UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 3

Centro de Informatización de Entidades

Migración de los módulos Persona y Combustible del Sistema Orbita a la arquitectura de referencia en PHP Bosón.

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

Autor: Raciel Ferrin González

Tutores: Ing. Yaniris Blanco Zamora

Ing. Ernesto Mató Roque

La Habana, 29 de junio de 2016

"Año 58 de la Revolución"

Declaración de autoría

Declaro ser el autor 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 2016.

_____

Raciel Ferrin González

Firma del Autor

_____ _____

Ing. Yaniris Blanco Zamora **Ing. Ernesto Mató Roque**

Firma de la Tutora Firma delTutor

Dedicatoria

Con todo el amor de mi alma te dedico esta tesis a ti mami que siempre has sido mamá y papá, mi confidente, mi apoyo, mi guía y mi ejemplo.Que siempre me has complacido en todo, por formarme con tanta dedicación, Te amo. A tía Marcia, que no conozco una vida sin ella, por educarme, por enseñarme a ser un hombre de bien, por regañarme cuando tuviste que hacerlo, esto es por ustedes. A mi familia que todos pusieron su granito de arena, para formarmi camino hasta aquí. A mi papá, por su confianza y sus consejos.A mis abuelos, por acompañarme siempre. A todos los que me apoyaron y contribuyeron a mi formación.A Mary que siempre estuvo presente a pesar de la distancia. ¡Gracias por todo!

Agradecimientos

Hay momentos en la vida que nos marcan para siempre, hoy es uno de esos momentos, es la culminación de mi carrera universitaria que ha sido posible gracias a muchas personas.En este trabajo de diploma he contado con la colaboración de muchas personas a quienes no puedo obviar. Sería imperdonable.

A mi mamá y mi tía que sin ellas no hubiese sido posible nada de esto, gracias por toda mi vida. Las Amo.

A ti padre por siempre estar ahí cuando lo he necesitado, por tu confianza y tus palabras.

A mis abuelos que, aunque no están vivos siempre están presente en mi recuerdo, ustedes que fueron participes de los primeros años de mi vida.

A mi hermano por soportarme y corre conmigo, cuando fue necesario. Por los buenos momentos y los malos.

A Oscar mi hermano de años, gracias por todo.

A mis compañeros de aula, y de la UCI que hicieron de estos 5 años algo digno de recordar.

A César y Rigo, que hicieron esto posible, por todas las madrugas en el laboratorio.

A Mary por darme su amor incondicional y por cuidar de nuestro amor. Gracias por haber llegado a mi vida, y por dejarme ser parte de la tuya. Te amo

Resumen

En el presente trabajo se describe la migración de los módulos Combustible y Persona del Sistema Orbita a la arquitectura de referencia en PHP Bosón, para la dirección de Transporte de la UCI. Estos módulos permiten realizar actividades como la generación de reportes, la gestión de recursos ya sean humanos o materiales en la dirección de transporte de la UCI. La migración de los módulos en cuestión contribuye a corregir deficiencias presentadas por la solución que los antecede, dentro de la que se destaca la actualización de las tecnologías de desarrollo. El perfeccionamiento de esta solución 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.

Se realizaron pruebas de rendimiento para comprobar que los nuevos módulos mejoraron respecto al de sus antecesores. La implementación de los módulos Combustible y Persona para el nuevo Sistema Orbita contribuirá a mejorar el rendimiento del sistema completo debido a las actualizaciones que incluyen las tecnologías que conforman la arquitectura de referencia en PHP Bosón.

Palabras clave: mantenimiento, migración,perfeccionamiento, rendimiento.

Índice

Índice de Tablas

Índice de Figuras

Introducción

En la actualidad existe un gran avance tecnológico que ha posibilitado que a través de sistemas informáticos sean realizadas labores que anteriormente se hacían de forma manual. Debido a este avance tecnológico está surgiendo una tendencia por parte de las organizaciones de informatizar sus procesosen aras de tener la información centralizada, controlada y que sea de fácil acceso, contribuyendo en la toma de decisiones de los usuarios del sistema. En nuestro país la informatización de la sociedad cubana es una proyección que está expresada en los Lineamientos de la Política Económica y Social, aprobados en el Sexto Congreso del Partido Comunista de Cuba para impulsar la economía de nuestro país(MINCOM, 2016).

La Universidad de las Ciencias Informáticas (UCI) como apoyo a la política de informatización de la sociedad lleva a cabo en sus centros el desarrollo de sistemas que sean capaces de informatizar la supervisión y control de los procesos de diversas empresas de nuestro país. En la facultad 3 de la UCI se encuentra el Centro de Informatización de Entidades (CEIGE) el mismo se encarga del desarrollo de productos y servicios asociados a la gestión de entidades, contribuyendo a la formación integral de profesionales que respondan a las necesidades del progreso científico-técnico y socioeconómico del país.

Dentro del centro se desarrolló el Sistema de Control de Flota y Mantenimiento(Orbita) de la Dirección de Transporte de la UCI, el mismo se encarga de gestionar los procesos relacionados con el mantenimiento en una base de transporte, se compone por 7 módulos Planificación, Vehículo, Ejecución, Combustible, Persona, Estructura y Nomencladores estos 2 últimos siendo de configuración.

El módulo Combustible del Sistema Orbita, se desarrolló con el objetivo de mejorar el proceso control y análisis de índices de consumo de combustible de la dirección de transporte de la UCI, debido a que la información relacionada con este proceso se realizaba de forma manual en formato duro y a través de documentos Excel lo cual traía como consecuencia que se cometieran errores, afectando esto la toma de decisiones del personal que trabaja en el área de portadores energéticos. El módulo mantiene el control sobre la asignación de dicho recurso a los vehículos de la universidad(Saavedra Ayrado, et al., 2013).

Puesto a que la información relacionada a los recursos humanos era gestionada de forma manual, pues no existía un control sobre los datos de cada trabajador. Se crea otro de los módulos del sistema, el de Persona, donde se registra toda la información asociada a las personas para la gestión de los recursos humanos, tomando dicha información para elaborar el expediente de cada trabajador. El módulo permite llevar el control de las herramientas, las sanciones y las categorías de licencias en el caso de los choferes, asociadas al trabajador. El sistema brinda una serie de reportes que permiten el monitoreo de las personas como es el Relación de personas que se les vence el chequeo médico o Relación de choferes que deben pasar la escuela de recalificación. Otro de los objetivos de su creación fue contribuir a lograr su integración con el resto de los módulos del área de Transporte (UCI, 2014).

Con la implementación de ambos módulos se contribuyó al ahorro de recursos, como el tiempo en el cual se llevaba a acabo los procesos por parte de los clientes, facilitándoles su trabajo con la realización de una serie de reportes que le posibilitaron controlar la información referente a los datos de los trabajadores y el consumo del combustible asociado a los vehículos de la UCI.

La tecnología que se empleó para el desarrollo del Sistema Orbita es el marco de trabajo Sauxe en su versión 2.0. El mismo está basado en Zend Framework en su versión 1.2, utiliza como framework de persistencia Doctrine en su versión 1.2.2(Osorio, 2013) y ExtJS en su versión 3.4 en la capa de presentación, a petición del cliente. ExtJS carga todas las bibliotecas javascript al inicio atentando con el rendimiento del sistema y su licencia no es completamente libre, para ser comercializado debe de pagar su uso (Villa., 2016).

Otra desventaja de ExtJS es que se hace difícil hacer que el servidor haga push[1] de información desde el servidor web hacia el navegador, haciendo que el cliente tenga que constantemente hacer pooling[2] para obtener datos actualizados del servidor (Rosas, 2016).

[1] Operación de inserción.

[2] ***Polling*** es una forma de control en redes de área local, según la cual la unidad central de procesamiento pide, de acuerdo con una programación determinada a cada puesto de trabajo conectado a la red, si ha de enviar alguna información. El término proviene del inglés *poll*, que significa «sondeo». También se puede entender como el sondeo que realiza un servidor para comprobar el estado de cada terminal en una red.

En la capa de negocio esutilizado Zend Framework 1.2 lo que tiene como desventaja asociada según (Chase, 2006):

- Operaciones CRUD (Create, Retrieve, Update y Delete) asociadas a patrón Active Record.
- El mapeado de objetos a bases de datos relacionales (ORM).
- Estructura por defecto para aplicaciones (scaffolding).
- Almacenamiento de logs de funcionamiento del framework.
- Integración con otras herramientas a través de plugins.

En la capa de acceso a datos hace uso de Doctrine v 1.2.2, a pesar de ser esta una versión estable al manejar gran flujo de información se obtiene un rendimiento reducido en la aplicación. Por ejemplo, al realizar una consulta al sistema de base de datos, el sistema tendrá que: convertir la consulta al SQL del proveedor usado; enviar la consulta; leer registros, limpiarlos y convertirlos a objetos.

Al usar este ORM en sistemas como el Marco de Trabajo (MT) Sauxe con una alta complejidad en la base de datos el ORM también se hace complejo. Debido a la forma en que se maneja la herencia, las consultas son excesivamente grandes y complejas, provocando demoras en la implementación de los requisitos y la obtención de la solución final(Osorio, 2013).

También el uso de los llamados métodos mágicos, hace que cuando se presenta algún error sea muy difícil tracear[3] el código. Otro inconveniente que presenta esta versión de Doctrine es que al liberarse la versión 2 se deja de dar soporte a la versión 1 trayendo como consecuencia que no se corrijan vulnerabilidades en el código ya sean de seguridad, de rendimiento o de escalabilidad(Osorio, 2013).

Para validar los problemas que presentan en las herramientas antes mencionadas y como contribuyeron al deterioro del rendimiento del Sistema Orbita. Se realizó una encuesta (ver Anexo:3 Encuesta) a grupo de 4 especialistas que intervinieron en el desarrollo del Sistema Orbita, la cual arrojó como resultados:

- 100% de los encuestados alegan que el sitio tiene una respuesta rápida.

[3]Método en el cual se revisa la ejecución del código paso a paso.

❖ 75% de los encuestados opina que el tiempo de respuesta de las páginas de inicio relacionada con los módulos Combustible y Persona es de 0 a 12 segundos.

❖ 75% de los encuestados opina que el tiempo de respuesta de las páginas que hacen consulta a la base de datos es de 8 a 12 segundos.

❖ 100% de los encuestados opina que existe poca documentación relacionada con la arquitectura y el marco de trabajo Sauxe.

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, 2000-2016):

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

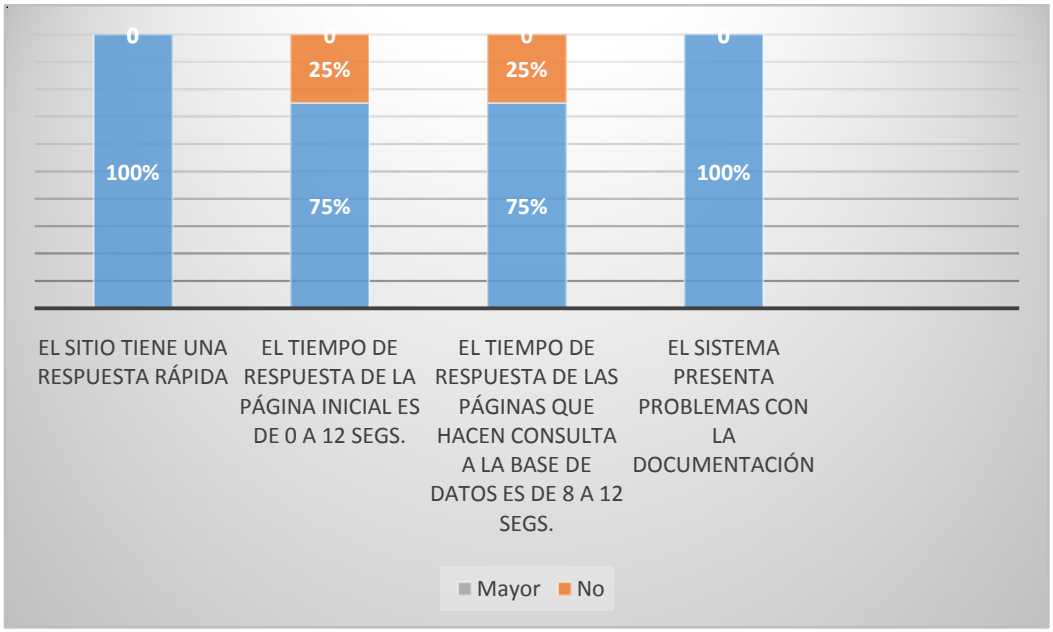


Figura 1 Resultados de la encuesta a los especialistas del sistema Orbita.

Atendiendo a estos resultados obtenidos en la encuesta se realizaron pruebas al sistema actual con la herramienta Jmeter. Dicha herramienta permite realizar pruebas unitarias para conexiones de bases de datos con JDBC, FTP, LDAP, Servicios web, JMS, HTTP y conexiones TCP genéricas(Apache JMeter™, 2016).

Los resultados (Ver Anexo:1 Resultados de la herramienta jmeter al Sistema Actual) de aplicar la prueba de carga y estrés con herramienta anterior fueron que para 100 usuarios conectados y 13900 peticiones al servidor el tiempo mínimo de respuesta fue 3 milisegundos y el máximo 6636 milisegundos, el 90% de las páginas respondió en 596 milisegundos, y el rendimiento fue 176,1 segundos por cada petición. Finalmente, la velocidad del Sistema Orbita fue de 8.8 Kb/segundos.

Los resultados obtenidos mediante la encuesta y la prueba de rendimiento realizada al Sistema Orbita confirmaron que los tiempos de respuesta del mismo entorpecen el trabajo de los usuarios que lo utilizan. Además, entre las tecnologías utilizadas para desarrollar el sistema se encuentran tecnologías privativas, obsoletas y sin soporte, lo cual impiden realizar mejoras sobre las mismas. La poca documentación del marco de trabajo empleado dificulta la corrección de alguna de estas dificultades.

A partir de la problemática antes planteada se define como **problema a resolver**: ¿Cómo contribuir a mejorar el rendimiento de los módulos Persona y Combustible del Sistema Orbita?

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

Campo de acción: migración tecnológica de los módulos Persona y Combustible del Sistema Orbita.

Objetivo: Migrar los módulos Persona y Combustible del Sistema Orbita a la arquitectura de referencia en PHP Bosón, contribuyendo a mejorar el rendimiento del sistema.

Objetivos específicos:

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

2. Analizar la ingeniería de requisitos de los módulos Persona y Combustible del sistema Orbita.

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

4. Implementar los módulos persona y combustible del sistema Orbita, siguiendo las técnicas de programación estudiadas en la tecnología de desarrollo selecciona para la migración.

5. Validar la solución propuesta.

Métodos Científicos:

- **Métodos Teóricos**

Histórico-Lógico: Se centró el estudio en la gestión de los procesos Administración de Recursos Humanos y Control de Índices de Combustible desde un enfoque histórico lógico, estudiando el desarrollo de este proceso desde el surgimiento del sistema actual.

Analítico-Sintético: Se utilizó para el análisis de la bibliografía existente relacionada con el tema, extrayendo los elementos más importantes y necesarios para dar solución al problema existente, de manera que permita sintetizar todo lo obtenido relacionado con los estándares pertinentes y trabajos ya realizados anteriormente por expertos en el tema.

Modelación: Se pone en práctica cuando se realiza un diseño del producto final que se va a obtener, con características específicas descritas. La modelación constituye el elemento intermedio entre el investigador y el objeto de investigación.

- **Métodos empíricos**

La entrevista: es una técnica de recopilación de información mediante una conversación profesional, con la que además de obtener información acerca de lo que se investiga, tiene importancia desde el punto de vista educativo; los resultados a lograr en la misión dependen en gran medida del nivel de comunicación entre el investigador y los participantes en la misma.

El trabajo está estructurado de la siguiente forma:

Capítulo 1: Fundamentación teórica. Se definen elementos teóricos que sustentan la investigación y se realiza un análisis sobre los procesos Persona y Combustible en el Sistema Orbita. También se definen las herramientas y lenguajes a utilizar en la implementación.

Capítulo 2: Análisis y diseño. Se analizan las historias de usuario del sistema actual y especifica los requisitos funcionales de la propuesta de solución. Además, abordan los productos de trabajo generados de los procesos los cuales muestran una visión clara del producto.

Capítulo 3: Implementación y validación. En este capítulo se abordan los elementos de estándares de codificación y la implementación, utilizado en el lenguaje de programación PHP. Se migran los módulos para darle solución a la situación problemática planteada. Se valida que la propuesta de solución cumple con el objetivo planteado. Además, son presentados los resultados de aplicar pruebas de caja negra, caja blanca, aceptación y rendimiento a la solución.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

El presente capítulo muestra las características principales de los módulos Combustible y Persona del Sistema Orbita. En el mismo se explicarán conceptos asociados a la investigación que servirán como base para una mejor comprensión del sistema. Además, se mostrará la justificación de la selección de cada una de las herramientas, metodología, lenguajes de desarrollo, servidor web, arquitectura de referencia y gestor de base de datos utilizados para el desarrollo de la propuesta de solución.

1.2 Sistema de gestión de flota y mantenimiento (Orbita)

El Sistema Orbita desarrollado por CEIGE de la UCI, permite gestionar los módulos relacionados con la gestión del mantenimiento que se desarrolla en cualquier empresa que cuente con una flota de vehículos. Este sistema fue desarrollado basado en los principios de independencia tecnológica utilizando software libre. Mediante su uso se puede gestionar toda la información de los vehículos, dígase asignaciones a las diferentes áreas de la empresa, accidentes, órdenes de trabajo, solicitudes de mantenimiento, control de hojas de ruta, control de combustible, autorizo de parqueo, día de la técnica control de neumáticos y baterías, todo esto de forma centralizada, eficiente y segura (UCI, 2014).

1.2.1 Proceso de gestión de combustible en el sistema Orbita

Se entiende por gestión de combustible al diseño y la puesta en práctica de un sistema de control, supervisión y seguimiento del consumo de combustible global e individual de los vehículos de una flota de transporte. La gestión de combustible permite aprovechar de la manera más rentable cada litro de combustible adquirido, contribuyendo con ello no sólo a la economía de la empresa, sino también al ahorro energético y a la conservación del medio ambiente. Una adecuada gestión de combustible(Saavedra Ayrado, et al., 2013) está asociada a:

- Una adecuada planificación de rutas y vehículos.
- La utilización de técnicas de conducción eficientes.
- Un correcto mantenimiento de los vehículos.
- La calidad del servicio prestado al cliente.

La gestión del combustible en la UCI se realiza en diferentes niveles, un nivel externo que incluye la solicitud y entrega del combustible a la institución por parte del Ministerio de Educación Superior y uno interno con el objetivo de distribuir el combustible por las diferentes áreas de la universidad que tienen vehículos asignados.

El proceso que maneja la gestión del combustible en esta área se denomina Control y análisis de índices de consumo de combustible y comienza con la elaboración de los planes de combustible por el especialista en portadores energéticos. Este plan se realiza mensual y por tipo de combustible (Gasol, diésel y gasolina) en función del kilometraje previsto a recorrer de los vehículos y los índices de consumo de cada tipo de vehículo según el fabricante. Este plan de combustible es aprobado por el rector de la universidad y en función de esto se procede a realizar la asignación de combustible para cada vehículo. Esta asignación comienza con la entrega de los documentos anexo de combustible y certificación de combustible habilitado y kilómetros recorridos, documentos estos con los cuales el chofer se presenta en la caja para que le entreguen la tarjeta de combustible(Saavedra Ayrado, et al., 2013).

En función de toda la información mencionada anteriormente el especialista realiza el análisis de eficiencia de cada uno de los vehículos de la flota, teniendo en cuenta los estados (Correcto, Distorsionado mayor que 5 y Distorsionado menor que 5). Luego procede a realizar el cierre de consumo del mes de todo el parque vehicular teniendo en cuenta los indicadores establecidos en la 1era edición de la norma cubana llamada: Transporte automotor. Servicio de Transportación de pasajeros y cargas. Términos, definiciones, símbolos y métodos de cálculos, emitida en febrero del 2011 por la Oficina Nacional de Normalización (Normalización Oficina Nacional de Transporte automotor., 2011).

Específicamente de esta norma se van a tener en cuenta los indicadores definidos para la intensidad energética (cantidad de combustible que se requiere para producir una unidad de tráfico) y el rendimiento energético (tráfico que se genera con un litro de combustible en la explotación del vehículo(Saavedra Ayrado, et al., 2013).

1.2.2 Proceso de gestión de los recursos humanos en el sistema Orbita

En el área de Transporte del UCI el objetivo fundamental de la gestión de los recursos humanos es registrar toda la información necesaria para elaborar el expediente de cada persona, dígase datos

generales como: nombres, apellidos, carné de identidad, sexo, área, cargo que ocupa, sanciones impuestas, herramientas que utiliza, fecha del último certificado médico, teléfono, dirección particular entre otros. El proceso Administración de Recursos se encarga de manejar la información de las personas que están relacionadas con la dirección de transporte de la universidad. Además, permite hacer uso de reportes los cuales permiten un control más eficiente sobre las personas. Ejemplo el reporte de relación personas por área, el cual muestra un listado con todos los datos de las personas de un área seleccionada (López, 2014).

El módulo Persona del Sistema Orbita se relaciona específicamente con los módulos: Ejecución del Mantenimiento y Vehículo, ya que estos necesitan manejar la información de las personas en sus procesos. Demostrando así la integración del sistema actual (López, 2014).

1.3 Estudio sobre la migración de sistemas

La primera ley de la ingeniería de sistemas de Bersoft (E.H. Bersoff, 1980) dice “Sin importar en qué momento del ciclo de vida del sistema nos encontremos, el sistema cambiará y el deseo de cambiarlo persistirá a lo largo de todo el ciclo de vida”. Es un hecho destacado por investigadores y profesionales[4] que los sistemas informáticos deben evolucionar para adecuarse a los siempre cambiantes requisitos del entorno. Las estadísticas indican que entre el 65% y el 75% de las fuerzas vivas relacionadas con el mundo de la informática (McKee) sustentan que aproximadamente el 80% de los gastos totales del software (Yourdon, 1989) se dedican al mantenimiento del software existente (J. A. Carsí, 2002).

Idealmente, a la hora de introducir una modificación en una aplicación informática se debe seguir un proceso (por ejemplo el ISO/IEEE 12207 (IEEE, 1992)) en el cual sea identificada la razón del cambio,se analiza cómo afecta el cambio a introducir en la aplicación[5], para posteriormente realizar los cambios allí donde sean necesarios, finalizando con una exhaustiva fase de pruebas que demuestre que la modificación no ha introducido errores. Todo el proceso anterior debe estar bajo un

[4]J. A. Carsí, I. Ramos, J. Silva, J. Pérez, V. Anaya Departamento de Sistemas Informáticos y Computación Universidad Politécnica de Valencia.

[5]Tanto en los documentos de análisis, diseño, como en el código.

estricto control de cambios que recoja todos los documentos que se generan y almacenen las diferentes versiones (J. A. Carsí, 2002).

La finalidad de la migración es trasladar un sistema de información, a un nuevo ambiente operativo, conservando los datos originales y su funcionalidad. Para que el mantenimiento y su posterior adecuación a nuevas demandas no tengan mayor impacto en el cambio. Una migración es normalmente un proyecto de ingeniería de sistemas, que se le da el calificativo de crítico, esto es por la importancia de la información que va a manejar a través de los datos. Las aplicaciones migradas deben brindar al final la misma operatividad y eficacia que el entorno anterior, por la necesidad de hacer mínimo el impacto en todos los niveles de la empresa (VIVAS, 2015).

El estudio de bibliografías de referencia en esta área del conocimiento, nos permite asumir como definición de migración la que limita a que se realice el redesarrollo completo del sistema, usando todos los precedentes de los cuales dispone como requisitos y diseños. Al hablar de una migración se hace mención a la necesidad de modificar un sistema a una nueva plataforma conservando sus funcionalidades y que el impacto en esta operación sea el mínimo en todos los niveles de la empresa.

1.4. Rendimiento

El rendimiento como atributo de calidad posibilita caracterizar el comportamiento de un sistema en su entorno de producción. Medir el rendimiento de una aplicación, le permite a las organizaciones tener una percepción de cuántos recursos deben dedicar para que el *software* realice sus funciones óptimamente o también, determinar cuánta carga de trabajo puede procesar el sistema por unidad de tiempo (Alvarez, 2013).

La IEEE define rendimiento como el grado en que un sistema o componente realiza sus funciones teniendo en cuenta determinadas restricciones como velocidad, precisión o uso de memoria (RAE, 2013). 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 transacción es por unidad de tiempo que procesa el sistema o por la cantidad de tiempo que se tarda en completar una transacción(Len, Paul, y Rick 2003).

En la norma ISO 9126-1 (Choque Aspiazu 2001) se utiliza la terminología eficiencia (efficiency), la cual se refiere a la cantidad de recursos utilizados por el sistema al proporcionar una funcionalidad requerida. Ejemplo: la cantidad de espacio en disco, memoria o ancho de banda utilizado ante una petición. Engloba los siguientes sub-atributos:

- Comportamiento en el tiempo (time behavior): Caracteriza los tiempos de respuesta para una capacidad de procesamiento determinada del sistema. Ejemplo: la tasa de transacción.

- Comportamiento de recursos (resource behavior): Caracteriza la cantidad de recursos utilizados y la duración de su uso en la ejecución de su función.

El Instituto de Ingeniería de Software(Collard 2005) (por sus siglas en inglés SEI) define que el rendimiento caracteriza la eficacia del servicio que provee el sistema. Se clasifican dentro de este atributo como requerimientos de rendimiento:

- Latencia[6] (latency): Intervalo de tiempo que transcurre desde que se produce un evento hasta que se ejecuta una respuesta del sistema. Este intervalo está dado por un tiempo de inicio (latencia mínima) y un tiempo final (latencia máxima).

- Capacidad de procesamiento (throughput): Define el número de respuestas de eventos que deben ser completadas totalmente en un intervalo de tiempo especificado.

- Capacidad (capacity): Es una medida de la cantidad de trabajo máximo que un sistema puede realizar. En términos de redes se le conoce como ancho de banda, lo cual usualmente se mide en megabits por segundos (Mbps). Una definición más práctica puede ser la máxima capacidad de procesamiento que se puede lograr sin violar los requerimientos de latencia especificados («HowManyThreadsi» 2016).

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

[6]Tiempo que transcurre entre un estímulo y la respuesta que produce, siendo un estímulo una petición al sistema y la respuesta, el servicio que provee el software para satisfacerla.

Teniendo en cuenta cada una de las taxonomías propuestas anteriormente para el rendimiento, se decide proponer la definición formal de rendimiento dada por el Instituto de Ingeniería de Software(Collard 2005) ,sobre la cual se acoge a la presente investigación y cuáles atributos serán evaluados para caracterizar el desempeño del sistema Orbita.

Rendimiento: Define el grado en que un sistema o componente realiza sus funciones teniendo en cuenta determinadas restricciones como la capacidad de procesamiento, el tiempo de respuesta y la utilización de los recursos(Collard, 2005).

❖ Capacidad de procesamiento: Define el número de peticiones que pueden ser procesadas totalmente en un intervalo de tiempo especificado o la cantidad de información procesada por unidad de tiempo. Usualmente se mide en transacciones por segundos (tps) o mensajes procesados por segundos (mps), aunque puede medirse también en términos de (Mb/s).

❖ Tiempo de respuesta o latencia: Intervalo de tiempo que transcurre desde que se realiza una petición hasta que se ejecuta una respuesta del sistema. Se mide en milisegundos (ms).

❖ Utilización de recursos: Caracteriza la cantidad de recursos utilizados y la duración de su uso en la ejecución de su función. Los recursos pueden ser memoria, procesador, disco y red.

1.5 Metodología, lenguajes y herramientas de desarrollo

1.5.1 Metodología de desarrollo

Una metodología de desarrollo de software es un conjunto de técnicas, herramientas, procedimientos y soporte documental que permite a los desarrolladores definir los elementos necesarios para la construcción de un nuevo producto de software. Mediante la metodología de desarrollo de software se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además, detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla (Rodríguez, 2015).

Se define como metodología a emplear la variación de la AUP-UCI (Agile Unified Process, por sus siglas en inglés) apoyándose en el modelo CMMI-DEV V1.3, la cual está definida para a la actividad productiva de la UCIpor su adaptabilidad a diversos proyecto (Rodríguez, 2015). La variación AUP-

UCI (Rodríguez Sánchez, 2014) indica que se mantiene la fase de Inicio, el resto de las fases se agrupa en una sola llamada Ejecución, y se adiciona una tercera fase Cierre. De estas etapas se estará haciendo énfasis en la fase de **Ejecución**. Dentro de esta fase se emplearán las disciplinas: Análisis y Diseño, Implementación y en Pruebas Internas. Se hará uso del **Escenario 4** de la metodología, por lo cual el método de encapsulación de requisitos es a través de **Historias de Usuario**(Rodríguez, 2015).

1.5.2 Bosón: arquitectura de referencia para PHP

Un gran porciento de los productos que se desarrollan en la UCI utilizan como lenguaje de programación PHP. Debido a que la infraestructura productiva de la universidad se encuentra separada en centros con dominios de aplicaciones diferentes, y que no existe una definición tecnológica para el desarrollo, ha traído como consecuencia una diversidad de tecnologías incluso utilizando el propio lenguaje PHP. Esto provoca que el nivel de complejidad para la integración de soluciones sea alto, así como la reutilización sea prácticamente nula. Actualmente en la UCI existen proyectos que al definir su arquitectura base suelen implementar los mismos requisitos, lo que trae como consecuencia la duplicación de esfuerzo y la dilatación de los cronogramas de desarrollo (Torres, 2016).

El desarrollo de la arquitectura de referencia PHP Bosón, haciendo uso de tecnologías libres garantiza que todas las soluciones que se desarrollen sobre PHP cumplan con la versatilidad necesaria para construir sistemas de distintos dominios y proporcione la capacidad de integración con un mínimo esfuerzo. Con el desarrollo de este proyecto se ahorra tiempo y recursos materiales en la construcción de soluciones informáticas, ya que provee un conjunto de componentes tecnológicos que son indispensables para la mayoría de los proyectos.

El hecho de utilizar como marco base a Symfony2 garantiza la presencia de una comunidad mundial que brinda soporte y actualizaciones a diario. Permite a los equipos de desarrollo centrarse en la implementación solo de la lógica de su negocio abstrayéndose de elementos como la gestión de la caché, el manejo de estructuras de datos, el registro de trazas, la seguridad de los sistemas, la gestión de excepciones, la definición de componentes visuales, el manejo de conceptos globales, la validación de datos y la gestión de componentes (Torres, 2016).

La arquitectura de referencia PHP Bosón materializa los requisitos comunes de las arquitecturas base para sistemas de gestión web a partir del desarrollo de los siguientes componentes:

Componente Caché: Permite a las aplicaciones que se están desarrollando en 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(Abraham Calas, 2016b).

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(Abraham Calas 2016).

Componente Excepciones: Componente que permite controlar de forma centralizada los diferentes tipos de excepciones y tratarlas según el tipo especificado. Garantiza además la internacionalización de los mensajes de las excepciones, el manejo declarativo del comportamiento, el mecanismo extensible de creación de nuevos tipos y la codificación(Abraham Calas, 2016).

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(Entidad Atributo Valor) para gestionar las estructuras lo cual disminuye la creación de grandes volúmenes de tablas destinadas a estos temas(Abraham Calas, 2016).

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(Abraham Calas, 2016).

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(Abraham Calas, 2016).

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(Abraham Calas 2016).

Componente Seguridad: Provee un mecanismo de autenticación haciendo uso del estándar abierto SAML[7] así como un mecanismo de autorización mediante la extensión del patrón RBAC[8]. Permite además la autenticación a partir de varias fuentes como base de datos, servidores LDAP, Facebook, Google, Linkedln, entre otras(Abraham Calas, 2016).

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(Abraham Calas, 2016).

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 HTML5, CS3, DHTML, JS, y mecanismos de comunicación asíncrona con el servidor como Ajax y Websockets(Abraham Calas, 2016).

1.5.3 Lenguajes de desarrollo

PHP v5.5

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. PHP es ampliamente utilizado para fines generales, aunque es especialmente adecuado para el desarrollo web y puede ser embebido en páginas HTML. PHP a partir de su versión 5 permite el uso del paradigma de programación orientada a objetos (POO) y también el de programación procedimental, o una mezcla de ambos. En la versión 5.5 se incluyeron nuevas características que mejoran el rendimiento y la seguridad de las aplicaciones desarrolladas, por ejemplo:

[7]Security Assertion Markup Language

[8]Control de Acceso con base en roles

❖ Se añadió la extensión OPcache: La caché de códigos de operación de Zend Optimiser+ se ha añadido a PHP como la nueva extensión OPcache. OPcache mejora el rendimiento de PHP almacenando códigos de byte de un script pre compilado en la memoria compartida, eliminando así la necesidad por parte de PHP de cargar y analizar scripts en cada petición(«PHP: New features - Manual» 2016).

❖ Según (Ortuondo ,2014) la mejora del rendimiento de PHP 5.5 se traduce en tiempos de carga y consumo de memoria RAM menores que en versiones anteriores. En las pruebas que se han hecho la mejora de rendimiento de PHP 5.5 respecto a PHP 5.3 suele ser del 20% y de un 5% respecto a PHP 5.4.

❖ Gestor de procesos FastCGI(FPM): Mejoras de rendimiento y más información en los log cuando se produce un error durante la invocación de una *Syscall*[9](Puertas ,2013).

Teniendo en cuenta la existencia de una amplia comunidad de programadores expertos en el lenguaje, la gran cantidad de bibliotecas orientadas a múltiples propósitos, la alta disponibilidad de marcos de desarrollo, y la orientación del presente trabajo a uno de estos marcos de trabajo, se concluye PHP como el lenguaje ideal para la implementación de la solución propuesta (Group y others 2012).

JavaScript v 1.8

JavaScript es un lenguaje de programación que se utiliza para crear páginas web dinámicas. Es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlo. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Responde a eventos en tiempo real. Este lenguaje permite interactuar con el navegador de manera dinámica y eficaz, proporcionando a las páginas web dinamismo y vida («- JavaScript - The Definitive Guide.pdf» 2016). Se utilizó este lenguaje en su versión 1.8.

HTML v5

[9] Llamada del sistema

El lenguaje de marcas HTML (del inglés: HyperText Marckup Language, en español Lenguaje de Marcado de Hipertexto), se utiliza para definir las páginas web y permite describir hipertexto, es decir, texto presentado de forma estructurada y agradable, con enlaces que conducen a otros documentos o fuentes de información relacionadas, y con inserciones multimedia (gráficos, fotos, sonido, entre otros). Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web, es decir, se utiliza para crear las páginas web y les indica a los navegadores cómo deben mostrar el contenido(«The HTML sourcebook » 2016).

El lenguaje HTML contiene dos partes:

- Contenido, que es el texto que se verá en la pantalla de un ordenador.
- Etiquetas y atributos que estructuran el texto de la página web en encabezados, párrafos, listas, enlaces, entre otros y normalmente no se muestra en pantalla.

XML

El Lenguaje de Etiquetado Extensible (Extensible Markup Language, XML por sus siglas en inglés) es muy simple, pero estricto, pues juega un papel fundamental en el intercambio de una gran variedad de datos. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones. Dicha tecnología es un conjunto de módulos que ofrece servicios útiles a las demandas más frecuentes por parte de los usuarios. XML sirve para estructurar, almacenar e intercambiar información. Siendo su función principal describir datos y no mostrarlos como es el caso de HTML(Bray et al. 1998).

Symfony2

Es un marco de trabajo para el desarrollo de aplicaciones en el lenguaje PHP. Fue creado por el francés Fabien Potencier, quien es cofundador de la compañía Sensio Labs. Este marco de trabajo permite la utilización de componentes genéricos, evitando la implementación de tareas comunes y enfocando a los desarrolladores en los verdaderos desafíos de la aplicación. Symfony2 introduce el concepto de bundle, este concepto se refiere a un conjunto de archivos (PHP, hojas de estilo, JavaScript, o cualquier otro) que implementan una funcionalidad específica(Eguiluz 2013).

Un bundle viene a ser lo que un plugin en otro software, pero con la diferencia de que todo en Symfony2 es un bundle, incluyendo el código del marco de trabajo, así como el código de la

aplicación desarrollada. Esto permite la utilización de funciones empaquetadas dentro de bundles de terceros y la distribución de bundles propios, así como la selección de las prestaciones a habilitar en la aplicación para optimizarla en la manera deseada(Eguiluz, 2013).

Entre las características de Symfony2 se pueden citar («Symfony, High Performance PHP Framework for Web Development» 2016):

- **Rápido y menos abarcador**: Symfony2 fue concebido desde el inicio para ser rápido y favorecer su desempeño. A manera de comparación Symfony2 es alrededor de 3 veces más rápido que en su versión 1.4 o que Zend Framework 1.10, mientras toma hasta 2 veces menos memoria(« Manual de Symfony2 en Español» 2016).

- **Flexibilidad ilimitada**: Su inyector de dependencias y su manejador de eventos, lo hacen enteramente configurable, con cada una de las partes siendo completamente independientes. Es un marco de trabajo 3 en 1 que puede tener («Manual de Symfony2 en Español» 2016):

  - Todas sus funciones: para desarrollar una aplicación compleja y cuando se necesiten muchas funcionalidades.

  - Parte por parte: construir el marco de trabajo parte a parte según las funcionalidades que se necesiten.

  - Micro marco de trabajo: Symfony2 también puede ser usado para implementar una funcionalidad específica dentro de un proyecto, sin tener que re implementar todo ni tener que instalar el marco de trabajo completo, sino solo la parte necesitada.

- **Expandible**: desde la pequeña parte que compone el núcleo en sí misma, todo se presenta como un bundle en Symfony2. Cada bundle intenta agregar funcionalidad al marco de trabajo, y cada bundle puede ser usado en otro proyecto(«Documentación de Symfony2 — Manual de Symfony2 en Español» 2016a).

- **Estable y sostenible**: desarrollado por Sensio Labs, la mayor parte de las versiones son soportadas por 3 años por la compañía(«Documentación de Symfony2 — Manual de Symfony2 en Español» 2016a).

❖ **Confort al desarrollar**: como un entorno altamente funcional, Symfony2 también garantiza un cierto nivel de confort para los desarrolladores. Ocupándose de ciertas tareas poco placenteras Symfony2 permite a los desarrolladores centrarse en los hitos de la aplicación(« Manual de Symfony2 en Español» 2016).

❖ **Facilidad de uso**: completamente flexible para satisfacer las necesidades de profesionales y usuarios avanzados, también Symfony2 es muy accesible. Documentación abundante, soporte en comunidad y profesional, y buenas prácticas "embebidas" dentro del marco de trabajo (buenas prácticas nativamente aplicadas, que no es necesario tener en cuenta y ni siquiera tener pleno conocimiento de ellas) permiten a un principiante sentirse a gusto muy rápido(«Documentación de Symfony2 — Manual de Symfony2 en Español» 2016b).

Este marco de trabajo cada día gana más seguidores dentro de la UCI dadas sus amplias ventajas para el desarrollo de aplicaciones complejas, por eso se escoge como destino de la solución propuesta.

CSS 3.0

La hoja de estilos en cascadas (CSS, por sus siglas en inglés) es un lenguaje creado para controlar la presentación de los documentos electrónicos definidos con HTML. CSS es la mejor forma de separar los contenidos de su presentación y optimiza el trabajo para la creación de páginas web complejas. Es un lenguaje que trabaja junto a HTML para proveer estilos visuales a los elementos del documento como tamaño, color, fondo y borde (Gauchat, 2012). CSS3 es seleccionado por las ventajas y facilidades que aporta a la hora de trabajar las plantillas que se van a mostrar en la aplicación, permitiendo ahorro de código y tiempo a los desarrolladores en la elaboración de varios trucos para lograr confeccionar las interfaces como son requeridas por el cliente (CSS3, 2015).

Motor de Plantillas Twig

Es un motor de plantillas, que posee un ambiente amigable para el diseñador y el desarrollador, permitiendo añadir funcionalidades a los entornos de plantillas. Posee una sintaxis corta y concisa, similar a la de otros lenguajes de programación. Además implementa un mecanismo de herencia de plantillas, para acelerar el rendimiento del sistema que se desarrolla (Pacheco, 2015)

Este lenguaje permite compilar las plantillas hasta código PHP optimizado, lo que proporciona rapidez durante la implementación. Garantiza la seguridad del código de las plantillas ya que posee un modo de recinto de seguridad para evaluar el código de plantilla que no es confiable. Permite al desarrollador definir sus propias etiquetas y filtros personalizados, garantizando flexibilidad a la aplicación. Posibilita la depuración de las plantillas creadas, mediante mensajes de error con el nombre del archivo y el número de línea donde se produjo el problema (Pacheco, 2015).Además, permite agilizar la construcción de plantillas de la vista, brindándoles estructuración y velocidad de ejecución del lado del cliente. Este lenguaje permite compilar las plantillas hasta código PHP optimizado, lo que proporciona rapidez durante la implementación (Twig, 2016).

Doctrine

Doctrine es capaz de convertir aproximadamente el 70-80% de la información asignada basándose en los campos, índices y restricciones de clave externa. Doctrine no puede descubrir asociaciones inversas, tipos de herencia, entidades con claves externas como claves principales u operaciones semánticas en asociaciones tales como eventos en cascada o ciclo de vida de los eventos. Posteriormente, será necesario algún trabajo adicional sobre las entidades generadas para diseñar cada una según tus características específicas del modelo de dominio(« Doctrine 2 ORM 2 documentation» 2016).

Doctrine es un marco de trabajo que proporciona persistencia transparente de objetos PHP, el cual se sitúa en la parte superior de una capa de abstracción de base de datos (DBAL, por sus siglas en inglés, Database Abstraction Layer). Una de sus principales características es que cuenta con la opción de escribir las consultas de base de datos en un lenguaje de consulta estructurado (SQL, por sus siglas en inglés Structured Query Language) orientado a objetos, llamado lenguaje de consulta Doctrine (DQL[10], por sus siglas en inglés, Doctrine Query Language) (Doctrine, 2016). Viene integrada por defecto con Symfony2 y presenta entre sus características principales las siguientes:

- Generación automática del modelo.

[10]DQL: Es un lenguaje creado para ayudar al programador a extraer objetos de la base de datos. Es importante considerar su uso para mejorar el rendimiento.

- Posibilidades de trabajar con YAML.
- Relaciones entre entidades.

1.5.4 Herramientas de desarrollo

Apache server 2.4: Apache es un servidor web de software libre, cuyo objetivo es servir o suministrar páginas web (en general, hipertextos) a los clientes web o navegadores que las solicitan, funciona sobre cualquier plataforma, permite que otros ordenadores vean la web mediante un navegador. Se integra perfectamente con varias tecnologías, lenguajes, plataformas, bases de datos(«The Apache Software Foundation» 2016). Esta versión presenta varias mejoras en función del rendimiento del programa, ejemplo:

Cargas de MPM en Tiempo de Ejecución: Múltiples MPMs ahora se pueden construir como módulos dinámicos de forma que pueden ser cargados en tiempo de compilación. El MPM de elección se puede configurar en tiempo de ejecución a través de LoadModule(«The Apache Software Foundation» 2016).

Variables de los Archivos de Configuración: Ahora es posible definir variables en la configuración, lo que permite una representación más clara si el mismo valor se utiliza en muchos lugares en la configuración(«The Apache Software Foundation» 2016).

Reducción del Uso de Memoria: A pesar de muchas de las nuevas características, 2.4.x tiende a usar menos memoria que la versión 2.2.x(«The Apache Software Foundation» 2016).

Analizador de Expresión de Uso General: Un nuevo analizador de expresiones permite especificar condiciones complejas utilizando una sintaxis común en directivas como SetEnvIfExpr, RewriteCond, Header, <If>, entre otras(«The Apache Software Foundation» 2016).

KeepAliveTimeout en Milisegundos: Ahora es posible especificar KeepAliveTimeout enmilisegundos(«The Apache Software Foundation» 2016).

Por las características anteriormente expuestas se decide utilizarlo para el desarrollo del módulo Combustible y Persona del sistema Orbita.

PhpStorm 9

PhpStorm es perfecto trabajando con Symfony, Drupal, WordPress, Zend Framework, Laravel, Magento, CakePHP, Yii, y otros marcos de trabajo. Todas las herramientas de PHP. El editor actual de toma el código y entiende su estructura, soporta las características de PHP tanto para proyectos modernos como antiguos. Proporcionalos mejores completamientos de código, refactorizaciones, la prevención enla marchade error, y más(«PhpStorm IDE :: JetBrains PhpStorm» 2016)Entre sus características se encuentran :

- **Tecnologíasde front-endincluido**

Aprovecha al máximolastecnologíasde front-endde vanguardia, como HTML5, CSS, Sass, Menos, Stylus, CoffeeScript, Emmety JavaScript, conrefactorizaciones, depuración y pruebas de unidaddisponible.Vealos cambiosal instanteen el navegadorgraciasa LiveEdit.Built- en las herramientas de desarrollo.

- **Realizamuchas tareas rutinariasdesde elIDE**

Gracias a la integraciónde sistemasde control de versiones, el apoyo a la implementación remota, bases de datos/SQL, herramientas de líneade comandos, Vagrant, Compositor, REST Client, y muchas otras herramientas. PhpStorm = WebStorm + PHP + DB/SQL. Todas las características de WebStorm se incluyen en PhpStorm, y soporte completo para PHP y bases de datos de soporte / SQL se añaden en la parte superior(«PhpStorm IDE :: JetBrains PhpStorm» 2016)

- **Asistencia de codificación inteligente**

Cientos de inspecciones se encargan de verificar el código a medida que escribe, el análisis de todo el proyecto. Apoyo PHPDoc, código de arreglista y formateador, soluciones rápidas, y otras características ayudan a escribir código limpio que es fácil de mantener(«PhpStorm IDE :: JetBrains PhpStorm» 2016).

- **Fácilde depurary hacer probar.**

PhpStormes famoso por suDepuradorVisual, proporcionando una visiónextraordinariaenlo que sucede ensu aplicacióna cada paso.Funciona conXdebugyZendDebugger,y se puede utilizar tanto local como remotamente. Pruebas unitariasconPHPUnit, BDDconBehaty la integraciónde perfilestambién están disponibles(«PhpStorm IDE :: JetBrains PhpStorm» 2016)

Visual Paradigm 8.0

Visual Paradigm es una herramienta CASE. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas.

Brinda numerosas ventajas tales como: cuenta con una licencia gratuita y comercial, es multiplataforma, contiene soporte para varios idiomas y es fácil de instalar y actualizar. («Software Design Tools for Agile Teams, with UML, BPMN and More» 2016).

Entre sus principales características se encuentran:

- Entorno de creación de diagramas para UML.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo. Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs.
- Generación de bases de datos.
- Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Importación y exportación de ficheros XML.

Se selecciona Visual Paradigm como herramienta CASE, para la construcción de diagramas y modelos en la definición del proceso de desarrollo.

Sistema gestor de base de datos PostgreSQL (9.2.4)

Para implementar los módulos Combustible y Persona del Sistema Orbita se mantiene la utilización del sistema de gestión de bases de datos PostgreSQL en su versión 9.2.4. Es un sistema gestor de base de datos relacional orientado a objetos de software libre que puede ser utilizado en los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows(«The PostgreSQL Global Development Group» 2016).

Navegador Mozilla Firefox 43.0

Mozilla Firefox es un navegador de Internet libre y de código abierto, desarrollado para Microsoft Windows, Android, OS X y GNU/Linux. Coordinado por la Corporación Mozilla y la Fundación Mozilla. Usa el motor Gecko para renderizar páginas web, el cual implementa actuales y futuros estándares web. Es usado para visualizar páginas web. Incluye corrector ortográfico, búsqueda progresiva, marcadores dinámicos y un sistema de búsqueda integrado que utiliza el motor de búsqueda que desee el usuario. Además, se pueden añadir funciones a través de complementos desarrollados por terceros (Mozilla Hispano, 2016).

1.6 Patrones

1.6.1 Patrones de diseño

Un patrón es una solución probada que se puede aplicar con éxito a determinados tipos de problemas que aparecen repetidamente en el desarrollo de sistemas software. Puede ser empleado muchas veces, en diferentes contextos, sin tener que duplicar el diseño. Se trata de un elemento de diseño que puede ser reutilizado (Garis, Riesco, y Montejano 2006).

Los diseñadores expertos no resuelven los problemas desde el principio, reutilizan soluciones que han funcionado en el pasado. Se encuentran patrones de clases y objetos de comunicación recurrentes en muchos sistemas orientados a objetos. Estos patrones resuelven problemas de diseño específicos y hacen el diseño flexible y reusable. Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptado para resolver un problema de diseño general en un contexto particular (Addison Wesley Helm, y otros, 1995).

La utilización de patrones de diseño, permite ahorrar grandes cantidades de tiempo en la construcción de software. El producto obtenido es más fácil de comprender, mantener y extender. Existen versiones ya implementadas de estas funcionalidades comunes (marcos de trabajo) que permiten centrarse en desarrollar solo la funcionalidad específica requerida por cada aplicación y que, además, dan mejor imagen de profesionalidad y calidad. Los patrones de diseño se agrupan en dos grandes categorías:

- GRASP[11]
- GOF[12]

Los primeros describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Dentro de este grupo de patrones se encuentran los siguientes: Experto, Creador, Bajo Acoplamiento, Controlador y Alta Cohesión. Los patrones de diseño GOF son 23 y se clasifican según su propósito en Creacionales, Estructurales, Comportamiento y según su ámbito en Objeto y de Clase (Patrones, 2010).

1.6.2 Patrones de arquitectura

Los patrones arquitectónicos, o patrones de arquitectura, también llamados arquetipos ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. En comparación con los patrones de diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor (Pressman, 2010).

MVC

El patrón arquitectónico Modelo Vista Controlador (Model View Controller, MVC por sus siglas en inglés) es un estilo arquitectónico que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. El estilo arquitectónico MVC se ve frecuentemente en aplicaciones web, donde la vista es la interface de usuario y el código es el que provee de datos dinámicos a la página; el modelo es el Sistema de Gestión de Base de Datos y la lógica de negocio; y el controlador es el responsable de recibir los eventos de entrada desde la vista (MVC, 2010).

El Modelo es el responsable de:

[11]GRASP: Acrónimo de “General Responsibility Assignment Software Patterns” en español Patrones generales de software para asignación de responsabilidades.

[12]Gang of Four, en español Banda de los Cuatro, formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides.

❖ Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.

❖ Define las reglas de negocio (la funcionalidad del sistema).

❖ Lleva un registro de las vistas y controladores del sistema.

❖ Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos puedan producir un agente externo.

El Controlador es responsable de:

❖ Recibir los eventos de entrada.

❖ Contiene reglas de gestión de eventos, estas acciones pueden suponer

❖ peticiones al modelo o a las vistas.

Las Vistas son responsables de:

❖ Recibir datos del modelo mediante el controlador y las muestras al usuario.

❖ Tienen un registro de su controlador asociado.

❖ Pueden dar el servicio de "Actualización ()", para que sea invocado por el controlador o por el modelo cuando es un modelo activo (MVC, 2010).

Un Framework MVC te ayuda a controlar los recursos del servidor, evitando Bugs que puedan repercutir en el rendimiento del sistema, por ejemplo, muchas veces olvidamos cerrar conexiones a la base de datos, sobrecargando el servidor.

1.7 Validación y Verificación

Los objetivos de las actividades de verificación y validación son valorar y mejorar la calidad de los productos del trabajo generados durante el desarrollo y modificación del software. Los atributos de la calidad deben ser la corrección, la perfección, la consistencia, la confiabilidad, la utilidad, la eficacia, el apego a los estándares y la eficacia de los costos totales. Según la IEEE (IEEE, 2015) se define las pruebas de software como una actividad en la que un sistema o un componente es ejecutado bajo condiciones especificadas. El objetivo es diseñar una serie de casos de pruebas que tengan una alta probabilidad de encontrar errores. Para ello se aplican las técnicas de pruebas del software, las cuales facilitan una guía sistemática para diseñar pruebas que comprueben la lógica interna de los componentes de software y verifiquen los dominios de entrada y salida del programa para descubrir errores en la funcionalidad, el comportamiento y rendimiento (Pressman, Roger, 2000).

1.7.1 Prueba de Caja Negra

Las pruebas de caja negra son aquellas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del funcionamiento del sistema sin tener en cuenta la estructura interna del software (Peña, 2010). 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.

Según(Pressman, Roger, 2002), para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

- Técnica de la Partición de Equivalencia: divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- Técnica del Análisis de Valores Límites: prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- Técnica de Grafos de Causa-Efecto: permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

En la presente investigación se escoge dentro del método de Caja Negra la técnica de la Partición de Equivalencia que permite examinar los valores válidos e inválidos de las entradas existentes en el software. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así en número de casos de pruebas que hay que desarrollar.

1.7.2 Prueba de Caja Blanca

El método de prueba de caja blanca, denominado a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los

casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero de software puede obtener casos de prueba que (Pressman, Roger, 2002):

1. Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
2. Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites y con sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez.

Para ejecutar este tipo de pruebas existen diversas técnicas entre ellas se encuentran la prueba de camino básico, la prueba de condición, la prueba de flujo de datos, prueba de bucles (Pressman, Roger, 2002). En esta investigación se utilizará la técnica del camino básico: técnica que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución.

1.7.3 Prueba de Aceptación

La prueba de aceptación es generalmente desarrollada y ejecutada por el cliente o un especialista de la aplicación y es conducida a determinar cómo el sistema satisface sus criterios de aceptación validando los requisitos que han sido levantados para el desarrollo, incluyendo la documentación y procesos de negocio. Está considerada como la fase final del proceso para crear una confianza en que el producto es el apropiado para su uso en explotación (Pressman, Roger, 2000).

Una prueba de aceptación puede ir desde un informal caso de prueba hasta la ejecución sistemática de una serie de pruebas bien planificadas. De hecho, la prueba de aceptación puede tener lugar a lo largo de semanas o meses, descubriendo así errores acumulados que pueden ir degradando el sistema (Pressman, 2010).

1.7.4 Prueba de Rendimiento

Las pruebas de rendimiento son realizadas para evaluar el cumplimiento por parte de un sistema o de sus componentes si cumple los requisitos de rendimiento especificados. Con ellas se puede determinar cómo responde un sistema ante una cierta carga, así como validar otros atributos

relacionados con la calidad, como pueden ser la escalabilidad o el uso de recursos entre otros. Dentro de las pruebas rendimiento se encuentran las pruebas de carga y las de estrés (Ávila, 2013).

Prueba de carga

Una prueba de carga se ejecuta para comprender el comportamiento de una aplicación ante una carga determinada (Ávila, 2013). Modela el uso previsto de un programa de softwaresimulando varios usuarios que obtienen acceso al programa al mismo tiempo (MSDN 2010).Las pruebas de carga, se relacionan directamente con las pruebas de estrés siendo su principalobjetivo probar un nivel específico de capacidad de concurrencia de usuarios generalmenteespecificados en los requisitos de una aplicación de software.

En el siguiente fragmento Boris Beizer[13] describe de forma muy clara qué significa prueba de carga:
"La prueba de carga somete a un sistema a una carga estadísticamente representativa. [...] la carga se varía de un mínimo (cero) hasta el nivel máximo que el sistema puede sostener sin quedar sin recursos o tener transacciones sufriendo (dependiendo de la aplicación) demoras excesivas. Otro uso del término de prueba de carga es como una prueba cuyo objetivo es determinar la carga máxima sostenible que el sistema puede manejar. En este uso, prueba de carga es meramente comprobar con las tasas más altas de llegada de transacciones." Es importante destacar que en una prueba de carga el sistema debe responder al nivel de carga sometido de forma adecuada, en otras palabras, según sus requerimientos. Si el sistema no responde de forma esperada, los resultados obtenidos no corresponden a una prueba de carga, sino a una de estrés (IGNACIO ABEL, 2005).

Prueba de estrés

Las pruebas de estrés son utilizadas normalmente para someter a la aplicación al límite de su funcionamiento, mediante la ejecución de un número de usuarios muy superior al esperado. Esta prueba tiene como finalidad determinar la robustez de una aplicación cuando la carga es

[13]Boris Beizer es un ingeniero de software y escritor estadounidense. Ha escrito numerosos libros y artículos sobre temas como la arquitectura del sistema y pruebas de software.

extrema y ayuda a administradores a determinar si la aplicación se comportará correctamente en dichas situaciones (Ávila, 2013)

La prueba de estrés determina el punto en que un sistema comienza a brindar un desempeño inaceptable. Este punto puede ser llamado punto de quiebre. La idea es estresar a un sistema al punto de quiebre para encontrar errores que podrían ser potencialmente perjudiciales. Esta es la única forma de poder detectar este tipo de errores. No se espera que el sistema procese la sobrecarga sin los recursos adecuados, pero sí que se comporte de una manera razonable.

1.8 Conclusiones Parciales

Hasta el punto actual del presente trabajo se arribó a las siguientes conclusiones:

- Se realizó un estudio detallado de los conceptos necesarios para la compresión del Sistema Orbita.
- Se decidió la utilización del marco de trabajo Bosón, ya que garantiza que se cumpla con la versatilidad necesaria para construir sistemas de distintos dominios y proporcione la capacidad de integración con un mínimo esfuerzo. Al mismo tiempo, que reúne un conjunto de tecnologías actualizadas necesarias para el desarrollo de la solución.
- Se seleccionó la metodología (AUP-UCI) ya que se adapta a las características del sistema a desarrollar. Además, es la metodología definida para la actividad productiva en la universidad la cual garantizará diseñar y elaborar la documentación necesaria para guiar el desarrollo del sistema de forma ágil y metódica.
- Se definió la utilización de múltiples herramientas para el desarrollo de la solución, como el IDE de desarrollo Phpstorm, como herramienta case Visual Paradigm y los lenguajes anteriormente descritos.

Capítulo 2: Análisis y Diseño

2.1 Introducción

En este capítulo se presenta la explicación de la migración, la especificación de los requisitos y se muestran las historias de usuarios generadas en la presente investigación. Luego, se consideran puntos claves en el desarrollo como patrones de diseño utilizados y el mapeo de la base de datos. También se muestran los productos de trabajos que propone la metodología seleccionada y el diseño de clases con estereotipos web del sistema.

2.2 Propuesta de solución

Teniendo en cuenta las dificultades que presentan el Sistema Orbita, mencionadas en la fundamentación teórica. Se decide desarrollar dos módulos que mantenga la lógica de los módulos anteriores, pero que aumenten su rendimiento y contribuyan a mejorar el rendimiento del sistema completo. Inicialmente partimos del análisis de los requisitos con los cuales ya cuenta el sistema y que perdurarán en el ciclo de vida de la migración de los módulos Combustible y Persona del Sistema Orbita a la arquitectura de referencia en PHP Bosón. El sistema contará con una interfaz amigable e intuitiva lo cual permitirá una fácil interacción, estará dividido por módulos, restringiendo el uso de los módulos de acuerdo al contexto operacional de cada usuario específico. El sistema utiliza el patrón arquitectónico MVC, el cual nos permite separar el modelo, el negocio y las vistas. Dando la posibilidad a estas últimas ser reutilizables. Además, nos permite hacer uso del motor de plantillas twig el cual es rápido, seguro y flexible para el trabajo con las vistas de la solución. También se hace uso de la librería de Doctrine la cual será la encargada del manejo de las operaciones con el modelo de datos. El uso de la arquitectura de referencia PHP Bosón que tiene como base el marco de trabajo symfony2, permite un menor uso de memoria por parte del sistema.

2.3 Requisitos del sistema

La especificación de requisitos del software es una descripción completa del comportamiento del sistema software a desarrollar. Incluye la descripción de todas las interacciones que se prevén que los usuarios tendrán con el software. También contiene requisitos no funcionales (o suplementarios). Los requisitos no funcionales son los requisitos que imponen restricciones al diseño o funcionamiento del sistema software (tal como requisitos de funcionamiento, estándares de calidad, o requisitos del

diseño). Las estrategias recomendadas para la especificación de los requisitos del software están descritas por la norma IEEE 830-1998. Este estándar describe las posibles estructuras, el contenido deseado, y cualidades de una especificación de requisitos del software. Los requisitos se dividen en tres: (a) funcionales: son los que el usuario necesita que efectúe el software; (b) no funcionales: son los "recursos" para que trabaje el sistema de información (redes, tecnología); y (c) empresariales u organizacionales: son el marco contextual en el cual se implantará el sistema para conseguir un objetivo macro (Pytel et al. 2011).

2.3.1. Requisitos Funcionales

El tratamiento de requisitos es el proceso mediante el cual se especifican y validan los servicios que debe proporcionar el sistema, así como las restricciones sobre las que se deberá operar. Consiste en un proceso iterativo y cooperativo de análisis del problema, documentando los resultados en una variedad de formatos y probando la exactitud del conocimiento adquirido. La importancia de esta fase es esencial puesto que los errores más comunes y más costosos de reparar, así como los que más tiempo consumen se deben a una inadecuada ingeniería de requisitos (Escalona y Koch 2002).

A partir de los resultados de aplicar las técnicas de obtención de requisitos se obtuvieron los siguientes requisitos funcionales:

Tabla 1: *listado de requisitos funcionales.*

Número	Requisitos	Descripción
RF#1	Autenticar usuarios	Permite que los usuarios registrados en el dominio se puedan autenticar en el sistema.
RF#2	Listar persona	Permite listar los datos de todas las personas registradas en el sistema.
RF#3	Buscar persona	Permite realizar una búsqueda de las personas registradas en el sistema por su nombre, apellidos, CI, área y licencia.
RF#4	Adicionar persona	Permite adicionar los datos una persona en el sistema.
RF#5	Modificar persona	Permite modificar de forma permanente los datos de una persona

		en el sistema.
RF#6	Eliminar persona	Permite eliminar de forma permanente una persona en el sistema.
RF#7	Listar índice de combustible	Permite listar los datos de todos los índices de combustibles registrados en el sistema.
RF#8	Buscar índice de combustible	Permite realizar una búsqueda todos los índices de combustible por el valor de cualquier columna de la tabla.
RF#9	Adicionar índice de combustible	Permite adicionar un índice de combustible a un vehículo determinado en el sistema.
RF#10	Modificar índice de combustible	Permite modificar un índice de combustible a un vehículo determinado en el sistema.
RF#11	Adicionar datos de la escuela de recalificación	Permite adicionar los datos de la escuela de recalificación de un trabajador.
RF#12	Adicionar la categoría de la licencia	Permite adicionar el tipo de categoría de la licencia a los datos de trabajador.
RF#13	Modificar la categoría de la licencia	Permite modificar el tipo de categoría de la licencia a los datos de trabajador
RF#14	Eliminar la categoría de la licencia	Permite eliminar el tipo de categoría de la licencia a los datos de trabajador.
RF#15	Adicionar sanción	Permite adicionar una sanción al trabajador.
RF#16	Modificar sanción	Permite eliminar una sanción al trabajador.
RF#17	Eliminar sanción	Permite adicionar una sanción al trabajador.
RF#18	Asignar herramienta	Permite asignar una o varias herramientas al trabajador.
RF#19	Quitar herramienta	Permite quitar una o varias la herramienta al trabajador.

RF#20	Listar vehículos	Permite listar todos los vehículos registrados en el sistema.
RF#21	Mostrar reporte de personas por áreas	Permite generar un reporte con la relación de todas las personas de un área determinada y guardarlo en pdf.
RF#22	Mostrar reporte de personas por cargo	Permite generar un reporte con la relación de todas las personas de un cargo determinada y guardarlo en pdf.
RF#23	Mostrar reporte de personas por áreas	Permite generar un reporte con la relación de todas las personas de un área determinada y guardarlo en pdf.
RF#23	Mostrar reporte de choferes que se le vencen el chequeo médico	Permite generar un reporte con la relación de todos que se le vencen el chequeo médico en un periodo de tiempo determinado y guardarlo en pdf.
RF#24	Mostrar reporte de análisis de índice de consumo por tipo de combustible	Permite generar un reporte con el análisis de índice de consumo por tipo de combustible seleccionado y guardarlo en pdf.
RF#25	Mostrar reporte de análisis de índice de consumo por tipo de vehículo	Permite generar un reporte con el análisis de índice de consumo por tipo de vehículo seleccionado y guardarlo en pdf.
RF#25	Mostrar reporte de análisis de índice de consumo por tipo de actividad	Permite generar un reporte con el análisis de índice de consumo por tipo de actividad seleccionado y guardarlo en pdf.
RF#26	Mostrar reporte de análisis de intensidad energética	Permite generar un reporte con el análisis de intensidad energética y guardarlo en pdf.
RF#27	Mostrar reporte de análisis de	Permite generar un reporte con el análisis de rendimiento

	rendimiento energético	energético y guardarlo en pdf.
RF#29	Mostrar reporte análisis del comportamiento de los índices de consumó en un mes	Permite mostrar en una gráfica el comportamiento de los índices de consumó en un mes determinado y guardarlo en pdf.
RF#30	Mostrar reporte análisis del comportamiento de los índices de consumó en un año	Permite mostrar en una gráfica el comportamiento de los índices de consumó en un año determinado y guardarlo en pdf.

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

Las Historias de Usuario (HU) son un instrumento para el levantamiento de requerimientos para el desarrollo de un software, también conducen el proceso de las pruebas de aceptación, que son empleados para verificar que las historias han sido implementadas correctamente. Existen diferencias entre las HU y la tradicional especificación de requisitos, siendo su principal diferencia el nivel de detalle. A continuación, se muestran las historias de usuario de los requisitos adicionar índice de combustible (Ver tabla 1) y modificar índices de combustible (Ver tabla 2).

Tabla 1 Historia de usuario Adicionar índice de combustible

Historia de usuario Adicionar índice de combustible	
Número: 1	**Nombre del requisito:** Adicionar índice de combustible.
Programador: Raciel Ferrín González	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 10 días
Riesgo en Desarrollo: Poca experiencia por parte de los estudiantes con las tecnologías de desarrollo	**Tiempo Real:** 9 días

Descripción: El sistema debe mostrar la opción de adicionar Índice de Combustible, donde debe llenar los campos Matricula vehículo en el cual se da clic en el botón de buscar y se selecciona el vehículo al que se le quiere asignar el Índice de Combustible, luego se rellenan los otros campos Asignado plan UCI, Asignado extra UCI, Serviciando otros, residual en tanque, inicio en tanque, distancia recorrida, fecha de registro y índice de consumo plan, después debe seleccionar el botón Adicionar para guardar los datos. Luego de insertado los campos el sistema efectuará las validaciones pertinentes, en caso que existan errores muestra un mensaje indicando los campos incorrectos. Si toda la información brindada es correcta se muestra un mensaje indicando el éxito de la operación.

Observaciones:

Prototipo de interfaz:

Tabla 2 Historia de usuario Modificar índice de combustible

Historia de usuario Modificar índice de combustible

Número: 2	**Nombre del requisito:** Modificar índice de combustible.
Programador: Raciel Ferrín González	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 8 días
Riesgo en Desarrollo: Poca experiencia por parte de los estudiantes con las tecnologías de desarrollo	**Tiempo Real:** 7 días

Descripción: El sistema debe mostrar la opción de modificar un Índice de Combustible, donde debe llenar los campos Matricula vehículo en el cual se da clic en el botón de buscar y se selecciona el vehículo al que se le quiere asignar el Índice de Combustible, luego se rellenan los otros campos Asignado plan UCI, Asignado extra UCI, Serviciando otros, residual en tanque, inicio en tanque, distancia recorrida, fecha de registro y índice de consumo plan, después debe seleccionar el botón Adicionar para guardar los datos. Luego de insertado los campos el sistema efectuará las validaciones pertinentes, en caso que existan errores muestra un mensaje indicando los campos incorrectos. Si toda la información brindada es correcta se muestra un mensaje indicando el éxito de la operación.

Observaciones:

Prototipo de interfaz:

2.3.2. Requisitos no Funcionales

Los requisitos no funcionales, son restricciones de los servicios ofrecidos por el sistema. De forma alternativa, se utilizan para definir las restricciones de la aplicación como la capacidad de los dispositivos de entrada/salida y restricciones de tiempo sobre el sistema (Requerimiento, 2015).

Usabilidad

RNF#3 El sistema podrá ser usado por personas con conocimientos básicos en el manejo de computadoras.

RNF#4 Los usuarios deberán tener conocimiento de la forma en que se realizan los procesos que maneja el sistema.

Eficiencia

RNF#5 Tiempo de respuesta: el promedio de las peticiones que se realizan al servidor no deben ser mayor de 3 segundos. En el caso de información que involucre consultas a las bases de datos los tiempos de respuestas no deben exceder los 10 segundos.

Soporte

RNF#6 La aplicación debe estar documentada y proveer el código fuente, previendo futuras modificaciones en el mismo para potenciar su alcance o eficiencia.

RNF#7 Consta con la documentación necesaria para el aprendizaje sobre el uso de la herramienta informática.

Seguridad

RNF#10 La seguridad está a nivel de gestión de roles con el fin de mantener la integridad de los datos, por el cual el acceso a la herramienta será a través de estos roles, trayendo consigo además la protección de la información.

Portabilidad

RNF#11 El sistema debe ser multiplataforma con enfoque en tecnologías basadas en software no propietario.

Requisitos de hardware requeridos para utilizar la aplicación web:

RNF#12 La comunicación entre el servidor de aplicaciones y la base de datos se lleva a través del protocolo TCP/IP.

RNF#13 La comunicación entre el cliente y el servidor de aplicaciones se lleva a través del protocolo HTTP.

- **Cliente**

Deben cumplir con los siguientes requisitos de hardware:

RNF#14 Ordenador dual core con 2.0 GHz de velocidad de microprocesador o superior. RNF#15 Memoria RAM mínimo 2GB.

- **Servidor**

Debe cumplir con los siguientes requisitos de hardware:

RNF#16 Ordenador core i3 con 2.4 GHz de velocidad de microprocesador o superior RNF#17 Memoria RAM mínimo 4GB.

Requisitos de software

- **Cliente**

RNF#18 Navegador web Mozilla Firefox 43.0 o superior.

- **Servidor**

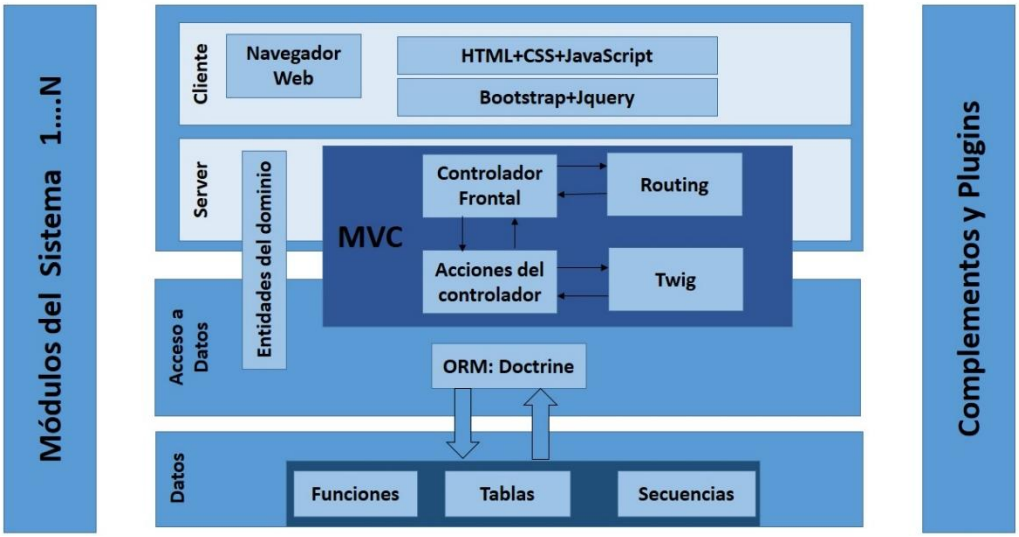
RNF#19 Servidor Web Apache 2.4 o superior, con módulo PHP 5.4 o superior.

RNF#20 Gestor de base de datos PostgreSQL 9.2 o superior (para el pc servidor).

2.4 Patrones

2.4.1. Patrón Arquitectónico

El sistema se implementa sobre una arquitectura en capas la cual proporciona unaorganización jerárquica, donde cada capa proporciona servicios a la capa inmediatasuperior y se sirve de las prestaciones que le brinda la inmediata inferior. La arquitectura está compuesta por: la capa de presentación, la capa de acceso a datos y la capa de datos. Es importante destacar el uso del patrón Modelo Vista Controlador (MVC) en las capas de presentación y acceso a datos. En la figura se puede apreciar la estructura arquitectónica en capas y la ubicación de los componentes en cada una de ellas,



destacando los elementos del patrón utilizado. La arquitectura propuesta posee la característica de tener varios complementos transversales a las capas para garantizar seguridad, tratamiento de excepciones, entre otros aspectos. Cada uno de los módulos se estructura arquitectónicamente igual. Las entidades del dominio son accesibles en la sub-capa servidor de la capa de presentación y la capa de acceso datos.

Figura 2Modelo Vista Controlador (MVC).

La Capade Presentación es la que contiene los componentes con los que va a interactuar el usuario (páginas). Permite alcanzar las funcionalidades que brinda el controlador y mostrar o capturar la información a través de los diferentes elementos que comprende: twigs y formularios. Se divide en dos sub-capas: cliente y servidor.

- En la primera se visualiza mediante el navegador web (utilizando tecnologías como HTML5, CSS3, JavaScript, Bootstrap y JQuery) datos procesados.
- En la segunda se maneja la lógica de control, así como la construcción de páginas y formularios.

La Capa del Negocio o Controladora contiene lo que se llama la lógica de negocio, donde cada controlador se encarga de una funcionalidad de la aplicación. Esta capa es la encargada de comunicarse con las vistas y la capa del modelo de datos.

La Capa de Acceso a Datos contiene las entidades y repositorios que mapea Doctrine como marco de trabajo para la comunicación con el servidor de datos. Gestiona las peticiones de la capa de negocio consultando la base de datos y retorna los datos correspondientes. Es la encargada de comunicarse con la Capa de Datos que se encuentra el gestor de base de datos PostgreSQL y en él un conjunto de esquemas y tablas que permiten persistir la información con la que trabaja la aplicación y manejar su almacenamiento.

Los Módulos o (bundles) para cada módulo se constituye una estructura de carpetas que se organizan según la arquitectura que se describe en la siguiente figura. Permiten utilizar funcionalidades construidas por terceros o empaquetar sus propias funcionalidades para distribuirlas y reutilizarlas.

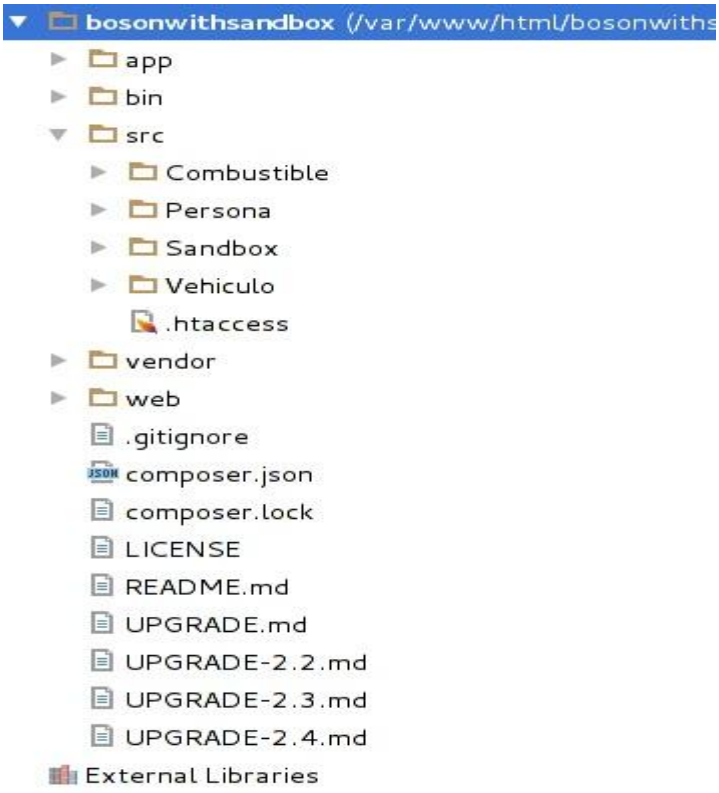


Figura 3 Estructura de carpetas de la aplicación

Los Complementos son un grupo de facilidades y mejoras que permiten lograr una arquitectura más flexible y adaptable tales como gestión de seguridad, de validaciones, de mensajería y de configuración, así como tratamiento de excepciones y otras.

En las capas de presentación y acceso a datos se evidencia el uso del patrón Modelo Vista Controlador (MVC) que se explica a continuación:

Este patrón de arquitectura de software se encarga de separar los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes e incrementa la reutilización y flexibilidad. Divide una aplicación interactiva en tres partes:

❖ ElModelo: contiene los datos y la funcionalidad esencial, las vistas despliegan la información al usuario y los controladores manejan las entradas.

❖ La Vista: encapsula las interfaces de usuario, entidades de presentación, plantillas twig y formularios necesarios para la interacción con el cliente.

❖ El Controlador: se encarga de recibir una petición, procesar la información, hacer un pedido al modelo y devolver una respuesta a los controladores los cuales a su vez la envían a la vista. Contiene las clases Controller (Controladoras) que se encarga de dar respuesta a las peticiones realizadas por el usuario.

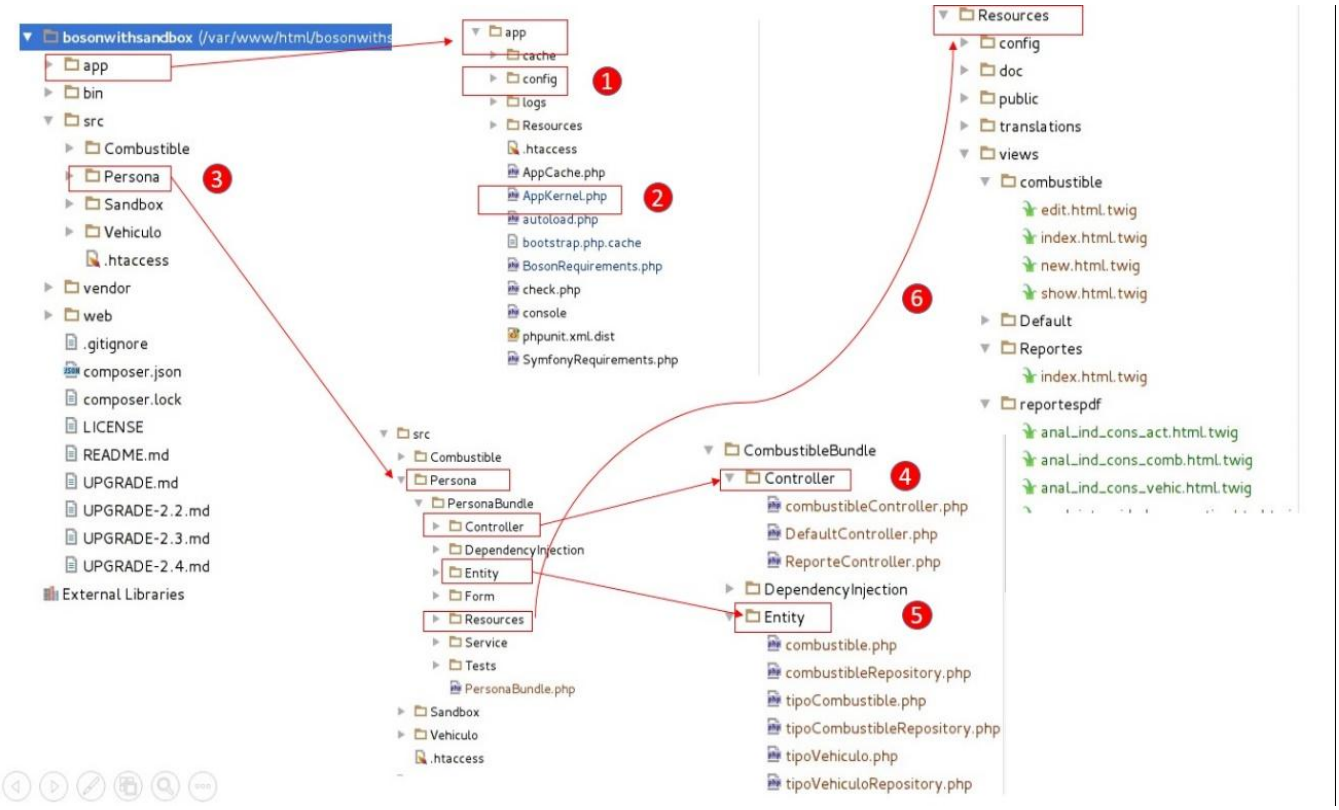
Modelo: contiene las clases Entity (Entidad) y las clases Repository (Repositorio), quienes se encargan del manejo de los datos para visualizarlos. Inicialmente el usuario interactúa con la interfaz (Vista). El controlador recibe la petición de la acción solicitada y gestiona el evento. El controlador accede al modelo actualizándolo o buscando la información requerida. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario y mostrar los datos del modelo para generar la interfaz apropiada.

En la figura 3 se muestra la estructura de carpetas del proyecto siguiendo el patrón arquitectónico MVC obtenido con la ayuda del marco de trabajo Symfony2.

Symfony 2 posee un controlador frontal el cual es el encargado de crear el kernel de la aplicación mediante una instancia de la clase **2**y es la responsable de toda la configuración. Ofrece las rutas de los módulos o bundles que el usuario necesita para satisfacer su necesidad y que se encuentran en la carpeta config **1**. Es el núcleo de Symfony 2 y por tanto uno de los componentes fundamentales para su correcto funcionamiento. Dentro de la estructura del proyecto se puede observar la carpeta Persona la cual contiene el sistema para evaluar la satisfacción del cliente a través de encuestas.

Dentro de la carpeta src se encuentra los bundles de la aplicación. Los módulos poseen una organización común, la cual se muestra en la figura anterior dentro de la misma se localizan las carpetas con los componentes específicos de la arquitectura:

Figura 4 Estructura de carpetas del proyecto



Controller **4:** posee los ficheros con los códigos de las clases Controladoras.

Entity **5:** en esta carpeta se encuentran las Entidades del Modelo.

Resources **6:** dentro de esta se encuentran las carpetas con los CSS, los JavaScript y las Twig que conforman la Vista.

2.4.2. Patrones GRASP

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y proveen facilidades para crear un software reutilizable de buena calidad. Cada patrón describe un problema que ocurre repetidamente en el entorno, y describe el núcleo de la solución a ese problema, de tal forma que esta pueda ser usada un millón de veces, sin hacer el mismo trabajo dos veces (Asenjo González, 2003). A continuación, se muestran los patrones utilizados en la solución.

Fachada:provee de una única interfaz para acceder a un sistema completo, que actúa como único punto de acceso al mismo, y hace que este sea más fácil de utilizar. Ejemplo de esta, es la interfaz de inicio del sistema llamada portal. Por la necesidad de integración entre los módulos del Sistema de Control de Flota y Mantenimiento, era necesaria la implementación de una clase fachada donde fueran publicados los servicios necesarios por estos, facilitando su interacción.

Controlador: el empleo de este patrón se puede observar en la creación de una clase controladora para la gestión de las personas, el control y análisis de índices de consumo de combustible. También, para mostrar sus respectivos los reportes (salidas del sistema) relacionado con el área de transporte de la UCI.

```php
/**
 * trabajador controller.
 *
 *
 */
class trabajadorController extends Controller implements AutenticableController
{

    public function indexAction()
    {
        $em = $this->EM();
        $entities = $em->getRepository('PersonaBundle:trabajador')->findAll();
        return $this->render('Persona:trabajador:index', array(
            'entities' => $entities

        ));
    }
```

Figura 5 Clase controladora trabajadorController, módulo Persona

Experto:

Fue usado en todas las clases definidas ya que estas contaban con la información necesaria para cumplir con la responsabilidad asignada.

```php
use Doctrine\ORM\Mapping as ORM;

/**
 * herramientas
 *
 * @ORM\Table(name="mod_mantenimiento.nom_herramienta")
 * @ORM\Entity(repositoryClass="Persona\PersonaBundle\Entity\herramientasRepository")
 */
class herramientas
{
    /** @var integer ...*/
    private $idherramienta;

    /** @var string ...*/
    private $codigo;

    /** @var string ...*/
    private $descripcion;

    /** @var string ...*/
    private $preciomn;
```

Figura 6 Entidad herramientas, módulo Persona

Alta cohesión:

En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Su empleo se puede observar en el desarrollo del módulo Combustible ya que se les asignaron responsabilidades a las clases de forma tal que cada una se encarga de realizar solamente las funciones que estén en correspondencia con su responsabilidad. Se puede apreciar el uso de este patrón en la imagen siguiente ya que demuestra que la clase combustible es capaz de resolver todas sus necesidades si ayuda de otras clases.

Bajo acoplamiento:

Un ejemplo donde se puede observar la utilización de este patrón es que la clase ReportecombustibleController, se integra con el componente Nomencladores y no conoce la existencia de la clase Nomtipovehiculo, que se encuentra en dicho componente y contiene los datos que son solicitados, sino que interactúa con la fachada, encargada de pasarle los valores pedidos.

2.5 Diagrama de clases de diseño

En el siguiente diagrama (Ver figura 7) se muestra la estructura que tiene el sistema. El cual cuenta con un conjunto de páginas web, formularios y librerías que son las encargadas de presentarle al

usuario la información referida al módulo. Cuando el usuario interactúa con estos elementos el servidor se encarga de la comunicación con el núcleo de la aplicación (AppKernel), el cual envía los datos de la petición del usuario al controlador (combustibleController). Este controlador utilizando diversos métodos que tiene definidos para procesar dichos datos, prepara y envía la respuesta al usuario inicial. De igual forma ocurre con los demás módulos.

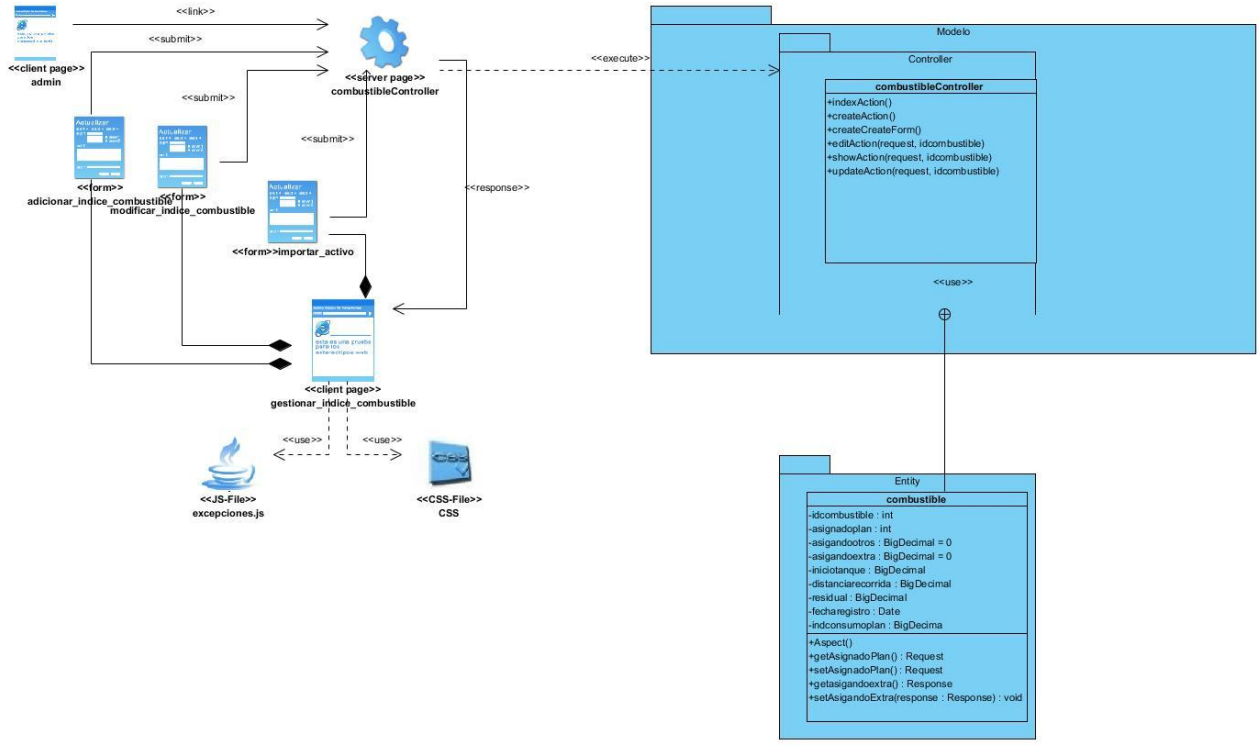


Figura 7 Diagrama de clases del diseño, módulo Combustible

2.6 Modelo de datos

Un modelo de datos es una serie de conceptos que puede utilizarse para describir los datos de acuerdo con reglas y convenios predefinidos y luego ser manipularlos (Jiménez, 2011). Se utiliza para describir la estructura lógica y física de la información persistente gestionada por el sistema, así como la correlación entre las clases de diseño y las estructuras de datos persistentes. En otras palabras, permite describir las estructuras de datos de la base de datos, su tipo, descripción y la forma en que se relacionan. A continuación, se presenta el modelo de base de datos del módulo Persona.

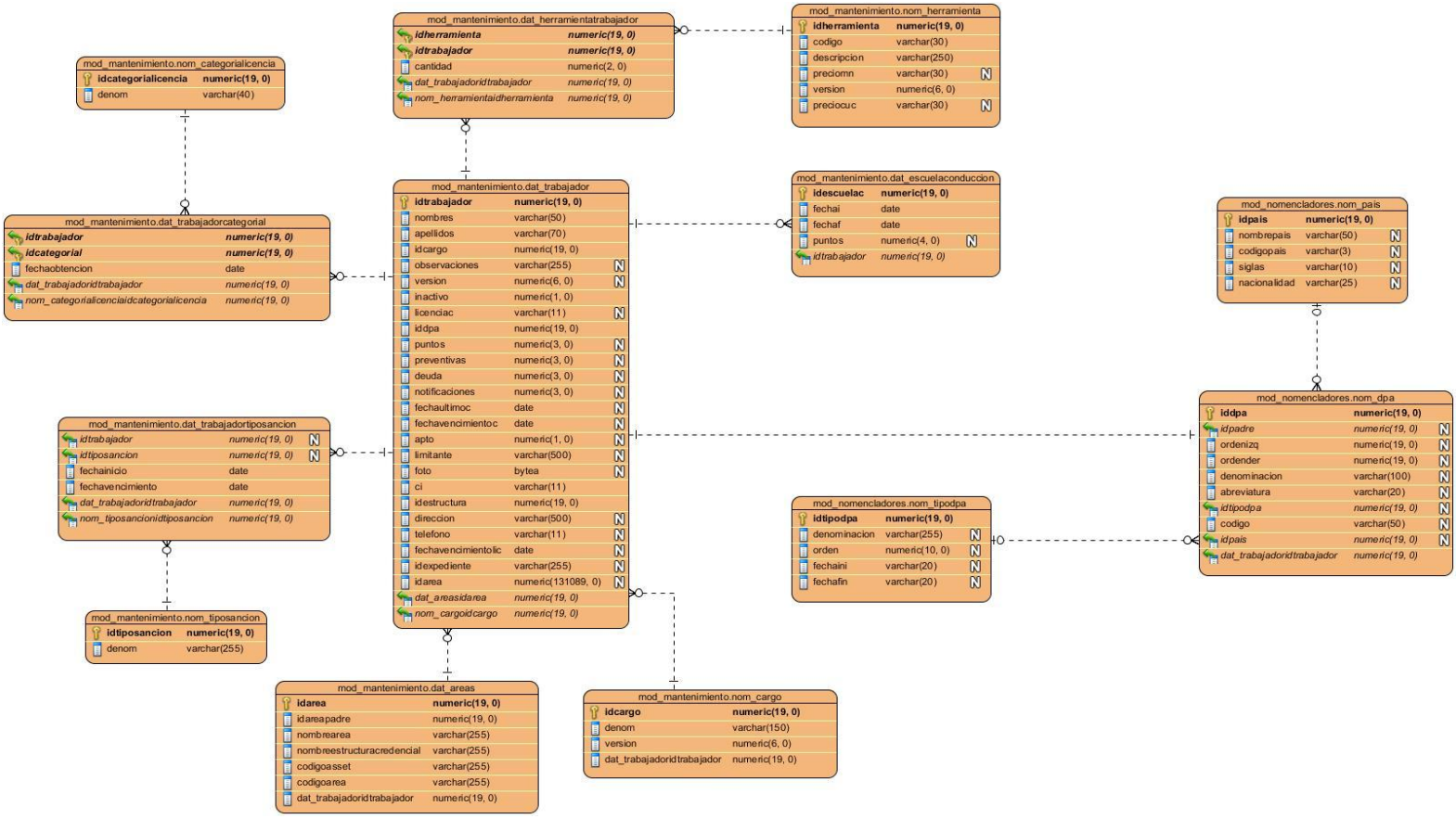


Figura 8**:**Modelo de Base de Datos del Módulo Persona

Para la migración de los módulosCombustible y Persona se empleó la propia base de datos del sistema actual. El primer paso, fue mapear todas las tablas de la base de datos utilizando los

componentes de Symfony2 mediante la librería doctrine. Tarea la cual se realizó mediante los siguientes de comandos.

- php app/console doctrine:mapping:convert xml ./src/Persona/PersonaBundle/Resources/config/doctrine/metadata/orm --from-database --force
- php app/console doctrine:mapping:import PersonaBundle annotation
- php app/console doctrine:generate:entities PersonaBundle

La primera línea de comando es en la quesele específica a Doctrine lea la información de la base de datos seleccionada y la procese, generándose un archivo de configuración con la descripción de la base de datos mapeada. Este archivo se puede guardar en los formatos xml, yml o php.Luego con el segundo comando le especificamos que nos importe ese archivo de configuración a nuestro bundle con el formato de anotaciones[14]. Finalmente, le pedimos a symfony2 mediante el último comando que nos genere las entidades que conformaran el modelo de datos con la información del archivo importado.

El proceso de mappeado tomo un tiempoconsiderable para su realización, más allá de su complejidad, por la cantidad de objetos a mapear y por la cantidad de funcionalidades que se re implementaron.Se mapearon un total de 15 clases, localizadas en la carpeta Entity de cada módulo con sus respectivas clases Repository[15]. Con la nueva versión del ORM Doctrine 2 el funcionamiento interno transita desdelas clases controladoras, accediendo a las entidades, hasta las clases Repository.

Todas las clases entidadnecesitan una clave primaria que las identifique unívocamente. Sedesigna la propiedad que se usará como identificador con la anotación @ORM\Id. Además, se defineel valor es generado automáticamente para ello se utiliza la anotación @ORM\GeneratedValue.Se utiliza comoestrategia SEQUENCE porque se maneja a través una secuencia la generación del id en la base de datos. Se detallael nombre de la secuencia utilizando @ORM\SequenceGenerator (Ver Figura 9).

[14]Las anotaciones se parsean mediante Doctrine Annotations, una librería que nos permite mapear información en las clases, atributos o métodos mediante comentarios llamados anotaciones.

[15]En esta última clase se definen las consultas que necesita cada entidad.

```
/**
 * @var integer
 *
 * @ORM\Column(name="idtrabajador", type="integer")
 * @ORM\Id
 * @ORM\GeneratedValue(strategy="SEQUENCE")
 * @ORM\SequenceGenerator(sequenceName="mod_mantenimiento.sec_trabajador_seq", allocationSize=1, initialValue=1)
 *
 */
private $idtrabajador;
```

Figura 9 Ejemplo del uso de anotaciones, módulo Persona

Para que exista una relación donde se obtiene una única instancia de un objeto se utiliza:@ManyToOne y @JoinColumn (Ver Figura 10). Doctrine2 soporta las relaciones uno-a-uno,uno-a-muchos, muchos-a-muchos.

```
/**
 *
 * @ORM\OneToMany(targetEntity="trabajadorcategorialic", mappedBy="trabajadorc", cascade={"persist","remove"})
 * @ORM\JoinColumn(name="idtrabajador", referencedColumnName="idtrabajador")
 */
private $categorialicencia;
```

Figura 10 Ejemplo del uso de anotaciones, módulo Persona

El uso de Object-Relational mapping(ORM)[16], consiste en la transformación de las tablas de una base de datos, en una serie de entidades que simplifiquen las tareas básicas de acceso a los datos para el programador. 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. Permite abstraerse al programador de la base de datos y le centra en el desarrollo de la aplicación. También permite la facilidad de trabajo, un ORM, ya que nos facilita las labores básicas de cualquier acceso a datos.

[16] **Mapeo de Objeto-Relacional**

2.7 Conclusiones parciales

Al finalizar el presente capítulo se arriba a las siguientes conclusiones:

- Después de realizar el análisis del sistema en términos de solución, quedaron definidas las Historias de Usuarios proporcionando una comprensión detallada de las funcionalidades de la aplicación.
- La propuesta de arquitectura del sistema se sustenta en un conjunto de componentes reutilizables que tienen como base el patrón arquitectónico MVC, lo que conforma un sistema flexible a cambios.
- La obtención de los requisitos funcionales y no funcionales, permitió definir el comportamiento y restricciones del sistema para su implementación.
- El empleo de patrones de diseño garantizó una solución que tiene como premisa la reutilización de código durante la fase de implementación del software.

CAPÍTULO 3: Implementación y Validación

3.1 Implementación

En esta fase se genera todo el código fuente necesario para satisfacer las HU definidas para la solución. Al inicio de cada iteración, se lleva a cabo una revisión del plan de iteraciones y se modifica de ser necesario. Todas las HU son traducidas en tareas de programación.La implementación constituye una de las fases más importantes del desarrollo de *software*. En ella se toman como punto de partida los resultados obtenidos en el diseño, implementándose el sistema en términos de componentes como ficheros de código binario, código fuente, scripts y ejecutables. Su importancia reside en que se obtiene como consecuencia una herramienta informática o un sistema ejecutable, siendo esto uno de los principales objetivos en el desarrollo de *software*(Arizaca, R. E, 2011).

3.2 Diagrama de componente

Un diagrama de componentes representa la separación de un sistema de *software* en componentes físicos (por ejemplo, archivos, módulos, paquetes, base de datos, código fuente, binario o ejecutable, y muestra las dependencias y organización existente entre estos componentes (Arizaca, R. E, 2011). A continuación, se muestra el diagrama de componente del módulo de Persona (Ver Figura 9).

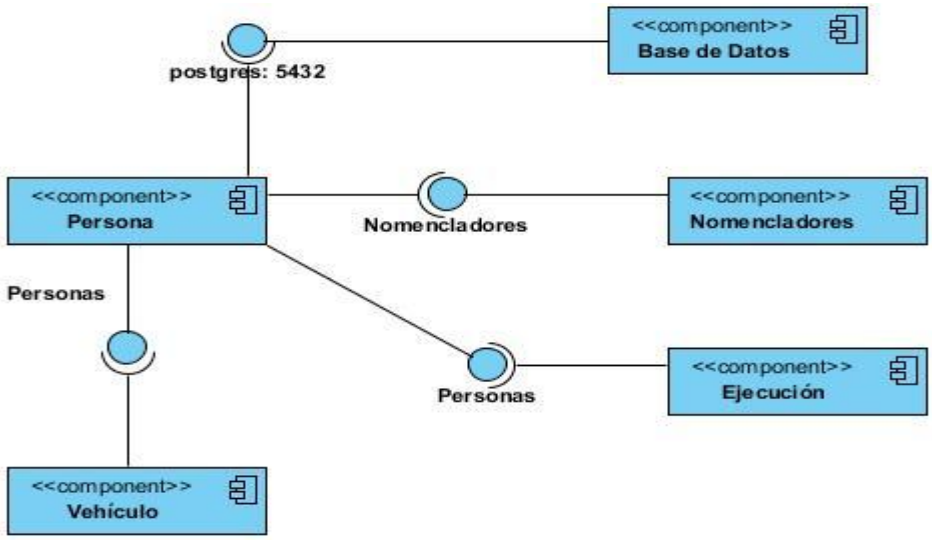


Figura 9: Diagrama de Componente del Módulo Persona

El módulo Persona no consume ningún servicio, pero si brinda servicios a los módulos Vehículo y Ejecución. Esto se evidencia en ambos módulos siendo este servicio el encargado de brindar la información de las personas existentes en la Base de Datos. En el módulo de Ejecución con el objetivo de asignarle un Conductor y un Responsable (Ver Figura 10) al vehículo en cuestión.

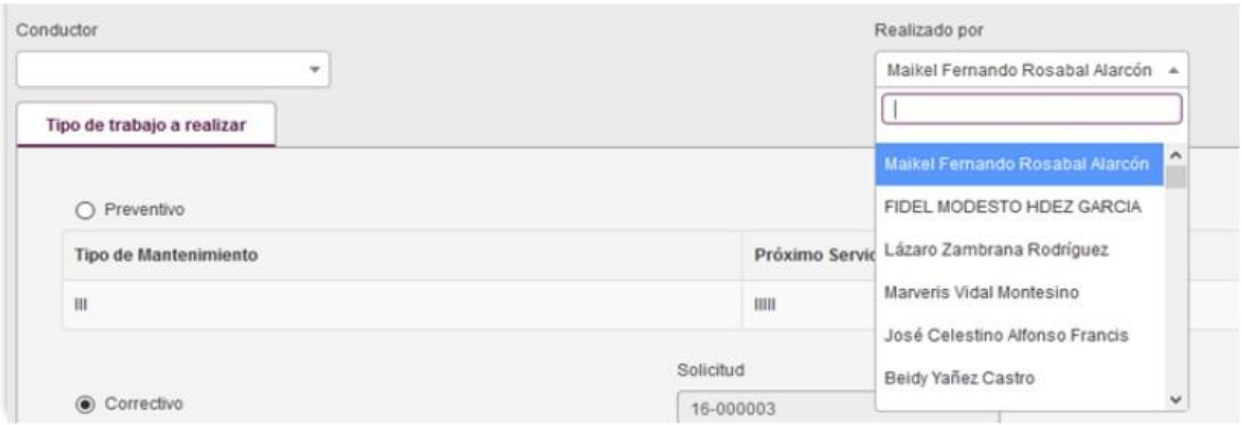


Figura 11 Módulo Ejecución, consumo de servicio personas del módulo Persona

En el módulo Vehículo se consume este servicio para obtener los datos de las personas para poder asignar un responsable de vehículo, así como su conductor (Ver Figura 11).

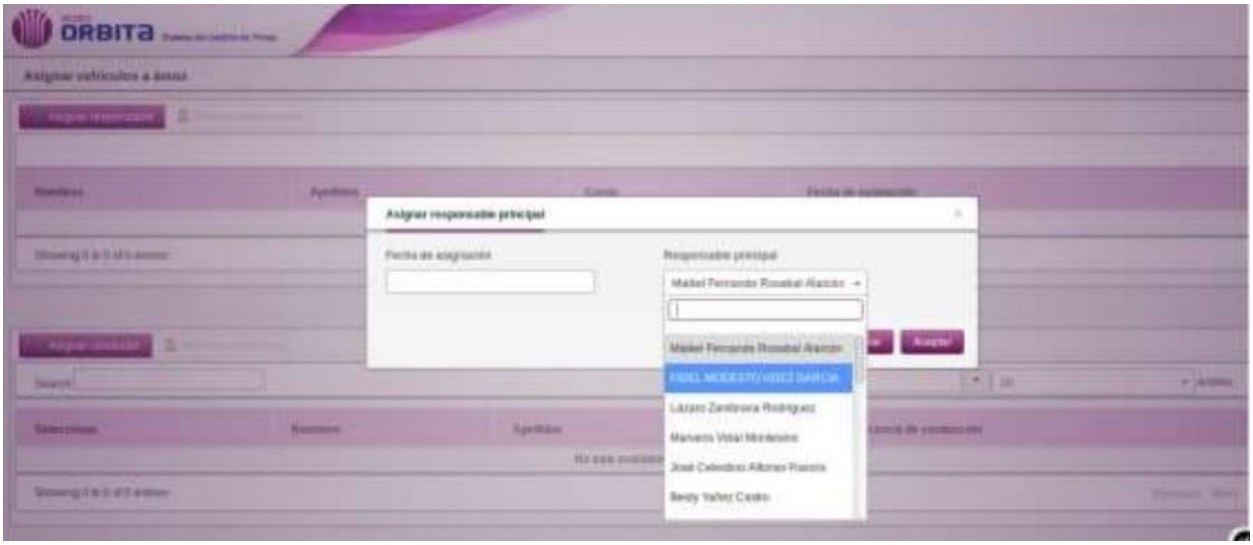


Figura 12 Módulo Vehículo, consumo del servicio de personas del módulo Persona

En el diagrama de componentes del módulo Combustible (Ver Figura 12), se muestra el consumo por parte del módulo Combustible del servicio del módulo Vehículo. Dicho servicio traer toda la información asociada a los vehículos del sistema.

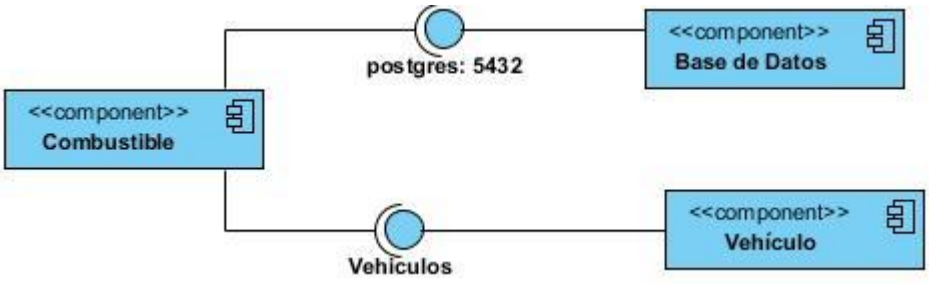


Figura 13 Diagrama de componente del módulo Combustible

El uso de este servicio en el módulo de Combustible se muestra en la funcionalidad adicionar índice de combustible, el cual necesita un listado de vehículos (Ver Figura 13) para permitirle a usuario seleccionar el vehículo al cual le quiere asignar el índice de consumo que se está creando.

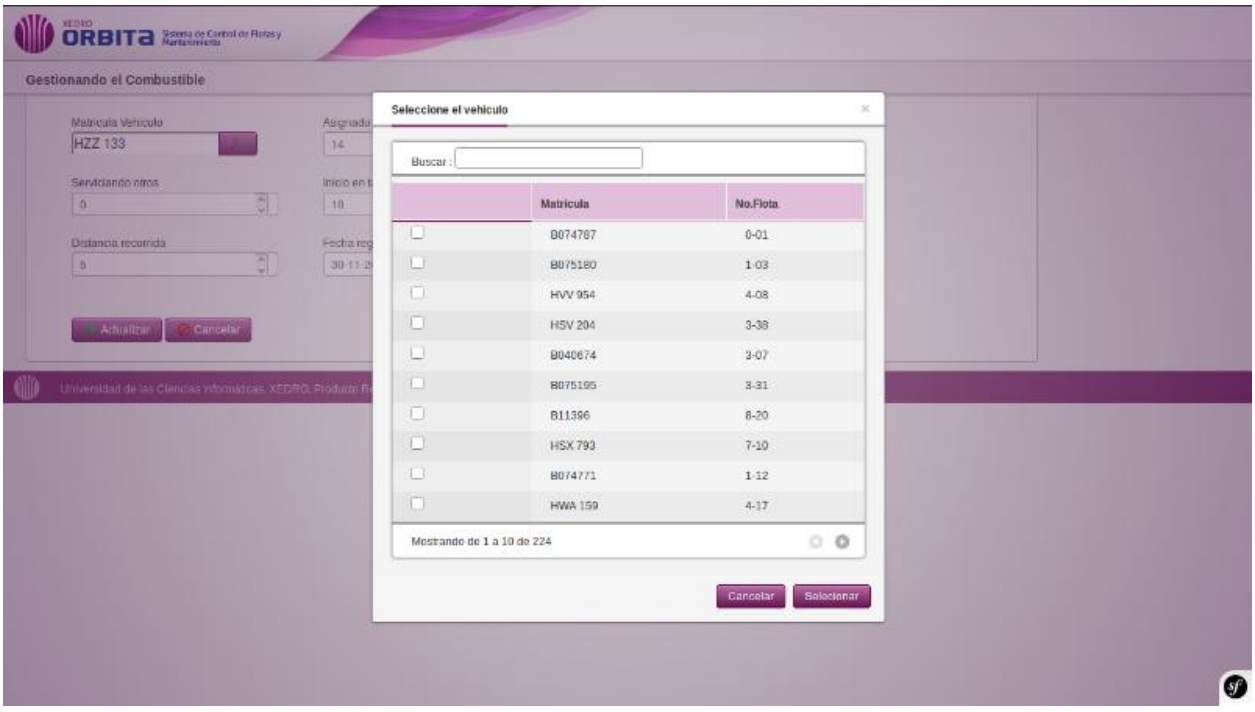


Figura 14 Módulo Combustible, funcionalidad adicionar índice de combustible, seleccionando el vehículo

3.3 Tratamiento de errores

El tratamiento de errores se hizo de ambos lados cliente y servidor. En el lado cliente a través de los lenguajes twig, javascript para validar algunos requisitos, permitiendo que la aplicación funcione de la manera esperada (Ver figura 15 y 16).

```
function validarBotones() {
    $(':checkbox').change(function () {


        if ($('input:checked').length > 1) {

            $('#btnEditar').attr('disabled', true);
            $('#btnImprimir').attr('disabled', true);
            $('#btnEliminar').attr('disabled', false);

        } else if ($('input:checked').length < 1) {
            $('#btnEditar').attr('disabled', true);
            $('#btnImprimir').attr('disabled', true);
            $('#btnEliminar').attr('disabled', true);

        } else {
            $('#btnEditar').attr('disabled', false);
            $('#btnImprimir').attr('disabled', false);
            $('#btnEliminar').attr('disabled', false);
            if (this.checked) {
                loaded = $(this).val();
                setSelectedUrls();

            }


        }

    });
}
```

Figura 15 Módulo Persona, parte del el código javascript

El tratamiento de errores en el lado del servidor se realiza utilizando el componente *validator* de symfony2 que nos permite validar los campos de los formularios a través de un grupo de restricciones establecidas en las entidades que modelan nuestro negocio. Si se detectan errores el componente los guarda para que luego con la ayuda delaclase controladora sean mostrados en la vista.

En la siguiente imagen se muestra el uso del componente*validator* en la clase controladora.

```
$validator = $this->get('validator');
$errors = $validator->validate($trabajador, null, array('datoslicencia'));
$erroresEscuela = $validator->validate($escuelac,null,array('datoslicencia'));
foreach ($erroresEscuela as $e) {
    $errors->add($e);
}
if (count($errors) == 0) {
    if($fechai>$fechaf){
        $error='La fecha inicio no puede ser mayor que la fecha fin';
        return $this->render("PersonaBundle:trabajador/wizard:datos_licencia_form.html.twig", array(
            'frm' => $form->createView(),
            'error' => $error,
            'escuelac' => $form_escuela->createView(),
            'categoriaslicencias' => $arrayCategorias,
            'editando' => $editando,
            'idtrabajador' => $idtrabajador,
        ));
```

Figura 16 Módulo Persona, muestra el uso del componente validator de symfony2

3.4 Estándares de codificación

Los estándares de codificación son aquellos que permiten entender de manera rápida y sencilla el código empleado en el desarrollo de un software y establecen un estilo de programación homogéneo permitiendo que cualquier persona que consulte el código, lo pueda entender en menos tiempo. Además, garantizan el mantenimiento óptimo de dicho código por parte del programador. El estándar de codificación PSR-1 es el estándar básico de estilos de código está orientado al contenido de los ficheros PHP y a los nombres de las clases y métodos. Su objetivo es garantizar un alto nivel técnico de interoperabilidad entre el código PHP(«PSR-u1: Basic Coding Standard »,2016).

En cambio, PSR-4 es el estándar de auto carga orientado a los nombres de los Namespaces, Clases y Ficheros. Su objetivo es facilitar la carga automática de clases(«PSR-4: Autoloader », 2016).A continuación, se muestran algunas pautas de los estándares anteriores definidos para el lenguaje PHP, por el equipo de desarrollo.

- Todas las nomenclaturas a utilizar se definen en idioma español.
- Los identificadores para las variables, los métodos y los parámetros se establecen que **DEBEN** declararse en notación camelCase. La letra minúscula y en caso de ser un nombre compuesto se escriben juntos y de la segunda palabra en adelante se escriben con letra inicial mayúscula.

Ejemplo: $licencia; $licenciaCategoria.

❖ Las clases formularios comienzan con el nombre del formulario según su función,seguido de la palabra Type

Ejemplo: trabajadorType.

❖ Las funciones deben ser llamadas sin espacios entre el nombre de la función, el signo de paréntesis y el primer parámetro, espacios entre cada coma por parámetro y sin espacios entre el último paréntesis, el signo de paréntesis cerrado y el signo de punto y coma.
Ejemplo: public adicionarPersonaDatosLicencia($idtrabajador){};

❖ Hacer uso de llaves para ganar en claridad del código.

❖ Los Namespaces y las clases **DEBEN** cumplir el estándar PSR-4.

❖ Las rutas completas de los Namespaces **DEBEN** tener la estructuraProveedor\Namespace\NombreDeLaClase. (VendorName\Namespace\ClassName)
Por ejemplo:
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
Proveedor:Illuminate
Namespace:Database\Migrations
Nombre de la Clase: Blueprint y Migration

❖ Todas las rutas **DEBEN** tener un Proveedor (Vendor Name).

3.5 Pruebas realizadas al sistema

Una vez finalizada la implementación del producto que se solicita, es necesario realizarle pruebas, con el objetivo de detectar errores en la aplicación y la documentación e identificar posibles fallos de implementación, calidad, o usabilidad del *software*. Además, medir el grado en que el software cumple con los requerimientos. Este proceso resulta de gran importancia porque proporciona una medida de la calidad de dicho sistema, siempre que se ejecute de manera correcta.

3.5.1. Prueba de caja negra

Para la realización de las pruebas de caja negra se empleó la técnica partición de equivalencia definida en el marco teórico. A continuación, se brinda un ejemplo de un caso de prueba de uno de

los requisitos que componen la funcionalidad adicionar persona que por su complejidad fue dividido en 6 partes. La siguiente tabla solo muestra la primera parte.

Escenario	Descripción	Categorías	Fecha de Obtención	Respuesta del sistema	Flujo central
EC 1.1 Adicionar Categoria Licencia	Se adicionan nuevas categorias correctamente	V seleccionado	V 26-05-16	El sistema almacena los datos y nos redireciona para la página datos de licencia	Seleccione la categoría y rellene la fecha de obtención -Se presiona el botón Aceptar Se validan los datos
EC 1.2 Adicionar Categoria Licencia dejando campos vacíos	No se insertan parámetros en los campos mostrados al adicionar las categorías	I no seleccionado V seleccionado	V 24-04-16 I	El sistema muestra los errores cometidos en una alerta y que se debe insertar algún parámetro	Se rellenan los campos mostrados -Se presiona el botón Aceptar
EC 1.2 Adicionar las categorías con caracteres inválidos	Se insertan parámetros inválidos en los campos mostrados al adicionar categorías	I no seleccionado V seleccionado I no seleccionado	V 26-05-16 I 26 jlas I dia 1	El sistema muestra los errores en una alerta indicando contiene carácteres inválidos indicando que debe corregirse	Se rellenan los campos mostrados -Se presiona el botón Aceptar

Figura 17 Descripción del caso de prueba para el caso de uso “Adicionar Categoría Licencia”.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Categorías	Campo de selección	No	Lista todas las categorías de licencia para la selección de una
2	Fecha de obtención	Campo de selección	No	Lista un calendario para seleccionar la fecha

Figura 18Descripción de las variables correspondiente al caso de prueba para el caso de uso “Adicionar Categoría Licencia”.

3.5.2. Prueba de caja blanca

Para aplicar las pruebas de caja blanca se empleó la técnica del camino básico. Esta permitió obtener una medida de la complejidad lógica para el diseño de los casos de pruebas y usar dicha medida como guía para la definición de un conjunto básico de caminos de ejecución. Para esto se utilizará la métrica de software complejidad ciclomática, la cual consiste en determinar una medida cualitativa de la complejidad lógica del programa (Pressman, 2010).

Se tomó como ejemplo el método adicionarPersonaDatosLicenciaAction(), perteneciente a la clase trabajadorController. Para ello se procede a enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo asociado. Los números que aparecen en las ilustraciones son bloques de códigos que se muestran en las ilustraciones posteriores.

```
public function adicionarPersonaDatosLicenciaAction(Request $peticion, $idtrabajador, $editando)
{
    /** @var trabajador $trabajador */
    $trabajador = $this->EM()->getRepository('PersonaBundle:trabajador')->find($idtrabajador);
    $form = $this->createForm(new datos_licenciaType(), $trabajador, array());
    $escuelac = $trabajador->getEscuelaconduccion();
    if (is_null($escuelac))
        $escuelac = new escuelaconduccion();

    $form_escuela = $this->createForm(new escuelaconduccionType(), $escuelac , array());
    $arrayCategorias = $trabajador->getCategorialicencia();

    if ($peticion->getMethod() == "POST") {
        1
    } else {
        //Muestra el formulario en blanco
        return $this->render("PersonaBundle:trabajador/wizard:datos_licencia_form.html.twig", array(
            'frm' => $form->createView(),
            'escuelac' => $form_escuela->createView(),
            'categoriaslicencias' => $arrayCategorias,
            'idtrabajador' => $idtrabajador,
            'editando' => $editando
        ));
    }
}
```

Figura 19 Método adicionarPersonaDatosLicencia parte1

```
$all = $peticion->request->all();
$fechai = $all['persona_personabundle_escuelaconduccion']['fechai'];
$fechaf = $all['persona_personabundle_escuelaconduccion']['fechaf'];
$form->handleRequest($peticion);
$form_escuela->handleRequest($peticion);
$escuelac->setTrabajador($trabajador);

$validator = $this->get('validator');
$errors = $validator->validate($trabajador, null, array('datoslicencia'));
$erroresEscuela = $validator->validate($escuelac,null,array('datoslicencia'));
foreach ($erroresEscuela as $e) {
    $errors->add($e);
}
if (count($errors) == 0) {
    2
    $this->EM()->persist($trabajador);
    $this->EM()->persist($escuelac);
    $this->EM()->flush();
    return $this->redirectToRoute('trabajador_chequeo_medico', array(
        'idtrabajador' => $trabajador->getIdtrabajador(),
        'editando' => $editando
    ));
} else {
    //Errores
    return $this->render("PersonaBundle:trabajador/wizard:datos_licencia_form.html.twig", array(
        'frm' => $form->createView(),
        'errors' => $errors,
        'escuelac' => $form_escuela->createView(),
        'categoriaslicencias' => $arrayCategorias,
        'editando' => $editando,
        'idtrabajador' => $idtrabajador,
    ));
}
```

Figura 20 Método adicionarPersonaDatosLicencia parte2

```
if($fechai>$fechaf){
    $error='La fecha inicio no puede ser mayor que la fecha fin';
    return $this->render("PersonaBundle:trabajador/wizard:datos_licencia_form.html.twig", array(
        'frm' => $form->createView(),
        'error' => $error,
        'escuelac' => $form_escuela->createView(),
        'categoriaslicencias' => $arrayCategorias,
        'editando' => $editando,
        'idtrabajador' => $idtrabajador,
    ));

}elseif($fechaf==$fechai){
    $error = 'La fecha inicio y la fecha fin no pueden ser iguales';

    return $this->render("PersonaBundle:trabajador/wizard:datos_licencia_form.html.twig", array(
        'frm' => $form->createView(),
        'error' => $error,
        'escuelac' => $form_escuela->createView(),
        'categoriaslicencias' => $arrayCategorias,
        'editando' => $editando,
        'idtrabajador' => $idtrabajador,
    ));
}
```

Figura 21 Método adicionarPersonaDatosLicencia parte3

Para el cálculo de la complejidad ciclomática se tiene en cuenta que esta corresponde al número de regiones. La complejidad ciclomática puede ser calculada de 3 formas:

- $V(G) = (\alpha - n) + 2$, siendo α el número de arcos o aristas del grafo y n el número de nodos.
- $V(G) = r$, siendo r el número de regiones cerradas del grafo.
- $V(G) = c + 1$, siendo c el número de nodos de condición.

Tabla 3 prueba de caja blanca

Prueba estructural de	Probador: Raciel Ferrin González

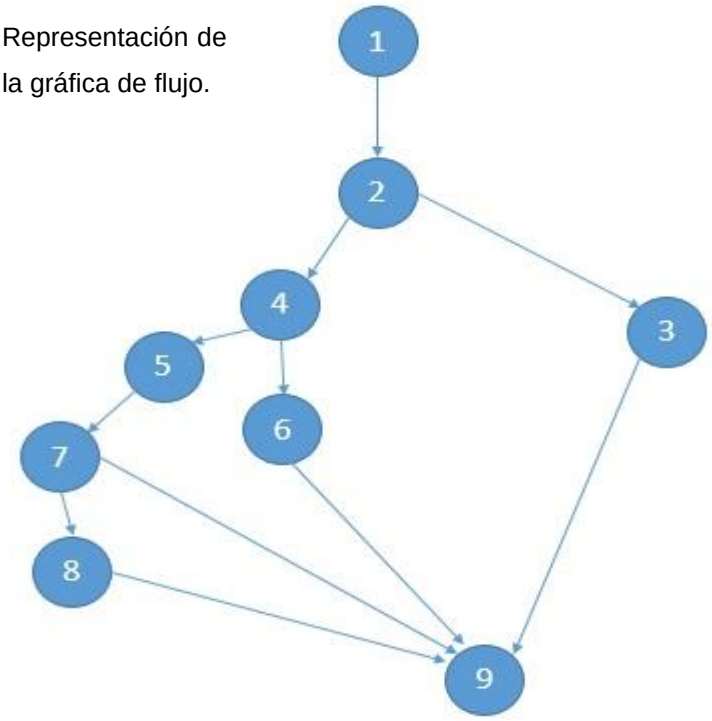
caja blanca	
El número de regiones corresponde a la complejidad ciclomática. **A=11** **N=9** **V (G) = (A – N) + 2=4** **Rutas linealmente independientes:** **1) 1-2-3-9** **2) 1 -2-4-6-9** **3) 1 -2-4-5-7-9** **4) 1 -2-4-6-7-8-9**	Representación de la gráfica de flujo. 

Tabla 4 Caso de prueba, camino básico 1

Caso de prueba para el camino básico 1	
Descripción: Se espera un objeto request que haya sido enviado por el método POST	
Condición de ejecución: Si el objeto request no ha sido enviado por el método POST	
Procesamiento de la prueba	
Dato de entrada:	petición, Idtrabajador, editando

Tipo de dato esperado:	peticion sea enviado por el método POST, idtrabajador, editando
Resultados esperados:	Se redireciona para un formulario de datosLicencia en blanco

Para consultar los demás casos de prueba(Ver Anexo 9).

Luego de aplicar la prueba del camino básico se comprobó que el métodoadicionarPersonaCategoriaLicenciaAction() se comportó de la manera esperada mostrando los resultados de una manera correcta.

En la siguiente tabla se muestra las no conformidades halladas en las diferentes iteraciones de las pruebas.

Tabla 5 No conformidades de las pruebas de caja blanca

Número de iteraciones	Número de no conformidades	Asociadas a
1ra	25	Errores en el sistema, errores de interfaz, errores de validación y errores ortográficos.
2da	13	Errores en el sistema, errores de interfaz, errores de validación y errores ortográficos.
3ra	9	Errores de validación y errores ortográficos.
4ta	0	

3.5.3. Prueba de rendimiento a la solución

Para la realización de estas pruebas se utilizó como servidor una máquina con las siguientes prestaciones: microprocesador Intel(R) Core(TM) i3 a 1.60GHz, 4Gb de memoria RAM y 40Gb de disco duro dedicados al sistema.

Resultados de las pruebas de carga y estrés

A continuación se muestran, en la siguiente tabla los resultados obtenidos de la aplicaciónantes y después de la migración.

Indicadores

Cantidad de usuarios: Cantidad de usuarios conectados.

Cantidad de peticiones: Cantidad de peticiones realizadas al servidor.

Tiempo mínimo: Tiempo mínimo de respuesta.

Tiempo máximo: Tiempo máximo de respuesta.

Respuesta 90%: Tiempo de respuesta del 90% de las páginas.

Rendimiento por petición: velocidad del sistema por petición.

Rendimiento: velocidad del sistema.

Tabla 6 Comparación de los sistemas

Indicadores	Sistema Viejo	Sistema Nuevo
Cantidad de usuarios	100	100
Cantidad de peticiones	13900	14576
Tiempo mínimo	3ms	0ms
Tiempo máximo	6636ms	337ms
Respuesta 90%	596ms	404ms
Rendimiento por petición	176.1/sec	102.4/sec
Rendimiento	881.3 kb/sec	522.6 kb/sec

En la tabla anterior se puede evidenciar que existe una gran diferencia en cuanto a las respuestas de ambos sistemas Orbita viejo y nuevo, tras realizarles las pruebas de carga y estrés. Los resultados

arrojados (anexos 1 y 2) por la herramienta jmeter demuestran una disminución en los tiempos de respuestas del sistema. A pesar de que la cantidad de usuarios conectados fueron los mismos, las peticiones hechas al sistema nuevo fueron mayores.

Podemos evidenciar valores son menores que los del sistema anterior como los tiempos de respuesta máx y min de la aplicación nueva (337 y 0ms) respectivamente.Finalmente, se constata que la velocidad de respuesta del sistema anterior según la herramienta es de 8.8segundos y en el nuevo sistema 5.2 segundos. De esta manera se observa una reducción de 3.6 segundos (3600 milisegundos).

3.5.4. Prueba de aceptación

El objetivo fundamental de estas pruebas es validar si las HU fueron implementadas según las especificaciones de los módulos Combustible y Persona.

Tabla 7 No conformidades encontradas en la pruebas de aceptación

Iteración	No conformidades
1ra	25
2da	13
3ra	9
4ta	0

Terminadas estas pruebas y corregidas las no conformidades el cliente avaló la solución quedando entonces el Certificado de Aceptación del Producto por parte del mismo.

3.6 Conclusiones del capítulo

En este capítulo de realizó la implementación y validación de la propuesta de solución arrojando resultados importantes tales como:

❖ Se validaron las fases de desarrollo abarcada durante el proceso de implementación de la herramienta informática con buenas prácticas, permitiendo obtener un producto de calidad.

- La definición y utilización del estándar de codificación a emplear durante la implementación, permitió el desarrollo de los módulos con un alto grado de legibilidad; lo que provee una guía para el mantenimiento y actualización del sistema, con código claro y bien documentado.
- Se realizaron pruebas de *software* que facilitaron identificar y solucionar las deficiencias detectadas en el sistema concibiendo resultados satisfactorios en el desarrollo del sistema como producto final y solución al problema planteado.
- Se constituyeron los diagramas de componentes de los módulos lo que facilitó una mejor organización y comprensión de las dependencias entre los módulos que conforman el sistema.
- Por último, se realizó una valoración del comportamiento de los atributos de la variable que forman parte del problema de la investigación, demostrando que, con el sistema desarrollado, se logró mejorar el rendimiento de los módulos Combustible y Persona.

Conclusiones Generales

Con la realización de este trabajo se demostró que el análisis del Sistema Orbita implementado anteriormente mostraba dificultades en su funcionamiento debido a que las tecnologías utilizadas en su desarrollo, ya que algunas atentaban contra su rendimiento por características propias y otras se encontraban obsoletas y sin soporte. Atendiendo a estas dificultades se decidió migrar el sistema a una nueva tecnología y por consiguiente se hizo necesario la migración de los módulos Combustible y Persona.

Sobre la base del análisis, interpretación y sistematización de las investigaciones teóricas y empíricas, a continuación, se presentan las siguientes conclusiones de la investigación:

- Se realizó el marco teórico de la investigación donde se establecieron conceptos necesarios para la compresión del software Orbita. Se decide la utilización del marco de trabajo Bosón, para garantizar que se cumpla con la versatilidad necesaria para construir sistemas de distintos dominios y proporcione la capacidad de integración con un mínimo esfuerzo.
- El análisis de las historias de usuarios generadas en la implementación del sistema anterior, así como de los requisitos derivados de las mismas, permitieron redefinir las nuevas historias de usuarios y los nuevos requisitos.
- La adopción de la arquitectura Bosón conllevó al rediseño de la estructura de los componentes y clases. Las diferencias entre ZendFramework y Symfony 2.7 se introducen cambios sustanciales en la estructura de los componentes.
- Se comprobó que los nuevos módulos Combustible y Persona cumplía con todas las especificaciones del cliente y se obtuvo el certificado de aceptación.
- Se comprobó que los nuevos módulos Combustible y Persona mejoraron su rendimiento respecto al módulo del sistema anterior a través de las pruebas realizadas.

Recomendaciones

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

Referencias Bibliográficas

"RE-3: Re-engineering, Restructuring and Reverse Engineering". **Yourdon, . 1989.** 1989, American Programmer, pp. pp. 3-10.

"La implantación de ISO 9001 en el desarrollo de software". **Osorio, Gloria Quintanilla. Septiembre (1999).** Septiembre (1999), Revista Soluciones, p. p.31.

Alvarez, Arianne Valdés. 2013.*Métricas de rendimiento para el Gestor de Documentos Administrativos eXcriba 2.0 en la Red de Centros de Desarrollo de la Universidad de las Ciencias Informáticas.* La Habana,junio : s.n., 2013.

Anabel Fé León Mendoza, Ebrik René López Ramirez. 2015.*Sistema de Gestión Integral de la Calidad de Software para el Centro de Gobierno Electrónico.* La Habana : s.n., 2015.

Arelys Saavedra Ayrado, Rolando Rodríguez Lastra. 2013.*Desarrollo del módulo Control de combustible del Sistema de Control de Flotas y Mantenimiento de la dirección de transporte de la Universidad de la Ciencias Informáticas.* La Habana : s.n., 2013.

Arizaca, R. E. 2011.*Artefacto: Diagrama de Componentes.* La Paz, Bolivia. : s.n., 2011.

Asenjo González, Diego Andrés and Ríos Peña, Alejandro. 2003.*Patrones de diseño.* La Habana : s.n., 2003.

Ávila, Antonio Mario Padrón. 2013.*Herramienta de pruebas de rendimiento a los servicios de obtención de variables, alarmas y eventos del Servidor de Comunicación con Terceros del SCADA "Guardián del ALBA".* La Habana,Cuba : s.n., 2013.

C. D. J. C. Comparación de las características de algunas herramientas de software para pruebas de carga. **CARLOS MARIO ZAPATA. 2011.** 2011, Avances en Sistemas e Informática , p. vol. 8.

Cataldi, Ing. Zulma. 2000.*Metodología de diseño, desarrollo y evaluación del software educativo.* La Plata : s.n., 2000.

CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DE ENTIDADES. 2014. Xedro orbita. Sistema de Control de Flotas y Mantenimiento. La Habana, La Habana, Cuba : s.n., 8 septiembre 2014.

Chase, . 2006. Understanding the Zend Framework, Part 1: The basics. 27 Junio 2006.

CSS3. 2015. Características de Css3. *[Online] 2015.* [Online] 9 Febrero 2015. http://www.css3.com..

Dayna Caridad Flores Hernádez, Adrés Sellés González. 2013.*Aplicación para la migración dinamica entre formatos bibliograficos.* La Habana : s.n., 2013.

Doctrine. 2016. doctrineproject.org. *doctrineproject.org.* [Online] 9 junio 2016. http://docs.doctrineproject.org/en/2.0.x7reference/introduction.html.

E.H. Bersoff, V.D. Henderson, S.G. Siegel. 1980.*Software Configuration Management.* s.l. : Prentice-Hall, 1980.

Eduardo, Laorden Fiter. 2012.*Descripción, comparación y ejemplos de uso de la funciones del toolbox de procesado digital de imágenes de Matlab.* Madrid : s.n., 2012.

Gauchat, Juan Diego. 2012.*El gran libro de HTML5, CSS3 y JavaScript.* Barcelona : MARCOMBO, 2012.

IEEE. 1992.*Standard for Software Maintenance,IEEE Computer Society Press.* s.l. : IEEE Std 12207, 1992.

IGNACIO ABEL, P. G. 2005.*Análisis e implementación de un Toolkit para Testing de Preformance.* s.l. : Edtion ed., , 2005.

Ing. Inoelkis Velazquez Osorio, Ing.Lisandra Cordero Estrada, Ing.René Rodrigo Bauta Camejo. 2013.*PROCEDIMIENTO PARA ACTUALIZACIÓN DE LA CAPA DE ACCESO A DATOS DEL MARCO DE TRABAJO SAUXE DE DOCTRINE 1.2.2 A DOCTRINE 2.2.* La Habana,Cuba : s.n., 2013.

J. A. Carsí, I. Ramos, J. Silva, J. Pérez, V. Anaya. 2002. Un Generador Automático de Planes de Migración de Datos. *Departamento de Sistemas Informáticos y Computación,Univ. Politécnica de de Valencia.* Valencia , España : s.n., junio 2002.

J.A. Carsí, I.Ramos,J.Silva,J. Peréz,V. Anaya. 2002.*Un Generador Automático de Planes de Migración de Datos.* Valencia, Valencia, España : s.n., 2002.

López, Alejandro Guerra. 2014.*Análisis, Diseño e Implementación del proceso Administración de Recursos Humanos del Sistema de Gestión de Mantenimiento Vehicular v 1.0.* La Habana, La Habana, Cuba : s.n., junio 2014.

McKee, .*"Maintenance as a Function of Design".* s.l. : Proc. AFIPS National Computer Conf.,pp187-93.

Mozilla Hispano. 2016. Mozilla Hispano. *Mozilla Hispano.* [Online] 9 junio 2016. https://www.mozilla-hispano.org/documentacion/Preguntas_frecuentes_sobre_la_organizaci%C3%B3n_Mozilla.

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

Normalización Oficina Nacional de Transporte automotor. 2011.*Servicio de Transportación de pasajeros y cargas. Términos, definiciones, símbolos y métodos de cálculos.* 2011.

Pacheco, N. 2013. 2015. Manual de Twig. [Online] 2013. *Manual de Twig. [Online] 2013.* [Online] 16 Abril 2015. http://gitnacho.github.io/Twig/ .

Patrones. 2010. Patrones de diseño. [Online] 24 marzo 2010. patronesdediseno.net16.net..

Pérez, Susell Fernández. Junio 2012.*Complemento de Configuración para la gestión de pruebas de carga y estrés a sistemas de gestión web con la utilización automatizada de la herramienta Jmeter.* La Habana,Cuba : s.n., Junio 2012.

Pressman, Roger. 2000.*Ingeniería de Software un enfoque Práctico,6ta edición.* New York : s.n., 2000.

—. 2002.*Ingeniería del Software. Un enfoque práctico 6ta edición.* New York : McGraw-Hill, 2002.

Pressman, Roger S. 2007.*Pressman.* 2007. Vol. VI.

Requerimiento. 2015. Metodología Gestión de Requerimientos. *sites.google.com.* [Online] 10 Junio 2015. https://sites.google.com/site/metodologiareq/capitulo-ii/tecnicas-para-identificarrequisitos-funcionales-y-no-funcionales..

Rodríguez, Tamara. 2015.*Metodología de desarrollo para la Actividad productiva de la UCI.* Universidad de las Ciencias Informáticas, La Habana, La liza : s.n., 2015.

Rosas, Juan Eladio Sánchez. 2016. ExtJS lo bueno, lo malo y lo feo. *Desarrollo en Web.* [Online] 11 junio 2016. http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

Sánchez, Tamara Rodríguez. 2014.*Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana, La Habana, Cuba : s.n., 21 11 2014.

Software Quality.A Framework for Success in Software Development and Support. **Sanders, Joc & Eugene Curran.** Addison Wesley.

T, . 1981.*Entropic thresholding: a new approach.* s.l. : Comput. Graph. Image Process, 1981.

Torres, Abraham Calás. 2016. dragones. *dragones.* [Online] 3 2016. [Cited: 18 Enero 2016.] https://dragones.uci.cu.

Twig. 2016. TWIG, EL MOTOR DE PLANTILLAS PARA PHP. *TWIG, EL MOTOR DE PLANTILLAS PARA PHP.* [Online] 10 junio 2016. http://www.acens.com/wp-content/images/2014/06/twig-plantillas-wpacens.pdf..

UCI. 2014.*Xedro orbita. Sistema de Control de Flotas y Mantenimiento.* La Habana, La Habana, Cuba : s.n., 8 septiembre 2014.

Villa., . 2016. quizzpot.com. *quizzpot.com.* [Online] 12 4 2016. https://quizzpot.com. https://quizzpot.com.

Vivas, Jenny Beatriz Vera. 2015.*DISEÑO DE LA METODOLOGÍA PARA LA MIGRACIÓN DE LOS MÓDULOS HISTORIA CLÍNICA FAMILIAR E HISTORIA CLÍNICA UNITARIA DEL SISTEMA "CERRITOS DE LOS MORREÑOS" .* 2015.

VIVAS, JENNY BEATRIZ VERA. 2015.*DISEÑO DE LA METODOLOGÍA PARA LA MIGRACIÓN DE LOS MÓDULOS HISTORIA CLÍNICA FAMILIAR E HISTORIA CLÍNICA UNITARIA DEL SISTEMA "CERRITOS DE LOS MORREÑOS" .* GUAYAQUIL – ECUADOR : s.n., 2015.

«25. Tools — Doctrine 2 ORM 2 documentation». 2016. Accedido junio 9. http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/reference/tools.html#reverse-engineering.

Abraham Calas. 2016. «Welcome to Boson's documentation». Accedido junio 8. http://docs.prod.uci.cu/online/Boson.docset/Contents/Resources/Documents/docs/index.html.

Aitor Ortuondo. 2014. «Acelera tu web cambiando al nuevo PHP 5.5». *Blog de guebs*. febrero 21. http://blog.guebs.com/2014/02/21/ofrecemos-php5-5-en-hosting-revendedor/.

«- JavaScript - The Definitive Guide.pdf». 2016. Accedido junio 9. https://www.arenahome.org/dir/B%20Per%20imparare%20e%20capire/informatica/musica/JavaScript%20-%20The%20Definitive%20Guide.pdf.

«Apache JMeter - Apache JMeter™». 2016. Accedido junio 7. http://jmeter.apache.org/.

Bray, Tim, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, y François Yergeau. 1998. «Extensible markup language (XML)». *World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210* 16. http://www.w3pdf.com/W3cSpec/XML/2/REC-xml11-20060816.pdf.

Choque Aspiazu, Guillermo. 2001. *Ingeniería de requerimientos*.

Collard, Ross. 2005. «System Performance Testing: A Case Study». Technical Report, Collard & Company, Feb.

«Documentación de Symfony2 — Manual de Symfony2 en Español». 2016a. Accedido junio 9. http://gitnacho.github.io/symfony-docs-es/.

———. 2016b. Accedido junio 9. http://gitnacho.github.io/symfony-docs-es/.

Eguiluz, J. 2013. «Desarrollo web ágil con Symfony2».

Escalona, María José, y Nora Koch. 2002. «Ingeniería de Requisitos en Aplicaciones para la Web–Un estudio comparativo». *Universidad de Sevilla*. https://www.lsi.us.es/docs/informes/LSI-2002-4.pdf.

Española, Real Academia De La Lengua. 2013. *Real academia de la lengua española*. Obtenido de http://lema. rae. es/drae.

Garis, Ana Gabriela, Daniel Eduardo Riesco, y Germán Antonio Montejano. 2006. «Perfiles UML para definición de Patrones de Diseño». En *VIII Workshop de Investigadores en Ciencias de la Computación*. http://sedici.unlp.edu.ar/handle/10915/20784.

Group, P. H. P., y others. 2012. *Php*. PHP Group.

«HowManyThreads - Jmeter Wiki». 2016. Accedido junio 2. http://wiki.apache.org/jmeter/HowManyThreads.

«Información General sobre las Nuevas Características en Apache HTTP Server 2.4 - Servidor HTTP Apache Versión 2.5». 2016. Accedido junio 10. https://httpd.apache.org/docs/trunk/new_features_2_4.html.

«Las bases estratégicas de la informatización cubana | MINCOM». 2016. Accedido junio 20. http://www.mincom.gob.cu/?q=node/803.

Len, Bass, Clements Paul, y Kazman Rick. 2003. «Software architecture in practice». *Boston, Massachusetts Addison*.

«PHP: New features - Manual». 2016. Accedido junio 9. http://php.net/manual/en/migration55.new-features.php.

«PhpStorm IDE :: JetBrains PhpStorm». 2016. *JetBrains*. Accedido junio 9. https://www.jetbrains.com/phpstorm/.

«PostgreSQL 9.2 Documentation - postgresql-9.2-A4.pdf». 2016. Accedido junio 9. https://www.postgresql.org/files/documentation/pdf/9.2/postgresql-9.2-A4.pdf.

«PSR-1: Basic Coding Standard - PHP-FIG». 2016. Accedido junio 18. http://www.php-fig.org/psr/psr-1/.

«PSR-4: Autoloader - PHP-FIG». 2016. Accedido junio 18. http://www.php-fig.org/psr/psr-4/.

Pytel, Pablo, C. Uhalde, Hugo Dionisio Ramón, H. Castello, M. Tomasello, María Florencia Pollo Cattaneo, Paola Verónica Britos, y Ramón García Martínez. 2011. «Ingeniería de requisitos basada en técnicas de ingeniería del conocimiento». En *XIII Workshop de Investigadores en Ciencias de la Computación*. http://sedici.unlp.edu.ar/handle/10915/20070.

«Snapshot». 2016. Accedido junio 9. http://dl.acm.org/citation.cfm?id=526978.

«Software Design Tools for Agile Teams, with UML, BPMN and More». 2016. Accedido junio 9. https://www.visual-paradigm.com/.

«Symfony, High Performance PHP Framework for Web Development». 2016. Accedido junio 8. https://symfony.com/.

Víctor,Puertas. 2013. «PHP 5.5 RC1 y actualizaciones en las ramas 5.3.x y 5.4.x». *YoSymfony*. mayo 9. http://yosymfony.com/php-5-5-rc1-y-actualizaciones-en-las-ramas-5-3-x-y-5-4-x/.

«Welcome to Boson's documentation! — Boson 0.1 documentation». 2016. Accedido junio 8. http://docs.prod.uci.cu/online/Boson.docset/Contents/Resources/Documents/docs/index.html.

ANEXOS

Anexo 1: Prueba de rendimiento al Sistema Viejo con la herramienta jmeter.

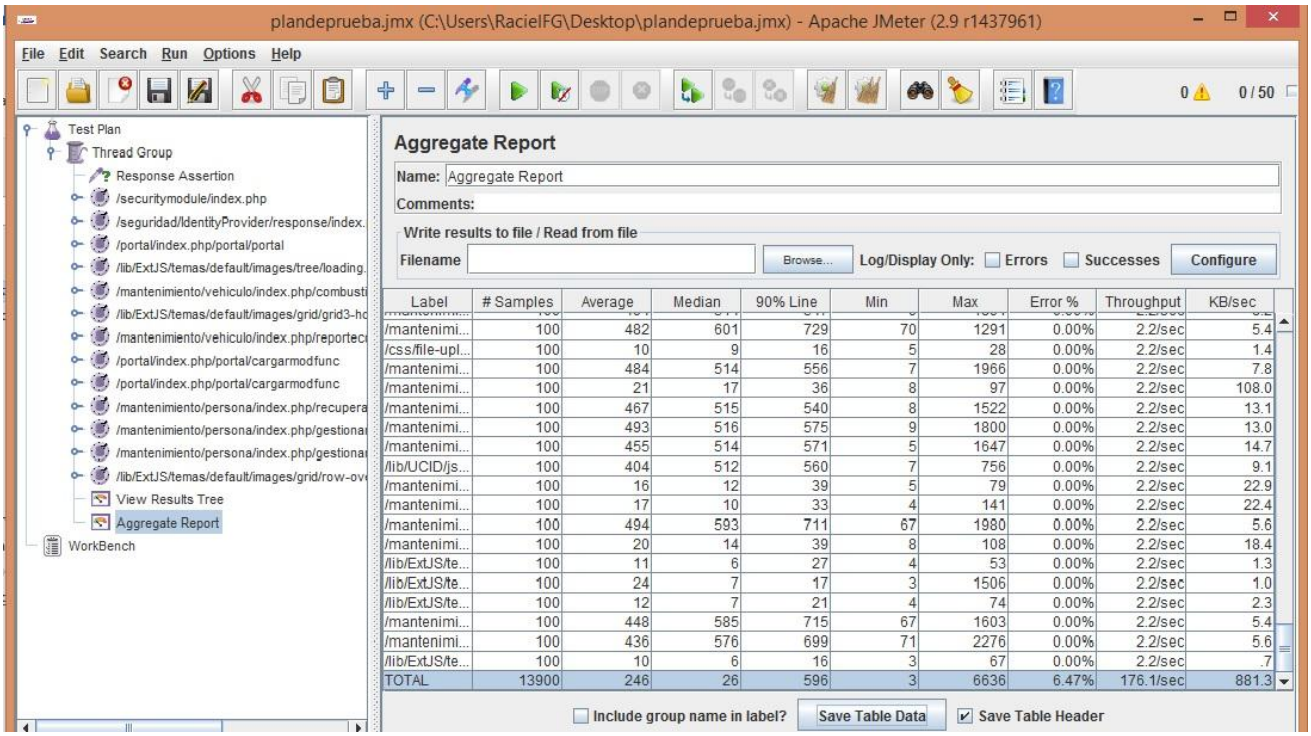


Figura 22 Prueba de rendimento al sistema viejo

Anexo 2: Prueba de rendimiento al Sistema Nuevo con la herramienta jmeter.

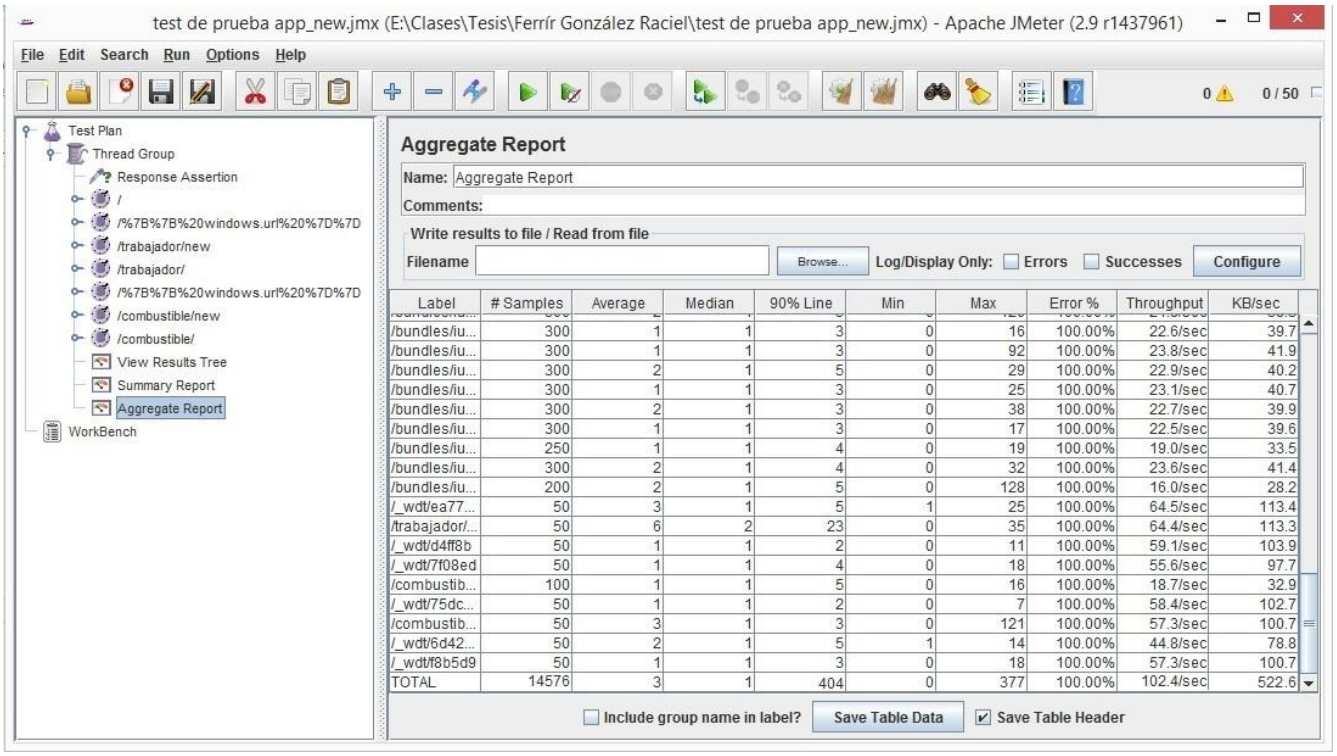


Figura 23 Prueba de rendimiento al Sistema Nuevo con la herramienta jmeter

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, Persona y Combustible. 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 es de:
   _ 0 a 12segs __de 13 a 27segs __más de 27segs
3. Tiempo de respuesta de las demás páginas.
   _de 0 a 7 segs _de 8 a 12 segs _más de 13 segs
4. ¿Se cargan correctamente todos los componentes visuales del sistema?
   _Sí _ No
   ¿En caso de NO, cuáles no se cargan? _____
   _____
   _____
   _____
5. ¿Se cargan correctamente los elementos de las listas?

   _Sí _No

¿En caso de NO, cuáles no secargan?

_____

_____

6. ¿El sitio evita que los usuarios se registren de manera innecesaria?
   _____
   _____
7. ¿En qué criterio se basó para elegir EXTJs en su versión 3.4 y Sauxe en su versión 2.0 para el desarrollo del sistema?
   _____
   _____

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

   Sí_ No_

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

   Sí _ No_

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

    secciones?

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

    Sí_ No_

12. ¿El sitio provee una clara retroalimentación cuando una tarea ha sido

completada exitosamente?

Sí _No_

13. ¿Los campos en los formularios contienen ayudas, ejemplos o modelos

de respuestas para demostrar el dato que se debe introducir?

Sí_ No_

14. ¿La interfaz de búsqueda está ubicada donde los usuarios esperan

encontrarla (en la parte superior derecha de la página)?

Sí_ No_

¿En caso de NO, dónde se encuentra ubicada?

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

_____

_____

_____

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

_____

_____

_____

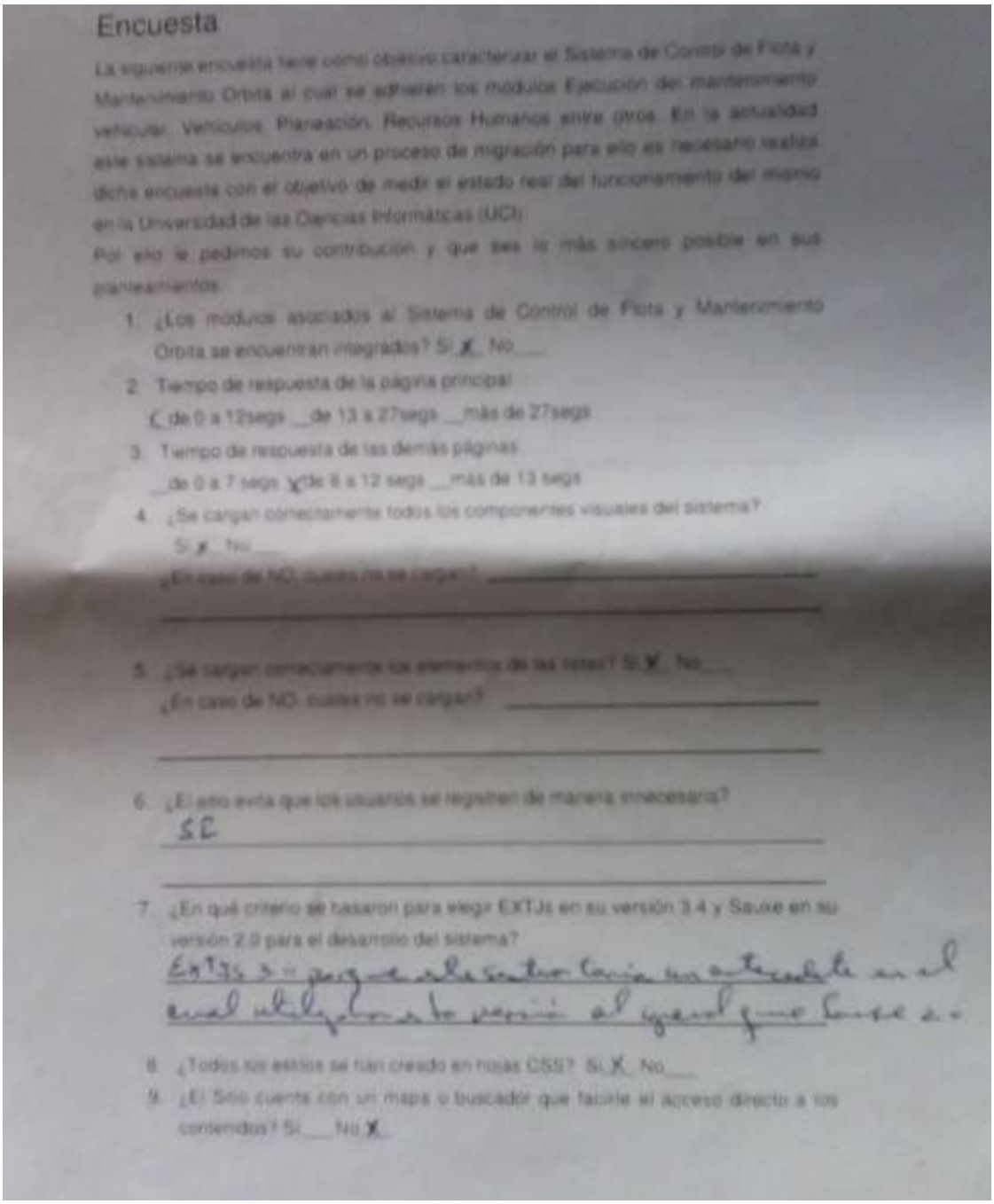
17. Al sistema Orbita se le implementó un sistema de auditorías al código fuente o al cumplimiento de las definiciones arquitectónicas. Argumente.

_____

_____

_____

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

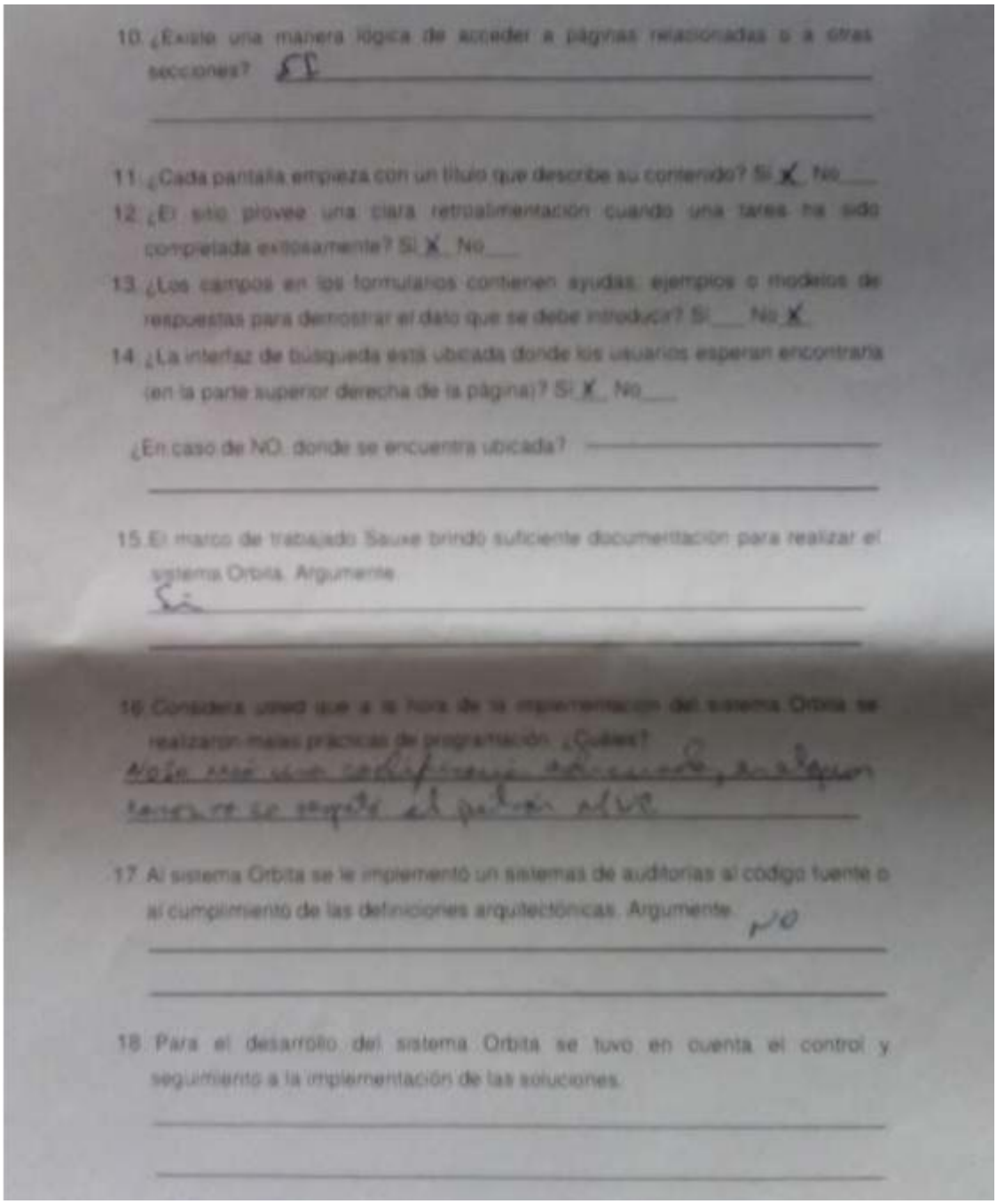


Encuesta

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

Por ello le pedimos su contribución y que sea lo más sincero posible en sus planteamientos.

1. ¿Los módulos asociados al Sistema de Control de Flota y Mantenimiento Orbita se encuentran integrados? Si X No___
2. Tiempo de respuesta de la página principal
   X de 0 a 12segs __de 13 a 27segs __más de 27segs
3. Tiempo de respuesta de las demás páginas
   __de 0 a 7 segs X De 8 a 12 segs __más de 13 segs
4. ¿Se cargan correctamente todos los componentes visuales del sistema?
   Si X No___
   ¿En caso de NO, cuales no se cargan? _____
5. ¿Se cargan correctamente los elementos de las tablas? 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 porque la tecnología es estable en el cual ubicaron esta versión al igual que Sauxe 2.
8. ¿Todos los estilos se han creado en hojas CSS? Si X No___
9. ¿El Sitio cuenta con un mapa o buscador que facilite el acceso directo a los contenidos? Si___ No X



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

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

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

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

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

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

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

Si

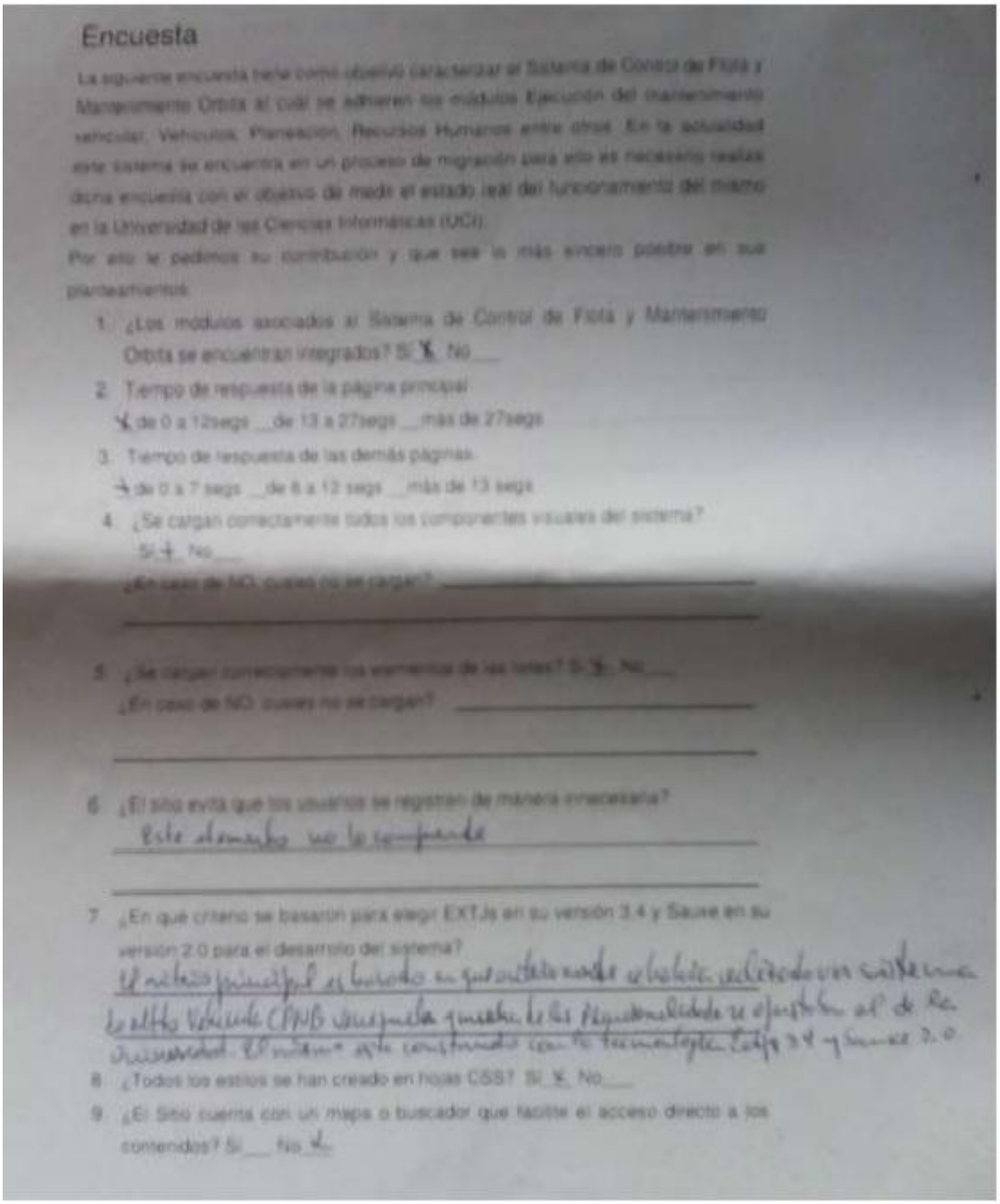
16. Considera usted que a la hora de la implementación del sistema Orbita se realizaron malas prácticas de programación. ¿Cuáles?

No se muy bien explicarme adecuadamente, a alguien

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.

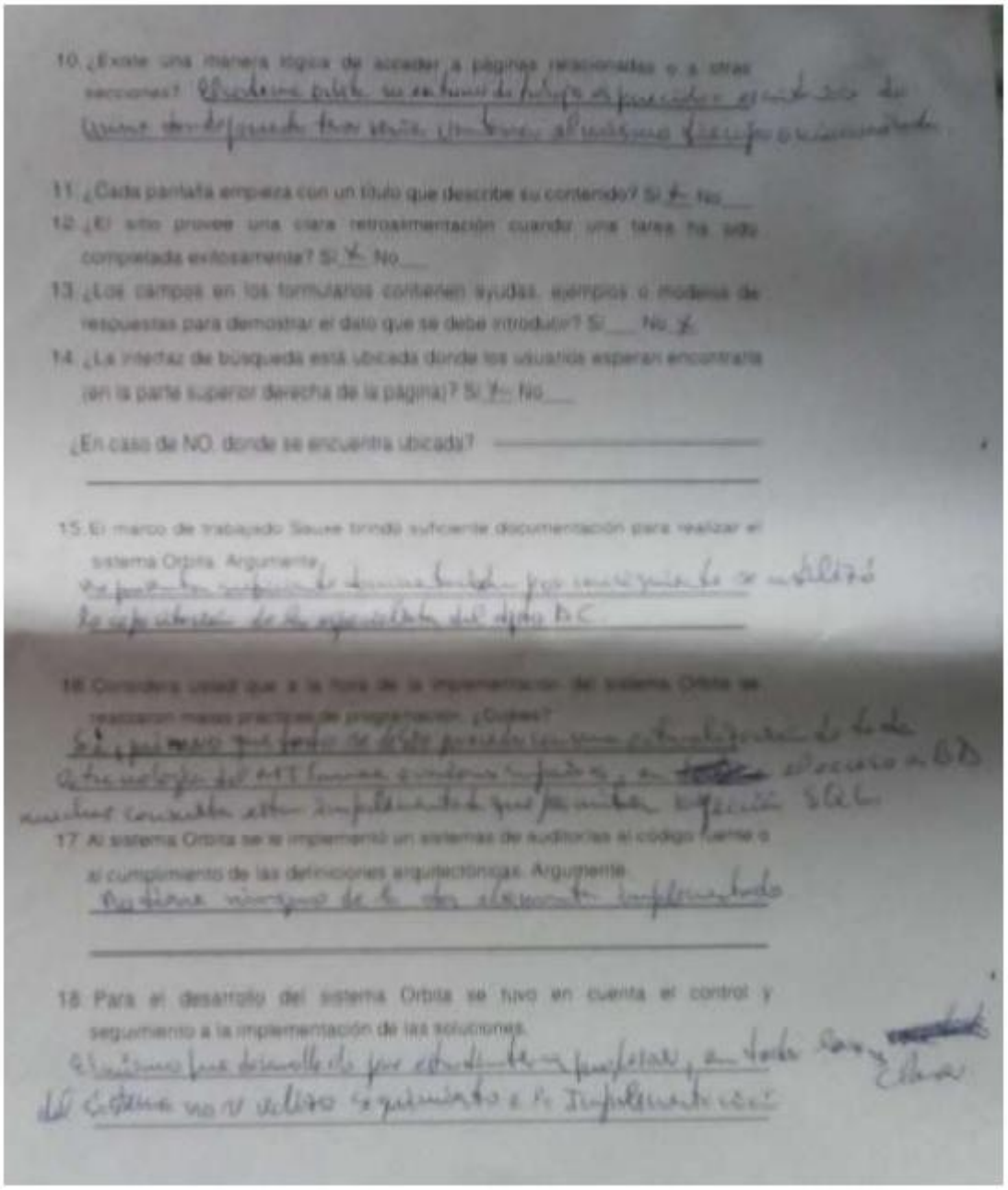


Encuesta

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 realizar dicha encuesta con el objetivo de medir el estado real del funcionamiento del mismo en la Universidad de las Ciencias Informáticas (UCI).

Por ello le pedimos su contribución y que sea lo más sincero posible en sus planteamientos.

1. ¿Los módulos asociados al Sistema de Control de Flota y Mantenimiento Orbita se encuentran integrados? Si X No___
2. Tiempo de respuesta de la página principal
   X de 0 a 12segs __de 13 a 27segs __más de 27segs
3. Tiempo de respuesta de las demás páginas
   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?
   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?
   Este demasiado no la comprendo
   _____
7. ¿En qué criterio se basaron para elegir EXTJs en su versión 3.4 y Sauve en su versión 2.0 para el desarrollo del sistema?
   El criterio principal es basado en que existe mucha información sobre el framework Extjs en su versión 3.4 y Sauve 2.0 porque la Universidad El sistema está construido con la tecnología Extjs 3.4 y Sauve 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? Existe una barra de búsqueda en la parte superior de cada una de las páginas donde se encuentra el sistema.

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 Sauve brindó suficiente documentación para realizar el sistema Orbita. Argumente

Se pudo contar con la documentación proporcionada por el cliente y la especificación de los requisitos del ciclo RUP.

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

Se pudo notar que hubo malas prácticas de programación en la arquitectura del MVC donde algunas consultas a la BD se encontraban en el controlador en vez de estar en el modelo.

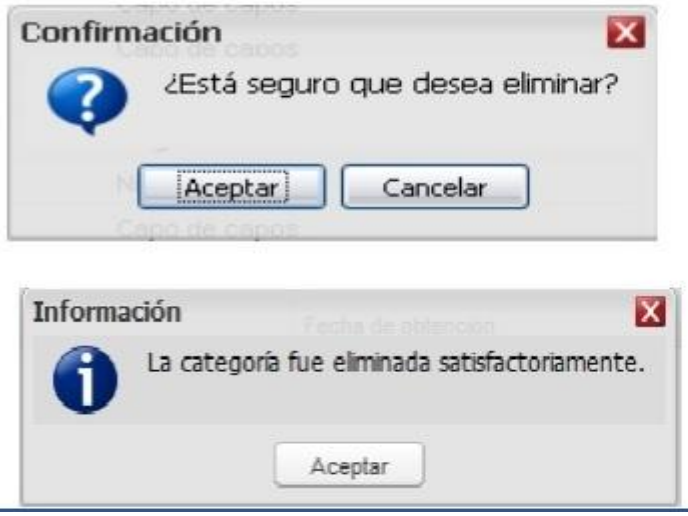
17. Al sistema Orbita se le implementó un sistemas de auditorías al código fuente o al cumplimiento de las definiciones arquitectónicas. Argumente

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 sistema fue desarrollado por estudiantes practicantes, en todo caso el sistema no tuvo ningún seguimiento a la implementación.

Anexo 4: Historia de Usuario del requisito Eliminar Categoría de Licencia

Historia de Usuario	
Número: HU_14	**Nombre del requisito:** Eliminar Categoría de Licencia
Programador: Raciel Ferrin González	**Iteración Asignada:** 4 días
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: Poca experiencia por parte de los estudiantes con las tecnologías de desarrollo	**Tiempo Real:** 7 horas
Descripción: Permitirá eliminar de forma permanente del sistema la categoría de licencia asociada a una personas seleccionadas previamente por el usuario.	
Observaciones: N/A	
Prototipo de interfaz: 	

Anexo 5: Historia de Usuario del requisito Adicionar Categoría de Licencia

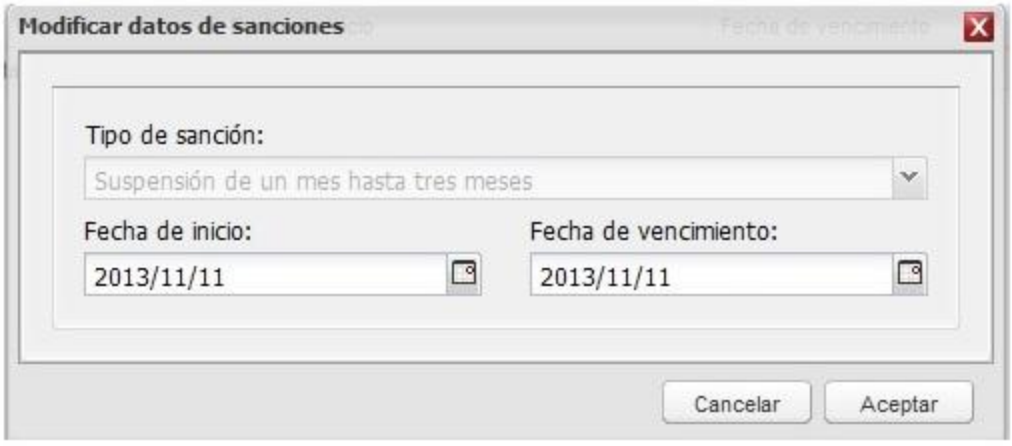
Tabla 8 Historia de usuario Adicionar Categoría de Licencia

Historia de Usuario	
Número: HU_6	**Nombre del requisito:** Adicionar Categoria Licencia
Programador: Raciel Ferrin González	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 4 días
Riesgo en Desarrollo: Poca experiencia por parte de los estudiantes con las tecnologías de desarrollo	**Tiempo Real:**4 días
Descripción: permitirá adicionar a la persona una categoría de licencia después de haber insertado cada uno de los parámetros necesarios.	
Observaciones: N/A	

Prototipo de interfaz:

Anexo 6: Historia de Usuario del requisito Modificar Sanción

Tabla 9 Historia de usuario Modificar Sanción

Historia de Usuario	
Número: HU_12	**Nombre del requisito:** Modificar Sanción
Programador: Raciel Ferrin González	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 7 días
Riesgo en Desarrollo: Poca experiencia por parte de los estudiantes con las tecnologías de desarrollo	**Tiempo Real:** 4 días
Descripción: Permitirá modificar las sanciones de la persona seleccionada después de haber insertado cada uno de los parámetros necesarios.	
Observaciones: N/A	
Prototipo de interfaz: 	

Anexo 7: Diagrama de Caso de Prueba Adicionar Persona Datos Licencia.

Tabla 10 Diagrama de Caso de Prueba Adicionar Persona Datos Licencia

Escenario	Descripción	Categoría	Fecha de Obtención	Fecha inicio	Fecha fin	puntos	Respuesta del sistema	Flujo central
EC 1.1 Adicionar persona datos licencia	Se adicionan nuevas personas datos licencia correctamente	V	V	V	V	V	El sistema almacena los datos y nos redireciona para la página chequeo médico	Se rellenan los campos mostrados -Se presiona el botón Acepta -Se validan los datos
		NA	26/05/2016	92136565431	Facultad 3	Especialista Superior		
EC 1.2 Adicionar personas datos licencia dejando campos vacíos	No se insertan parámetros en los campos mostrados al adicionar nuevas personas datos licencia	I	V	V	V	V	El sistema muestra los errores cometidos en una alerta y que se debe insertar algún parámetro	Se rellenan los campos mostrados -Se presiona el botón Aceptar
			24/04/2016	24/04/2016	04/05/2016	23		
		V	I	V	V	V		
		NA		12/04/2016	Facultad 3	32		
		V	V	I	V	V		
		NA	24/04/2016		04/05/2016	56		
		V	V	V	I	V		
		NA	12/04/2016	24/04/2016		34		
		V	V	V	V	I		
		NA	26/05/2016	24/04/2016	04/05/2016			
EC 1.2 Adicionar personas datos licencia caracteres inválidos	Se insertan parámetros inválidos en los campos mostrados al adicionar nuevas personas datos licencia	V	V	I	V	I	El sistema muestra los errores en una alerta indicando contiene carácteres inválidos indicando que debe	Se rellenan los campos mostrados -Se presiona el botón Aceptar -Se validan los datos
		NA	26/05/2016	921asd65431	Rectorado			
		V	I	V	V	V		
		NA	26 junio	04/03/2016	04/05/2016	5		
		V	V	I	V	V		
		NA	24/04/2016	"js21	04/03/2016	32		

Anexo 8: Diagrama de Caso de Prueba Exportar reporte del análisis de rendimiento energético.

Tabla 11 Diagrama de Caso de Prueba Exportar reporte del análisis de rendimiento energético

Escenario	Descripción	Tipo de combustibl	Fecha Inicio	Fecha fin	Respuesta del sistema	Flujo central
EC 1.1 Exportar reporte del análisis de rendimento energético	Se exporta el reporte el correspondiente al análisis de intensidad energética en una fecha determinada previamente seleccionado por el usuario	V Gazol	V 12/01/2015	V 12/11/2015	El sistema exporta el reporte generado del historial en formato PDF en la ruta definida por el usuario	Se hace click en Reportes localizado en la barra de menú -Se seleccionan la opción análisis de intensidad energética -Se seleciona el tipo de combustible y las fechas corresponndientes. -Se presiona el botón Aceptar -Se validan los datos.
EC 1.2 Exportar reporte del análisis de rendimento energético con campos vacíos	Se exporta el reporte el correspondiente al análisis de rendimento energético en una fecha determinada previamente seleccionado por el usuario	I	V 12/11/2015	I 12/01/2015	El sistema muestra una notificación al usuario indicándole que faltan o hay error en los datos	Se hace click en Reportes localizado en la barra de menú -Se seleccionan la opción análisis de intensidad energética -Se seleciona el tipo de combustible y las fechas corresponndientes. -Se presiona el botón Aceptar
		V Gazol	I	V 12/01/2015	El sistema muestra una notificación al usuario indicándole que faltan o hay error en los datos	
		V Gazol	V 12/11/2015	I	El sistema muestra una notificación al usuario	

Por la cantidad de productos de trabajos generados solo se muestra una pequeña parte en la presente investigación, para consultar los demás productos de trabajo revisar el Expediente de Proyecto.

Anexo 9:*Caso de prueba para el camino básico 2, 3, 4*.

Tabla 12 Caso de prueba, camino básico 2

Caso de prueba para el camino básico 2
Descripción: Trata los errores del formulario como campos vacíos

Condición de ejecución: Si el objeto request ha sido enviado por el método POST y el formulario recibido tiene errores	
Procesamiento de la prueba	
Dato de entrada:	objeto request, Idtrabajador, editando
Tipo de dato esperado:	objeto request sea enviado por el método POST, idtrabajador, editando
Resultados esperados:	Se redireciona para un formulario de datosLicencia con los errores encontrados

Tabla 13Caso de prueba, camino básico 3

Caso de prueba para el camino básico 3	
Descripción: Trata el error de que la fecha de inicio sea mayor que la fecha fin	
Condición de ejecución: Si el objeto request ha sido enviado por el método POST y el formulario recibido no tiene errores y las fecha de inicio es mayor que la fecha de fin(datos de las escuelas de recalificación)	
Procesamiento de la prueba	
Dato de entrada:	objeto request, Idtrabajador, editando
Tipo de dato esperado:	objeto request sea enviado por el método POST, idtrabajador, editando
Resultados esperados:	Se redireciona para un formulario de datosLicencia con el error encontrado

Tabla 14 Caso de prueba, camino básico 4

Caso de prueba para el camino básico 4	
Descripción: Se espera un objeto request que haya sido enviado por el método POST	
Condición de ejecución: Si el objeto request ha sido enviado por el método POST y el formulario recibido no tiene errores y las fecha de inicio no es igual que la fecha de fin(datos de las escuelas de recalificación)	
Procesamiento de la prueba	
Dato de entrada:	Objeto request, Idtrabajador, editando
Tipo de dato esperado:	Objeto request sea enviado por el método POST, idtrabajador, editando
Resultados esperados:	Se redireciona para un formulario de datosLicencia con el error encontrado