones complementarias para el análisis y comprobación de los sistemas:

- ✓ Inspecciones del software: Analizan y comprueban las representaciones del sistema tales como documentos de requerimientos, los diagramas de diseño y el código fuente del programa (Sommerville, 2005).
- ✓ Las pruebas del software: consiste en contrastar las respuestas de una implementación del software a series de datos de prueba y examinar las respuestas

del software y su comportamiento operacional, para comprobar que se desempeñe conforme a lo requerido (Sommerville, 2005).

Las pruebas se realizan en cuatro etapas (Pressman, 2007):

Prueba de unidades (prueba de métodos y clases): se prueba cada método y clase de forma independiente.

Prueba de integración o de subsistemas: se prueban agrupaciones de clases relacionadas.

Prueba de sistema: se prueba el sistema como un todo.

Prueba de validación (o de aceptación): prueba del sistema en el entorno real de trabajo con intervención del usuario final.

Métodos de Prueba:

Caja blanca:

La prueba de caja blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de la caja blanca el ingeniero del software puede obtener casos de prueba que:

- ✓ Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- ✓ Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- ✓ Ejecuten todos los bucles en sus límites operacionales.
- ✓ Ejerciten las estructuras internas de datos para asegurar su validez.

Caja Negra

La prueba de caja negra no es una alternativa a las técnicas de prueba de la Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores encontrados en los métodos de la Caja Blanca. Estas pruebas permiten encontrar:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y terminación.

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien. Para

desarrollar la prueba de caja negra se utilizó la técnica de la Partición de Equivalencia. (Pressman, 2007)

1.10 Conclusiones parciales

En este capítulo se hace un análisis de los principales aspectos y conceptos relacionados con el Sistema Orbita, para dejar sentadas las bases teóricas del trabajo a desarrollar. Como resultado de este análisis se llegó a las siguientes conclusiones:

Se identificaron los elementos esenciales del marco de trabajo Symfony2 que optimizan el desarrollo de aplicaciones web, el mismo fue seleccionado luego de un estudio realizado a la arquitectura de referencia Bosón perteneciente al Departamento de Desarrollo de Componentes del Centro CEIGE, la cual se encuentra desarrollada sobre la base de este framework.

Se identificó la metodología de desarrollo para la actividad productiva en la UCI, la cual permite aumentar la calidad del software que se produce, debido a que se apoya en el modelo CMMI-DEV v1.3 que constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora.

Se explicaron las tecnologías y herramientas propuestas por la dirección del proyecto para el desarrollo del módulo Ejecución.

Se identificaron los problemas existentes en el sistema Orbita donde quedaron expuestos un conjunto de deficiencias que hacen necesaria la migración de dicho sistema.

CAPÍTULO 2: ANÁLISIS Y DISEÑO

2.1 Introducción

El presente capítulo recopila los resultados obtenidos durante el proceso de desarrollo de la solución, así como los productos de trabajo generados por el ciclo de vida del software. Se exponen los requisitos funcionales mediante historia de usuarios para proponer una visión de lo que se quiere lograr, además de la estrategia a seguir para la implementación. Es necesario que se traten puntos clave del desarrollo como la arquitectura de la aplicación, patrones de diseño utilizados, algunos productos de trabajo que proponen la metodología seleccionada y el diseño de clases con estereotipos web del sistema.

2.2 Migración del sistema Orbita

Al realizar la migración del módulo Ejecución, se realizan varios cambios en la estructura y arquitectura del módulo en sí. Un elemento necesario a la hora de realizar la migración de una aplicación con Symfony2 es conocer la estructura predefinida de los proyectos. Symfony2 proporciona una estructura en forma de árbol de archivos para organizar de forma lógica todos los contenidos de un proyecto web, además de ser consistente con la arquitectura MVC utilizada. Cada vez que se crea un nuevo proyecto, aplicación o módulo, se genera de forma automática la parte correspondiente de esa estructura.

A continuación, se muestra un ejemplo de la estructura de los proyectos de Symfony2 (Ver Figura 2) y la descripción de sus elementos (Ver Tabla 2):

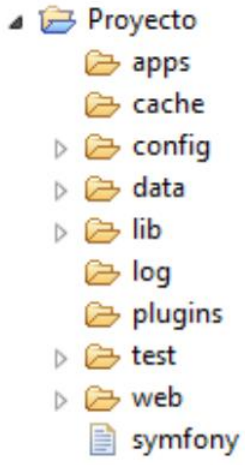


Figura 2: Estructura de los proyectos de Symfony2. Fuente (LibrosWeb, 2006-2016)

Tabla 3: Descripción de la estructura de los proyectos de Symfony2. Fuente (LibrosWeb, 2006-2016)

Directorio	Descripción
apps/	Contiene un directorio por cada aplicación del proyecto (normalmente, frontend y backend para la parte pública y la parte de gestión respectivamente).
cache/	Contiene la versión cacheada de la configuración y (si está activada) la versión cacheada de las acciones y plantillas del proyecto. El mecanismo de cache utiliza los archivos de este directorio para acelerar la respuesta a las peticiones web. Cada aplicación contiene un subdirectorio que guarda todos los archivos PHP y HTML preprocesados.
config/	Almacena la configuración general del proyecto.
data/	En este directorio se almacenan los archivos relacionados con los datos, como por ejemplo el esquema de una base de datos, el archivo que contiene las instrucciones SQL para crear las tablas e incluso un archivo de bases de datos de *SQLite*.
lib/	Almacena las clases y librerías externas. Se suele guardar todo el código común a todas las aplicaciones del proyecto. Contiene un subdirectorio llamado: model/ que guarda el modelo de objetos del proyecto.
log/	Guarda todos los archivos de log generados por *Symfony*. También se puede utilizar para guardar los logs del servidor web, de la base de datos o de cualquier otro componente del proyecto. Symfony crea un archivo de log por cada aplicación y por cada entorno.
plugins/	Almacena los *plugins* instalados en la aplicación.
test/	Contiene las pruebas unitarias y funcionales escritas en PHP y compatibles con el *framework* de pruebas de *Symfony*. Cuando se crea un proyecto, *Symfony* crea algunas pruebas básicas.
web/	La raíz del servidor web. Los únicos archivos accesibles desde Internet son los que se encuentran en este directorio.

Debido a que Symfony2 toma lo mejor de la arquitectura MVC y la implementa de manera que el desarrollo de aplicaciones sea rápido y sencillo, el sistema que se estará migrando

contará con varios cambios en la arquitectura del módulo Ejecución del Sistema Orbita, los mismos se explican a continuación:

Dentro del patrón o arquitectura MVC la vista es la encargada de proporcionar la verdadera interfaz a nuestros usuarios, aunque EXTJs es una librería Java Script Open-Source (código abierto) de alto rendimiento para la creación y desarrollo de aplicaciones web dinámicas este se encuentra bajo una licencia comercial, por lo cual se propone Symfony2 que contiene un lenguaje de plantillas llamado Twig. Twig permite escribir plantillas concisas y fáciles de leer, también contiene filtros**,** los cuales modifican el contenido antes de reproducirlo, la potencia de Twig reside sobretodo en la posibilidad de crear plantillas que heredan de otras**,** de forma que se pueden ir rellenando huecos de una plantilla padre (Potencier, 2013).
Por otra parte también se utilizan otras tecnologías tales como Bootstrap, jQuery y Angular.

- ✓ Bootstrap es un framework CSS, esta es una excelente herramienta para crear interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas, sea cual sea su tamaño. Además, Bootstrap ofrece las herramientas que permite dar forma a un sitio web mediante librerías CSS que incluyen tipografías, botones, cuadros, menús y otros elementos que pueden ser utilizados (W3SchoolsT, 2016).
- ✓ jQuery es una biblioteca de JavaScript, que nos permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX[26] a páginas web. jQuery permite además programar nuevas funcionalidades por medio de plugins para hacer cosas tan variadas como validación de formularios, sistemas de plantillas, pases de diapositivas e interfaces de usuario avanzadas (Álvarez, 2012).
- ✓ AngularJS es un framework MVC de JavaScript para el Desarrollo Web Front End que permite crear aplicaciones SPA (*Single-Page Applications*). AngularJS permite extender el vocabulario HTML con directivas y atributos, manteniendo la semántica y sin necesidad de emplear librerías externas como jQuery o Underscore.js para que funcione (Álvarez, 2012).

Teniendo en cuenta que el modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio una de las tareas más comunes y a la vez más complejas de la programación web, consiste en la persistencia de la información en una base de datos. Afortunadamente, Symfony incluye la librería Doctrine, que proporciona herramientas para

[26] Es el acrónimo de Asynchronous Javascript and XML, es decir: Javascript y XML.

simplificar el acceso y manejo de la información de la base de datos, además proporciona persistencia transparente de objetos (LibrosWeb, 2015).

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. El sistema actual hace uso de Zend Framework, este a pesar de permitir la creación de componentes de manera independiente no presenta facilidades para crearlos de manera dinámica. Teniendo en cuenta que los bundles generados automáticamente por Symfony2 incluyen un controlador frontal por defecto llamado Default, este es el encargado de atender todas las peticiones del usuario, las sesiones de los usuarios, la autenticación, etc., además este ofrece un punto de entrada único para toda la aplicación. Una aplicación implementada con Symfony2 tiene un controlador frontal de producción (por ejemplo, app.php) y un controlador frontal de desarrollo (por ejemplo, app_dev.php) (LibrosWeb, 2015).

2.3 Explicación del negocio y requisitos

Debido a que el sistema que se va a migrar parte de la segunda fase (elaboración) de la metodología variación de AUP para la UCI, es necesario ejecutar las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura.

La migración de este sistema se basa en obtener un módulo que sea capaz de mantener la lógica de los componentes existente con el objetivo de poder brindar con un mayor rendimiento el mantenimiento a los vehículos de la base de transporte de la UCI. Inicialmente se parte del análisis de los requisitos con los cuales ya cuenta el sistema y que perdurarán en el ciclo de vida de la migración del módulo Ejecución del Sistema de Control de Flota y Mantenimiento a la arquitectura de referencia en PHP Bosón.

2.3.1 Requisitos del sistema

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer (Sommerville, 2005).

Los requisitos funcionales que se tomaron en cuenta para la migracion del modulo Ejecucion fueron definidos en el trabajo de diploma titulado: Diseño e Implementación del proceso Ejecución del sistema Mantenimiento Vehicular v 1.0 .Con el objetivo de verificar que los requisitos extraídos del trabajo de diploma eran los requeridos por el cliente, se realizaron varios talleres y entrevistas, donde quedaron establecidos los siguientes requisitos para la

implementación.Ver tabla 3:

2.3.1.1 Requisitos funcionales

Tabla 4: Descripción de los requisitos funcionales. Fuente (Elaboración propia, 2016)

Número	Nombre	Descripción
RF1	Listar solicitud	Muestra la información general de las solicitudes de mantenimiento que se les realizarán a los vehículos en forma de lista.
RF2	Adicionar solicitud	Permite adicionar solicitudes de mantenimiento con un vehículo determinado.
RF3	Modificar solicitud	Permite modificar una solicitud al seleccionar una de las existentes en la lista de solicitudes.
RF4	Eliminar solicitud	Permite al seleccionar una solicitud eliminar la misma.
RF5	Buscar solicitud	Permite realizar la búsqueda de una solicitud de mantenimiento determinada.
RF6	Imprimir solicitud	Permite imprimir una solicitud de mantenimiento.
RF7	Emitir orden de trabajo	Permite al seleccionar una solicitud de mantenimiento del listado de vehículos existente emitir una orden de trabajo.
RF8	Programar mantenimiento	Permite al seleccionar un vehículo programar el mantenimiento.
RF9	Modificar orden de trabajo	Permite al seleccionar una orden de trabajo en estado abierta modificarla.
RF10	Cancelar orden de trabajo	Permite al seleccionar una orden de trabajo en estado abierta cancelarla.
RF11	Asociar recursos humanos.	Permite asociar los recursos humanos empleados en el mantenimiento a la orden.
RF12	Asociar repuestos	Permite asociar los repuestos empleados en el mantenimiento a la orden.
RF13	Asociar herramientas	Permite asociar las herramientas empleadas en el mantenimiento a la orden.

RF14	Imprimir orden de trabajo	Permite imprimir una orden de trabajo.
RF15	Listar batería	Muestra un listado de las baterías existentes, asociadas a los vehículos.
RF16	Cambiar batería	Permite cambiarla batería al seleccionar un vehículo.
RF17	Imprimir Análisis de fallas por vehículos	Permite imprimir a partir de la fecha de inicio, fecha de fin, análisis de fallas por vehículos.
RF18	Imprimir Historial de mantenimiento de un vehículo	Permite imprimir a partir de la fecha de inicio y fecha de terminación, el historial de mantenimiento de un vehículo.
RF19	Imprimir cambios de baterías realizados por un período.	Permite imprimir a partir del tipo de vehículos y la fecha, los cambios de baterías realizados por un período.
RF20	Imprimir repuestos utilizados por vehículos	Permite imprimir a partir del grupo de vehículos, fecha de inicio, fecha de fin, repuestos utilizados por vehículos.

2.3.2 Historia de usuarios para los requisitos funcionales de la aplicación

Las Historia de Usuarios (HU) son un instrumento para el levantamiento de requerimientos para el desarrollo de un software, que ha emergido con la aparición de los nuevos marcos de trabajo de desarrollo ágil, son descripciones cortas de una necesidad de un cliente del software que se esté desarrollando.

A continuación, se muestra la HU del requisito funcional Listar solicitud de mantenimiento, el resto de las HU no se incluyen en el presente documento, estas pueden ser consultadas en el expediente de proyecto Entregables con el nombre HU_módulo Ejecución.docx (Ver Tabla 4 y 5):

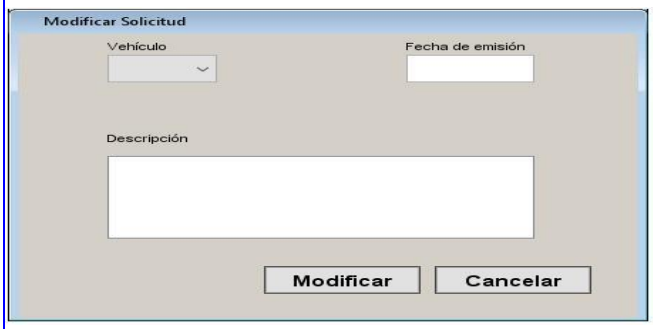
Tabla 5: Historias de usuario: Adicionar solicitud. Fuente (Elaboración propia, 2016)

Número:2	**Nombre del requisito:** Adicionar solicitud
Programador: Yudelyn Reyes Navarro	**Iteración Asignada:** 1

Prioridad: Alta	**Tiempo Estimado:** 50 horas
Riesgo en Desarrollo: Poca experiencia con la arquitectura de referencia que se usa.	**Tiempo Real:** 50 horas
Descripción: El sistema debe brindar la posibilidad al usuario de adicionar solicitudes de mantenimiento con un vehículo determinado. Permitirá al usuario añadir una solicitud de mantenimiento a un vehículo determinado. La misma contará con los campos **Vehículo**: el mismo despliega un listado de los vehículos existentes en la Base de Datos, el segundo campo será **Fecha emisión**: es un campo de tipo date que proporciona el calendario del cual se ha de elegir la fecha, este tendrá el mismo estado de la pc en la cual se encuentre el servidor de la aplicación con la siguiente estructura *dd/mm/aaaa*, el tercer campo **Descripción**: el mismo admitirá letras y números con un tamaño indefinido, además esta interfaz cuenta con dos botones para realizar la acción ya sea de **Adicionar** o **Cancelar** la nueva solicitud de mantenimiento que se esté creando. Cuando se adicione una nueva solicitud muestra un mensaje en la interfaz de los listados de las solicitudes.	
Observaciones: A un vehículo solo se le puede adicionar una solicitud.	
Prototipo de interfaz: 	

Tabla 6:Tabla 4: Historias de usuario: Modificar solicitud. Fuente (Elaboración propia, 2016)

Número:3	**Nombre del requisito:** Modificar solicitud
Programador: Yudelyn Reyes Navarro	**Iteración Asignada:** 1

Prioridad: Alta	**Tiempo Estimado:** 35 horas
Riesgo en Desarrollo: Poca experiencia con la arquitectura de referencia que se usa.	**Tiempo Real:** 35 horas
Descripción: El sistema debe brindar la posibilidad al usuario de modificar solicitudes de mantenimiento con un vehículo determinado. Permitirá al usuario modificar una solicitud al seleccionar una de las existentes en la lista de solicitudes. La misma contará con los campos **Vehículo**: este campo contiene los datos de la matrícula, el mismo admitirá letras y números con un tamaño definido entre 5 o 6 caracteres este es un elemento que no es modificable por el usuario, **Fecha emisión:** campo de tipo date el cual deberá mostrar la fecha en el formato *dd/mm/aaaa* **Descripción**: el mismo admitirá letras y números con un tamaño indefinido, además esta vista cuenta con dos botones para realizar la acción ya sea de **Modificar** o **Cancelar** la solicitud de mantenimiento que se esté modificando. Se muestra un mensaje luego de haber realizado la modificación	
Observaciones: N/A	
Prototipo de interfaz: 	

2.3.3 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. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema (Sommerville, 2005).

Para el siguiente trabajo, atendiendo a lo anterior, se especifican los requisitos no funcionales de acuerdo a las categorías que propone la norma ISO 9126.Ver Tabla 7:

Tabla 7:Listado de requisitos no funcionales del sistema. Fuente (Elaboración propia, 2016)

Usabilidad	
RNF1	El módulo debe presentar consistencia y estándares, en el mismo se puede encontrar el botón Inicio ubicado en la zona superior izquierda que permite regresar al inicio del sistema.
RNF2	El módulo brinda una interfaz fácil en un contexto desconocido, para las personas que no tienen conocimiento de los procesos del sistema.
RNF3	El módulo debe proporcionar mensajes de error que sean informativos y orientados a usuario final.
Eficiencia	
RNF4	Toda funcionalidad del módulo debe responder al usuario en menos de 5 segundos.
RNF5	Los datos que se adicionan/modifican/eliminan en la base de datos deben ser actualizados para todos los usuarios que acceden en menos de 5 segundos.
Seguridad	
RNF6	Los permisos de acceso al módulo podrán ser cambiados solamente por el administrador de acceso a datos a través de los roles.
RNF7	El acceso al módulo se encontrará protegidos contra accesos no autorizados utilizando para la autenticación el servicio de usuarios y contraseñas de la UCI.
RNF8	Para acceder al módulo debe estar el usuario autenticado y con el rol definido.
Software	
RNF9	Para el funcionamiento en la PC cliente será necesario Firefox 40 o superior.
Hardware	
RNF10	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.
Portabilidad	
RNF11	El módulo puede ser adaptado a cualquier ambiente sin tener que recibir modificaciones.
RNF12	El sistema debe ser fácil de instalar, en la carpetas de módulo Ejecución Orbita/app/vendor/Bosón/ComponentesBosón/resources/doc se encuentran documentos que explican la instalación paso a paso, la misma a tener en cuenta para el funcionamiento del módulo.

2.4 Patrones de diseño

Para la migración del módulo ejecución se emplearon los siguientes patrones de diseño, con el objetivo de proporcionan una estructura conocida, además que el software construido es más fácil de comprender, mantener y extender.

Patrones *GRASP* usados:

Creador: El sistema cuenta con diferentes clases controladoras (BateriaController.php, SolicitudController.php, OrdenTrabajoController.php) las cuales contienen las acciones o funciones que hacen al sistema funcional. En estas clases las acciones se encargan de crear los objetos de las clases que representan las entidades, evidenciando de este modo que cada clase controladora es el "creador" de las entidades.

Experto: Se evidencia este patrón puesto que Doctrine es la librería externa que utiliza Symfony para realizar su capa de abstracción al modelo de datos, encapsulando toda la lógica de los datos y generando las clases con funcionalidades comunes de las entidades. Por tanto, cada clase creada por Doctrine a partir de una entidad es experta en manejar su información.

Controlador: Todas las peticiones web son manejadas por un solo controlador frontal (app.php), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario. Este patrón se evidencia en las clases AppKernel, Controller, los "actions" y el app.php del ambiente.

Alta Cohesión: es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión garantiza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Symfony2 permite asignar responsabilidades con una alta cohesión, por ejemplo, la clase Actions tiene la responsabilidad de definir las acciones para las plantillas y colabora con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades, es decir, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas proporcionando que el software sea flexible frente a grandes cambios.

Bajo Acoplamiento: Una clase con bajo acoplamiento no depende de muchas clases. El patrón propone que cada clase debe tener un bajo grado de dependencia con otras clases en la medida de lo posible.

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.

Ejemplo del patrón bajo acoplamiento (Ver Figura 4):

Controlador

```
class SolicitudController extends Controller
```

Clase del dominio

```
class Solicitud
{

    private $id;
    private $nosolicitud;
    private $fechaemision;
    private $descripcion;
    private $idvehiculo;
    private $estadosolicitud;
    private $orden;


    public function getId(){...}
    public function setNosolicitud($nosolicitud){...}
    public function getNosolicitud(){...}
    public function setFechaemision($fechaemision){...}
    public function getFechaemision(){...}
    public function setDescripcion($descripcion){...}
    public function getDescripcion(){...}
    public function setEstadosolicitud(\Taller\TallerBundle\Entity\EstadoSolicitud $estadosolicitud = null){...}
    public function getEstadosolicitud(){...}
    public function setOrden(\Taller\TallerBundle\Entity\OrdenTrabajo $orden = null){...}
    public function getOrden(){...}
    public function setIdvehiculo($idvehiculo){...}
    public function getIdvehiculo(){...}
    public function __toString() {...}

}
```

Figura 3: Ejemplo del patrón bajo acoplamiento. Fuente (Elaboración propia, 2016)

Patrones *GOF* usados:

Estructurales:

Decorador: Este patrón añade dinámicamente nuevas responsabilidades a un objeto. Este método pertenece a la clase abstracta View, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente. El archivo base.html.twig, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página.

Otros Patrones usados:

Data Mapper: con este patrón las filas de una tabla de una base de datos son manejadas como objetos de una clase, las actualizaciones del objeto luego se reflejan como actualizaciones en la respectiva fila de la tabla.

En Symfony la estructura de la base de datos es mapeada a clases del sistema mediante un ORM (Object-Relational Mapping). Utilizando el patrón de diseño Data Mapper el desarrollador puede interactuar con el contenido de la base de datos a través de las instancias de estas clases. Todo el sistema utiliza clases del modelo generadas por Doctrine, ejemplos de estas son las entidades: Bateria.php, Solicitud.php y OrdenTrabajo.php las cuales permiten acceso a los datos de las tablas correspondientes a estas clases.

2.5 Patrón arquitectónico

Modelo Vista Controlador es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, la lógica de control en tres componentes distintos. El estilo de llamada y retorno MVC, se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, el controlador es el responsable de recibir los eventos de entrada desde la vista. Los beneficios de la utilización de dicho patrón van desde la imposición de decisiones tempranas en el desarrollo hasta la reutilización. Teniendo en cuenta las características del sistema y la utilización de la arquitectura de referencia en PHP Bosón que utiliza como marco base a Symfony2, la arquitectura MVC es la más adecuada para la representación del módulo Ejecución del Sistema Orbita ya que el *framework* basa su funcionamiento interno en la misma (Librosweb, 2013). (Ver Figura 5):

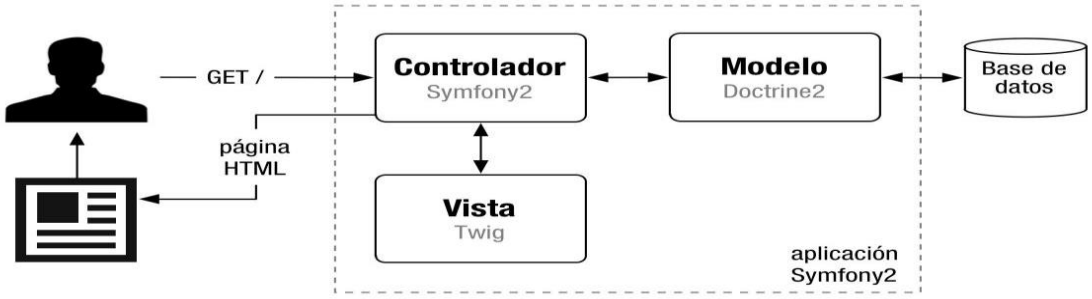


Figura 4: Arquitectura del sistema. Fuente (Fabien Potencier, 2016)

Modelo: Encapsula los datos y las funcionalidades. Se encuentran las clases, que son generadas de forma automática según la estructura de la base de datos. En Symfony, el acceso y la modificación de los datos que se almacenan en la base de datos, se realiza mediante objetos. Doctrine es un mapeador de BD que se encarga de esta generación automática para construir sus clases, creando la estructura y generando el código de las mismas. A medida que el desarrollo de un proyecto va avanzando, puede ser necesario agregar métodos y propiedades personalizadas en los objetos del modelo, esto trae consigo que se aumenten las tablas o columnas. Asimismo, cada vez que se modifica se deben

regenerar las clases del modelo de objetos (Librosweb, 2013).(Ver Figura 6).

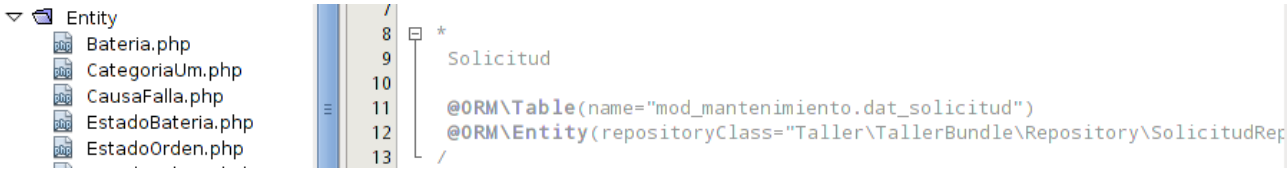


Figura 5: Ejemplo de las entidades mapeadas. Fuente (Elaboración propia, 2016)

Vista: Pueden existir múltiples vistas del modelo, cada una teniendo asociado un componente controlador. La vista es lo que utilizan los usuarios para interactuar con la aplicación (el motor de plantillas Twig pertenece a esta capa, es decir transforma el modelo en una página web. En Symfony 2 la capa de presentación está formada principalmente por plantillas (un ejemplo es index.html.twig) y el layout (base.html.twig) que contiene la información visual común a todas las páginas (Librosweb, 2013).

Las plantillas se encuentran contenidas en nuestros Bundles y en la Aplicación. Las mismas se renderizan desde el controlador con la siguiente estructura (Librosweb, 2013):

- ✓ Bundles: src/Vendor/MyBundle/Resources/views.
- ✓ Aplicación: app/Resources/views.

Cuando renderizamos las vistas (Ver Figura 7) desde el Controlador utilizamos un patrón definido (Ver Tabla 8): **[Bundle]: [Controller]:** template [._format].template_engine.

Tabla 8: Descripción de las vistas Twig. Fuente (Elaboración propia, 2016)

Definiciones	
Template	Nombre de archivo de la plantilla.
Format	Formato real que representará la plantilla.
Template_engine	Extensión real de la plantilla.

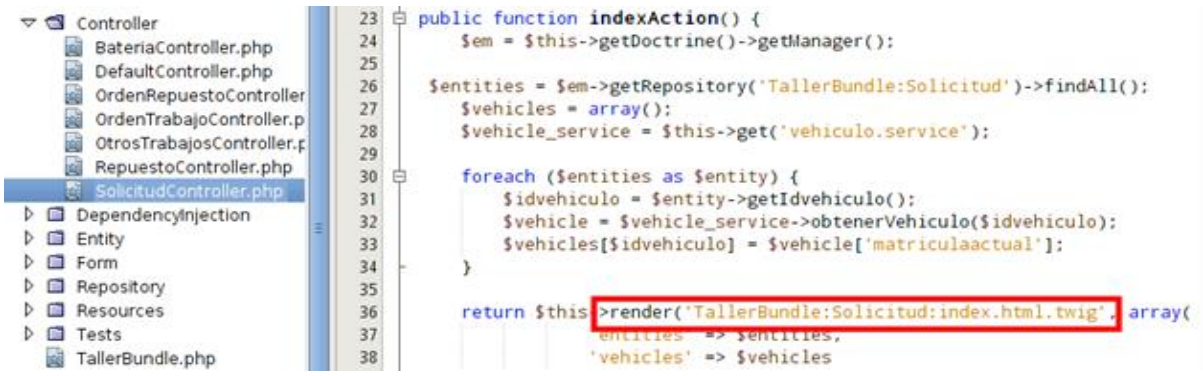


Figura 6: Ejemplo de cómo se renderizan vista Twig. Fuente (Elaboración propia, 2016)

Controlador: El controlador es el que realiza llamadas al modelo para obtener los datos y se los pasa a la vista para que los muestre al usuario. En Symfony 2 todas las peticiones se canalizan a través de los controladores frontales (app.php y app_dev.php). Recibe las entradas, normalmente como eventos. En este se encuentran las acciones, las cuales son el núcleo de la aplicación, pues contienen toda la lógica de la aplicación (Ver Figura 8). Estas acciones utilizan el modelo y precisan las variables para la vista. Al realizarse una petición web en una aplicación Symfony2, la URL define una acción y los parámetros de la petición (Librosweb, 2013).

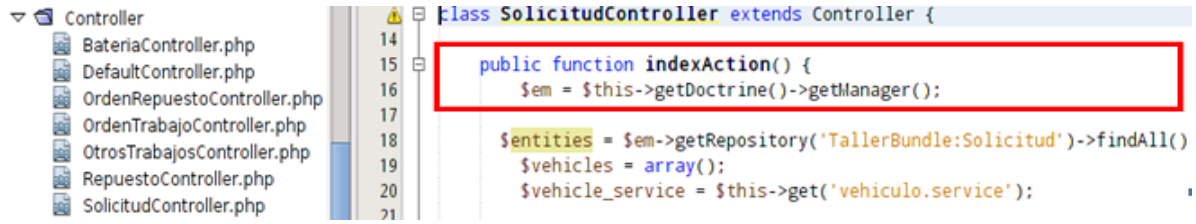


Figura 7: Acción del controlador Solicitud. Fuente (Elaboración propia, 2016)

2.6 Diagrama de clases con estereotipos web

Los diagramas de clases establecen un modelo que relaciona clases, interfaces y colaboraciones, así como sus relaciones, utilizándose para la modelación de la vista de diseño de un sistema. Cuando se quiere modelar aplicaciones web los diagramas de clases cambian debido a la necesidad de modelar páginas, los enlaces y el contenido dinámico de las mismas (Franco, 2005). En este caso se utilizan los estereotipos web: las páginas clientes, las servidoras y los formularios.

En el diagrama de clase del diseño con estereotipos web Gestionar Solicitud la Client Page (CP_Principal) le hace una petición a la Server Page (SP_GestionarSolicitud), esta es la encargada de construir las CP_ListarSolicitud, CP_ModificarSolicitud, CP_AdicionarSolicitud, CP_EliminarSolicitud con la colección de elementos de entrada (FR), para luego mostrarlos a través de la vista correspondiente a la función que se esté solicitando, dígase listar/adicionar/eliminar/modificar solicitud (Ver Figura 9).

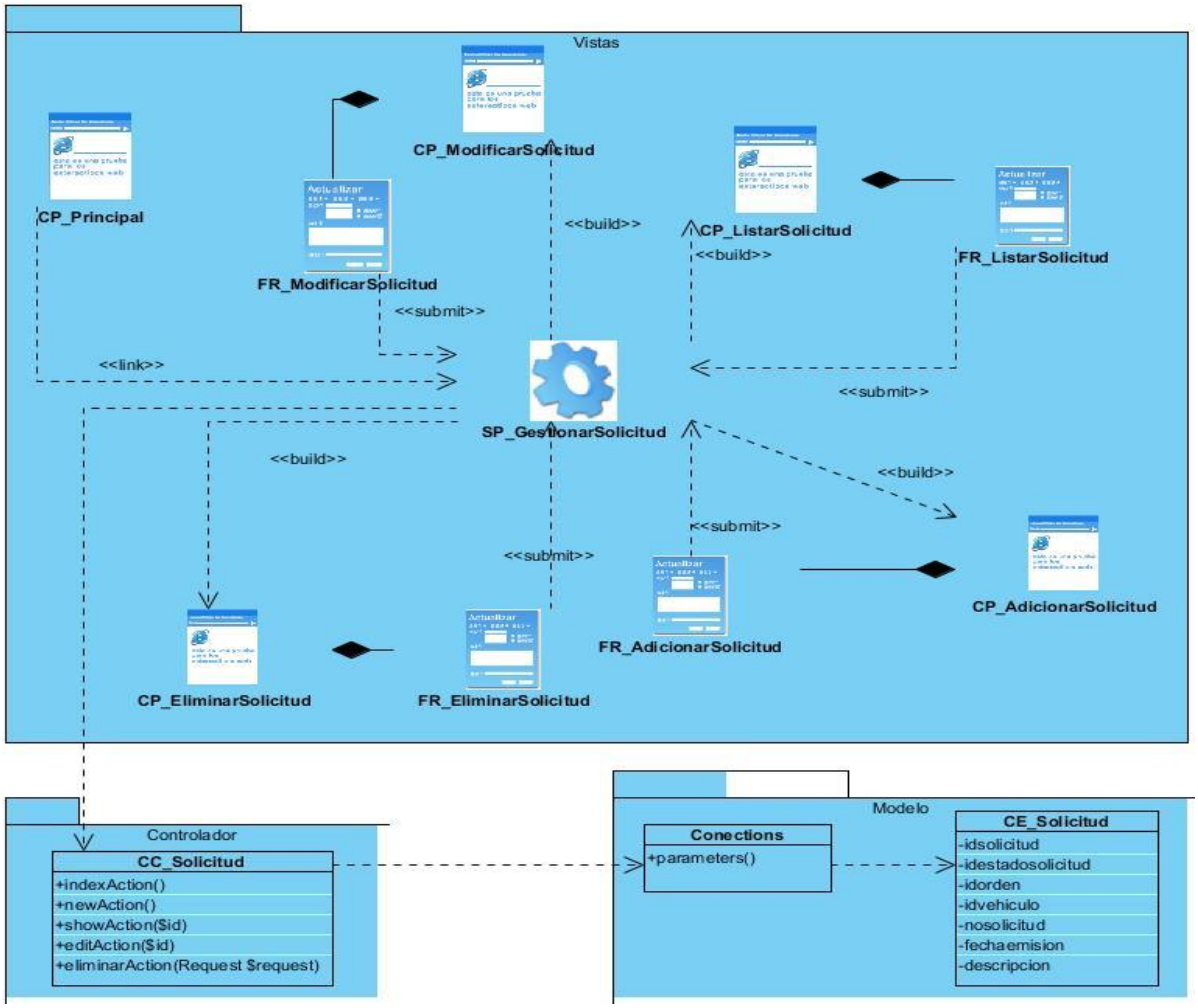


Figura 8: Diagrama de clase del diseño con estereotipos web Gestionar Solicitud . Fuente (Elaboración propia, 2016)

2.6.1 Mapeo de la base de datos.

Para la migración del módulo Ejecución se empleó la misma base de datos del Sistema Orbita. La primera acción a realizar en este paso, es mapear todas las tablas utilizando los mecanismos de Symfony2 mediante la librería doctrine. Esta tarea requiere de un tiempo considerable para su realización, más allá de su complejidad, por el monto de objetos que se deben mapear y por la cantidad de funcionalidades que se implementan nuevamente.

Se mapearon un total de 17 clases, utilizando el formato anotación[27], a su vez fueron creadas las Entitys con sus respectivos ficheros Repository que son los encargados de hacer las consultas. Con la nueva versión del ORM Doctrine 2 el funcionamiento interno transita desde las clases controladoras, accediendo a las entidades, hasta los ficheros Repository. En este

[27] Tipo de datos que guarda y reconoce en la base de datos.

archivo se establece una conexión a la base de datos, se realizan las peticiones y se envían los objetos necesarios o que fueron solicitados (Eguiluz, 2014).

Con el fin de marcar una clase para ser persistida en la base de datos, se hace con la anotación @ORM\Entity, por defecto la entidad será persistida con el mismo nombre de la clase, pero se puede modificar utilizando la anotación @ORM\Table (Ver Figura 10).

```
/**
 * Solicitud
 *
 * @ORM\Table(name="mod_mantenimiento.dat_solicitud")
 * @ORM\Entity(repositoryClass="Taller\TallerBundle\Repository\SolicitudRepository")
 */
```

Figura 9: Ejemplo de entidad. Fuente (Elaboración propia, 2016)

Ahora las instancias de Solicitud serán persistidas en una tabla llamada dat_solicitud en el esquema mod_mantenimiento en la base de datos.

Cada clase Entity necesita una clave primaria que identifique unívocamente a la entidad. Se designa la propiedad que se usará como identificador con la anotación @ORM\Id y si es un valor generado automáticamente se utiliza la anotación @ORM\GeneratedValue y como estrategia SEQUENCE a través de @ORM\SequenceGenerator (Ver Figuar 11).

Figura 10: Ejemplo de clave primaria. Fuente (Elaboración propia, 2016)

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

Para que exista una relación donde se obtenga una única instancia de un objeto se utiliza: @ManyToOne y @JoinColumn (Ver Figura 12). Doctrine2 soporta las relaciones uno-a-uno, uno-a-muchos, muchos-a-muchos, etc.

```
/**
 * @ORM\ManyToOne(targetEntity="EstadoSolicitud", fetch="EAGER")
 * @ORM\JoinColumn(name="idestadosolicitud", referencedColumnName="idestadosolicitud")
 */
```

Figura 11:Ejemplo de relación mucho-a-uno. Fuente (Elaboración propia, 2016)

La manipulación de la información de Doctrine2 (buscar, crear, modificar y borrar registros en las tablas) se realiza a través de un objeto especial llamado getManager.

El uso de un ORM es una alternativa efectiva a la hora de trasladar el modelo conceptual (orientado a objetos) al esquema relacional nativo de las bases de datos SQL. Evita la inclusión de sentencias SQL embebidas en el código de la aplicación, lo que a su vez facilita la migración hacia otro sistema gestor de bases de datos.

2.7 Conclusiones parciales.

En el presente capítulo se inició el desarrollo de la propuesta de solución para el problema definido con perspectivas de darle cumplimiento al objetivo general planteado en el marco teórico de la investigación. Se espera una mayor compresión de los procesos de negocio y un mayor entendimiento de por qué la necesidad de realizar el análisis y diseño del sistema a migrar. Con la realización de un exhaustivo análisis de los procesos, se logró:

Realizar un análisis de los requisitos existentes donde se identificaron las principales funcionalidades que deben mantenerse en el sistema que se va a migrar, dando como resultado historia de usuarios completa y sin ambigüedades.

Realizar el diseño del sistema representado por los productos de trabajo que brinda la metodología de desarrollo para la actividad productiva en la UCI en la fase de ejecución, donde quedaron elaborados los diagramas de clases de diseño con estereotipos web para un mejor entendimiento de las clases e interfaces, así como sus relaciones.

Realizar la implementación del módulo Ejecución aplicando los patrones de diseño y arquitectónico establecidos, facilitando con estos que el diseño orientado a objetos sea más flexible, elegante y en algunos casos reusable, además de mejorar la organización del código.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

3.1 Introducción.

Luego de culminadas las diferentes etapas de diseño y concepción de la herramienta propuesta como solución, se introduce en el presente capítulo el proceso de verificación de la calidad. Para ello se realiza la descripción de los productos de trabajo que son generados durante la implementación del sistema. Se muestra la implementación de los componentes que se utilizan en la aplicación, así como los estándares de codificación utilizados para lograr el desarrollo del mismo. También se notificarán los resultados de un conjunto de pruebas unitarias realizadas durante las diferentes fases de construcción.

3.2 Diagrama de componentes.

En los diagramas de componentes se muestran los elementos de diseño de un sistema de software. Un diagrama de componentes permite visualizar con facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. Se puede usar un diagrama de componentes para describir un diseño que se implemente en cualquier lenguaje o estilo. Solo es necesario identificar los elementos del diseño que interactúan con otros elementos del diseño a través de un conjunto restringido de entradas y salidas. (RUMBAUGH, et al., 2000)

A continuación, se muestra el diagrama de componentes correspondiente al módulo Ejecución (Ver Figura 13), para el mismo es necesario consumir varios servicios de otros módulos logrando con esto la integración del Sistema Orbita.

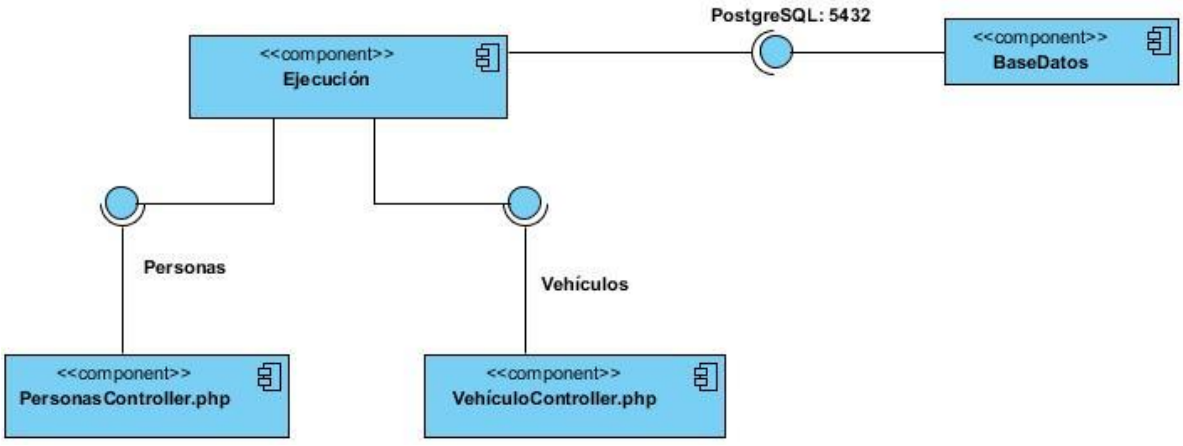


Figura 12: Diagrama de componentes. Fuente (Elaboración propia, 2016)

El módulo Ejecución consume servicios del módulo Recursos Humanos a través de la clase PersonasController.php este servicio es el encargado de brindar la información de las

personas existentes en la Base de Datos, con el objetivo de asignarle un Conductor y un Responsable (Ver Figura 14) al vehículo en cuestión.

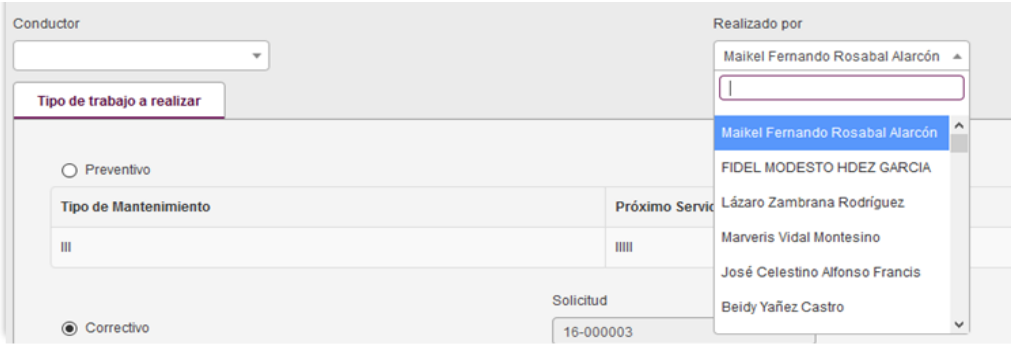


Figura 13: Consumo del servicio del módulo Recursos Humanos, Realizado por. Fuente (Sistema Orbita, 2016)

El módulo Ejecución consume servicios del módulo Vehículos a través de la clase VehiculoController.php este servicio es el encargado de brindar la información de los vehículos existentes en la Base de Datos, con el objetivo de asignar un Vehículo a la solicitud de mantenimiento en cuestión (Ver Figura 15), iniciando así el proceso del módulo Ejecución.

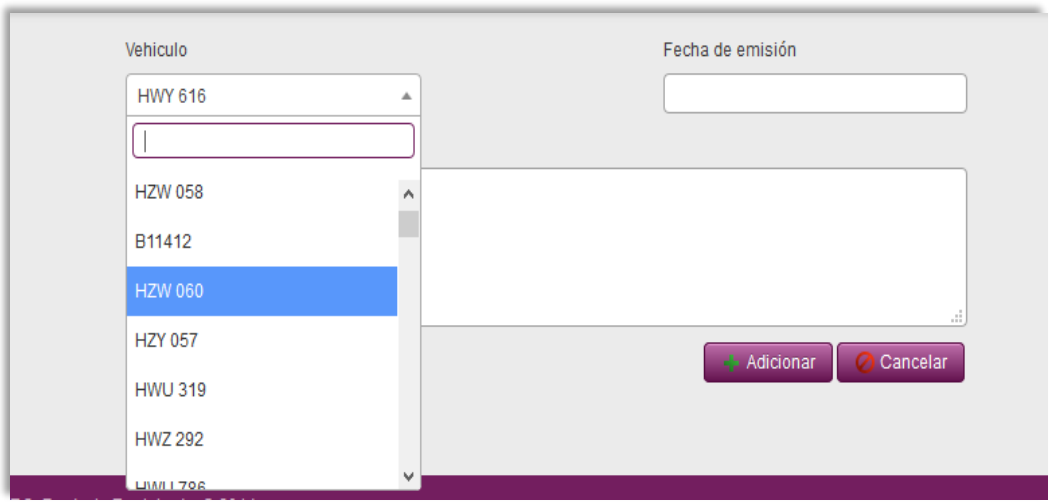


Figura 14: :Consumo del servicio del módulo Vehículos, Vehículo. Fuente (Sistema Orbita, 2016)

3.3 Estándares de codificación.

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada.

La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. El mantenimiento del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento (Pacheco, 2013).

Symfony2

Entre las normas las normas seguidas por el código de Symfony, se encuentra el estándar UpperCamelCase para el nombre de clases y variables. Solamente existen dos excepciones: las clases del núcleo de Symfony empiezan por sf (por tanto, en minúsculas) y las variables utilizadas en las plantillas que utilizan la sintaxis de separar las palabras con guiones bajos. Es necesario comentar todo el código para facilitar su revisión y entendimiento, lo que aumentará su legibilidad y por consiguiente que sea más fácil de darle mantenimiento a la aplicación (Pacheco, 2013).

Plugins

Symfony especifica que la notación de los plugins siempre debe estar atada al sufijo Plugin. El nombre de un Plugin debe indicarse con la menor cantidad de palabras cuál es el objetivo del Plugin e iniciarse con el prefijo sf .

Ejemplo: sfMiPropioPlugin

Nombre de las clases (Pacheco, 2013)**:**

- ✓ Los nombres de las clases deben estar expresados en notación UpperCamelCase.
- ✓ No se deben utilizar guiones bajos en su nombre —_.
- ✓ Deben expresar con claridad cuál es el alcance y la responsabilidad de la clase.
- ✓ Los nombres de las clases no deben estar atados a las clases de las que se deriva, cada clase debe tener un significado por ella misma, no en dependencia de la clase de la que deriva.
- ✓ En los nombres compuestos por más de tres palabras se debe revisar el diseño, no sea que se le estén dando a la clase más responsabilidades de las que realmente tiene.
- ✓ Definir las clases abstractas con Abstract.
- ✓ Denominar las interfaces con Interface.
- ✓ Denominar las excepciones con Exception.
- ✓ Utilizar caracteres alfanuméricos y guiones bajos para los nombres de archivo.

Negocio

Las clases del negocio (Ver Figura 16) deben cumplir con claridad las reglas de nomenclatura de las clases.

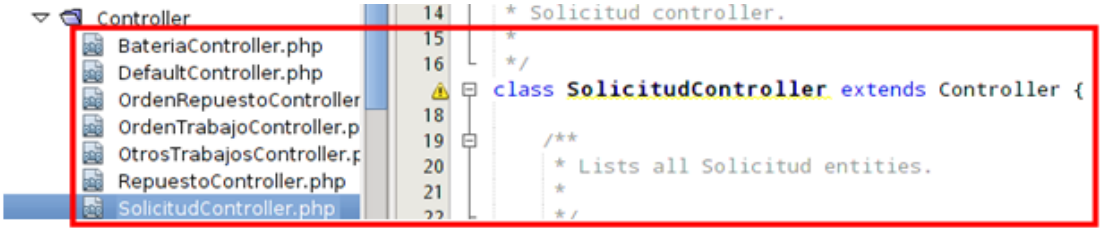


Figura 15: Ejemplo de las clases del negocio. Fuente (Elaboración propia, 2016)

Formularios

Los formularios tendrán el sufijo Form (Ver Figura 17) para identificarlos.

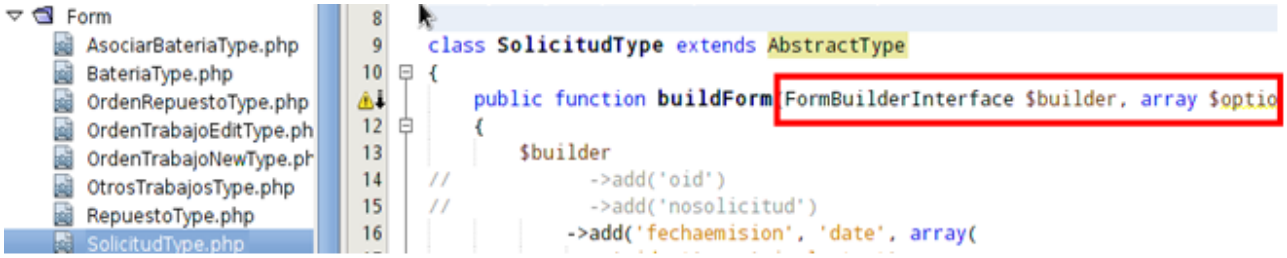


Figura 16: Ejemplo de formulario. Fuente (Elaboración propia, 2016)

3.4 Fase de prueba.

Luego de concluido un proyecto de desarrollo de software es necesario verificar la calidad del mismo antes de realizar su entrega. En el flujo de trabajo de pruebas es donde se examina el resultado de la implementación. Estas no son más que un conjunto de técnicas y métodos que garantizan el buen desempeño de una aplicación, para la validación del producto y representan un elemento crítico en la garantía de la calidad del producto.

3.4.1 Pruebas unitarias.

Las pruebas unitarias se centran en un esfuerzo de verificación de la unidad más pequeña del diseño de software: el componente y el módulo del software. “Las pruebas de unidad se centran en la lógica del procesamiento interno y en las estructuras de datos dentro de los límites de un componente” (Pressman, 2007).

Caja Blanca: Se realiza un análisis riguroso de los detalles procedimentales, comprobando las rutas lógicas del programa. El objetivo principal de este método es probar que todos los caminos del código están correctos (Pressman, 2007). Para desarrollar la prueba de caja blanca se utilizó la técnica del camino básico. A continuación se muestra un ejemplo realizado a el método EliminaBateriasAction.

Camino básico: La prueba del camino básico se basa en derivar casos de prueba a partir de un conjunto dado de caminos independientes (Ver Figura 18). Para obtener el conjunto de caminos independientes se construirá el Grafo de Flujo asociado y se calculará su Complejidad Ciclomática (Pressman, 2007).

```
public function eliminarBateriasAction(Request $request) {
    $em = $this->getDoctrine()->getManager();
    $baterias_ids = $request->request->get('choose');
    $cantidad = 0;
    if ($baterias_ids) {
        foreach ($baterias_ids as $id_bateria) {
            $entity = $em->getRepository('TallerBundle:Bateria')->find($id_bateria);
            if (!is_null($entity)) {
                $em->remove($entity);
                $em->flush();
                $cantidad++;
            }
        }
        if ($cantidad == 0) {
            $mensaje = 'No fue eliminada ninguna batería';
        } else if (count($baterias_ids) == $cantidad) {
            $mensaje = 'Fueron eliminadas todas las baterías';
        } else {
            $mensaje = 'No fueron eliminadas todas las baterías';
        }
    } else {
        $mensaje = 'No fue eliminada ninguna batería';
    }

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

Figura 17: Camino Básico. Método Eliminar Baterías. Fuente (Elaboración propia, 2016)

Complejidad Ciclomática.

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición. El camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente (Pressman, 2007).

Hay tres formas fundamentales de calcular la complejidad:

1-El número de regiones del grafo de flujo coincide con la complejidad ciclomática.

La complejidad ciclomática, V(G), se define como:

V(G) = A - N + 2 donde: A es el número de aristas del grafo y N es el número de nodos.

2-La complejidad ciclomática, V(G), también se define como:

V(G) = P + 1 donde: P es el número de nodos predicado contenidos en el grafo G.

3-V(G)= Nodos predicados, estos se definen como aquellos donde emergen dos o más aristas.

A continuación, se presenta la gráfica del flujo resultante. Ver Figura 16.

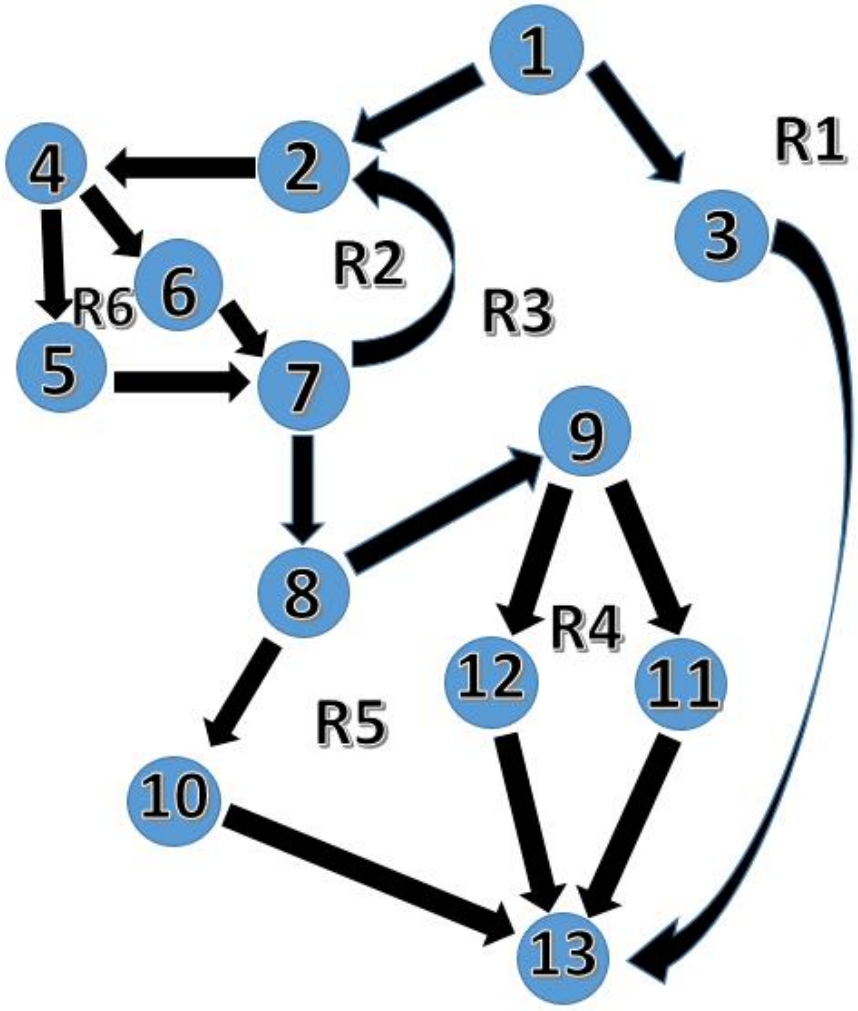


Figura 18: Grafo de Flujo resultante. Fuente (Elaboración propia, 2016)

- ✓ V(G)= 6 regiones.
- ✓ *V(G)* = 17 aristas - 13 nodos + 2 = 6.
- ✓ *V(G)* = 5 nodos predicado + 1 = 6.

Por tanto, la complejidad ciclomática del grafo de flujo de la Figura 19 es 6.

Camino 1: 1-3-11-13

Camino 2: 1-2-4-5-7-4-6-7-8-10-13

Camino 3: 1-2-4-5-7-4-6-7-8-9-12-13

Camino 4: 1-2-4-5-7-4-6-7-8-9-11-13

Tabla 9: Casos de prueba para camino básico 1. (Elaboración propia, 2016)

Caso de prueba para el camino básico 1	
Descripción: Permite eliminar una batería según la cantidad existente.	
Condición de ejecución: La cantidad de batería sea igual a cero.	
Procesamiento de la prueba	
Dato de entrada:	Id_bateria, cantidad
Tipo de dato esperado:	(interger, integer)
Resultados esperados:	Mensaje: No fue eliminada ninguna batería.

Tabla 10: Casos de prueba para camino básico 2. (Elaboración propia, 2016)

Caso de prueba para el camino básico 2	
Descripción: Permite eliminar una batería según la cantidad existente.	
Condición de ejecución: La cantidad de batería sea igual a cero.	
Procesamiento de la prueba	
Dato de entrada:	Id_bateria, cantidad
Tipo de dato esperado:	(integer, integer)
Resultados esperados:	Mensaje: No fue eliminada ninguna batería.

Tabla 11: Casos de prueba para camino básico 3. (Elaboración propia, 2016)

Caso de prueba para el camino básico 3
Descripción: Permite eliminar una batería según la cantidad existente.

Condición de ejecución: La cantidad de batería sea mayor que cero y cantidad de batería se igual a la cantidad encontrada.	
Procesamiento de la prueba	
Dato de entrada:	Id_bateria, cantidad
Tipo de dato esperado:	(integer, integer)
Resultados esperados:	Mensaje: Fueron eliminadas todas las baterías.

Tabla 12:Casos de prueba para camino básico 4. (Elaboración propia, 2016)

Caso de prueba para el camino básico 4	
Descripción: Permite eliminar una batería según la cantidad existente.	
Condición de ejecución: La cantidad de batería sea mayor que cero y cantidad de batería no es igual a la cantidad encontrada.	
Procesamiento de la prueba	
Dato de entrada:	Id_bateria, cantidad
Tipo de dato esperado:	(integer, integer)
Resultados esperados:	Mensaje: No fueron eliminadas todas las baterías.

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

3.4.2 Pruebas de funcionalidad

Para validar la solución propuesta se realizaron pruebas funcionales, las cuales permiten probar el correcto funcionamiento de los requisitos funcionales, lo que incluye la entrada de datos, el procesamiento de estos y la obtención de los resultados. Para la creación de casos de prueba de sistema se usa el método de caja negra mediante la técnica partición de equivalencia, procedimiento que permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata los errores (Pressman, 2007).

Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de

valores relacionados o una condición lógica. Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices (Pressman, 2007):

- ✓ Si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.
- ✓ Si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.
- ✓ Si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.
- ✓ Si una entrada es booleana, hay 2 clases: si o no.

Los mismos criterios se aplican a las salidas esperadas.

A continuación, se describe una de las pruebas realizada al Caso de Prueba Modificar solicitud, especificando la información de entrada, los resultados esperados y obtenidos una vez realizado el Caso de Prueba.

En el expediente de proyecto Entregables en el documento Diseño de casos de pruebas de Caja Negra.docx se muestran el resto de los casos de pruebas de caja negra aplicados a la solución.

EC1.Modificar solicitud.

- ✓ **Flujo Central.**

Seleccionar la opción Modificar solicitud de mantenimiento.

- ✓ **Condiciones de ejecución.**

El usuario debe estar autenticado en el sistema.

El usuario debe tener permisos para realizar esta acción.

El usuario debe seleccionar el módulo Talleres/Gestionar solicitud y luego la opción Modificar solicitud.

Debe existir al menos una solicitud para poder seleccionarla.

Iteración 1. Ver Tabla 13 y 14

Tabla 13:Casos de prueba para el EC2 Modificar solicitud. (Elaboración propia, 2016)

Escenario	Descripción	Vehículos	Fecha de emisión	Descripción

EC1.1 Modificar solicitud insertando valores correctos presionando el botón **Modificar.**	Se modifica una solicitud al seleccionar una de las existentes, insertando valores correctos.	V	V	V
		HYR 093	26/05/2016	Cambio de focos delanteros
EC1.2 Modificar una solicitud insertando valores correctos presionando el botón Cancelar.	Se modifica solicitud al sistema insertando valores correctos.	V	V	V
		HYR 093	26/05/2016	Cambio de focos delanteros
EC1.3 Modificar una solicitud dejando campos vacíos presionando el botón **Modificar.**	Se dejan campos vacíos al modificar solicitud.	V	I	V
		HYR 093	Vacío	Roto
		I	V	I
		Vacío	Roto	Vacío

ID Escenarios	**Respuestas del sistema**	**Flujo central**

EC 1.1	El sistema modifica una solicitud a la estructura seleccionada y muestra el mensaje de Aceptación.	-Se selecciona la estructura **Gestionar Solicitud** -Se selecciona la opción **Modificar** en la parte superior izquierda de la interfaz. -Se abre la interfaz **Modificar solicitud**. -El usuario inserta los valores deseados. -Presiona el botón **Modificar.** -El sistema valida los datos. -Se registran los datos insertados y se redirecciona para la interfaz de **Gestionar Solicitud.** -Se muestra un mensaje de **Aceptación** de la operación.
EC 1.2	El sistema no modifica una solicitud al presionar el botón **Cancelar**.	-Se selecciona la estructura **Gestionar Solicitud.** -Se selecciona la opción **Modificar** en la parte superior izquierda de la interfaz. -Se abre la interfaz **Modificar Solicitud**. -El usuario inserta los valores deseados. -Presiona el botón **Cancelar** para abortar la operación. -El sistema se redirecciona para la interfaz de Gestionar Solicitud.
EC 1.3	El sistema muestra un mensaje en el campo que corresponde cuando está vacío.	-Se selecciona la estructura **Gestionar Solicitud.** -Se selecciona la opción **Modificar** en la parte superior izquierda de la interfaz. -Se abre la interfaz **Modificar** solicitud. -El usuario inserta los valores deseados. -Presiona el botón **Modificar.** -El sistema muestra un mensaje en los campos vacíos. -El usuario llena todos los campos correctamente -Se registran los datos insertados y se redirecciona para la interfaz de **Gestionar Solicitud.** -Se muestra un mensaje de **Aceptación** de la operación.

Tabla 14:Descripción de las variables del casos de prueba Modificar. (Elaboración propia, 2016)

No	Nombre de campo	Clasificación	Valor Nulo

1	Fecha de emisión	Componente	No
2	Descripción	Text Área	No
3			No

Resultado de la prueba

El método de prueba aplicado a las funcionalidades demostró resultados satisfactorios desde el punto de vista funcional. Las no conformidades detectadas fueron analizadas y corregidas debidamente, logrando un correcto comportamiento de la funcionalidad antes diferentes situaciones (entradas válidas y no válidas). A continuación se muestra una tabla con el número de no conformidades detectadas tras cada iteración de prueba realizada. Ver Tabla 15

Tabla 15: Cantidad de no conformidades detectadas. (Elaboración propia, 2016)

Iteraciones	Cantidad de no conformidades	Tipo de no conformidades
1ra	14	Errores de interfaz, de validación y ortografía.
2da	6	Errores de interfaz, de validación y ortografía.
3ra	0	

3.4.3 Pruebas de sistema.

Las pruebas del sistema son similares a las pruebas de caja negra, solo que éstas buscan probar al sistema como un todo. Están basadas en los requerimientos generales y abarca todas las partes combinadas del sistema. Como parte del cumplimiento del procedimiento de la migración se realizan pruebas de rendimiento, las cuales se centran en determinar la velocidad con la que el sistema, bajo pruebas, realiza una tarea en condiciones particulares del escenario de pruebas.

Las pruebas de rendimiento tienen que diseñarse para asegurar que el sistema pueda procesar su carga esperada. Esto normalmente implica planificar una serie de pruebas en las que el sistema se hace inaceptable. Estas pruebas implican estresar el sistema realizando demandas que están fuera de los límites del diseño del software. (Pressman, 2007)

Existen varios tipos de pruebas de rendimientos de los cuales los más mencionados son (ISTQB, 2014):

Pruebas de Capacidad: Su objetivo es encontrar los límites de funcionamiento del sistema y detectar el cuello de botella o elemento limitante para poder actuar en caso de ampliación del servicio.

Ejemplo: el consumo de CPU en los servidores de base de datos alcanza el 100% con un nivel de servicio de 3.950 operaciones por hora.

Pruebas de estrés: Someten al sistema a una carga por encima de los límites requeridos de funcionamiento.

Ejemplo: Puesta a la venta de un producto estrella en un canal de venta.

Pruebas de Carga: Intentarán validar que se alcanzan los objetivos de prestaciones a los que se verán sometido el sistema en un entorno productivo.

Ejemplo: Un sistema web debe soportar 3.500 reservas de viajes por hora con un tiempo de respuesta no superior a 6 segundos por página.

El empleo de este tipo de pruebas sirve para diferentes propósitos tales como demostrar que el sistema cumple los criterios de rendimiento, comparar dos sistemas para encontrar cuál de ellos funciona mejor o medir qué partes del sistema o de carga de trabajo provocan que el conjunto rinda mal. Para su diagnóstico se utilizan herramientas que pueden ser monitorizaciones que midan qué partes de un dispositivo o software contribuyen más al mal rendimiento o para establecer niveles que mantenga un tiempo de respuesta aceptable. (Madeja.)

Resultados de las pruebas de rendimiento

Para la realización de este tipo de pruebas se utiliza la herramienta Apache JMeter en su versión 2.10, específicamente para realizar pruebas de rendimiento al módulo Ejecución del Sistema Orbita implementado con marco de trabajo Sauxe y al módulo Ejecución del Sistema Orbita implementado con marco de trabajo Symfony2 (Ver Anexos 2). 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. A continuación, se muestra una leyenda para comprender la tabla que ofrece la información referente a la comparación entre ambos sistemas. Ver Tabla 16

Leyenda:

- ✓ Sistema 1: Módulo Ejecución del Sistema Orbita implementado con el marco de trabajo Sauxe.
- ✓ Sistema 2: Módulo Ejecución del Sistema Orbita implementado con el marco de trabajo Symfony2.

- ✓ Máximo (Máx.): tiempo máximo de ejecución utilizado para una petición con 1 usuario realizando peticiones de manera simultánea.
- ✓ Media: tiempo de ejecución promedio de una petición con un usuario.
- ✓ Rendimiento (Kb/seg): velocidad del sistema.

Tabla 16: Comparación del Sistema Orbita (1) y el Sistema Orbita (2) . (Elaboración propia, 2016)

Indicadores	Sistema 1	Sistema
Media	193 ms	268 ms
Máximo	11997 ms	907 ms
Rendimiento (Kb/seg)	5,1 s	3,7s

En la tabla se muestran los tiempos de respuestas del Sistema 1 y el Sistema 2. A pesar de que el tiempo de ejecución promedio de una petición con un usuario en el Sistema 1 fue de 193 milisegundos y en Sistema 2 fue de 268; se puede evidenciar que existe una gran diferencia en cuanto al tiempo máximo de respuesta de las funcionalidades, lo que contribuye a una reducción de la velocidad del sistema a 3,7 segundos estableciendo una diferencia de 1,4 segundos entre ambos módulos, lo que contribuye una mejora en la velocidad del módulo Ejecución. Ver Anexos 2

3.4.4 Prueba de aceptación

Las pruebas de aceptación son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales.

Luego de la culminación de las pruebas unitarias y de sistema se procedió a realizar las pruebas de aceptación; la misma se llevaron a cabo por el cliente Ernesto Mató, fueron realizadas bajo un ambiente controlado para determinar si el software cumple con lo estipulado por el cliente y lo establecido en la historia de usuarios. Terminadas estas pruebas y corregidas las no conformidades el cliente avaló la solución quedando entonces el Certificado de Aceptación del Producto por parte del mimo.

3.5 Conclusiones del capítulo

Al concluir el desarrollo de este capítulo se obtuvo un sistema que cumple satisfactoriamente con los requerimientos definidos, para arribar a dicho resultado se tuvieron en cuenta los siguientes elementos:

Se estructuraron las clases del diseño en paquetes de componentes lo que facilitó la organización y las dependencias entre los componentes que conforman el módulo.

Se realizaron las pruebas necesarias para el control de la calidad del producto que permitieron la corrección de las no conformidades detectadas.

Se detalló el uso de un estándar de codificación, lo que proporcionó mayor organización y claridad, contribuyendo así a realizar una programación explícita.

Se obtuvo la implementación de todas las funcionalidades, garantizando el correcto funcionamiento mediante la prueba de rendimiento, además que esta tributó a la prueba de aceptación donde el cliente avaló la solución.

CONCLUSIONES GENERALES

Con la investigación realizada, el diseño y la implementación de una solución informática para dirección de Transporte de la UCI se obtuvieron resultados que permiten arribar a las siguientes conclusiones:

- ✓ El análisis de los requisitos existentes en el sistema antecesor permitió establecer historia de usuarias concisas y sin ambigüedades, contribuyendo estas en la implementación de las funcionalidades requeridas por el cliente.
- ✓ La utilización de la arquitectura de referencia Bosón permitió realizar el diseño del nuevo módulo obteniendo interfaces más amigables para los usuarios, gracias a los beneficios de esta arquitectura.
- ✓ El proceso de implementación permitió desarrollar el producto a partir de los resultados del análisis y diseño donde quedó establecida la nueva estructura del proyecto en cuestión, contribuyendo así a darle cumplimiento a los requisitos establecidos por el cliente.
- ✓ La validación de la solución propuesta a través de las pruebas unitarias, funcionales, y de rendimiento, así como la resolución de las no conformidades detectadas durante este proceso, permitieron asegurar que el trabajo realizado se acerca (en lo posible) a la perfección en cuanto calidad y desarrollo seguro. Evidenciándose esto en el mejoramiento de los tiempos de respuestas del módulo Ejecución lo cual contribuye a un aumento del rendimiento del Sistema Orbita y a la aceptación por parte del cliente.

RECOMENDACIONES

Realizar la integración del módulo Ejecución con los otros los módulos del Sistema Orbita implementados sobre la arquitectura de referencia en PHP Bosón, para que el sistema pueda trabajar como un todo y brinde los servicios con la calidad requerida en el área de la dirección de Transporte de la UCI.

Bibliografía

ISTQB. 2014. PMOinformatica. [En línea] 29 de enero de 2014. [Citado el: 26 de febrero de 2016.] http://www.pmoinformatica.com/2014/01/tipos-de-pruebas-de-software-istqb.html.

Abella, Mª Belén Muñoz. 2008. *Mantenimiento industrial.* Universidad Carlos III de Madrid Área de Ingeniería Mecánica. Madrird : s.n., 2008. pág. 47.

Abraham Calás Torres, Katia Saria Preval,Ricardo E. Suarez Riquenes. 2016. *ARQUITECTURA DE REFERENCIA PARA PHP.* La Habana : s.n., 2016.

ALEGSA. 2010. Diccionario de informática y tecnología. [En línea] ALEGSA,Santa Fe-Argentina, 2010. [Citado el: 10 de febrero de 2016.] http://www.alegsa.com.ar/Dic/rendimiento.php.

Alegsa, Leandro. 2010. DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA. [En línea] 12 de octubre de 2010. [Citado el: enero de 25 de 2016.] http://www.alegsa.com.ar/Dic/sgbd.php.

Álvarez, Miguel Angel. 2012. Desarrolloweb. [En línea] 19 de septiembre de 2012. [Citado el: 03 de marzo de 2016.] http://www.desarrolloweb.com/manuales/manual-jquery.html..

Amat, Daniel A. Casals. 2015. Comunidad cubana de PHP. [En línea] Universidad de las Ciencias Informáticas, 2015. [Citado el: 27 de febrero de 2016.] http://php.uci.cu/comenzando-con-boson/.

Andi Gutmans, Stig Bakken,Derick Rethans. 2004. The ACM Digital Library. [En línea] 2004. [Citado el: enero de 20 de 2016.] http://dl.acm.org/citation.cfm?id=1238068. 013147149X.

Arturo, Daniel. 2015. PHPCuba Comunidad cubana de PHP. [En línea] 25 de noviembre de 2015. [Citado el: 13 de febrero de 2016.] http://php.uci.cu/el-componente-excepcionesbundle-de-boson/.

BarbaraPVN. 2013. Entendiendo HTML5: guía para principiantes. [En línea] 08 de mayo de 2013. [Citado el: 14 de noviembre de 2015.] http://hipertextual.com/archivo/2013/05/entendiendo-html5-guia-para-principiantes/..

Bass, Len, Clements, Paul y Kazman, Rick. 1998. *Software Architecture in Practice.* s.l. : Addison-Wesley Publishing Co, 1998. 0-321-15495-9.

Cáceres, Ing. María Beatriz. 2011. Mantenimiento mundial. [En línea] 2011. [Citado el: 10 de febrero de 2016.] http://www.mantenimientomundial.com/sites/mm/notas/competitividad.pdf.

Can, Carolina Nobelo. 2011-2016. Scrib. [En línea] Campeche, 06 de septiembre de 2011-2016. [Citado el: 21 de marzo de 2016.] http://es.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales#scribd..

Carnegie Mellon University. 2013. Software Engineering Institute.Carnegie Mellon University. [En línea] Carnegie Mellon University, diciembre de 2013. [Citado el: 25 de enero de 2016.] https://www.sei.cmu.edu/searchresults.cfm?PUBID=&Q=perfomance%20in%20software&client=sei_dam_2013_frontend&site=sei_dam_2013&output=xml_no_dtd&sort=date%3AD%3AL%3Ad1&exclude_apps=1&entqr=3&filter=p&getfields=*&start=10&num=10.

Carolina, Terna. 2005.. Pruebas de Funcionalidad. [En línea] 02 de mayo de 2005. [Citado el: 18 de febrero de 2016.] http://carolina.terna.net/ingsw3/datos/Tipos_Prueba.pdf.

CATRIAN. 2014. Las ventajas de usar GIT. [En línea] 2014. [Citado el: 05 de marzo de 2016.] http://www.catrian.com/las-ventajas-de-usar-git/..

César Leonidas Padilla Valdez, Ing Marco Amaya Pinos. 2012. *Carrera mecánica automotriz. Plan de gestión del mantenimiento vehicular de la ciudad de Cañar.* Cuenca-Ecuador : s.n., 2012.

Chagoya, Ena Ramos. 2008. GestioPolis. [En línea] WebProfit Ltda, 01 de julio de 2008. [Citado el: 20 de enero de 2016.] http://www.gestiopolis.com/metodos-y-tecnicas-de-investigacion/.

Ciberaula. 2010. Una Introducción a Apache . [En línea] 2010. [Citado el: 14 de noviembre de 2015.] http://linux.ciberaula.com/articulo/linux_apache_intro...

Ciolli, María Elena. 2008. Testing de migración de aplicaciones distribuidas a entornos Web. [En línea] mayo de 2008. [Citado el: 24 de febrero de 2016.] http://postgrado.info.unlp.edu.ar/Carreras/Magisters/Ingenieria_de_Software/Tesis/Ciolli_Maria_Elena.pdf.

Cornejo, José Enrique González. ¿Qué es UML? . [En línea] docIRS. [Citado el: 14 de noviembre de 2016.] http://www.docirs.cl/uml.htm..

Corporation, Borland Software. 2015. Silk Performer. [En línea] 2015. [Citado el: 25 de febrero de 2016.] http://www.borland.com/Borland/media/Resources/Data%20sheets/BDS-Silk-Performer-16-5.pdf?ext=.pdf.

Delgado, Luis Miguel. 2015. ¿Que es CSS? y ¿Qué me importa a mi? [. [En línea] 2015. [Citado el: 10 de noviembre de 2015.] http://tunegocioenlanube.net/que-es-css3-y-que-importa-mi/..

DESOFT. 2015. Las comunicaciones al servicio de la sociedad. [En línea] 25 de febrero de 2015. [Citado el: 27 de enero de 2016.] http://www.mincom.gob.cu/?q=node/821.

Doctrine. 2009. Doctrine. [En línea] 2009. http://docs.doctrine-project.org/projects/doctrine1/en/latest/en/index.html.

Doctrine Team. 2010. Documentation, D.O. Doctrine 2 ORM Documentation. [En línea] 2010. [Citado el: 22 de marzo de 2016.] http://www.doctrine-project.org/about.html.

Domínguez Medina, Lisdanay. 2010. *SISTEMA DE GESTIÓN DEL CAPITAL HUMANO.* 2010.

Donadello, Lic. Domingo. 2013. Glosario de Ingenería de Software. [En línea] Buenos Aires, 2013. [Citado el: febrero de 12 de 2016.] http://www.ub.edu.ar/catedras/ingenieria/ing_software/ubftecwwwdfd/glossary/glosary.htm.

E. Gamma, R. Helm, R.Johnson, and J. Vlissides. 1995. *Desing Patterns.Addison Wesley.* 1995.

Eguiluz, Javier. 2014. Desarrollo web ágil con Symfony2. 2014.

Fabien Potencier, François Zaninotto. 2011. Librosweb. [En línea] 2011. [Citado el: 12 de febrero de 2016.] http://librosweb.es/libro/symfony_1_4/capitulo_2/el_patron_mvc.html.

Franco, J.A. 2005. *"UML en acción. Modelando Aplicaciones Web".* 2005.

García, Juan David Rodríguez. 2012. Desarrollo de Aplicaciones web con symfony 1.4 . . [En línea] 2012. [Citado el: 06 de marzo de 2016.] http://juandarodriguez.es/cursosf14/unidad7.html..

Gèlvez, Hugo Alexander Parada. 2010. *Contribucion a la Gestion de los Procesos de Pruebas de Sofware y Servicios. .* 2010.

Gerner, Jason y et al. Professional LAMP. 2006. *Linux, Apache, MySQL and PHP 5 Web Development.* Indianopolis : Wiley Publishing , 2006. 978-0-7645-9723-7...

Git . 2016. Git --fast-version-control. [En línea] 2016. [Citado el: 7 de febrero de 2016.] https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones.

GitLab. 2016. GitLab. [En línea] 2016. [Citado el: 7 de febrero de 2016.] https://about.gitlab.com/community/.

Guerra, Frank Orlando Menéndez. 2010. *Migración del subsistema Gestión de Cuadros y Personal de Apoyo del proyecto Sistema de Gestión Fiscal a la versión 1.3 del framework Symfony. .* La Habana : s.n., 2010.

Hernández, Ariel Portal. 2012. *Diseño e Implementación del proceso Ejecución del sistema Mantenimiento Vehicular v 1.0.* La Habana : Universidad de las Ciencias Informáticas., 2012.

Ian Gorton, Anna Liu. 2003. Evaluating the performance of EJB components. [En línea] mayo de 2003. [Citado el: 26 de mayo de 2016.] https://scholar.google.com/citations?view_op=view_citation&hl=en&user=SqC-kTwAAAAJ&citation_for_view=SqC-kTwAAAAJ:Y0pCki6q_DkC.

2010. . Industrial, Portal de mantenimiento.Solo_Mantenimiento. . [En línea] 20 de octubre de 2010. . [Citado el: 20 de febrero de 2016.] www.solomantenimiento.com.

Informática. 2016. Application Consolidation and Migration. [En línea] Informática LLC., 2016. [Citado el: 26 de mayo de 2016.] https://www.informatica.com/pe/solutions/application-consolidation-and-migration.html.

2011. InteraSystem. [. [En línea] InteraSystem, 2011. [Citado el: 01 de diciembre de 2016.] http://www.usa.interasystem.com/index.php/mnucaracteristicasproducto..

Jiménez Quintana, Jorge Yosmiel y Rosa Castellano, Luis Mariano. 2015. *Procedimiento Pruebas de Rendimiento.* La Habana : s.n., 2015.

1999-2016. . JMeter, Apache.The Apache Software Foundation, . [En línea] 1999-2016. . [Citado el: 25 de febrero de 2016.] http://jmeter.apache.org/..

Larman, Craig. 1999. *Uml y Patrones.Introducción al análisis y diseño orientado a objetos.* s.l. : PRENTICE HAL, 1999. 970-17-0261-1..

Librosweb. 2013. Librosweb. [En línea] 2013. http://librosweb.es/libro/symfony_1_2/capitulo_2/el_patron_mvc.html.

LibrosWeb. 2015. LibrosWeb. [En línea] 2015. [Citado el: 12 de febrero de 2016.] http://librosweb.es/libro/symfony_1_2/capitulo_1/symfony_en_pocas_palabras.html.

LP, Hewlett Packard Enterprise Development. 2016. LoadRunner. [En línea] 2016. [Citado el: 25 de febrero de 2016.] http://www8.hp.com/es/es/software-solutions/loadrunner-load-testing/..

Madeja. *Marco de Desarrollo de la Junta de Andalucía. Buenas prácticas en el diseño.*

Mark Otto, Jacob Thornton. 2015. Librosweb. [En línea] 2015. [Citado el: 28 de enero de 2016.] https://librosweb.es/libro/bootstrap_3/.

Marón, Walter René Calisaya. 2015. Curso Ingeniería de mantenimiento hospitalario. [En línea] 25 de abril de 2015. [Citado el: 20 de febrero de 2016.] http://www.mailxmail.com/curso-ingenieria-mantenimiento-hospitalario.

Mozilla. 2016. Mozilla. [En línea] 2016. https://www.mozilla.org/en-US/foundation/.

Muñoz Tamayo, Francisco y Roberto Aballí Mor, Félix. 2009. *Herramienta para la migración de datos de Microsoft Access a PostgreSQL.* La Habana : Universidad de las Ciencias Informáticas, 2009.

NetBeans. 2016. NetBeans. [En línea] 2016. [Citado el: 25 de enero de 2016.] https://netbeans.org/about/history.html.

Nielsen, Jakob. 2005. *Designing Web Usability: The Practice of Simplicity.* s.l. : Prentice-Hall, 2005. pág. 432.

Pacheco, Nacho. 2013. Estándares de codificación¶. [En línea] 22 de abril de 2013. [Citado el: 16 de abril de 2016.] http://gitnacho.github.io/symfony-docs-es/contributing/code/standards.html.

PIÑERO, P.Y., TORRES, S., PESTANO, H., VAZQUEZ, M., IZQUIERDO , M. and JORRÍN , M. 2010. Paquete de Herramientas para Gestión de Proyecto. [En línea] 2010. https://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&sqi=2&ved=0ahUKEwiS9oX745vNAhVCHx4KHZCxDAoQFggmMAI&url=http%3A%2F%2Fpublicaciones.uci.cu%2Findex.php%2FSC%2Farticle%2Fdownload%2F1373%2F718&usg=AFQjCNEzXiMP99HFXgdyQmfyccHhfxCf_w&sig2=5igb.

PostgreSQL. 2011. PostgreSQL. [En línea] 2011. [Citado el: 14 de noviembre de 2015.] http://www.postgresql.org/about/..

Potencier, Fabien. 2013. symfony. [En línea] abril de 22 de 2013. [Citado el: 23 de marzo de 2016.] http://gitnacho.github.io/symfony-docs-es/book/templating.html..

Prado, Francisco González de. 2010. *Gestion de flotas.Planes de mantenimiento de vehículos y organización del tráfico.* 2010.

Pressman, Roger. 2007. *Ingeniería del Software, Un enfoque práctico.* Nueva York, E.U.A : McGraw-Hill, 2007. págs. pp. 383-385, 419-432). Vol. 6ta Edición.

2012. QA Técnico. Pruebas de rendimiento: Tipos y objetivos Pruebas de rendimiento:. [En línea] 03 de marzo de 2012. [Citado el: 05 de mayo de 2016.] http://qatecnico.blogspot.com/2012/03/pruebas-de-rendimiento-tipos-y.html..

Real Academia Española. 2016. Real Academia Española. [En línea] Madrid, España, 2016. [Citado el: 25 de enero de 2016.] http://www.rae.es/.

Rouse, Margaret. 2014. TechTarget. [En línea] 2014. [Citado el: enero de 25 de 2016.] http://searchsoftwarequality.techtarget.com/definition/performance-testing.

RUMBAUGH, James, JACOBSON, Ivar y BOOCH, Grady,Addison Wesley. 2000. *"El lenguaje unificado de modelado. Manual de referencia" .* Capítulos 4 y 13 . 2000. págs. Páginas 42-48122-125, 131-143, 156, 162-170, 191- 197, 211, 299-303, 367-373, 399-402, 427, 434-435, 446-4.

Saldaña, Gabriel. 2014. Introduccion a Git: Un sistema control de versiones bien hecho. [En línea] 2014. [Citado el: 12 de marzo de 2016.] http://www.gabrielsaldana.org/platica_git.pdf..

Sánchez, Alonso. 2014. Prezi. [En línea] 19 de septiembre de 2014. [Citado el: 25 de enero de 2016.] https://prezi.com/7wmx8_d6ertl/entorno-desarrollo-integrado-ide/.

Sánchez, Tamara Rodríguez. 2014. *Metodología de desarrollo para la Actividad productiva UCI.* s.l. : UCI, 2014.

Sencha. 2016. Sencha. [En línea] 2016. [Citado el: 23 de marzo de 2016.] https://www.sencha.com/legal/sencha-software-license-agreement/.

servicios, Equipo del Producto CMMI Mejora de los procesos para el desarrollo de mejores productos y. 2010. CMMI para Desarrollo,Version 1.3 CMMI-DEV.TECHNICAL REPORT CMU/SEI-2010-TR-033 ESC-TR-2010-033 Software Engineering Process Manage. [En línea] 2010. [Citado el: 10 de febrero de 2016.] https://www.sei.cmu.edu.

Sommerville, Ian. 2005. *Ingeniería de Software.* 7ma Edición. Madrid : Pearson Education S.A, 2005.

Stallman, Richard M. 2015. Free Software, Free Society. [En línea] second, 05 de junio de 2015. [Citado el: 20 de 01 de 2016.] https://www.gnu.org/doc/fsfs-ii-2.pdf.

2016. Symfony2 tutorial. Que son las entidades en Symfony 2. [En línea] 2016. [Citado el: 12 de febrero de 2016.] http://symfony2tutorial.com/symfony-2/entidades/que-son-las-entidades-en-symfony-2/..

2013.. Testeando Software.Asegurando la Calidad del Software. LoadUI. Pruebas de carga hechas arte. . [En línea] 10 de diciembre de 2013. [Citado el: 25 de febrero de 2016.] http://testeandosoftware.com/loadui-pruebas-de-carga-hechas-arte/..

Torres, Abraham Calas. 2015. Excriba. [En línea] 2015. [Citado el: 06 de junio de 2016.] https://excriba.prod.uci.cu/proxy/alfresco//api/node/content/workspace/SpacesStore/e5032d4e-1472-46bf-b554-e37016ed0009/CEIGE_Boson_Ficha%20tecnica%20del%20proyecto.odt?a=true.

Varela-Pérez, L. M. 2014.. *SISTEMA WEB PARA EL ANÁLISIS DE ESCENARIOS BASADO EN MAPAS COGNITIVOS DIFUSOS.* 2014.

W3SchoolsT. 2016. THE WORLD'S LARGEST WEB DEVELOPER SITE. [En línea] 2016. [Citado el: 22 de marzo de 2016.] http://www.w3schools.com/bootstrap/.

Wilson, Jose Manuel Ayala. 2011. Blog sobre arquitectura, multitarea, programación y tecnología web y mobile. . [En línea] 2011. [Citado el: 06 de marzo de 2016.] http://jmaw.blogspot.com/2011/04/h2-margin-bottom-0.html..

Xavier Molero, Carlos Juiz,Miguel Jesús Rodeño. 2015. Evaluación y Modelado del Rendimiento de los Sistemas Informáticos. [En línea] 01 de octubre de 2015. [Citado el: 25 de noviembre de 2016.] https://www.researchgate.net/profile/Carlos_Juiz/publication/266565524_Evaluacion_y_Modelado_del_Rendimiento_de_los_Sistemas_Informaticos/links/560cf01b08ae6c9b0c42dfc1.pdf. 84-205-4093-5.

Zend Technologies Ltd. 2016. Zend Framework2. [En línea] 2016. [Citado el: 16 de enero de 2016.] http://framework.zend.com/downloads/archives.

ANEXOS 1

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, cuales no se cargan? _____

   _____
5. ¿Se cargan correctamente los elementos de las listas? Sí___No___
   ¿En caso de NO, cuales no se cargan? _____

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

   _____

   _____
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?

   _____

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

9. ¿El Sitio 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, donde se encuentra ubicada? _____

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

    _____

    _____
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.

    _____

    _____

Resultados de las encuestas:

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? 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 listas? 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 ya existía un antecedente en el cual utilizaban esta versión al igual que Sauxe 2.0
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 usó una codificación adecuada y algunas veces no se respetó el patrón MVC

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 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 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 le 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?
   El criterio principal es basado en que actualmente el Polo cuenta con un sistema llamado Vehículo CPNB Venezuela y muchas de las funcionalidades se ajustan al de la Universidad. El mismo está construido con la tecnología 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? El sistema orbita su entorno de trabajo es parecido a escritorio de Linux donde puede tener varias ventanas al mismo tiempo o minimizada.

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

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

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

14. ¿La interfaz de búsqueda está ubicada donde los usuarios esperan encontrarla (en la parte superior derecha de la página)? Sí 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.
No puesto suficiente documentación por consiguiente se utilizó la capacitación de los especialistas del dpto DC.

16. Considera usted que a la hora de la implementación del sistema Orbita se realizaron malas prácticas de programación. ¿Cuáles?
Sí, primero que todo se debe poseer con una actualización de la de tecnología del MT Sauxe a versiones superiores, en el acceso a BD nuevas consulta están implementado que permiten inyección SQL.

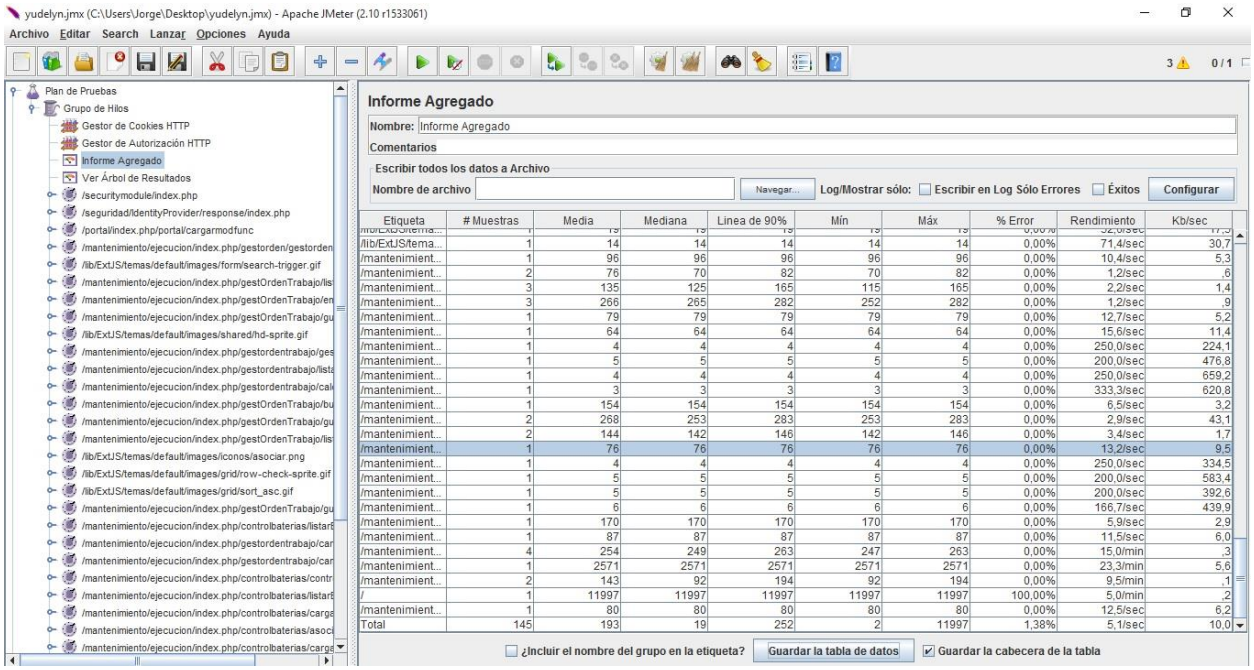
17. Al sistema Orbita se le implementó un sistemas de auditorías al código fuente o al cumplimiento de las definiciones arquitectónicas. Argumente.
No tiene ninguno de los dos elementos implementado.

_____

18. Para el desarrollo del sistema Orbita se tuvo en cuenta el control y seguimiento a la implementación de las soluciones.
El mismo fue desarrollado por estudiantes y profesores, en todas las clases del sistema no se utilizo seguimiento a la implementación

ANEXOS 2

Prueba de rendimiento con la herramienta Jmeter al sistema Orbita implementado con el marco de trabajo Sauxe.



Prueba de rendimiento con la herramienta Jmeter al sistema Orbita implementado con el marco de trabajo Symfony2.