Universidad de las Ciencias Informáticas

Facultad 3

Migración de los módulos Configuración, Clasificadores, y el proceso Vehículo del sistema Xedro-Orbita a la arquitectura de referencia ODOO

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

Autores: Esmirna Milians Recio

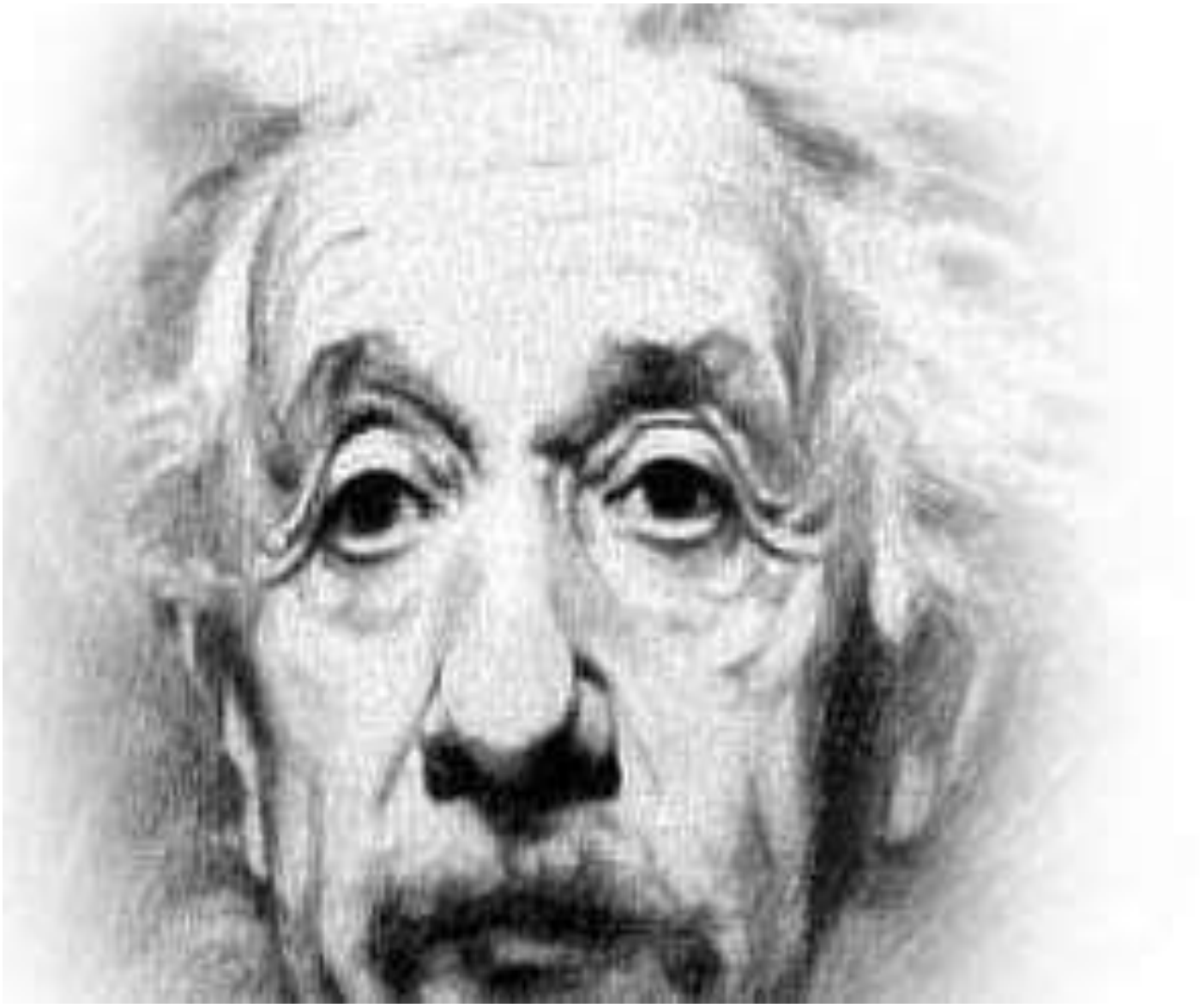
Jibsan Joel Rosa Toirac

Tutor: Ing. Virtudes Milagro Figueredo Lara

Co-tutor: Ing. Ernesto Mató Roque

Junio 2018

"Año 60 de la Revolución"



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

Albert Einstein

Declaración de autoría

Declaramos ser los autores de este trabajo titulado: Migración de los módulos Configuración, Clasificadores y el proceso Vehículo del sistema Xedro-Orbita a la arquitectura de referencia Odoo, y se otorga al Centro CEIGE de la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste se firma la presente a los ____ días del mes de _____ del año_____.

Esmirna Milians Recio

_____

Autor

Jibsan Joel Rosa Toirac

_____

Autor

Ing. Virtudes Milagro Figueredo Lara

_____

Tutor

Ing. Ernesto Mató Roque

_____

Co_Tutor

Agradecimientos

Esmirna Milians Recio

A mis padres por existir, por darme la vida, por educarme, por hacer de mí la persona que soy hoy, por brindarme su apoyo incondicional, por estar siempre a mi lado, y por tantas cosas más que no cabrían en mil páginas.

A mi familia en general, en especial a mi hermano Ricardo que lo quiero muchísimo, a mi abuelita Juana, y a todos en general por ser la familia más unida del mundo.

A mi pareja Alexander, que a pesar de los problemas y dificultades me ha ayudado en el transcurso de todos estos años.

A todas las amistades, que he tenido la oportunidad de conocer durante el trayecto de mi carrera, en especial a mi amiga Yuliet, que me brindó su hombro cuando lo necesité, me aconsejó para no fracasar en la vida, y de ella me llevo una gran amistad para toda la vida, pues personas como ella, existen pocas. Entre otras amistades están: la loca de Dianeliz, con la cual siempre estaba discutiendo; Robinson y Yudiel, amigos de fiesta; Omarito, con el que me desahogaba cada vez que tenía un problema y me aconsejaba.

A mi tutora Virtudes, que me ayudó incondicionalmente y le voy a estar agradecida por el resto de la vida; a Ernesto Mato, que me ayudó muchísimo en el desarrollo de la tesis, y al tribunal , que de una forma u otra, contribuyeron a mi formación como profesional.

A Jibsan por ser mi amigo y compañero de tesis, por todo su apoyo incondicional en todo este tiempo.

Agradecimientos

Jibsan Joel Rosa Toirac

A mi papá y mi hermano por sus sacrificios en estos 5 años para poder permitirme estudiar esta carrera.

A mi familia tanto dentro y fuera de Cuba por su apoyo y contribución a que este sueño se volviera realidad.

A mis tías, tíos y abuelos por acogerme cada vez que hizo falta.

A todos los amigos que me brindaron su apoyo para que pudiera seguir adelante.

A mi mamá quien falleció hace 5 años, especialmente quiero agradecerle y dedicarle esta tesis porque gracias a ella soy quien soy en este día y sus enseñanzas vivirán eternamente conmigo.

A mi tutora Virtudes por su gran ayuda y dedicación en el desarrollo de la tesis.

A Mató y Yaicel por todo su tiempo invertido y su aporte a la conclusión de esta tesis.

A Esmirna por ser mi amiga y compañera de tesis, por todo su apoyo incondicional en todo este tiempo.

Dedicatoria

A nuestras familias por su amor, dedicación y apoyo incondicional.

A la UCI, Fidel y la Revolución.

Resumen

Un proceso de desarrollo de software para su práctica debe regirse por guías, modelos, metodologías y/o marcos de trabajo que garanticen la implementación de un producto de alta calidad. Es la ingeniería de software quien fundamenta, define y establece las bases para realizar una implementación viable de acuerdo a las necesidades de los clientes. Con el propósito de contribuir efectivamente a una coherencia adecuada entre lo que se especifica y representa, con una migración estructurada y organizada de tecnologías, la presente investigación persigue como objetivo migrar los módulos Configuración, Clasificadores y el proceso Vehículo del Sistema Xedro-Orbita, del marco de trabajo Sauxe para la arquitectura de referencia Odoo. Estos módulos forman parte del control de un parque vehicular, en los cuales se realizan actividades como: gestión de usuarios, accesorios, documentos técnicos, grupo de vehículos, herramientas, repuestos, rutas, baterías, combustible, mantenimiento, neumáticos, vehículos, entre otros. La realización de estas actividades contribuyen a la generación de productos de trabajo ingenieriles, como salidas fundamentales a partir del uso de la metodología AUP-UCI, que combina las buenas prácticas del nivel 2 del Modelo de Madurez de la Capacidad Integrado. La validez de la migración fue comprobada con la ejecución de pruebas internas y pruebas de aceptación, obteniendo para ambos casos, las actas de aceptación que así lo acreditan. Además, fue demostrada y comprobada la mejoría en cuanto a rendimiento y seguridad de la migración del sistema con respecto al anterior, a través de los resultados obtenidos al aplicar las herramientas Jmeter y Acunetix.

PALABRAS CLAVE

Palabras claves: mantenimiento, migración, módulos, rendimiento, seguridad.

Tabla de Contenido

Tabla de Contenido

Índice de Figuras

Índice de Tablas

Introducción

INTRODUCCIÓN

Las empresas desarrolladoras de software independientemente de su tamaño, capital o presencia en el mercado persiguen un objetivo común: desarrollar software de calidad a un costo adecuado y en tiempo reducido. Este es un reto difícil de lograr debido a que las demandas de los clientes en muchos casos son superiores a las capacidades productivas de las empresas. La solución a este conflicto se ha buscado en todas las direcciones: se han perfeccionado los procesos y las metodologías de desarrollo, se trabaja en la capacitación de los recursos humanos y constantemente se desarrollan nuevas tecnologías y marcos de trabajo.

Cuba no se ha quedado atrás en el avance y desarrollo de las Tecnologías de la Información y las Comunicaciones (TICS), las cuales han evolucionado y están cambiando el modo tradicional de realizar las actividades. Las empresas de transporte no están exentas de estos cambios, la utilización de sistemas de control de flotas le permiten a las mismas ahorrar dinero por concepto de reducción del uso de combustible, reducción de distancias y manejo de cargas. Estas empresas además buscan mantener un control sobre la información mecánica de los vehículos, su estado técnico y la gestión de su mantenimiento.

La informatización del control de las flotas en Cuba fue una tarea que se comenzó en el año 2006. La misma utiliza tecnología GPS[1] y posee una aplicación Web central combinada con aplicaciones locales. El proyecto ha alcanzado actualmente a más de 25 bases de transporte y más de 1 090 medios instalados reportando un ahorro de combustible promedio de más del 15%. Los proyectos que componen esta informatización del control de flotas según la Oficina Nacional de Estadísticas (ONE, 2007) son:

- Móvil Web.
- CartoSIG.
- Infraestructura Informática.

En la actualidad son numerosas las instituciones que se dedican al desarrollo de software, entre las que se encuentra la Universidad de las Ciencias Informáticas (UCI). Dentro de esta, radican proyectos productivos que tienen como tarea principal la producción de aplicaciones y servicios informáticos, a partir de la vinculación estudio - trabajo como modelo de formación. La UCI, como apoyo a la política de

[1] Global Positioning System(Sistema de Posicionamiento Global).

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 en el país. En la facultad 3, 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.

En esta universidad se desarrolló el Sistema de Control de Flota y Mantenimiento de la Dirección de Transporte de la UCI, el cual permite gestionar los módulos relacionados con la gestión del mantenimiento que se desarrollan 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 utilización 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 rutas, control de combustible, autorizo de parqueo, día de la técnica control de neumáticos y baterías, todo esto de forma centralizada, eficiente y segura.

Xedro-Orbita es un sistema de gestión a través del cual se puede manejar toda la información referente a los módulos de mantenimiento preventivo planificado y correctivo de una flota de vehículos, sin importar el tamaño de esta y garantizando una correcta planificación, organización y control en la gestión del mantenimiento de sus vehículos.

La tecnología empleada para el desarrollo del sistema fue Sauxe en su versión 2.2, este marco de trabajo en la capa presentación utiliza la actualización de ExtJs 3.4. Aunque se empleó dicha actualización, el mismo se encuentra atado a una licencia de pago para su uso comercial (Sencha, 2016).

En la capa de negocio hace uso del marco de trabajo Zend Framework 1.2 que no permite la generación automática de CRUD[2], la generación de código PHP[3] mediante líneas de comandos para la creación, mantenimiento de aplicaciones y el almacenamiento en la caché de las vistas. Dichas características deben tenerse en cuenta para un desarrollo ágil y un mayor rendimiento del sistema (Zend, 2016).

En la capa de acceso a datos hace uso de Doctrine 1.2.2 lo cual provoca bajo rendimiento y un mayor acoplamiento entre sus clases, por lo que se pierde la independencia y es más costoso a la hora del cambio hacia otra tecnología. Otro inconveniente que presenta el uso de esta versión de Doctrine es que a

[2] CRUD es el acrónimo de “Crear, Leer, Actualizar y Borrar” (del original del inglés: Create, Read, Update, and Delete).

[3] Hypertext Preprocessor(Procesador de Hipertexto).

partir del 1 de junio del 2011 se dejó de dar soporte, trayendo consigo que no se corrijan posibles errores y vulnerabilidades que atentan contra su estabilidad y rendimiento (Doctrine, 2009).

Se identificaron además, varios factores subjetivos que influyen negativamente en el proceso de desarrollo como por ejemplo:

- No se le brindará más soporte al marco de trabajo Sauxe, por tanto ya no se actualizarán las tecnologías utilizadas en su desarrollado. Esto trae consigo que existan brechas de seguridad en los sistemas desarrollados y la información almacenada.
- Poco control y seguimiento a la implementación de las soluciones.
- No se implementa un sistema de auditorías al código fuente o al cumplimiento de las definiciones arquitectónicas.
- Uso de bibliotecas no integradas al marco de trabajo.

Por tal motivo se realiza el proceso de migración de los módulos Configuración, Clasificadores y el proceso Vehículo del Sistema Xedro-Orbita a la arquitectura de referencia ODOO[4], pues provee un conjunto de funcionalidades y módulos para gestionar cada parte de la organización y la sintetiza como si fuera un todo, permitiendo la organización, el análisis y la toma de decisiones.

A partir de la problemática antes planteada se define como **problema a resolver**: ¿Cómo contribuir a mejorar el rendimiento y la seguridad de los módulos Configuración, Clasificadores y el proceso Vehículo del Sistema Xedro-Orbita?

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

Objetivo general: Realizar la migración de los módulos Configuración, Clasificadores y el proceso Vehículo del Sistema Xedro-Orbita a la arquitectura de referencia ODOO, para contribuir a la mejora del rendimiento y la seguridad.

El **campo de acción** se enmarca en: Migración de tecnologías de desarrollo en sistemas de mantenimiento.

Para dar cumplimiento al objetivo general planteado se han definido los siguientes **objetivos específicos**:

[4] Odoo, es un sistema de planificación de recursos empresariales (ERP), integrado de código abierto actualmente producido por la empresa belga Odoo S.A.

- Realizar un estudio del estado del arte para la elaboración del marco teórico teniendo en cuenta el objeto de estudio.
- Redefinir la ingeniería de requisitos de los módulos Configuración, Clasificadores y el proceso Vehículo del Sistema Xedro-Orbita.
- Rediseñar la estructura de componentes y clases, aplicando los patrones de diseños y métricas definidas en el estudio realizado.
- Implementar los módulos Configuración, Clasificadores y el proceso Vehículo del Sistema Xedro-Orbita, utilizando la arquitectura de referencia ODOO.
- Validar la migración del sistema utilizando pruebas de caja blanca, caja negra, pruebas internas y pruebas de aceptación; y las pruebas de rendimiento y pruebas de seguridad para validar la investigación.

Se define como **Idea a defender**: Si se realiza la migración de los módulos Configuración, Clasificadores y el proceso Vehículo del Sistema Xedro-Orbita a la arquitectura de referencia ODOO, se contribuye a la mejora de su rendimiento y seguridad.

Posibles resultados: Módulos Configuración, Clasificadores y el proceso Vehículo del Sistema Xedro-Orbita sobre la arquitectura de referencia ODOO.

Métodos teóricos:

- **Histórico-Lógico**: a través de este método se realizó un estudio de la teoría conocida hasta el momento, y el análisis de la trayectoria completa de los módulos Configuración, Clasificadores y el proceso Vehículo del Sistema Xedro-Orbita, obteniendo las características que hacen necesaria la migración de dichos módulos.
- **Analítico-Sintético**: se utilizó con el fin de analizar el comportamiento de los módulos Configuración, Clasificadores y el proceso Vehículo del sistema Xedro-Orbita, sintetizando los elementos importantes para realizar el proceso de migración hacia la arquitectura de referencia Odoo.
- **Modelación**: este método permitió crear una representación del flujo de las principales actividades de los módulos Configuración, Clasificadores y el proceso vehículo del Sistema Xedro-Orbita

permitiendo un mejor entendimiento por parte de los autores de la presente tesis. Además su aplicación se evidenció en la elaboración de diagramas, mapas conceptuales y prototipos durante el desarrollo del sistema.

Métodos empíricos:

- **Medición**: se utiliza en las técnicas aplicadas para validar los requisitos; el código, mediante las métricas TOC (tamaño operacional de clases), y RC (relaciones entre clases); y en las pruebas realizadas al sistema para verificar y validar su correcto funcionamiento.

El documento está compuesto por los siguientes capítulos:

Capítulo 1. Fundamentación teórica

En este capítulo se definen los elementos teóricos que sustentan la investigación y se realiza un estudio y análisis sobre los módulos Clasificadores, Configuración, y el proceso Vehículo del sistema Xedro-Orbita. Se estudian las necesidades actuales que permitirán una migración correcta del sistema; y además se identifican las herramientas, tecnologías y la metodología a utilizar para su desarrollo.

Capítulo 2. Migración del Sistema Xedro-Orbita

En el mismo se efectúa el proceso de migración de los módulos Configuración, Clasificadores y el proceso Vehículo al marco de referencia Odoo. Se hace alusión a las técnicas, mediante las cuales se validaron los requisitos redefinidos. Se obtienen los artefactos generados durante las Disciplinas: Requisitos, Análisis y diseño; e Implementación de la solución. Así como las métricas utilizadas para validar el diseño, las cuales aseguran el correcto funcionamiento de las funcionalidades del sistema.

Capítulo 3. Validación de la migración del Sistema Xedro-Orbita

Se realizan las pruebas de caja blanca, caja negra, pruebas internas y pruebas de aceptación para validar funcionalmente la propuesta de solución. Para validar la investigación se aplican las pruebas de rendimiento y pruebas de seguridad, las cuales permiten garantizar que el sistema migrado se ejecute a gran velocidad demostrando buen rendimiento; y además evidencie seguridad perfeccionando las vulnerabilidades existentes en el desarrollo del sistema con el marco de trabajo Sauxe.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

El presente capítulo se realiza un análisis y estudio sobre los módulos Configuración, Clasificadores, y el proceso vehículo del Sistema Xedro-Orbita. Este estudio permitirá apoyar el proceso de migración del marco de trabajo Sauxe a la arquitectura de referencia Odoo. Se estudian la metodología, arquitectura, patrones, tecnologías y herramientas a utilizar, que constituyen el basamento teórico necesario para el desarrollo de la propuesta de solución. Además se analizan una plataforma, un estándar y una guía para la migración de software, las cuales sustentarán el proceso de migración entre tecnologías.

1.1 Conceptos básicos

La sustitución de viejas tecnologías y herramientas por elementos más modernos se realiza fundamentalmente para mejorar uno o varios aspectos del software. La modificación de un software para hacer que algún aspecto del mismo funcione de manera más eficiente y/o utilice menos recursos (mayor rendimiento) es el principal objetivo de esta investigación.

Los recursos informáticos no son infinitos, por lo tanto la utilización de nuevas tecnologías permitirían disminuir el consumo de los mismos. En la investigación se hará uso de los dos conceptos (Migración y Rendimiento), ya que abarcan y expresan una definición con todo lo relacionado con un sistema informático.

Migración

La migración es el traslado de una aplicación de un ordenador a otro en condiciones de compatibilidad. Es elevar una versión de un producto de software a otra de más alto nivel, o bien el movimiento de una arquitectura a otra (Mastermagazine, 2015).

Migrar es el paso de los programas, archivos y datos de un sistema desde una determinada plataforma tecnológica a otra diferente (Fernández Pacheco, 2010).

En tecnología de la información, la migración es el proceso de pasar de la utilización de un único entorno operativo a otro entorno operativo, es decir, en la mayoría de los casos, se piensa que es mejor. La migración puede implicar la mudanza hacia el nuevo hardware, hacia el nuevo software, o ambos. La

migración puede ser a pequeña escala, tales como la migración de un sistema único, o en gran escala, con la participación de muchos sistemas, nuevas aplicaciones, o una red rediseñado (TechTarget, 2016).

Teniendo en cuenta las definiciones aportadas por estos autores se constata que coinciden en que migración es el traslado de una aplicación informática tanto en hardware como en software, persiguiendo una mejora en su desempeño. En el presente trabajo se adopta el concepto aportado por (Mastermagazine, 2015), el mismo se ajusta a las necesidades de la investigación como se describe en las secciones siguientes.

Rendimiento

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

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

El estudio de bibliografías de referencia en esta área del conocimiento, permite identificar que no existe una clasificación única que defina rendimiento; por lo que los autores de la presente investigación establecen que el rendimiento será medido como el tiempo de respuesta de las funcionalidades del sistema y se inclinan por el concepto propuesto por (Jacobson, y otros, 1999).

Mantenimiento Correctivo

Mantenimiento Correctivo o por Rotura consiste en la corrección de las averías, roturas o fallas cuando se presentan, impidiendo el diagnóstico fiable de las causas que provocan las fallas (Garrido, 2009).

Mantenimiento Preventivo Planificado

Es la programación de actividades de inspección de los equipos, tanto de funcionamiento como de limpieza y calibración, que deben llevarse a cabo en forma periódica con base en un plan de aseguramiento y control de calidad. Su propósito es prevenir las fallas, manteniendo los equipos en óptima operación (Garrido, 2010).

La característica principal de este tipo de mantenimiento es la de inspeccionar los equipos, detectar las fallas en su fase inicial y corregirlas en el momento oportuno (Renove Tecnología, 2009-2016).

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

Partiendo de los conceptos antes estudiados sobre los tipos de mantenimiento se escoge como concepto general el de (Stiven, y otros, 2011), el cual plantea que el mantenimiento es el conjunto de actividades que deben realizarse a los equipos, con el fin de corregir o prevenir fallas, buscando que estos continúen prestando el servicio para el cual fueron diseñados.

1.2 Sistema Xedro-Orbita: necesidad de migrar

El Sistema Xedro-Orbita está compuesto por nueve módulos principales, cada uno de ellos posee varias funcionalidades. Para el desarrollo de la investigación se realizará un estudio y análisis de los módulos Configuración, Clasificadores y el proceso Vehículo.

El módulo **Configuración**: Es el módulo a través del cual se definen los diferentes grupos de vehículos de acuerdo a su marca, modelo, régimen de mantenimiento, entre otras características. También se define el umbral de mantenimiento por el cual se van regir todos los vehículos para la realización de los mantenimientos preventivos planificados que le corresponden (Xedro-Orbita, 2014).

El módulo **Clasificadores**: Es el módulo donde se insertan, modifican y eliminan tipos de mantenimientos, accesorios, causas de fallas, repuestos, herramientas, tipos de vehículos, documentos técnicos, tipos de combustible, tipos neumáticos, tipos de batería, unidades de medida (Xedro-Orbita, 2014).

Proceso **Vehículo**: Es el proceso desde el cual se va a gestionar toda la información de los vehículos (Xedro-Orbita, 2014).

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, 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.

Actualmente se encuentra desplegado y funcionando en la Dirección de Transporte de la UCI el sistema Xedro-Orbita, el cual fue desarrollado en el marco de trabajo Sauxe en su versión 2.2, por el Departamento de Tecnología del centro CEIGE. A pesar de su uso y explotación, el sistema requiere de mejoras en cuanto a rendimiento y seguridad, para ello se realiza este análisis, que permitirá viabilizar este proceso, el cual estará enfocado en la migración del software hacia una nueva tecnología, que le permita realizar todas sus actividades de forma segura y en tiempo.

La migración de este sistema hacia una nueva tecnología, no solo estará centrado en la necesidad de mostrar un producto con mayor rendimiento, sino que también sea capaz de proveer mejoras en las funcionalidades que posee; y que además permita obtener un nuevo diseño arquitectónico, con el objetivo de proporcionar una mejor interfaz visual a los usuarios finales.

Sauxe brinda solución a un sin número de escenarios o aspectos arquitectónicos como: gestión y configuración dinámica de caché, integración de componentes de forma distribuida o no distribuida, acceso a bases de datos a través de una capa de abstracción, gestión de concurrencia de recursos, administración centralizada de transacciones, gestión dinámica de las trazas generadas por los sistemas, implementación de mecanismos de autenticación y autorización, implementación de mecanismos de mensajería y control de excepciones, gestión y configuración dinámica de pre condiciones, pos condiciones y validaciones de variables, gestión y configuración de flujos de trabajo, visualización de las funcionalidades de un sistema, entre otros escenarios de alta complejidad, garantizando los atributos de calidad de los sistemas que se desarrollen con el mismo. Cuenta con una arquitectura en capas que a su vez presenta en su capa superior una implementación Modelo-Vista-Controlador (Cañete Pupo, 2010).

En la capa de presentación actualmente el sistema cuenta con ExtJs en su versión 3.4, este marco de trabajo se encuentra atado a una licencia, lo que significa que para comercializar un software hay que

pagar su uso. Teniendo en cuenta que el software libre se ha convertido en una premisa de la independencia tecnológica por la cual aboga Cuba, la UCI se suma en el uso de estas por lo que se decide emplear tecnologías que no presenten privatización en su licencia, por lo cual se opta por la arquitectura de referencia Odoo, el cuál es un software libre de código abierto, que permite ser adaptado a las necesidades del cliente. Además es gratuito por lo que no tiene costo de licencias.

Sauxe utiliza en la capa de negocio el marco de trabajo Zend Framework en su versión 1.2, que no permite la generación automática de CRUD, la generación de código PHP mediante líneas de comandos para la creación, mantenimiento de aplicaciones y el almacenamiento en la caché de las vistas. Esta versión que pertenece al paquete ZF1 de la compañía Zend Technologies se encuentra sin soporte desde el año 2014 según el sitio oficial (Zend, 2016)

Sauxe además propone para la capa de acceso a datos el uso del ORM[5] Doctrine en su versión 1.2.2, lo cual provoca bajo rendimiento y un mayor acoplamiento entre sus clases, por lo que se pierde la independencia y es más costoso a la hora del cambio hacia otra tecnología. Otro inconveniente que presenta el uso de esta versión de Doctrine es que a partir del 1 de junio del 2011 se dejó de dar soporte, trayendo consigo que no se corrijan posibles errores y vulnerabilidades que atentan contra su estabilidad y rendimiento (Doctrine, 2009).

Además, se identificaron varios factores subjetivos que influyen negativamente en el proceso de desarrollo tales como:

- El marco de trabajo Sauxe no obtendrá más soporte, esto trae consigo que existan brechas de seguridad, debido a que no serán actualizadas las tecnologías con las que se encuentra desarrollado.
- Poco control y seguimiento a la implementación de las soluciones, trayendo consigo que no se corrijan posibles errores en el proceso de implementación.
- No posee un sistema de auditorías al código fuente o al cumplimiento de las definiciones arquitectónicas, permitiendo que al pasar el tiempo la tecnología se vuelva obsoleta y no se le brinde soporte.

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

- Uso de bibliotecas no integradas al marco de trabajo, como por ejemplo la biblioteca TCPDF[6] en su versión 2.1, lo cual requiere de integraciones adicionales para su funcionamiento, a pesar de las ventajas que TCPDF posee, el marco de referencia Odoo ya trae integrada la biblioteca de clases, constituyendo una gran ventaja.

Las características que distinguen a Odoo de Sauxe es que es un sistema integral, pues permite controlar el flujo de procesos de un negocio, entendiendo que todos los departamentos de una empresa se relacionan entre sí. Es modular, pues está compuesto por módulos interconectados que gestionan los procesos de un negocio y es adaptable, ya que permite ser adaptado a las necesidades del cliente de forma gratuita.

Además, posee las siguientes ventajas:

- **Fácil exportación**: permite exportar e importar los datos que aporta en formato ***.csv***[7] por lo que hace que su manejo sea sencillo y entendible, para cualquier usuario. Además, permite la visualización de informes en ***.pdf***[8].
- **Multiplataforma:** es compatible con todas las plataformas por lo que es posible acceder desde cualquier sistema operativo GNU/Linux, Mac OS X o Windows. Además dispone de versión para los celulares con sistema operativo Android[9] y iOS[10].
- **OpenObject**: este marco de trabajo, permite un desarrollo rápido de funcionalidades o conectividad con otras plataformas. La extensa documentación facilita la integración y mejora del programa de manera muy ágil. Además, incluye un sistema de reportes con integración con OpenOffice.org, lo que permite personalizar los informes. También hay motores de reportes alternativos utilizando Qweb o JasperReports, este último será el utilizado, dado la agilidad que posee para el desarrollo de reportes (Kelly Naranjo, 2016).

---

6 TCPDF es una Open Source Clase/Biblioteca para el Popular Lenguaje de Programación Web PHP.

7 La extensión de archivo CSV significa Comma Separated Values (Valores separados por comas).

8 Sigla del inglés Portable Document Format (formato de documento portátil).

9 Android se basa en Linux, un programa libre que, a su vez, está basado en Unix.

10 IOS es un sistema operativo para dispositivos móviles desarrollado por la empresa Apple.

- **Fácil migración**: la herramienta oficial importa y exporta datos maestros en formato ***.csv***, para que resulte más sencillo seguir trabajando con los datos de la aplicación actual.

Odoo está calificado como una de las mejores soluciones de código abierto por las características antes mencionadas, y especialmente por su licencia 100% libre. Ofrece en un solo paquete el componente básico para la construcción de una aplicación de negocios: multilenguaje, servicios web, campos traducibles, ingeniería de reportes, PostgreSQL, Python como lenguaje de programación y licencia GNU AGPL v3 [11] (ODOO, 2016).

La arquitectura del ERP[12] ODOO es cliente – servidor, lo que permite que todos los usuarios trabajen sobre el mismo repositorio de datos. Esto tiene la ventaja de que toda la información está disponible y sincronizada en todo momento además de que descarga la mayor parte del trabajo de procesamiento de datos de las máquinas cliente (donde trabajan efectivamente los usuarios). El intercambio de datos entre el servidor y el cliente puede realizarse mediante XML-RPC[13], Net-RPC y/o JSON[14].

Odoo tiene una gran variedad de ventajas, analizadas anteriormente, pero cuenta además con funcionalidades importantes como: Inteligencia de Negocios, Mapeador Relacional de Objetos (ORM), casos de pruebas, motores de flujos de trabajo, grabador de módulos, envases de módulos, entre otros. Permitiendo una fácil implementación mediante el lenguaje de programación Python, contribuyendo de esta forma a su seguridad, trayendo consigo la internacionalización por el constante soporte que este recibe.

Por todas estas bondades que Odoo posee, aplicando el uso de tecnologías libres en el cual está sumergida la UCI, cumpliendo con la premisa de la independencia tecnológica por la cual aboga Cuba; y teniendo en cuenta el estudio realizado sobre las desventajas actuales que posee el marco de trabajo Sauxe, se decide migrar el sistema Xedro-Orbita hacia la arquitectura de referencia Odoo.

---

11 Licencia pública general de Affero (en inglés, Affero General Public License, también Affero GPL o AGPL).

12 Sistemas de planificación de recursos empresariales ('ERP', por sus siglas en inglés, enterprise resource planning).

13 XML-RPC es un protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes.

14 JSON (JavaScript Object Notation - Notación de Objetos de JavaScript).

1.3 Metodología para el desarrollo de software

Metodología de desarrollo para la Actividad Productiva en la UCI (AUP[15]-UCI).

A raíz del crecimiento tecnológico se hace necesario el desarrollo de software dentro de la actividad productiva de la UCI. La elección de la metodología es un proceso minucioso, ya que guía el proceso para alcanzar la satisfacción del cliente y del equipo de desarrollo, además que de ello depende la calidad del producto final. Por tal motivo se decide utilizar una variación realizada por la UCI a la metodología AUP, la cual se aplica en los proyectos productivos de la universidad y de la misma parte el desarrollo de los módulos Configuración, Clasificadores y el proceso Vehículo del Sistema Xedro-Orbita.

Esta variación cuenta con 8 disciplinas: Modelado de Negocio, Requisitos, Análisis y Diseño, Implementación, Pruebas Internas, Pruebas de Liberación, Pruebas de Aceptación y el Despliegue, que se considera opcional, en la presente investigación solo se estarán desarrollando las siguientes: Requisitos, Análisis y Diseño, Implementación, Pruebas Internas y Pruebas de Aceptación. (Rodríguez Sánchez, 2014).

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) el ciclo de vida de los proyectos de la UCI mantiene la fase de Inicio, unifica las restantes 3 fases en una sola, llamada Ejecución y se agrega la fase de Cierre (Rodríguez Sánchez, 2014)

Inicio: se realizan las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto (Rodríguez Sánchez, 2014).

Ejecución: en esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se obtienen los requisitos, se elabora la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en dicha transición se capacita a los usuarios finales sobre la utilización del software (Rodríguez Sánchez, 2014)

[15] Proceso Unificado Ágil.

Cierre: en esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto (Rodríguez Sánchez, 2014).

La fase de Inicio tiene como objetivo principal realizar la planeación de los proyectos, en esta etapa fue realizada la planeación del proceso de migración. Luego como la metodología lo indica, se siguen las Disciplinas de Negocio, requisitos, análisis y diseño, implementación, pruebas internas y pruebas de aceptación, donde en cada una de ellas se realizaron los artefactos necesarios que forman parte del desarrollo de la presente investigación.

La metodología posee cuatro escenarios de acuerdo al desarrollo de los proyectos, el trabajo de diploma corresponde con el escenario # 3, el cual se basa en la descripción de requisitos por procesos de negocio (Rodríguez Sánchez, 2014).

Escenario No 3: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad. Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y la relaciones entre los procesos identificados (Rodríguez Sánchez, 2014).

1.4 Análisis de estándar, plataforma y guías para la migración de software

1.4.1 ISO/IEC 14764

La **ISO/IEC 14764** es un estándar internacional, titulado ‘‘Ingeniería de Software-Proceso del ciclo de vida del software-Mantenimiento’’, describe los requerimientos para el mantenimiento del software (International Standard ISO/IEC 14764 IEEE, 2006).

Es un estándar específico sobre mantenimiento del software publicado por la ISO en 1998. El estándar internacional ISO 14764 presenta los requerimientos para el proceso de mantenimiento del software, contiene las actividades y tareas del mantenedor, proporciona una guía que explica cómo realizar el proceso de mantenimiento y establece definiciones para los distintos tipos de mantenimiento. La guía es aplicable a la planificación, ejecución y control, mantenimiento, revisión y evaluación del proceso de mantenimiento. (EcuRed, 2018)

La norma propone un plan que forma parte de la estrategia de mantenimiento, dicho plan es usado para guiar a los mantenedores de software, explica la necesidad de realizar mantenimiento, refiriéndose a quién efectúa ese trabajo y cómo se hace, contiene la documentación y responsabilidades de todos los involucrados. Además, debe incluir qué recursos hay disponibles para el mantenimiento, dónde se hace y cuándo comienza. Una vez definido dicho plan, el estándar propone establecer una guía para desarrollar el mantenimiento. (EcuRed, 2018)

Requisitos de la Guía

Los requisitos que debe de contener esta guía para este estándar son:

- La descripción del sistema al que se le brinda soporte, aquí se especifican todos los detalles del sistema a mantener.
- Identificación del estado inicial del software, para saber cuáles son los cambios nuevos realizados.
- Descripción del soporte para facilitar el comienzo del desarrollo del mantenimiento del software.
- Identificación de la organización que debe hacer el soporte o mantenimiento para contemplar el objetivo del mantenimiento en el proceso de desarrollo del software.
- Descripción de cualquier acuerdo entre cliente y vendedor, se debe tener claro lo que quiere el cliente por escrito, de este modo el vendedor sabe lo que tiene que hacer para satisfacer al cliente. (EcuRed, 2018)

Actividades de Mantenimiento

Estos son los aspectos fundamentales en cuanto a la estrategia de mantenimiento que propone el estándar. Las actividades que comprende el proceso de mantenimiento son:

- Implementación del proceso.
- Análisis de modificaciones y problemas.
- Implementación de modificaciones.
- Revisión y aceptación del mantenimiento.
- Migración.

- Retiro. (EcuRed, 2018)

Básicamente éste es el enfoque que brinda la norma ISO 14764 para realizar la actividad de mantenimiento de software. Esta norma identifica adecuadamente qué hacer en las actividades y tareas a desarrollar, pero solo para el proceso de mantenimiento del software no para el proceso de migración.

1.4.2 Plataforma Cubana de Migración a Código Abierto (PCMCA)

La PCMCA es una aplicación web que automatiza la gestión de la información generada en el proceso de migración a plataformas de código abierto de las empresas o instituciones. Cuenta con subsistemas que interactúan entre sí en un esquema de servicios, compartiendo datos y funcionalidades. Comparten una interfaz única de fácil manejo y usabilidad para el usuario, que permita la gestión de los procesos de migración y el acceso a las distintas aplicaciones de la plataforma (Revista Cubana de Ciencias informáticas, 2014)

La plataforma permite:

- Planificar el proceso de migración.
- Crear un entorno seguro de almacenamiento, organización, integración, análisis, gestión y disponibilidad de los datos.
- Recopilar la información necesaria en el levantamiento de información.
- Buscar herramientas alternativas para la migración de software.
- Medir el avance de los procesos de migración.
- Realizar reportes del estado del proceso de migración.
- Diversificar el alcance de las actividades propuestas por la Guía de Migración a Software Libre.
- Mejorar la toma de decisiones.
- Brinda la posibilidad de elaborar el Plan de migración a aplicaciones de Código Abierto, que es el documento en el que se precisan los detalles que orientarán el proceso (Fajardo, 2011), deberá estar escrito en un lenguaje claro y dejar reflejado todos los elementos resultantes del análisis de la información que se realiza en los diferentes subsistemas.

1.4.3 Guía Cubana para la Migración al Software Libre:

La estrategia para la migración de software, tiene una estructura basada en la propuesta de la "Guía cubana para la migración al software libre". Esta estará compuesta por Etapas, un total de tres, agrupadas en dos fases: Fase de Preparación y Fase de Migración. Las etapas previstas son la de Preparación, Migración y Consolidación. Dentro de las etapas se irán desarrollando un grupo de Flujos de Trabajos, los cuales se muestran a continuación (Feal Delgado, y otros, 2014):

- **Evaluación**: Hacer una evaluación de todos los procesos, tecnologías y personal y adaptarlas al entorno de la Universidad.
- **Formación**: Formación del personal y certificación del mismo en dependencia del tipo y niveles de los usuarios.
- **Pruebas**: Creación de ambientes reales de pruebas donde validar las experiencias prácticas
- **Implementación**: Instalación, migración de servicios y estaciones de trabajo.
- **Soporte y Asistencia Técnica**: Brindará atención y soporte a las infraestructuras, servicios instalados, y al personal vinculado a la migración.

El objetivo de la metodología propuesta para la migración de software, es ordenar dicho proceso a través de fases, que incrementen la probabilidad de éxito en momentos en que los sistemas serán reemplazados por nuevas alternativas. Se establece una Fase Inicial que comprende el análisis del sistema. Luego se analizan los datos a migrar, lo que cuenta con el análisis del dominio, alcance y objetivos, los que quedan documentados en Descripciones de requisitos, para luego realizar el Modelado de la Base de Datos. La propuesta termina con la aceptación, documentación, y aplicación, completando el proceso de migración.

Es necesario establecer una metodología de trabajo que incremente la probabilidad de éxito en el proceso de migración de software, en el cual se adopta el establecimiento de pasos, tareas y actividades que aportarán a la consecución de los objetivos.

Por tal motivo el presente trabajo de tesis utilizará de la "Guía Cubana para la Migración al Software Libre", pues desde el punto de vista social, los usuarios implicados en este proceso, al notar que la migración se ejecuta de manera más rápida y con mayor calidad, se involucran de manera directa, aceptando con mayor rapidez las nuevas tecnologías. Esto impacta positivamente en la ejecución de los

procesos de la UCI, en tanto es menor el trauma inherente a un proceso de cambio tecnológico. Por lo tanto, este documento de tesis será guiado con los elementos básicos propuestos por esta guía, para realizar una migración de software de forma exitosa.

1.5 Lenguajes y herramientas

1.5.1 Lenguaje de modelado UML

UML (Unified Modeling Language, por sus siglas en inglés) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema de software. Proporciona una forma estándar de escribir los planos de un sistema, cubriendo elementos conceptuales y concretos tales como: procesos del negocio, funciones del sistema, clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes de software reutilizables. Además brinda las siguientes ventajas: mayor rigor en la especificación, verificación y validación del modelo realizado, automatización de procesos y generación de código a partir de los modelos y viceversa (a partir del código fuente generar los modelos). Esto permite que el modelo y el código estén actualizados, manteniendo la visión en el diseño de la estructura de un proyecto (Jacobson, y otros, 2000).

1.5.2 Lenguaje de Programación Python 2.7.6

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, funcional. Es un lenguaje interpretado, legible, utiliza tipado dinámico, es multiplataforma y usa conteo de referencias para la administración de memoria (Python, 2016).

Para el desarrollo de la aplicación se utilizó Python debido a que es el lenguaje de programación que utiliza Odoo.

1.5.3 Lenguaje de Marcas Extensible XML 1.2

XML, siglas en inglés de Xtensible Markup Language (Lenguaje de marcas extensible), es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.

Proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información (XML, 2016).

Para el desarrollo dela aplicación se utilizó XML debido a que es el lenguaje de marcas que utiliza Odoo.

1.5.4 Lenguaje de programación Python 2.7.6

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, funcional. Es un lenguaje interpretado, legible, utiliza tipado dinámico, es multiplataforma y usa conteo de referencias para la administración de memoria (Python, 2016).

Para el desarrollo de la aplicación se utilizó Python debido a que es el lenguaje de programación que utiliza Odoo.

1.5.5 Visual Paradigm 8.0

Visual Paradigm es una herramienta para desarrollo de aplicaciones utilizando modelado UML de alta confiabilidad y estabilidad en el desarrollo orientado a objetos. Tiene soporte multiplataforma y proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Permite modelar el negocio orientado a procesos usando la notación BPMN (Paradigm, 2012).

Esta herramienta será utilizada para modelar los prototipos de interfaz de usuario del sistema. Estos formarán parte de cada requisito y serán descritos dentro de cada artefacto de descripción de requisitos en el capítulo 2.

1.5.6 Biblioteca de clases JasperReports 5.6

JasperReports es una biblioteca de creación de informes que tiene la habilidad de entregar contenido enriquecido al monitor, a la impresora o a ficheros PDF[16], HTML[17],XLS[18], CSV[19] y XML[20]. Su propósito

16 PDF (sigla del inglés Portable Document Format, formato de documento portátil).

principal es ayudar a crear documentos de tipo páginas, preparados para imprimir en una forma simple y flexible. JasperReports se usa comúnmente con iReport, para la edición de informes, si bien a partir de la versión 5.5.0 iReport ha sido sustituido por Jaspersoft Studio. Se encuentra bajo licencia GNU, por lo que es Software libre. (JasperSoft, 2014).

1.5.7 Arquitectura de referencia Odoo 10.0

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

Odoo es un sistema de ERP[21] integrado de código abierto producido por OpenERP s.a. Actualmente han cambiado el nombre de OpenERP por el de Odoo.

Entre las principales características de Odoo se encuentra su escalabilidad, modularidad y flexibilidad. Posee una importante comunidad de desarrolladores, con presencia en varios países de América y Europa, así como Asia y Australia, que están constantemente ampliando y mejorando el proyecto (amplia documentación, foros, listas de correo y desarrollo comunitario (Kelly Naranjo, 2016).

La arquitectura que utiliza Odoo es cliente-servidor donde el servidor se ejecuta independientemente del cliente y maneja la lógica de negocio y comunica con la aplicación de base de datos. El desarrollo de módulos se realiza editando archivos Python y XML.

No hay un editor oficial, aunque los tutoriales descantan Eclipse o PyCharm + PyDev. Odoo también incluye un sistema de reportes con integración con OpenOffice.org, lo que permite personalizar los informes. También hay motores de reportes alternativos utilizando Qweb o JasperReports (Kelly Naranjo, 2016).

---

17 HTML, sigla en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto).

18 XLS es una extensión para los archivos de hoja de cálculo utilizados en la aplicación Microsoft Excel.

19 Comma Separated Values,(Valores Separados por Columnas).

20 Extensible Markup Language (lengua de marcas extensible).

21 Planificación de Recursos Empresariales.

1.5.8 Marco de trabajo OpenObject 1.0

Un marco de trabajo simplifica el desarrollo de las aplicaciones, ya que automatiza muchos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas (Gerner, 2016)

Marco de trabajo de código abierto, inteligente, profesional y rápido en el desarrollo de aplicaciones en Python. Está basado en la arquitectura modelo-vista-controlador, además de poseer Inteligencia de Negocios, Mapeador Relacional de Objetos(ORM), casos de pruebas, motores de flujos de trabajo, grabador de módulos, envases de módulos, entre otros. OpenObject ofrece, en un solo paquete el componente básico para la construcción de una aplicación de negocios: multilenguaje, servicios web, campos traducibles, ingeniería de reportes, PostgreSQL, Python como lenguaje de programación y licencia GNU AGPL v3 (ODOO, 2016).

Se utilizó como marco de trabajo OpenObject por las ventajas que posee al estar integrado dentro de Odoo.

1.5.9 Gestión de la Base de Datos

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

PgAdmin 1.18.1

Es una aplicación gráfica para gestionar y administrar las bases de datos PostgreSQL. PgAdmin se diseña para responder a las necesidades de la mayoría de los usuarios, desde escribir simples consultas SQL hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de

PostgreSQL y hace simple la administración. Está disponible en más de una docena de lenguajes y para varios sistemas operativos, incluyendo Microsoft Windows, Linux, Mac OSX y Solaris (PgAdmin, 2016).

Postgresql 9.3

Es un sistema de gestión de bases de datos relacional orientado a objetos, el cual incluye características como herencia, restricciones, tipos de datos, reglas e integridad transaccional. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados, almacenamiento de objetos de gran tamaño. Se destaca en ejecutar consultas complejas, consultas sobre vistas, sub-consultas y joins. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Utiliza el modelo cliente servidor y es un manejador de base de datos de código abierto liberado bajo licencia. Postgresql está diseñado para administrar grandes volúmenes de datos (PostgreSQL, 2016).

Se utilizó como sistema gestor de base de datos por las ventajas que posee al estar integrado dentro de Odoo.

1.5.10 Servidor de aplicaciones Apache 2.2.9

El servidor Apache es un servidor web HTTP[22] de código abierto, para la creación de páginas y servicios web. La utilización del mismo se debe a que es un servidor multiplataforma, gratuito, muy robusto y que destaca por su seguridad y rendimiento (Ibrugor, 2014).

Se utilizó como servidor de aplicaciones Apache porque es el que se define en Odoo.

1.5.11 Entorno de desarrollo integrado PyCharm 4.5

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

PyCharm es un entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica, integración con VCS / DVCS y soporte para el desarrollo web con Django, entre otras bondades. Entre las características fundamentales

[22] HyperText Transfer Protocol (Protocolo de transferencia de hipertexto).

que posee el PyCharm se encuentran el autocompletado, resaltador de sintaxis, herramientas de análisis y refactorización. Posee un depurador avanzado, además de la integración con lenguajes de plantillas como Mako, Jinja2 y Django. (Python, 2016).

Se utilizó el Pycharm porque es el IDE para la programación en Python y tiene como ventaja principal que es multiplataforma.

1.5.12 Navegador Mozilla Firefox 57.0.1 o superior

Un Navegador Web es un programa que permite visualizar páginas web en la red además de acceder a otros recursos, documentos almacenados y guardar información. El Navegador se comunica con el servidor a través del protocolo HTTP y solicita el archivo en código HTML, después lo interpreta y muestra en pantalla para el usuario (Masadelante, 2018).

Firefox es un navegador de Internet con interfaz gráfica de usuario desarrollado por la Corporación Mozilla y un gran número de voluntarios. Permite añadir funcionalidades mediante extensiones, Firebug es precisamente el complemento que se integrará a Mozilla Firefox para ayudar a desarrollar, evaluar y depurar la propuesta de solución, controlando el CSS[23] y HTML en tiempo real, midiendo el tiempo de carga para optimizar la página o corrigiendo los posibles inconvenientes con JavaScript. Es un navegador de código libre y multiplataforma. Además permite la identificación de sitios web de forma instantánea, posibilita navegar de forma privada y sin registro de lo accedido en internet (Mozilla, 2013).

1.6 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Los patrones ayudan a capturar conocimiento y a crear un vocabulario técnico, hacen el diseño orientado a objetos más flexible, elegante y en algunos casos reusable (Gamma, 1994).

Los **patrones GRASP**[24] son guías o principios que sirven para asignar responsabilidades a las clases, dentro de los patrones que se tuvieron en cuenta, en la presente investigación están (Larman, 1999):

---

23 Cascading Style Sheets (hojas de estilo en cascada).

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

Experto: Es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele ser útil en el diseño orientado a objetos. La aplicación del patrón Experto consiste en asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Este tiene como beneficio que se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece tener sistemas más robustos y de fácil mantenimiento. Además, el comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más fáciles de comprender y mantener (Larman, 1999).

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

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

Bajo acoplamiento: Asignar una responsabilidad para mantener bajo acoplamiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. Este patrón soluciona el problema de lograr una dependencia escasa y un aumento de la reutilización, asignando responsabilidades a las clases de tal forma que la dependencia entre las mismas sea la menor posible (Larman, 1999).

Alta cohesión: Asigna una responsabilidad de modo que la cohesión siga siendo alta. En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. La utilización de este patrón permite que las clases contengan las responsabilidades estrechamente relacionadas, sin tener que realizar un trabajo excesivo, posibilitando la claridad y facilidad con que se entiende el diseño, simplificando el mantenimiento y las mejoras en funcionalidad, logrando en ocasiones un bajo acoplamiento (Larman, 1999).

Patrones GOF

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

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

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

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

1.7 Patrones de arquitectura

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

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

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

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

El patrón MVC[25] está representado en 3 capas: el modelo que es el objeto de la aplicación, la vista que es su representación y el controlador que define el modo en que la interfaz reacciona a la entrada del usuario. De una manera poco ortodoxa se puede decir que el controlador controla el flujo de la aplicación, pide al modelo aquello que el usuario solicita, y le devuelve (al usuario) una representación del modelo a través de la vista. La siguiente figura ilustra la cooperación entre estos tres objetos (Rodríguez García, 2012). (Ver Figura 1).

[25] Modelo-Vista-Controlador

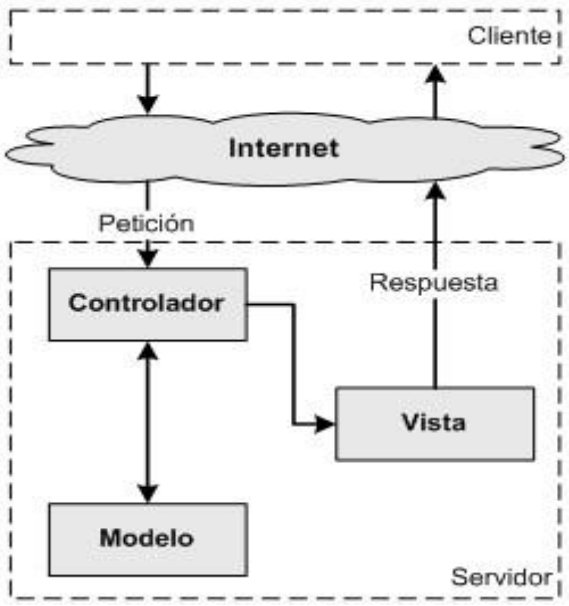


Figura 1. El patrón MVC. (Potencier, 2011)

Esta separación en capas con responsabilidades bien definidas permite, mejorar la organización del código.

1.8 Conclusiones parciales

- El estudio de los conceptos fundamentales abordados en el capítulo del trabajo de diploma, permitirá una mejor comprensión de los términos utilizados en la gestión vehicular y en el proceso de migración entre tecnologías.
- El análisis de metodologías, lenguajes y herramientas para el proceso de migración, permitió determinar los principales productos de trabajo ingenieriles, que estarán guiados por la metodología AUP-UCI.
- A partir del estudio y análisis de las tecnologías utilizadas en el desarrollo del Sistema Xedro-Orbita, se obtuvo información valiosa y necesaria, para realizar el proceso de migración hacia la arquitectura de referencia Odoo, lo que permitirá obtener un mejor funcionamiento del sistema.

CAPÍTULO 2: MIGRACIÓN DEL SISTEMA XEDRO-ORBITA

Introducción

En este capítulo se realiza el análisis, diseño e implementación de los módulos Configuración, Clasificadores, y el proceso Vehículo del sistema Xedro-Orbita, quedando plasmadas las descripciones de los principales artefactos generados en esta etapa, entre ellos se encuentran: descripción y especificación de los requisitos del sistema y diagrama de clases con estereotipos web. Se describen y obtienen los requisitos no funcionales. Además se propone una guía de pasos, que permitirá apoyar la migración del marco de trabajo Sauxe hacia la arquitectura de referencia Odoo.

2.1 Modelado de negocio de Xedro-Orbita

La metodología AUP-UCI consta de tres fases, la primera es Inicio donde se desarrollan todos los planes de proyecto y se construye el cronograma que contiene los hitos principales por los cuales pasará. La segunda fase Elaboración es donde se centrará la investigación, o sea, el proceso de migración del sistema Xedro-Orbita del marco de trabajo Sauxe a la arquitectura de referencia Odoo, donde es necesario ejecutar las actividades requeridas dentro de las Disciplinas Análisis y diseño, Implementación, Pruebas internas y Pruebas de aceptación.

En la Fase de Ejecución se ejecutan todas las Disciplinas exceptuando la de Modelado de negocio porque la migración se centra analizar el negocio existente no cambiarlo, por tanto se parte de su análisis para obtener una redefinición de los requisitos del sistema, los cuales cambiarán en funcionamiento y concordancia con la nueva tecnología, para que estos requisitos se apliquen positivamente, es necesario realizar un rediseño de las clases, con ello se podrá realizar la implementación de todas las funcionalidades a migrar.

La migración Xedro-Orbita se basa en obtener módulos que sean capaces de respetar y mantener la lógica del negocio; y con ello mejorar la carga de datos que poseen los componentes existentes, todo ello se realiza con el objetivo de brindar un mayor rendimiento de las funcionalidades del sistema.

2.2 Requisitos de software

Los requisitos de software describen los servicios ofrecidos por el sistema y sus restricciones. Estos reflejan las necesidades de los clientes. Los mismos pueden dividirse en dos categorías: requisitos funcionales y requisitos no funcionales (Pressman, 2010).

2.2.1 Requisitos Funcionales

Los requisitos funcionales son declaraciones de los servicios que el sistema debe proporcionar, cómo este debe reaccionar ante entradas específicas y cómo debe comportarse en determinadas situaciones (Almeida, 2014).

Los requisitos pueden ser agrupados de acuerdo a la relación funcional directa, por ejemplo según el tipo de datos que van a manejar o según los procesos de negocio que van a gestionar (Hernández Aguilar, y otros, 2010).

Los requisitos funcionales identificados en los módulos Configuración, Clasificadores, y el proceso vehículo son los siguientes:

1. Agrupación de requisitos Gestionar accesorio:
    - RF 1.1. Crear accesorio
    - RF 1.2. Editar accesorio
    - RF 1.3. Suprimir accesorio
    - RF 1.4. Imprimir accesorio
    - RF 1.5. Listar accesorio
    - RF 1.6. Buscar accesorio
2. Agrupación de requisitos Gestionar causa de falla:
    - RF 2.1. Crear causa de falla
    - RF 2.2. Editar causa de falla
    - RF 2.3. Suprimir causa de falla
    - RF 2.4. Imprimir causa de falla
    - RF 2.5. Listar causa de falla
    - RF 2.6. Buscar causa de falla
3. Agrupación de requisitos Gestionar documento técnico:

- RF 3.1. Crear documento técnico
- RF 3.2. Editar documento técnico
- RF 3.3. Suprimir documento técnico
- RF 3.4. Imprimir documento técnico
- RF 3.5. Listar documento técnico
- RF 3.6. Buscar documento técnico

4. Agrupación de requisitos Gestionar grupo de vehículos:
    - RF 4.1. Crear grupo de vehículos
    - RF 4.2. Editar grupo de vehículos
    - RF 4.3. Suprimir grupo de vehículos
    - RF 4.4. Imprimir grupo de vehículos
    - RF 4.5. Listar grupo de vehículos
    - RF 4.6. Buscar grupo de vehículos
5. Agrupación de requisitos Gestionar herramienta:
    - RF 5.1. Crear herramienta
    - RF 5.2. Editar herramienta
    - RF 5.3. Suprimir herramienta
    - RF 5.4. Imprimir herramienta
    - RF 5.5. Listar herramienta
    - RF 5.6. Buscar herramienta
6. Agrupación de requisitos Gestionar repuesto:
    - RF 6.1. Crear repuesto

- RF 6.2. Editar repuesto
- RF 6.3. Suprimir repuesto
- RF 6.4. Imprimir repuesto
- RF 6.5. Listar repuesto
- RF 6.6. Buscar repuesto

7. Agrupación de requisitos Gestionar ruta:
    - RF 7.1. Crear ruta
    - RF 7.2. Editar ruta
    - RF 7.3. Suprimir ruta
    - RF 7.4. Imprimir ruta
    - RF 7.5. Listar ruta
    - RF 7.6. Buscar ruta
8. Agrupación de requisitos Gestionar distancias:
    - RF 8.1. Crear distancias
    - RF 8.2. Editar distancias
    - RF 8.3. Suprimir distancias
    - RF 8.4. Imprimir distancias
    - RF 8.5. Listar distancias
    - RF 8.6. Buscar distancias
9. Agrupación de requisitos Gestionar tipo de batería:
    - RF 9.1. Crear tipo de batería
    - RF 9.2. Editar tipo de batería

- RF 9.3 Suprimir tipo de batería
- RF 9.4. Imprimir tipo de batería
- RF 9.5. Listar tipo de batería
- RF 9.6. Buscar tipo de batería

10. Agrupación de requisitos Gestionar tipo de combustible:
    - RF 10.1. Crear tipo de combustible
    - RF 10.2. Editar tipo de combustible
    - RF 10.3. Suprimir tipo de combustible
    - RF 10.4. Imprimir tipo de combustible
    - RF 10.5. Listar tipo de combustible
    - RF 10.6. Buscar tipo de combustible
11. Agrupación de requisitos Gestionar tipo de actividad:
    - RF 11.1. Crear tipo de actividad
    - RF 11.2. Editar tipo de actividad
    - RF 11.3. Suprimir tipo de actividad
    - RF 11.4. Imprimir tipo de actividad
    - RF 11.5. Listar tipo de actividad
    - RF 11.6. Buscar tipo de actividad
12. Agrupación de requisitos Gestionar tipo de mantenimiento:
    - RF 12.1. Crear tipo de mantenimiento
    - RF 12.2. Editar tipo de mantenimiento
    - RF 12.3. Suprimir tipo de mantenimiento

- RF 12.4. Imprimir tipo de mantenimiento
- RF 12.5. Listar tipo de mantenimiento
- RF 12.6. Buscar tipo de mantenimiento

13. Agrupación de requisitos Gestionar tipo de neumático:
    - RF 13.1. Crear tipo de neumático
    - RF 13.2. Editar tipo de neumático
    - RF 13.3. Suprimir tipo de neumático
    - RF 13.4. Imprimir tipo de neumático
    - RF 13.5. Listar tipo de neumático
    - RF 13.6. Buscar tipo de neumático
14. Agrupación de requisitos Gestionar tipo de vehículo:
    - RF 14.1. Crear tipo de vehículo
    - RF 14.2. Editar tipo de vehículo
    - RF 14.3. Suprimir tipo de vehículo
    - RF 14.4. Imprimir tipo de vehículo
    - RF 14.5. Listar tipo de vehículo
    - RF 14.6. Buscar tipo de vehículo
15. Agrupación de requisitos Gestionar unidad de medida:
    - RF 15.1. Crear unidad de medida
    - RF 15.2. Editar unidad de medida
    - RF 15.3. Suprimir unidad de medida
    - RF 15.4. Imprimir unidad de medida

- RF 15.5. Listar unidad de medida
- RF 15.6. Buscar unidad de medida

16. Agrupación de requisitos Gestionar vehículo:
    - RF 16.1. Crear vehículo
    - RF 16.2. Editar vehículo
    - RF 16.3. Suprimir vehículo
    - RF 16.4. Imprimir vehículo
    - RF 16.5. Listar vehículo
17. Agrupación de requisitos Gestionar usuario:
    - RF 17.1. Crear usuario
    - RF 17.2. Editar usuario
    - RF 17.3. Suprimir usuario
    - RF 17.4. Imprimir usuario
    - RF 17.5. Listar usuario
    - RF 17.6. Buscar usuario
    - RF 17.6. Buscar vehículo

2.2.2 Descripción de Requisitos Funcionales

La descripción de los requisitos funcionales de los módulos Configuración, Clasificadores, y el proceso Vehículo, permitió realizar una descripción textual de cada uno de los requisitos identificados, donde se reflejan los flujos por los que transita el requisito, la información que muestra, así como las restricciones que posee. Los requisitos migrados fueron necesario redefinirlos y plasmarlos dentro de cada descripción de requisitos, ya que el cambio de tecnología influye mayormente, en el prototipo de interfaz de usuario con que el cliente trabajará. A continuación, se muestra un ejemplo de la descripción del requisito Crear Vehículo, perteneciente a la Agrupación de requisitos Gestionar Vehículo:

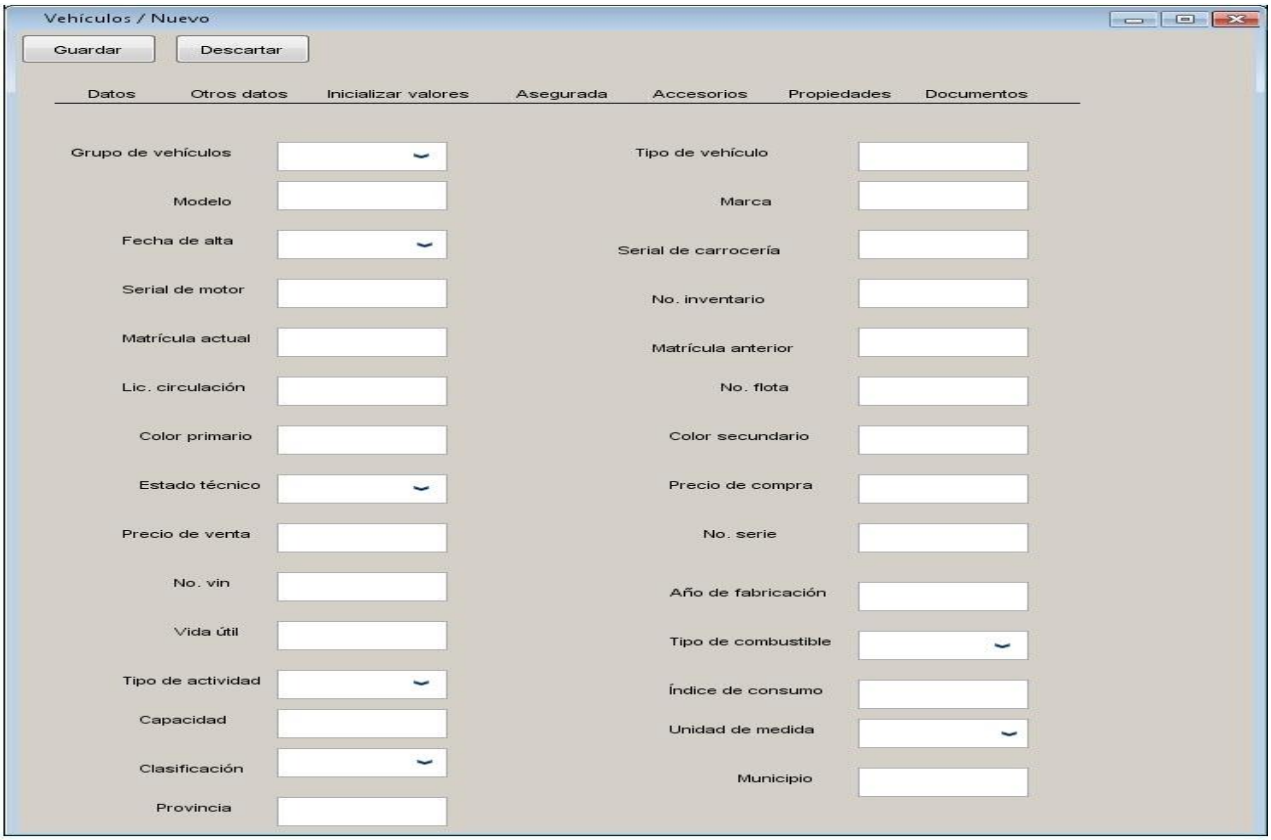
Tabla 1. Descripción del requisito funcional Crear Vehículo.

Precondiciones	El usuario se ha autenticado en el sistema y tiene permiso para realizar esta acción.
Flujo de eventos	
Flujo básico Crear	
1	Se registran los datos: Grupo de vehículos Modelo Fecha de alta Serial de motor Matrícula actual Li. circulación Color primario Estado técnico Precio de venta No. vin Vida útil Tipo actividad Capacidad Clasificación Provincia Tipo de vehículo Marca Serial de carrocería No. inventario Matrícula anterior No. flota Color secundario Precio de compra No. serie Año de fabricación Tipo de combustible Índice de consumo Unidad de medida Municipio
2	El sistema valida (ver validación 1) los datos introducidos.
3	Si los datos son correctos el sistema los registra.
4	El sistema confirma el registro de los datos.
5	Concluye el requisito.
Pos-condiciones	
1	Se ha registrado un nuevo vehículo.
Flujos alternativos	
Flujo alternativo 3.a Información errónea	
1	El sistema señala los datos erróneos y permite corregirlos.

2	El usuario corrige los datos.	
3	Volver al paso 2 del flujo básico.	
Pos-condiciones		
	N/A	
Flujo alternativo 3.b Información incompleta		
1	El sistema señala los datos vacíos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 2 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo *.a El usuario cancela la acción		
1	Concluye el requisito.	
Pos-condiciones		
1	No se registran los datos	
Validaciones		
1	Se validan los datos según lo establecido en el CEIGE-Xedro_Orbita-Modelo conceptual.doc.	
Conceptos	**Vehículo**	Visibles en la interfaz: Se registran los datos: Grupo de vehículos Modelo Fecha de alta Serial de motor Matrícula actual Licencia circulación Color primario Estado técnico Precio de venta No. vin Vida útil Tipo actividad Capacidad Clasificación Provincia Tipo de vehículo Marca Serial de carrocería No. inventario Matrícula anterior No. flota Color secundario Precio de compra No. serie Año de fabricación

		Tipo de combustible Índice de consumo Unidad de medida Municipio
Requisitos especiales	N/A	

Prototipo elemental de interfaz gráfica de usuario



La restante información sobre la descripción y la agrupación del requisito, anteriormente abordado, se encuentra en el documento entregable CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Vehículos.odt. Los requisitos funcionales de los módulos Configuración, Clasificadores y el proceso Vehículo se encuentran descritos en los documentos entregables:

- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Accesorio.odt

- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Causa de Falla.odt
- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Documentos Técnicos.odt
- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Grupo de Vehículos.odt
- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Herramienta.odt
- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Repuesto.odt
- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Ruta.odt
- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Tabla de Distancias.odt
- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Accesorio.odt
- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Tipo de Batería.odt
- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Tipo de Combustible.odt
- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Tipo de Actividad.odt
- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Tipo de Mantenimiento.odt
- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Tipo de Neumáticos.odt
- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Tipo de Vehículos.odt
- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Unidad de Medida.odt
- CEIGE-Xedro_Orbita-Descripción de la agrupación de requisitos Gestionar Usuario.odt

2.2.3 Requisitos no funcionales

Los requisitos no funcionales describen las restricciones de los servicios o funciones ofrecidas por el sistema. Pueden incluir restricciones de tiempo, de las normas de desarrollo y de procesos. Generalmente se aplican al sistema como un todo (Almeida, 2014).

Entre los requisitos no funcionales se encuentran:

RNF 1. Usabilidad

1.1 El idioma de todas las interfaces de la aplicación será el español.

1.2 Todos los mensajes de error del sistema deberán incluir una descripción textual del error.

1.3 En cada formulario del sistema los campos obligatorios serán señalados en azul.

RNF 2. Seguridad

2.1 El sistema manejará la seguridad de acceso y administración de usuarios mediante el otorgamiento de privilegios y roles, así como la asignación de perfiles.

2.2 Se concederá acceso al sistema a partir de un nombre de usuario y una contraseña.

RNF 3. Confiabilidad

3.1 El sistema debe prever en cada formulario, la entrada de datos incorrectos.

3.2 En cada formulario, el sistema impondrá campos obligatorios para garantizar la integridad de la información que se introduce por el usuario.

3.3 El sistema deberá detectar fallos internos y notificar al usuario de la ocurrencia de estos.

3.4 El sistema tendrá un respaldo de la información diaria, permitiendo la recuperación ante la pérdida parcial o total de la información.

RNF 4. Hardware

4.1 El sistema interactuará con impresoras para imprimir los diferentes documentos que genere la aplicación, como respuesta a las funcionalidades del sistema.

4.2 El sistema para su instalación en el servidor de aplicación requiere:

4.3.1 Procesador: 3.00 GHZ.

4.3.2 RAM: 1GB (recomendado 4GB).

4.3.3 Disco duro: 160 GB (recomendado 500GB).

4.3.4 UPS: 1.

4.3.5 Lector de CD: 1.

4.3.6 Tarjeta de Red: 1.

4.3 El sistema para su instalación en el servidor de base de datos requiere:

4.4.1 Procesador: 3.00 GHZ.

4.4.2 RAM: 1GB (recomendado 4GB).

4.4.3 Disco duro: 160 GB (recomendado 500GB).

4.4.4 UPS: 1.

4.4.5 Lector de CD: 1.

4.4.6 Tarjeta de Red: 1.

2.2.4 Técnicas de Validación de Requisitos

La validación de requisitos examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos, y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso, el proyecto y el producto (Pressman, 2010).

Para validar los requisitos descritos fueron utilizadas las técnicas siguientes:

- **Revisiones de requisitos**: los requisitos son analizados sistemáticamente por un equipo de revisores (Sommerville, 2005).
- **Construcción de prototipos**: se muestra un modelo ejecutable del sistema a los usuarios finales y a los clientes. Éstos pueden experimentar con este modelo para ver si cumple sus necesidades reales (Sommerville, 2005).
- **Generación de casos de prueba**: los requisitos deben poder probarse. Si las pruebas para éstos se conciben como parte del proceso de validación, a menudo revela los problemas en los requisitos (Sommerville, 2005).
- **Revisión técnica formal**: el equipo de revisión incluye ingenieros del sistema, clientes, usuarios y otros intervinientes que examinan la especificación del sistema buscando errores en el contenido o en la interpretación, áreas donde se necesitan aclaraciones, información incompleta,

inconsistencias, requisitos contradictorios, o requisitos imposibles o inalcanzables (Pressman, 2010).

Luego de analizar el uso de las técnicas antes mencionadas se aplican las siguientes:

- **Construcción de prototipos:** se realizaron los prototipos de interfaz con el objetivo de presentarle al cliente el resultado final esperado, de esta forma puede interpretar con claridad las necesidades solicitadas.
- **Generación de casos de prueba:** fueron generados los casos de pruebas con el objetivo de probar cada uno de los requisitos descritos y eliminar la existencia de errores.
- **Revisión técnica formal:** fue realizada una revisión técnica formal examinando detalladamente cada requisito, con el objetivo de obtener los errores en el contenido descrito o interpretado.

La validación de los requisitos permitió examinar cada una de las descripciones y especificación de requisitos realizada, para asegurar que todos los requisitos del sistema se establecieron sin ambigüedad, sin inconsistencias y sin omisiones. Los errores detectados fueron corregidos y el resultado obtenido se ajusta a los estándares establecidos para el proceso, el proyecto y el producto, según lo dictamina Pressman.

2.3 Diseño de la propuesta de migración

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

2.3.1 Diseño arquitectónico

El patrón arquitectónico Modelo-Vista-Controlador (MVC) se ve reflejado en el sistema a través de las siguientes tres capas:

Modelo: Objetos python cuyos datos son almacenados en una base de datos PostgreSQL. Los modelos se encuentran dentro de la carpeta models, divididos en dos componentes: clasificadores y vehículo. Para el caso del módulo Configuración, no se requiere de un componente específico, porque serán migrados sus requisitos para el módulo Clasificadores, por tanto pertenecerá al componente clasificadores,

volviéndose un nomenclador gestionable. Odoo gestiona el módulo Configuración dentro de las facilidades administrativas que posee. El mapeo de la base de datos es gestionado automáticamente por Odoo, y el mecanismo responsable por esto es el Modelo Objeto Relacional (ORM: Object Relational Model).

Vista: Las vistas en Odoo manejan la presentación visual de los datos representados por el Modelo a través de archivos XML. Se encuentran almacenadas dentro de la carpeta views.

Controlador: En Odoo el controlador se manifiesta a través del modelo controller.py, presente en el archivo base_import, el cual es el encargado de hacer peticiones al modelo cuando se hace alguna solicitud de la información por parte del cliente.

2.3.2 Patrones de diseño

Durante la migración del Sistema Xedro-Orbita se utilizaron varios patrones de diseño GRASP y GOF, los cuales permitieron asignar responsabilidades a las clases. En el diagrama de clases representado a continuación en la Figura 3 y Figura 4, se muestra la aplicación de cada uno de ellos.

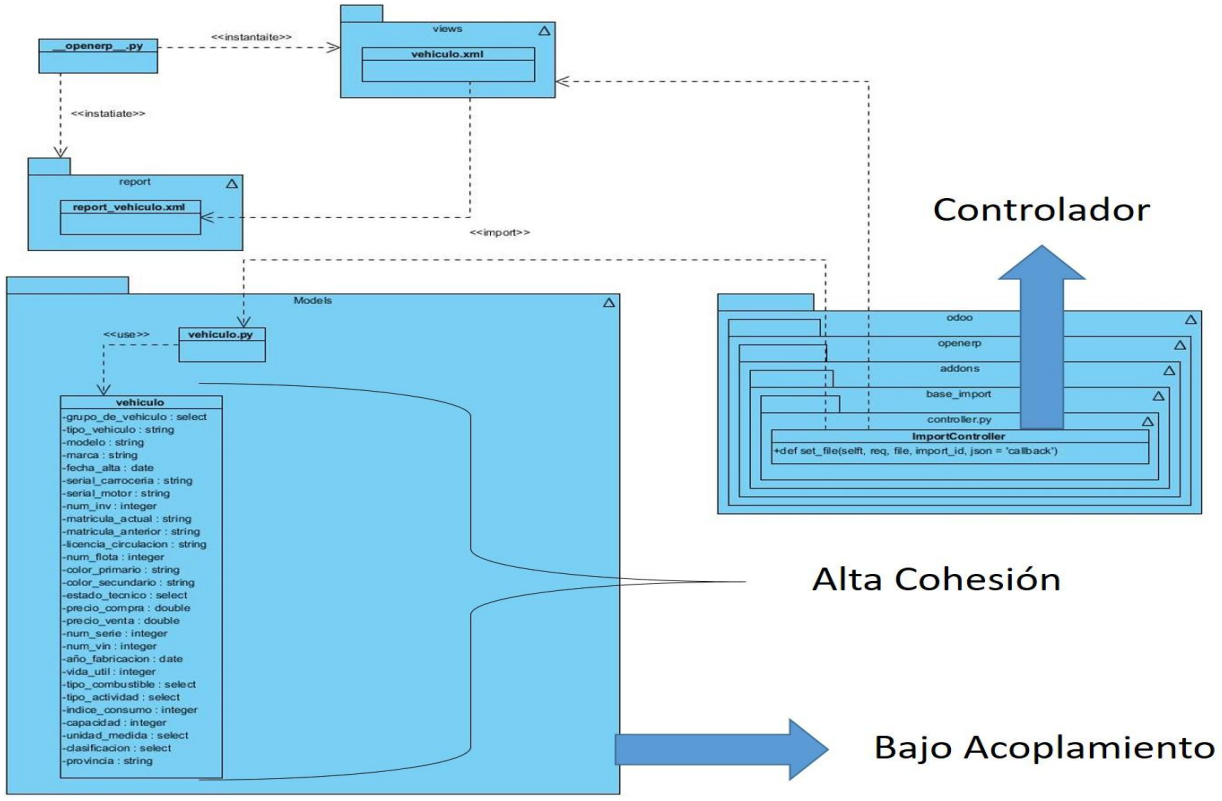


Figura 2. Proceso Vehículo. Aplicación de patrones de diseño GRASP.

Bajo acoplamiento: se aprecia el bajo acoplamiento existente en el sistema, ya que el diseño muestra como las clases del componente ***models*** no depende de ningún otro componente.

Alta Cohesión: se muestra como el modelo vehículo es el que posee la responsabilidad de crear los vehículos que se van a registrar en el sistema.

Controlador: se observa como la clase ***ImportController*** es la encargada de controlar los eventos del sistema, controlando tanto las vistas, como los modelos.

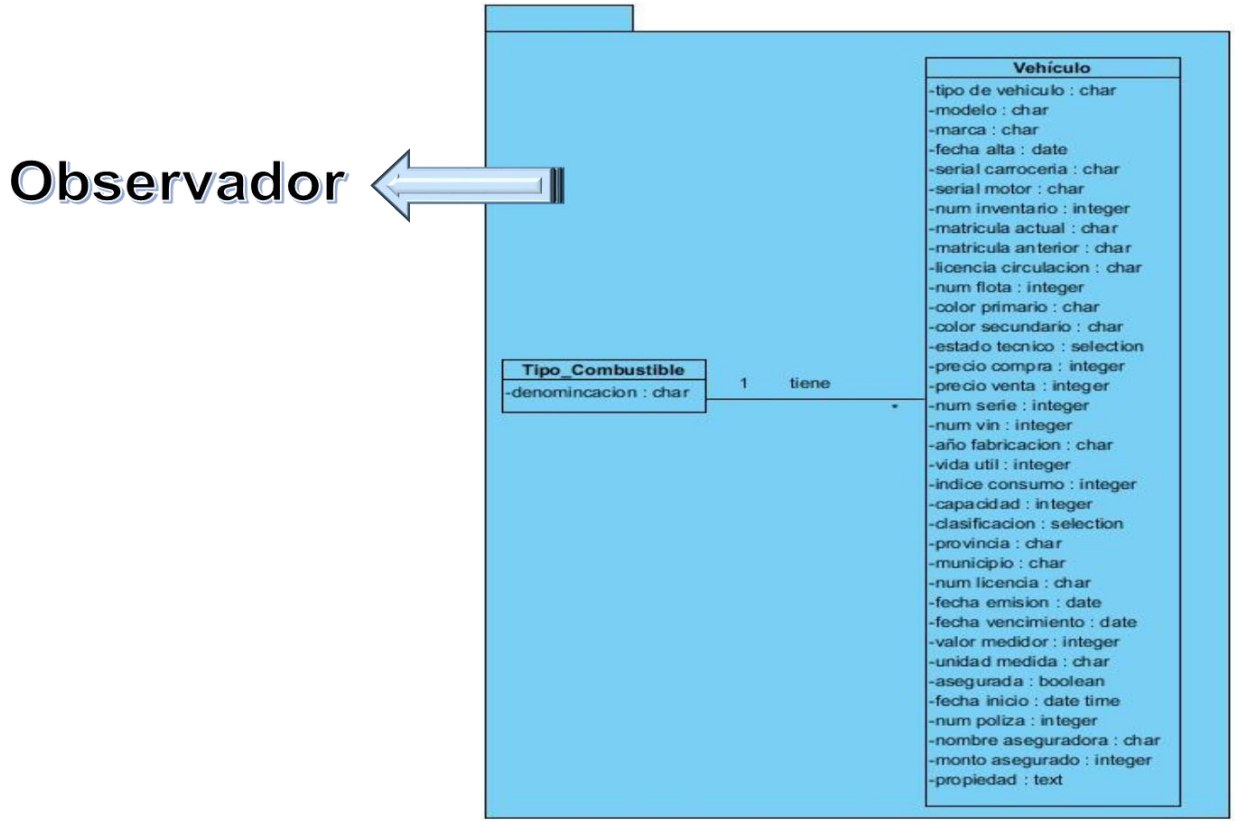


Figura 3. Aplicación de patrones de diseño GOF.

Patrón de Comportamiento

Observer (Observador): Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él. (EcuRed, 2018).

El patrón **Observador** puede ser utilizado cuando hay objetos que dependen de otro, en este caso **Vehículo** depende de **Tipo_Combustible**, donde si se modifica el tipo de combustible, este cambio se verá reflejado en los vehículos que dependen de él.

2.3.3 Diseño de clases con estereotipos web

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

En este caso se utilizan los estereotipos web: las páginas clientes, las servidoras y los formularios.

En el siguiente diagrama de clases del diseño con estereotipos web, la Client Page Principal (Página Cliente Principal) le hace una petición a la Server Page(Página del Servidor) (Gestionar Vehículo), esta es la encargada de construir las Client Page (Crear Vehículo, Editar Vehículo, Suprimir Vehículo, Imprimir Vehículo, Listar Vehículo y Buscar vehículo), con la colección de elementos de entrada FR (formulario), para luego mostrarlos a través de la vista correspondiente a la función que se esté solicitando, dígase: crear/editar/suprimir/imprimir/listar/buscar vehículo.

Para dar respuesta a las peticiones realizadas por la página cliente, el controlador tiene la responsabilidad de realizar las operaciones, sobre la información de las tablas de la base de datos accediendo a las clases del modelo y enviarle los resultados. (Ver Figura 3).

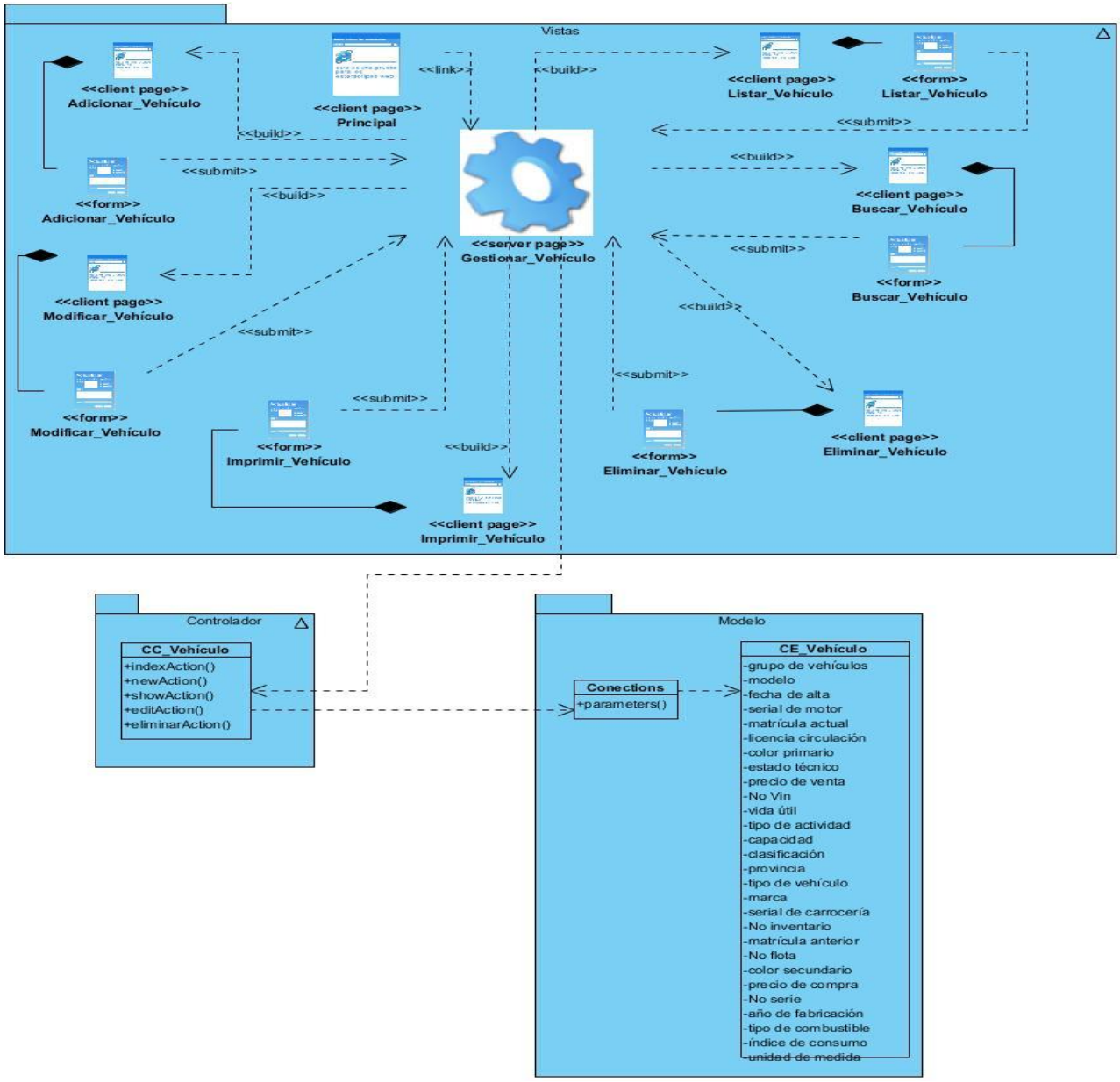


Figura 4. Diagrama de clases del diseño con estereotipos web del proceso Vehículo.

2.3.4 Métricas para la validación del diseño

Las métricas de diseño permiten medir de forma cuantitativa la calidad de los atributos internos del software. A nivel de componentes se centran en las características internas de los componentes del

software con medidas que pueden ayudar al desarrollador a juzgar la calidad de un diseño a nivel de componente (IngSoftwareII-CUFM, 2012).

Para la validación del diseño se aplicaron las métricas Tamaño operacional de clase (TOC) y Relaciones entre clases (RC) debido al conjunto de atributos de calidad de diseño que ambos miden.

Métrica de Tamaño Operacional de Clases (TOC)

Esta métrica se encuentra entre las planteadas por Lorenz y Kidd y pertenece al grupo de las métricas de tamaño de clase. La misma se centra en la cantidad de operaciones para cada clase individual y los valores promedio para el sistema como un todo. Se encarga de medir la calidad de acuerdo a los atributos Responsabilidad, Complejidad de implementación y Reutilización de las clases. (EcuRed, 2018).

La aplicación de la métrica TOC define los siguientes atributos de calidad:

- **Responsabilidad**: consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- **Complejidad de implementación**: consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- **Reutilización**: consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software (EcuRed, 2018).

Para evaluar las métricas son necesarios los valores de los umbrales para los parámetros de calidad. Los umbrales para esta métrica se basan en el promedio de operaciones por clases obtenidos, estos valores fueron los aplicados en el diseño del sistema y los mismos son reflejados en la siguiente tabla (EcuRed, 2018).

Tabla 2. Criterios para evaluar los atributos según el umbral de TOC.

Atributos	**Categoría**	**Criterio**
Responsabilidad	Baja	<= Promedio
	Media	Entre Promedio y 2*Promedio

	Alta	> Promedio
Complejidad de implementación	Baja	<= Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	> Promedio
Reutilización	Baja	<= Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	> Promedio

Resultados obtenidos al aplicar la métrica TOC

A partir de la evaluación de cada una de las clases según los atributos anteriormente mencionados se obtuvo el siguiente resultado:

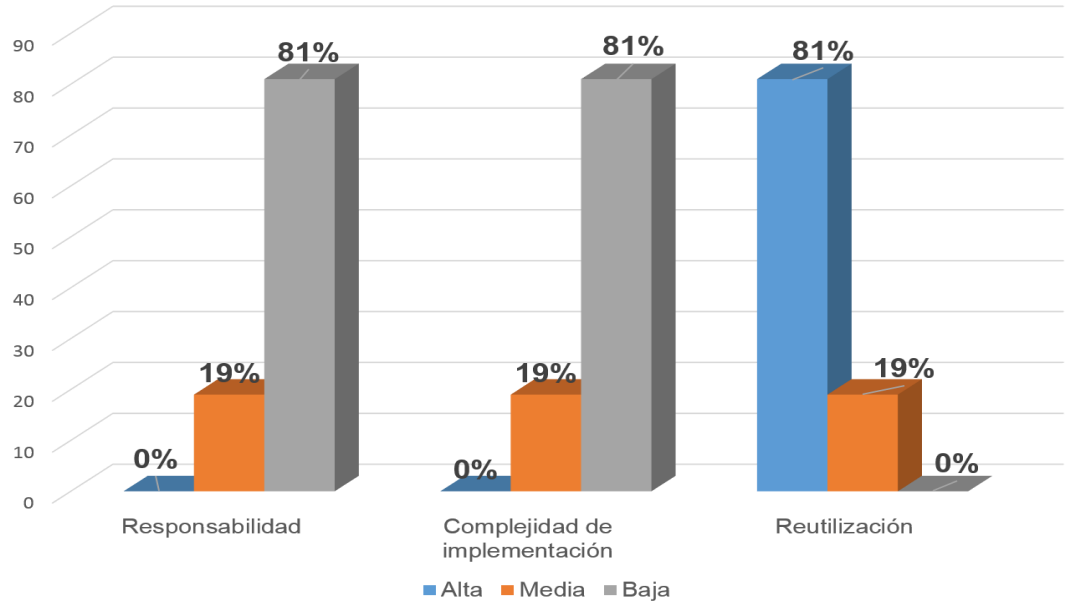


Figura 5. Gráfica con los resultados de la evaluación del diseño según TOC.

Como resultado de la aplicación de la métrica TOC se puede apreciar de manera general que el 81% de las clases del diseño tienen baja responsabilidad. Esto significa que cuentan con menor grado de complejidad de implementación, lo que favorece notablemente la reutilización de las clases. Mientras que el 19% restante de los atributos de calidad presenta una evaluación media, lo cual no influye en el resultado final al poseer el menor por ciento de evaluación.

Métrica de Relaciones entre Clases (RC)

Esta métrica se basa en la cantidad de relaciones de uso que presenta una clase determinada con las demás clases. Utiliza los atributos Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas para medir la calidad del diseño de la clase (EcuRed, 2018).

La aplicación de la métrica RC define los siguientes atributos de calidad:

- Acoplamiento: el aumento del RC implica un aumento del acoplamiento de la clase.
- Complejidad de mantenimiento: un aumento de la complejidad del mantenimiento de la clase implica un aumento del RC.
- Reutilización: un aumento del RC implica una disminución en el grado de reutilización de la clase.
- Cantidad de pruebas: el aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase. (EcuRed, 2018)

Los umbrales para evaluar el diseño según esta métrica se basan en el promedio de relaciones por clases y el caso del atributo Acoplamiento, utiliza la propia cantidad de relaciones, estos valores fueron los aplicados en el diseño del sistema y los mismos son reflejados en la siguiente tabla (EcuRed, 2018):

Tabla 3. Criterios para evaluar los atributos según el umbral de RC.

Atributos	**Categoría**	**Criterio**
Acoplamiento	Bajo	1
	Medio	2
	Alto	> 2

Complejidad de mantenimiento	Baja	<= Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	> Promedio
Reutilización	Baja	<= Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	> Promedio
Cantidad de pruebas	Baja	<= Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	> Promedio

Resultados obtenidos al aplicar la métrica RC

A partir de la evaluación de cada una de las clases según los atributos anteriormente mencionados se obtuvo el siguiente resultado:

Al aplicar la métrica RC se obtuvo como resultado un 62% de bajo acoplamiento, un 56% de baja complejidad de mantenimiento, un 56% de baja cantidad de pruebas y un 56% de alta reutilización que poseen las clases diseñadas para el sistema. Sin embargo, el 38% restante presenta un alto acoplamiento, lo que provoca un 44% de alto en los atributos complejidad de mantenimiento y cantidad de pruebas unitarias, induciendo que la reutilización sea menor con respecto a las clases.

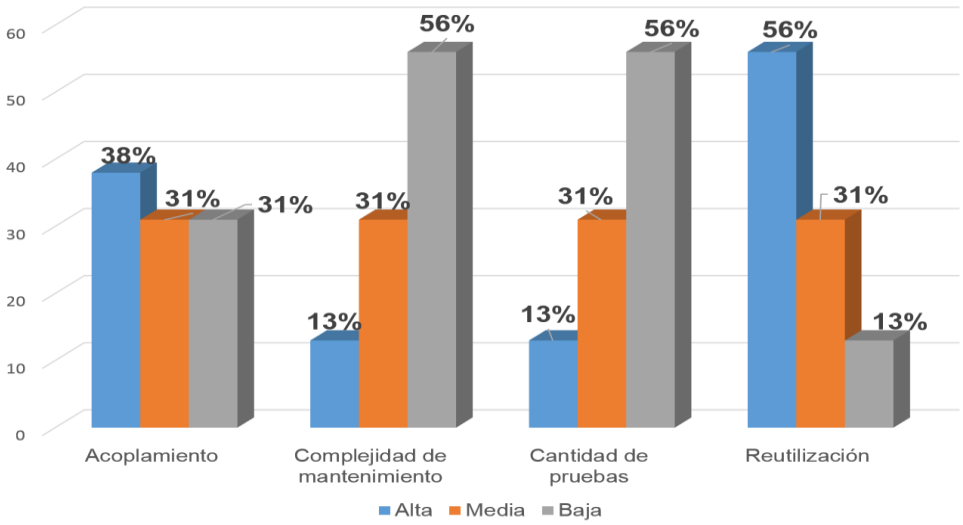


Figura 6. Gráfica con los resultados de la evaluación del diseño según RC

La aplicación de las métricas TOC y RC demuestra que las clases del sistema poseen una baja carga de responsabilidades, un bajo acoplamiento y una alta reutilización, permitiendo una fácil implementación, una baja complejidad de mantenimiento y una baja cantidad de pruebas de unidad necesarias para probar las clases. Los resultados obtenidos por la aplicación y evaluación de las métricas fueron positivos, quedando validado el diseño del sistema.

2.3.5 Modelo de Datos

Un diagrama o modelo entidad-relación (DER) es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información, así como sus interrelaciones y propiedades. El DER fue propuesto originalmente por Peter Chen para el diseño de sistemas de base de datos relacionales. Se identifica un conjunto de componentes primarios para el DER: objeto de datos, atributos, relaciones y varios indicadores de tipo. El propósito fundamental es representar objetos de datos y sus relaciones (Pressman, 2010).

La Figura 7 muestra el modelo de datos del sistema, el cual está compuesto por 17 entidades, las cuales poseen atributos y relaciones entre sí, siendo la entidad Vehículo la más significante debido a su alta responsabilidad en el sistema. El modelo de datos se encuentra normalizado, en tercera Forma Normal, cumpliendo con la normalización de la base datos que posee Odoo.

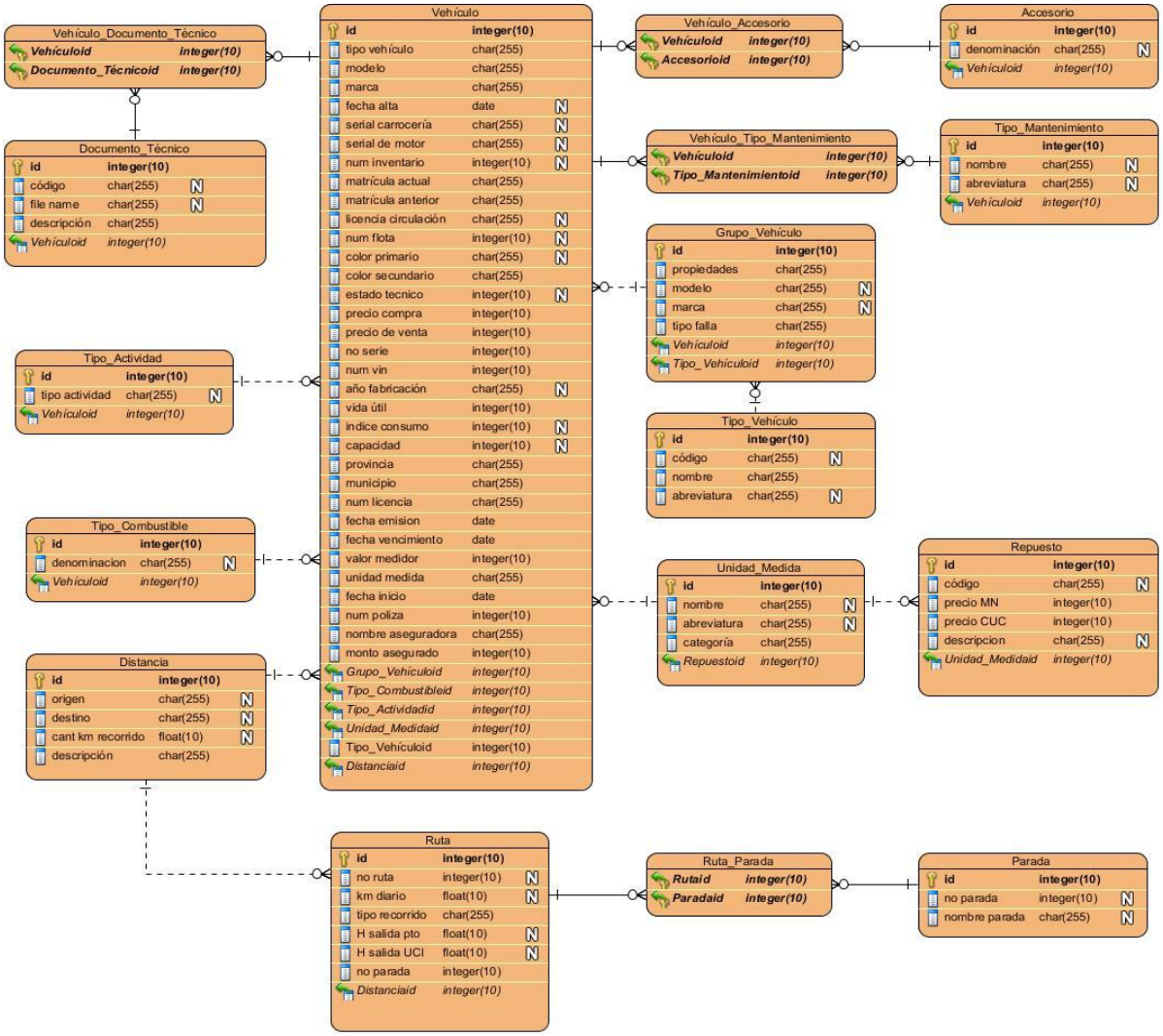


Figura 7. Modelo de datos.

2.4 Implementación de la migración del Sistema Xedro-Orbita a ODOO

La implementación de las funcionalidades de los módulos Configuración, Clasificadores y el proceso Vehículo permitió obtener como resultado un producto que cumple con las necesidades de la Dirección de Transporte de la UCI. Dentro de los principales aspectos analizados durante esta disciplina se encuentran los que a continuación se abordan.

2.4.1 Estándares de codificación

Los estándares de código son una forma de “normalizar” la programación de forma tal que, al trabajar en un proyecto, cualquier persona involucrada en el mismo tenga acceso y comprenda el código (Microsoft, 2016).

Definir los estándares de codificación a utilizar en la investigación permite tener un estilo único en la implementación de la solución, lo cual facilitará el estudio y comprensión del código haciendo que sea más fácil a la hora de realizar cambios o dar mantenimiento. A continuación, se muestran ejemplos de los estilos de codificación definidos para la implementación de la aplicación, a partir del análisis de los estándares utilizados en la arquitectura de referencia Odoo:

Nombre de los archivos:

- ✓ Vista: módulo_nombre: vehiculo.xml

Clases de un modelo:

- ✓ Si la clase es nueva: class módulo_nombre.

```
class dat_vehiculo(models.Model):
```

Vistas:

- ✓ vista Form: id = “módulo_modelo_form”.

```
<record model="ir.ui.view" id="vehiculo_form">
```

- ✓ vista Tree: id = “módulo_modelo_tree”.

```
<record model="ir.ui.view" id="vehiculo_tree">
```

Menú:

- ✓ id = “menú_módulo_nombre”.

```
<menuitem id="Inicio" parent="Vehiculo" name="Inicio"/>
```

2.4.2 Implementación de la migración del Sistema Xedro-Orbita

Como parte del proceso de implementación, se muestra ejemplificado un fragmento de código de la funcionalidad: ***accesorio***, correspondiente al sistema Xedro-Orbita, desarrollado en Odoo, específicamente la funcionalidad permite adicionar un accesorio de acuerdo con los parámetros definidos.

```
class nom_accesorio(models.Model):
    _name = 'nom.accesorio'

    name = fields.Char('Denominación', required=True)
```

Figura 8. Fragmento de código de la funcionalidad accesorio.

2.4.3 Diagrama de componentes

Un componente es la parte modular, desplegable y reemplazable de un sistema que encapsula implementación y expone un conjunto de interfaces (Pressman, 2010).

El diagrama de componentes del módulo Clasificadores y el proceso Vehículo se observa en la figura siguiente:

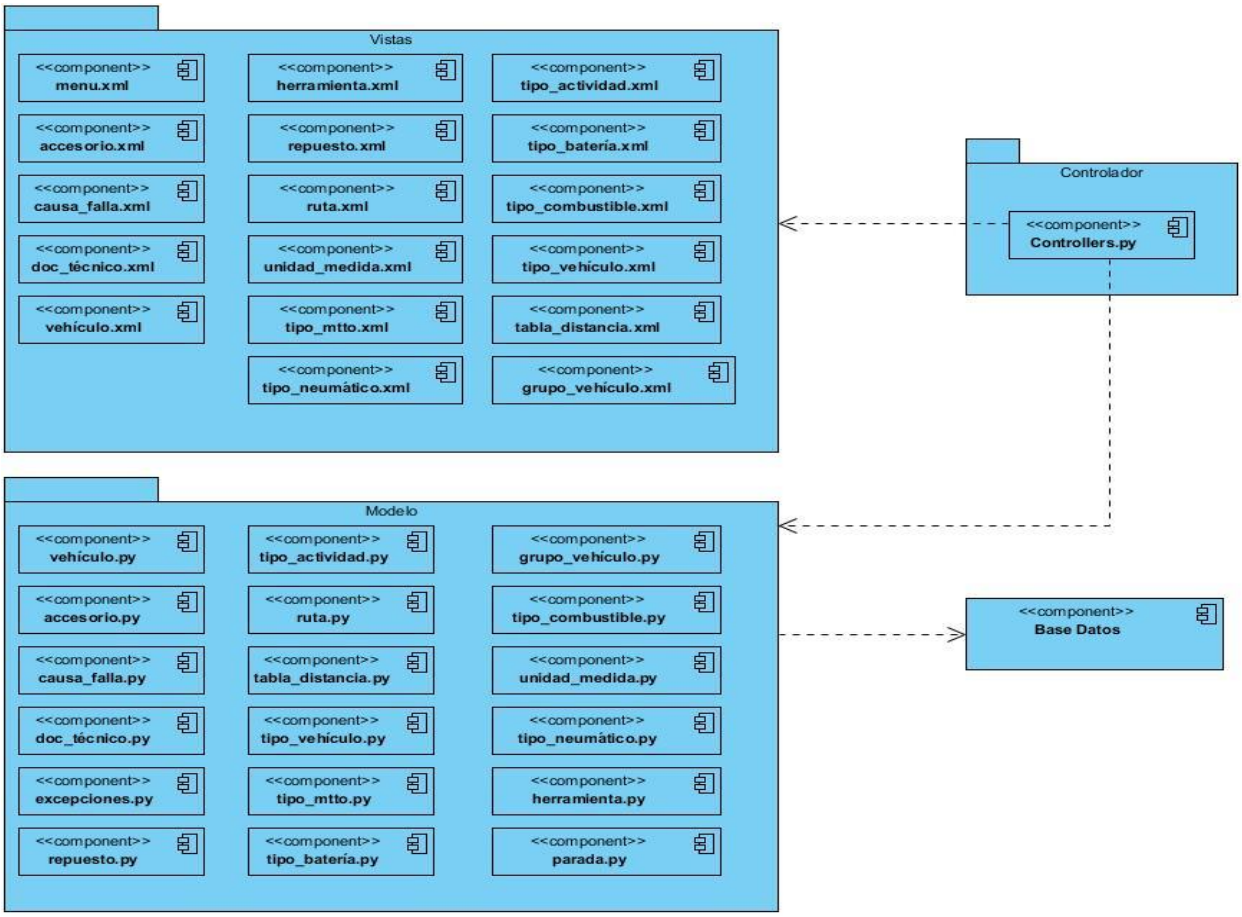


Figura 9. Diagrama de componentes.

Capítulo 2: Migración del Sistema Xedro-Orbita

2.5 Guía para migrar el Sistema Xedro-Orbita a la arquitectura de referencia ODOO

El presente trabajo de tesis está guiado por la Guía Cubana para la Migración al Software Libre (Feal Delgado, y otros, 2014), la cual se adopta al establecimiento de pasos, tareas y actividades que aportarán al proceso de migración del sistema Xedro-Orbita a la arquitectura de referencia Odoo.

A la hora de realizar la migración de la tecnología en la que fue desarrollado un sistema es importante conocer cuáles son las premisas, necesidades o problemas existentes, tal es el caso de la migración del marco de trabajo Sauxe al marco de referencia Odoo, del cual actualmente no existe documentación. Por ello surge la necesidad de realizar una guía que permita describir objetivamente el procedimiento para realizar la migración entre estos marcos de trabajo. Para ello es necesario cumplir con un conjunto de pasos claves:

Paso #1: La recopilación de requisitos anteriores permite conocer el "¿Qué?" con lo que se dispone de una completa visión del sistema que se va a migrar.

Paso #2: El siguiente paso es precisamente definir el "¿Cómo?". Para planificar la migración y tomar las decisiones técnicas más acertadas, será necesario estudiar y analizar los requisitos previos.

Paso #3: El próximo paso es fijar los objetivos de la migración. Más adelante estos objetivos podrán ser utilizados para evaluar el éxito de la migración:

- **Reducción de costes:** Uno de los objetivos principales de la migración, suele ser la reducción de costes. Puede reducirse de muchas maneras licencias, mantenimientos, etc.
- **Mejorar la seguridad del sistema:** Invertir en seguridad dentro de un sistema informático lo hace mucho más robusto y productivo. Por lo que la seguridad debe ser uno de los objetivos principales de la migración.
- **Mejorar la productividad del sistema:** Un sistema ha de funcionar correctamente, pero además debe hacerlo de manera eficiente. Un sistema lento puede producir pérdidas de tiempo de hasta una hora por trabajador y día, lo que a la larga conlleva grandes pérdidas a la empresa.

- **Ampliar la funcionalidad del sistema:** Al nuevo sistema se le pueden exigir nuevas funcionalidades, que el antiguo no cumplía, de esta manera se conseguirá una herramienta mucho más competitiva.
- **Obtener un sistema en constante evolución:** Los sistemas GNU/Linux están en constante evolución, todos los días se actualizan, tanto el sistema operativo como el software instalado, pero en ningún momento se exige pagar por actualizaciones ni el software se queda obsoleto y deja de funcionar. Además, los errores y fallos en el software se solucionan en horas, debido a la gran comunidad de desarrolladores que hay detrás de estas plataformas libres. (Feal Delgado, y otros, 2014)

Los pasos anteriormente planteados servirán como meta primaria para obtener una migración lo más certera posible. Con ellos, una vez analizados, se puede obtener una guía de pasos para migrar, como lo es en el caso del análisis: de Sauxe a Odoo, lo cual se muestra a continuación:

2.5.1 Guía de pasos para migrar del marco de trabajo Sauxe al marco de referencia Odoo

Para la migración del sistema Xedro-Orbita al marco de referencia Odoo, primeramente se realizó una revisión del sistema a migrar, con el fin de entender concretamente cuales son las causas que conllevan a realizar dicho proceso, además de realizar un profundo estudio sobre las tecnologías que componen ambos sistemas, para así poder encontrar una solución factible a la hora de realizar dicho proceso de migración. La guía de pasos que se muestra a continuación fue utilizada para migrar el sistema Xedro-Orbita, pero puede ser utilizada como guía para la migración hacia Odoo de otros sistemas desarrollados con Sauxe.

A continuación se listan los pasos a seguir en el proceso de migración de Sauxe a Odoo:

1. Se realiza un amplio estudio y análisis del sistema y las tecnologías a migrar, como resultado de dicho proceso se obtiene lo siguiente:
   1.1. El marco de trabajo **Sauxe** en la capa presentación utiliza **ExtJs**, el cual no es libre, implicando que para ser utilizado hay que pagar licencia. Odoo por su parte no paga licencia porque es libre; y para este caso, en Odoo se crea un archivo "***.xml***" y en este se introducen los datos.
   1.2. En la capa de negocio hace uso del marco de trabajo Zend Framework, el cual no permite la generación automática de CRUD, la generación de código PHP mediante líneas de comandos

para la creación, el mantenimiento de aplicaciones, ni el almacenamiento en la caché de las vistas. Por su parte Odoo gestiona estos problemas permitiendo la generación automática de CRUD. Para ello se deberán introducir los nombres de los campos que se almacenarán en la base de datos, específicamente en el archivo ".xml" de la vista a la cual pertenece.

1.3. En la capa de acceso a datos hace uso de Doctrine, lo cual provoca bajo rendimiento y un mayor acoplamiento entre sus clases, por lo que se pierde la independencia y es más costoso a la hora del cambio hacia otra tecnología. Posee además problemas de seguridad (errores y vulnerabilidad), los cuales atentan contra la estabilidad y el rendimiento, porque esta versión ya no recibirá más soporte ni mantenimiento. Odoo gestiona estos problemas implementando automáticamente una inter-relación entre los módulos desarrollados y los que ya vienen por defecto en el sistema, además con el lanzamiento de cada nueva versión de Odoo se mejora aún más el rendimiento y la seguridad del mismo. Se utilizarán las inter-relaciones automáticas que Odoo genera dentro de los módulos desarrollados. Además se deberá actualizar Odoo para obtener la versión más reciente.

1.4. Sauxe utiliza la biblioteca "TCPDF", la cual no está integrada al marco de trabajo, esto requiere de integraciones adicionales para su funcionamiento, bajándole el rendimiento al sistema. Odoo posee la biblioteca "QWEB" y el módulo "Report", ambos designados a la generación de reportes. Por ello es necesario instalar el módulo: "Report", el cual se encargará de estas funciones.

2. Para empezar el proceso de migración se debe empezar preparando el entorno de trabajo con las herramientas necesarias para el desarrollo de la migración:

2.1. Contar con un Editor de Texto Competente como es el caso de "***SublimeText3***" para la escritura del código o bien un **IDE** como "***Pycharm 2018.1"***. Ambos deben brindar la capacidad de leer el lenguaje de programación **Python**.

2.2. Instalar la versión de Odoo más estable y actualizada, e ir a la carpeta de instalación y buscar la carpeta "**addons**", en donde se creará la carpeta que contendrá los nuevos módulos a desarrollar.

2.3. Una vez creada la carpeta, deberá poseer el siguiente formato en su interior:

2.3.1. Una carpeta llamada "***models***", la cual contendrá los modelos del sistema.

2.3.2. Una carpeta llamada "***static***", que poseerá información como el icono del sistema.

2.3.3. Una carpeta llamada "***views***", donde se guardarán los archivos "***.xml***" que mostrarán las vistas del sistema.

2.3.4. Crear un archivo llamado "__***init***__***.py***", el cual permitirá al sistema saber cuál es la carpeta que contendrá las vistas y cuál los modelos.

2.3.5. Crear un archivo llamado "__***manifest***__***.py***", en donde se encontrará la información necesaria para que Odoo reconozca el sistema. Se le insertarán los módulos con los cuales se interconectará el que se desarrolle. Se introduce además, el nombre del módulo a desarrollar, para que luego pueda ser encontrado e instalado en las aplicaciones que Odoo trae por defecto.

3. Una vez preparado el entorno de trabajo, se empiezan a crear los archivos que contendrán el código de los campos que se van a mostrar en el sistema, así como las validaciones de los mismos:

   3.1. Dentro de la carpeta "***models***", se creará un archivo llamado "__***init***__***.py***", en el cual se importará el modelo o los modelos con los que cuente el sistema, un ejemplo de importación de modelos dentro de este archivo puede ser "***import vehiculo***".

   3.2. También se creará el modelo o los modelos del sistema, los cuales contendrán la información de los campos que se van a mostrar, además se definirá que tipo de campo son: **entero (Integer)**, **fecha (Date)**, **texto largo (Text)**, **varchar (Char)**, **float (Float)**, entre otros que puede ser encontrados en la web principal de Odoo (ODOO, 2016). Se define además, el nombre del modelo, tanto en la vista externa del archivo como en la interna. Un ejemplo de nombre en la vista interna puede ser: ***" _name = 'vehiculo' "*** y de externa "***vehiculo.py***".

   3.3. Al final de cada modelo "****.py***" se escribirá el código pertinente a la validación de uno o varios campos, de esta forma se garantiza la inserción correcta de los valores.

   Para realizar estas validaciones, es necesario crear funciones de validación, para ello se debe utilizar el siguiente encabezado: "***@api.constrains('nombre_del_campo_a_validar')***".

4. En la carpeta "***views***" se crean los archivos ***"*.xml***", que mostrarán la información al cliente en el sistema, en las cuales los datos mostrados serán recopilados de los modelos creados:

   4.1. Al crear un archivo "****.xml***", se puede nombrar como se desee, pero se recomienda utilizar un estándar de codificación. Un ejemplo es "**vehiculo.xml**".

   4.2. Dentro de cada archivo "****.xml***", se hará una llamada al modelo con el cual se desea trabajar. Cada archivo "****.xml***" solo puede relacionarse con un modelo, de esta forma se prevee la inserción doble de datos en la base de datos y producir un fallo del sistema. El modelo que se va a

relacionar al archivo "***.xml***", será llamado de la la siguiente forma: "***<field name=''model''>nombre_del_modelo</field>***".

4.3. Al final de cada archivo "***.xml***", se debe insertar el código para listar los elementos insertados, así brinda la posibilidad de gestionar el CRUD del modelo que la vista está proporcionando. El código utilizado para listar los elementos tiene este encabezado: "***<record model=''ir.ui.view*** ***id=''cualquier_id''>''***. Así mismo el encabezado del CRUD del modelo en el archivo "***.xml***" es el siguiente: "***<record model=''ir.action.act_window'' id=''cualquier_id''>***"

4.4. Una vez hecho esto en el campo "**menuitem**" del archivo "***.xml***", se van a definir tanto el **id** de la vista, como su posición en la lista de los menús, la acción de generar **CRUD**, el nombre de la vista que se va a mostrar en el sistema y si pertenece a un padre o no, o lo que es lo mismo, si está agrupada en un conjunto de vistas o no. Esto es útil a la hora de separar una equis cantidad de vistas en un sub-grupo distinto al principal.

5. Para la generación de los reportes, explicado de forma global en el paso 1.4, se puede utilizar el módulo que trae Odoo por defecto "**Report**", pero además está disponible el módulo "**Jasper Report**" que no viene incluido en Odoo, pero puede ser integrado fácilmente. El "**Report**" recibe un archivo "**.xml**", pero debe ser programado en su totalidad, convirtiéndolo en un proceso engorroso. Por ello se sugiere utilizar el "**Jasper Report**", porque es más sencillo de utilizar, ya que este recibe el archivo "***.xml***" diseñado en la herramienta "**iReport**"; esta herramienta también posee gran usabilidad por la interfaz gráfica que posee y además se integra fácilmente a Odoo.

5.1. La herramienta "**iReport**" brinda la posibilidad de diseñar los reportes que se deseen generar, mediante una interfaz gráfica donde se diseña el reporte con todos los datos deseados. Crea una conexión con la base de datos en donde el sistema guarda la información, de esta forma puede chequear si el reporte va a mostrar información verdadera o falsa, cuando se insertan los campos del reporte.

5.2. Al integrar el módulo "**Jasper Report**", instalándolo en Odoo, habilita en el apartado "**Configuración**", una pestaña para crear la llamada a los reportes y el formato que se van a exportar, ejemplo: "**.PDF**"; "**.OSD**" y "**.XML**". También se pueden introducir otros datos, sobre el nombre personalizado del reporte y la ruta en donde será guardado.

5.3. Para concluir este proceso de creación de reportes, es necesario crear una carpeta nombrada "***reports***" y dentro estarán los archivos generados por el "**iReport**". De esta forma se vincula el

modelo desarrollado con los reportes correspondientes, además para visualizarlo se debe habilitar en la vista del modelo la opción de generar este reporte.

Los pasos anteriormente descritos permitirán guiar las migraciones que se deseen realizar del marco de trabajo Sauxe al marco de referencia Odoo. De existir alguna particularidad del sistema a migrar, se recomienda utilizar como complemento la documentación de Odoo (ODOO, 2016).

2.6 Conclusiones parciales

- En las Disciplinas Requisitos, Análisis y diseño e Implementación, se obtuvieron los artefactos necesarios para guiar el proceso de migración del sistema. Estos fueron desarrollados bajo el expediente de proyecto 4.0 de la metodología AUP-UCI, certificada en el Nivel 2 de CMMI.
- Para validar el diseño fueron utilizadas las métricas TOC y RC, obteniendo resultados satisfactorios en cada uno de los indicadores aplicados.
- Se obtuvo una guía de pasos, con el objetivo de apoyar las futuras migraciones del marco de trabajo Sauxe a la arquitectura de referencia Odoo.

CAPÍTULO 3: VALIDACIÓN DE LA MIGRACIÓN DEL SISTEMA XEDRO-ORBITA

Introducción

En el presente capítulo se ejecutan varias pruebas: caja blanca, caja negra, pruebas internas y pruebas de aceptación; las cuales permitirán validar el proceso de migración del marco de trabajo Sauxe a la arquitectura de referencia Odoo. Se obtienen las actas de aceptación, que avalan la aprobación del correcto funcionamiento del sistema Xedro-Orbita en la nueva tecnología y verificando la calidad del proceso de migración. Como complemento importante se valida además la investigación con la aplicación de las pruebas de rendimiento y seguridad al sistema migrado, utilizando para ello las herramientas Jmeter y Acunetix, las cuales arrojan resultados positivos.

3.1 Validación de la migración del Sistema Xedro-Orbita

La verificación y la validación del software incluyen un conjunto de procedimientos, actividades, técnicas y herramientas que se utilizan paralelamente al desarrollo del software, para asegurar que el producto resuelve correctamente el problema para el que fuera diseñado. El objetivo es prevenir las fallas desde los requisitos hasta su implementación. Respecto de las pruebas a realizar en el software, ellas pueden ser dinámicas o estáticas, de acuerdo a si se realizan o no sobre el código. Entre las pruebas dinámicas, están las llamadas de caja blanca y las de caja negra, "testeando" los procedimientos estructurales o las salidas. (Cataldi, 2000).

Una vez implementadas las funcionalidades correspondientes a cada requisito, se realizó una serie de pruebas para garantizar que las mismas cumplieran correctamente con los requisitos aceptados por el cliente. A continuación, se hace alusión a los tipos de pruebas y técnicas utilizadas.

3.1.1 Pruebas de caja blanca

La prueba de caja blanca, en ocasiones llamada prueba de caja de vidrio, es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba. Al usar los métodos de prueba de caja blanca se revisan estructuras de datos internas para garantizar su validez (Pressman, 2007).

Para muchos casos donde a la hora de implementar una funcionalidad en una sentencia de código, ejecutar una prueba exhaustiva de todos los caminos, es poco práctico. Por ello existen otras técnicas para el diseño de este tipo de pruebas, que permiten decidir qué sentencias o caminos se deben examinar con los casos de prueba. Entre las más utilizadas se encuentran:

- **Prueba del camino básico**: La prueba del camino básico se basa en derivar casos de prueba a partir de un conjunto dado de caminos independientes. Para obtener el conjunto de caminos independientes se construirá el Grafo de Flujo asociado y se calculará su Complejidad Ciclomática (Pressman, 2007).
- **Prueba de condición:** es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa. (Pressman, 2007)
- **Prueba de flujo de datos:** se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. (Pressman, 2007)
- **Prueba de bucles**: es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles. (Pressman, 2007)

Prueba del Camino Básico

Para ejecutar una de las pruebas de caja blanca, fue aplicada la Técnica del Camino Básico. Para el desarrollo de la misma, es necesario realizar el cálculo de la complejidad ciclomática del algoritmo que será analizado. Partiendo de la enumeración las sentencias de código del mismo, se procede a elaborar el grafo de flujo correspondiente a esta funcionalidad.

La siguiente figura muestra las sentencias enumeradas del código de la funcionalidad **Validación_norma ()** del requisito Tipo de batería. Se escoge este fragmento de código por tener mayor complejidad, aunque de igual forma se puede escoger cualquier otro de este tipo, ya que presentan la misma complejidad.

```
def validacion_norma(self):
    excepcion = self.env['nom.excepciones']                                    }1
    if self.norma:                                                             }2
        if not patron_cero.match(str(self.norma)):                             }3
            excepcion.excepcion_modelos_fom('Atención!','El campo "Norma" solo puede contener }4
            números y no puede empezar en cero.')
```

Figura 10. Sentencias enumeradas de la funcionalidad Validación_norma ().

A continuación se muestra el grafo de flujo correspondiente a la funcionalidad Validación_norma () del requisito Tipo de batería.

Figura 11. Grafo de flujo de la funcionalidad Validación_norma ().

Cálculo de la complejidad ciclomática a partir de un segmento de código

La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y brinda un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez (Pressman, 2007).

La complejidad ciclomática será calculada de tres maneras diferentes. Al calcular la complejidad del método Validación_norma () del requisito Tipo de batería, se utilizarán las tres vías posibles para así verificar que el cálculo sea realizado correctamente. Se escoge este método porque al aplicar la prueba con otras funcionalidades del sistema el resultado obtenido arroja el mismo resultado, por tanto se puede escoger cualquiera de estas funcionalidades.

Las fórmulas para realizar dicho cálculo son las siguientes:

1. V (G) = (A - N) + 2, V (G) = (3 - 4) + 2, V (G) = 1.

Siendo A la cantidad total de aristas del grafo y N la cantidad de nodos.

2. V (G) = P +1, V (G) = 0 + 1, V (G) = 1.

Siendo P la cantidad de nodos predicado (son aquellos de los cuales parten dos o más aristas).

3. V (G) = R, V (G) = 1.

Siendo R la cantidad de regiones que posee el grafo.

En cada una de las fórmulas V (G) representa el valor del cálculo. A partir de los resultados obtenidos en cada uno, se puede determinar que la complejidad ciclomática del código analizado es 1, que a su vez es

el número de caminos posibles a transitar por el flujo y el límite superior de casos de prueba que se le pueden aplicar a dicho código.

A continuación, se muestra el camino básico por donde puede circular el flujo:

Camino básico: 1-2-3-4.

Para el camino básico obtenido se realiza el siguiente caso de prueba:

Tabla 4. Diseño de caso de prueba para el camino básico.

Descripción	A partir de añadir los datos correspondientes al campo Norma, se comprueba si estos datos son válidos.
Condición de ejecución	Llenar el campo Norma introduciendo parámetros incorrectos.
Entrada	self.norma
Resultados esperados	Se espera que muestre un mensaje de error de validación.
Respuesta del sistema	Respuesta satisfactoria

Prueba de Condición o Cobertura de Condición

Otra de las pruebas de caja blanca aplicada en el presente trabajo de diploma es la Prueba de Condición (Pressman, 2007), o Cobertura de Condición (SUPERIOR, 2018).

La Cobertura de Condición se encarga de que cada sentencia se ejecute al menos una vez; cada condición en la decisión toma todos los posibles resultados al menos una vez. Esto se refiere a la ejecución de todas las condiciones "***if***" que contenga la sentencia de código, cada condición "***if***" será analizada de acuerdo a los parámetros que se le pasen al método en análisis, obteniendo luego cada resultado válido o no para cada caso de prueba. Esta prueba se encarga de demostrar que el código implementado se valide correctamente.

Al aplicar la Cobertura de Condición al fragmento de código mostrado en la Figura 10, perteneciente al requisito Tipo de batería, se obtienen los siguientes resultados:

Tabla 5. Cobertura de Condición funcionalidad Validación_norma ().

Cobertura Condición	**self.norma**	**patron_cero.match (str (self.norma))**	**Casos de prueba**
IF-1	norma < 1 y > 9	**patron_cero.match (str (5))**	(1) 5
IF-1	*	****	(2)
IF-2	norma < 1 y > 9	**patron_cero.match (str (5))**	(1) 5
IF-2	norma = 0	**patron_cero.match (str (0))**	(2) 0

La tabla anterior muestra los resultados obtenidos al aplicar la prueba, la cual se encarga de analizar las condiciones de la funcionalidad Validación_norma() del requisito Tipo de Batería, esta funcionalidad contienen dos condiciones "***if***", para ello analiza primeramente la condición **IF-1** de acuerdo a la variable "***self.norma***" que corresponde a la norma de una batería, esta variable está validad para solo obtener como resultados válidos números enteros y el campo solo admite números enteros del 1 al 9, todo lo demás se consideran valores inválidos. Luego analiza la condición **IF-2** de acuerdo a la variable "***self.norma***", validada de la misma forma que la condición anterior, por lo cual compara que fue introducida correctamente de acuerdo a la variable.

Por tanto la primera validación que ejecuta la prueba es la siguiente:

- Para la condición "**IF-1**": se introducen los datos correctos de acuerdo a la validación de la variable "**self.norma**", obteniendo como resultado en este caso 5, ya que 5 está dentro de los parámetros establecidos para la norma que van del 1 al 9 (el 0 se incluye pero si y sólo si, se introduce a continuación de los número enteros del 1 al 9). En este caso devuelve lo planteado en la tabla anterior para el caso de prueba (1) dando un resultado válido.
- Para la condición "**IF-1**": se introduce cualquier dato incorrecto de acuerdo a la validación de la variable "**self.norma**", obteniendo como resultado siempre un mensaje de error. En este caso no

devuelve nada porque fue introducido un dato incorrecto, función que realiza el "*****" como lo sugiere la bibliografía consultada.

- Para la condición "**IF-2**": siempre que sea una validación correcta, devuelve como resultado el dato introducido, para este caso el 5, número correcto comprendido dentro de la norma.
- Para la condición "**IF-2**": siempre que sea una validación incorrecta, devuelve como resultado un mensaje de error, para este caso se traduce a lo mismo que devolver el 0, ya que el 0 si y sólo si está solo, no es un número correcto comprendido dentro de la norma.

3.1.2 Prueba de caja negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (Pressman, 2007).

Método de partición de equivalencia

Dentro de las pruebas de caja negra se utiliza el método de partición de equivalencia. Es un método que divide el dominio de entrada de un programa en clases de datos de los que pueden derivarse casos de prueba. Se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada (Pressman, 2007).

Para realizar las pruebas de caja negra se utilizó la técnica **Partición Equivalente**. Definiendo para cada uno de los requisitos identificados un caso de prueba. A continuación, el Diseño de Caso de Prueba (DCP), realizado para la validación del requisito funcional Crear Vehículo.

Descripción del caso de prueba Crear Vehículo:

El requisito funcional permite crear (adicionar) un nuevo vehículo en el sistema, a partir de los atributos que lo componen.

Condiciones de ejecución:

- El usuario autenticado en el sistema debe tener los permisos para ejecutar esta acción.

- El usuario debe seleccionar la opción Vehículo.

Tabla 6. Diseño de caso de prueba de Caja Negra del requisito Crear Vehículo

Escenario	**Descripción**	**Matrícula anterior**	**Matrícula actual**	**Tipo de Dato**	**Descripción**	**Respuesta del Sistema**	**Flujo Central**
EC 1.1 Crear vehículo introduciendo datos válidos.	El sistema debe permitir adicionar vehículo en el sistema.	HTR587	B034567	char	Matrícula con que se identifica el vehículo.	Se activa otras opciones para trabajar en el vehículo.	Se introducen los datos del vehículo correctamente. Se presiona el botón Guardar.
EC 1.2 Crear vehículo introduciendo datos inválidos.	El sistema debe permitir adicionar vehículo en el sistema.	HTR34	B0345	char		Se muestra los campos en rojo.	Se introducen los datos inválidos del vehículo. Se presiona el botón Guardar. Se muestra un mensaje de error. Se presiona el botón Guardar.
EC 1.3 Crear vehículo dejando campos vacíos.	Se dejan campos vacíos al adicionar un vehículo.					Se muestra los campos en rojo.	Se introducen los datos dejando algún campo en blanco. Se presiona el

							botón Guardar. Se muestran campos en rojo. Se presiona el botón Guardar.
EC 1.4 Descartar.	Se descarta la operación de adicionar un vehículo.						Se introducen o no los datos del vehículo. Se presiona el botón Descartar.

Tabla 7. Descripción de variables del requisito Crear vehículo

Nombre del Campo	Clasificación	Valor Nulo	Descripción
Serial de carrocería	Campo de texto	No	Caracteres alfanuméricos
Serial del motor	Campo de texto	No	Caracteres alfanuméricos
Número de inventario	Campo de texto	Si	Caracteres alfanuméricos
Matrícula actual	Campo de texto	Si	Caracteres alfanuméricos
Matrícula anterior	Campo de texto	Si	Caracteres alfanuméricos
Licencia de circulación	Campo de texto	No	Caracteres alfanuméricos
Número de flota	Campo de texto	Si	Solo números
Color primario	Campo de texto	No	Solo letra
Color secundario	Campo de texto	Si	Solo letra
Precio de compra	Campo de texto	Si	Solo números

Precio de venta	Campo de números	Si	Solo números
Número de serie	Campo de texto	Si	Solo números
Número de vin	Campo de números	Si	Caracteres numéricos
Año de fabricación	Campo de números	No	Caracteres numéricos
Vida útil	Campo de números	Si	Caracteres numéricos
Tipo de combustible	Fila de una tabla	No	N/A
Tipo de actividad	Fila de una tabla	No	N/A
Índice de consumo	Campo de números	Si	Solo números
Capacidad	Campo de números	Si	Caracteres numéricos
Unidad de medida	Fila de una tabla	No	N/A
Clasificación	Campo de texto	Si	Selección
Provincia	Campo de texto	Si	Solo letra
Municipio	Campo de texto	Si	Solo letra
Grupo de vehículo	Fila de una tabla	No	N/A
Tipo de vehículo	Campo de texto	Si	Solo letra
Marca	Campo de texto	Si	Solo letra
Modelo	Campo de texto	Si	Caracteres alfanuméricos
Fecha de alta	Componente fecha	Si	Selección
Estado técnico	Campo de texto	No	Selección

En los documentos entregables correspondientes al expediente de proyecto, se pueden encontrar los restantes diseños de casos de pruebas de caja negra aplicados a la solución.

Resultado de la prueba

La puesta en práctica el método de Partición de equivalencia perteneciente a las Pruebas de Caja Negra, arrojó resultados satisfactorios, luego de aplicar los DCP y con ellos corregir los errores detectados. Demostrando así, la consistencia del sistema desarrollado, a partir de los requisitos descritos y el correcto funcionamiento de las funcionalidades implementadas en los módulos Configuración, Clasificadores y el proceso Vehículo.

3.1.3 Pruebas internas

Se ejecutaron las Pruebas internas, definidas como disciplina en la metodología AUP-UCI que guía la investigación, estas pruebas fueron realizadas por el grupo de calidad del Centro CEIGE, a partir de los diseños de casos de prueba de los requisitos funcionales del sistema. El método de prueba aplicado demostró resultados satisfactorios desde el punto de vista funcional. Las no conformidades detectadas fueron analizadas y corregidas debidamente en el tiempo establecido, logrando un correcto comportamiento ante diferentes situaciones (entradas válidas y no válidas). A continuación, se muestra una tabla con el número de no conformidades detectadas en cada iteración de prueba realizada.

Tabla 8. Cantidad de no conformidades detectadas

Iteraciones	**Cantidad de No Conformidades**	**Tipo de No Conformidades**
1ra	14	Errores de interfaz y de validación.
2da	6	Errores de validación.
3ra	0	No se encontraron errores.

En la primera iteración fueron detectadas un total de ***14 No conformidades (NC)***: 1 de interfaz y 13 de validación. En la segunda iteración fueron detectadas ***6 NC***: todas de validación. En la tercera iteración no fueron detectados errores de ningún tipo, para un total de ***0 NC***, quedando de esta forma validado el sistema, emitiendo para ello un Acta de Aceptación (**Anexo 1**).

3.1.4 Pruebas de aceptación

Las pruebas de aceptación se realizan para que el cliente certifique que el sistema es válido según los requisitos solicitados. La planificación detallada de estas pruebas debe haberse realizado en etapas tempranas del desarrollo del proyecto, con el objetivo de utilizar los resultados como indicador de su validez: si se ejecutan las pruebas documentadas a satisfacción del cliente, el producto se considera correcto, y por tanto, adecuado para su puesta en producción. (Ramos, 2007).

El cliente, luego de haber revisado los productos de trabajo entregados, determina que está de acuerdo con el producto final mostrado, el cual fue desarrollado bajo los requisitos previamente definidos. La aceptación del producto arrojó un total de ***18 solicitudes de cambios*** en ***3 Iteraciones***, desglosadas en: ***10*** en la ***1ra iteración***, ***8*** en la ***2da iteración*** y ***0*** en la ***3ra iteración***, las cuales fueron resueltas en tiempo y forma cumpliendo con los requisitos planteados por el cliente. Estas solicitudes pueden ser consultadas en el documento entregable "***Solicitud de cambio de mejora.ods***", correspondiente al expediente de proyecto. Para obtener una mayor claridad de los resultados obtenidos, se muestra la siguiente gráfica:

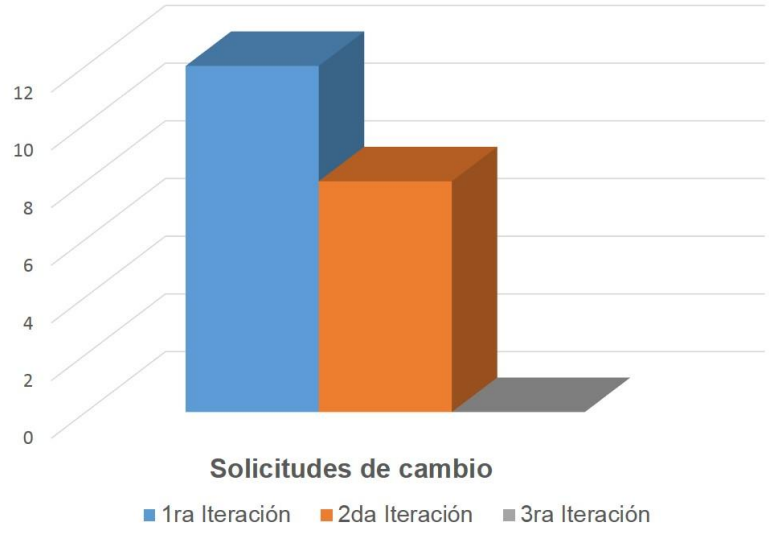


Figura 12. Solicitudes de cambio realizadas por el cliente.

Estas solicitudes fueron solucionadas para el correcto funcionamiento del sistema. Una vez comprobada la correcta implementación de los requisitos, se emitió el Acta de aceptación con el cliente, la cual puede ser consultada en el **Anexo 2**.

3.2 Validación científica aplicando Pruebas de rendimiento

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

Existen varios tipos de pruebas de rendimiento, entre los más utilizados se encuentran:

- Pruebas de Capacidad: Su objetivo es encontrar los límites de funcionamiento del sistema y detectar el cuello de botella o elemento limitante para poder actuar en caso de ampliación del servicio.
- Prueba de estrés: Someten al sistema a una carga por encima de los límites requeridos de funcionamiento.
- Pruebas de carga: Intentarán validar que se alcanzan los objetivos de prestaciones a los que se verá sometido el sistema en un entorno productivo. (ISTQB, 2014)

El empleo de este tipo de pruebas sirve para diferentes propósitos tales como demostrar que el sistema cumple los criterios de rendimiento, comparar dos sistemas para encontrar cuál de ellos funciona mejor o medir qué partes del sistema o de carga de trabajo provocan que el conjunto rinda mal. Para su diagnóstico se utilizan herramientas que pueden ser monitorizaciones que midan qué partes de un dispositivo o software contribuyen más al mal rendimiento o para establecer niveles que mantenga un tiempo de respuesta aceptable. (Pressman, 2007)

Apache JMeter

La aplicación **Apache Jmeter**, es un software de código abierto, una aplicación Java diseñada para cargar el comportamiento funcional de la prueba y medir el rendimiento. Originalmente fue diseñado para probar aplicaciones web, pero desde entonces se ha expandido a otras funciones de prueba. Puede ser utilizado para probar el rendimiento tanto en recursos dinámicos como estáticos, aplicaciones web dinámicas. Se puede usar para simular una gran carga en un servidor, grupo de servidores, red u objeto para probar su fortaleza o analizar el rendimiento general bajo diferentes tipos de carga. (Apache Software Foundation, 2018)

Las características de Apache JMeter incluyen:

- Capacidad para cargar y probar el rendimiento de diferentes aplicaciones / servidores / tipos de protocolos: Web: HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET), Servicios web SOAP / REST, FTP, Base de datos a través de JDBC, LDAP, Middleware orientado a mensajes (MOM) a través de JMS, Correo: SMTP (S), POP3 (S) e IMAP (S), Comandos nativos o scripts de Shell, TCP, Objetos de Java.
- IDE de prueba con todas las funciones que permite la grabación rápida de Plan de prueba (desde navegadores o aplicaciones nativas), compilación y depuración. Modo de línea de comandos (modo no guiado / sin cabeza) para cargar la prueba desde cualquier sistema operativo compatible con Java (Linux, Windows, Mac OSX).
- Fácil correlación mediante la capacidad de extraer datos de los formatos de respuesta más populares, HTML, JSON, XML o cualquier formato de texto. Fácil integración continua a través de bibliotecas de código abierto de terceros para Maven, Graddle y Jenkins. (Apache Software Foundation, 2018)

JMeter no es un navegador, funciona a nivel de protocolo. En lo que respecta a servicios web y servicios remotos, JMeter se parece a un navegador (o más bien, varios navegadores); sin embargo, JMeter no realiza todas las acciones admitidas por los navegadores. En particular, JMeter no ejecuta el Javascript encontrado en las páginas HTML. Tampoco representa las páginas HTML como lo hace un navegador (es posible ver la respuesta como HTML, pero los tiempos no se incluyen en ninguna muestra, y solo se muestra una muestra en un hilo a la vez). (Apache Software Foundation, 2018)

Resultados de las pruebas de rendimiento utilizando Apache JMeter

Para la aplicación de este tipo de prueba fue utilizada la herramienta Apache JMeter en su versión 2.10, específicamente para realizar pruebas de rendimiento a los módulos Configuración, Clasificadores y el proceso Vehículo del sistema Xedro-Orbita implementado con Odoo; y al sistema Xedro-Orbita desarrollado con Sauxe. Se utilizó como servidor una máquina con las siguientes prestaciones: microprocesador Intel(R) Core (TM) i3 a 2.10GHz, 4Gb de memoria RAM y 40Gb de disco duro dedicados al sistema.

A continuación, se muestra una leyenda para comprender la Tabla 9, que ofrece la información referente a la comparación entre ambos sistemas.

Leyenda:

- Sistema 1: Módulos Configuración, Clasificadores y el proceso Vehículo de Xedro-Orbita implementado con Sauxe.
- Sistema 2: Módulos Configuración, Clasificadores y el proceso Vehículo de Xedro-Orbita implementado con Odoo.
- Máximo (Máx.): Tiempo máximo de ejecución para una petición con un usuario realizando peticiones de manera simultánea.
- Media: Tiempo de ejecución promedio de una petición con un usuario.
- Rendimiento (Kb/seg): Velocidad del sistema.

Tabla 9. Comparación del rendimiento de Xedro-Orbita para Sauxe y Odoo.

Indicadores	**Sistema1**	**Sistema2**
Media	382	242
Máximo	743	307
Rendimiento (Kb/seg)	4,7/seg	1,3/seg

En la tabla se muestran los tiempos de respuestas del Sistema1 (con Sauxe) y el Sistema2 (con Odoo). Donde se puede apreciar que existen diferencias, tanto en el tiempo de ejecución promedio como en el tiempo máximo de respuesta de las funcionalidades de una petición, las cuales favorecen en ambos casos al Sistema2. Contribuyendo de esta forma reducir la velocidad, alcanzando el Sistema1 un rendimiento de 4,7/seg y el Sistema2 1,3/seg, evidenciando una gran diferencia de 3,4 segundos entre ambos sistemas, logrando así una mejora en la velocidad y rendimiento de los módulos Configuración, Clasificadores y el proceso Vehículo desarrollados con Odoo.

Los parámetros medidos muestran una mejora en cuanto al tiempo de respuesta, concluyendo que el rendimiento de los módulos migrados mejoró. Pero es válido señalar que, aunque se observa una mejora

en el rendimiento de estos módulos, se deben realizar pruebas de rendimiento una vez que Xedro-Orbita esté completamente migrado, lo que permitirá obtener una mejor comparación de los resultados.

3.3 Validación científica aplicando Pruebas de seguridad

Cualquier sistema basado en computadora, que maneje información sensible o realice acciones que puedan perjudicar (o beneficiar) impropiamente a las personas, es un posible objetivo para entradas impropias o ilegales al sistema. (Pressman, 2007)

Existen varias herramientas encargadas de controlar la seguridad de los sistemas web, como por ejemplo:

- **Acunetix**: analiza todos los puntos vulnerables de la Web, incluyendo Inyecciones de SQL[26], Cross Site Scripting[27] y otros. (Nápolez, 2010).
- **Nikto:** es un escáner de servidor web de código abierto (GPL) que realiza pruebas exhaustivas contra servidores web para varios elementos, incluidos más de 6700 archivos / programas potencialmente peligrosos, verificaciones de versiones obsoletas de más de 1250 servidores y problemas específicos de la versión en más de 270 servidores. También comprueba los elementos de configuración del servidor, como la presencia de varios archivos de índice, las opciones del servidor HTTP, e intentará identificar los servidores web y el software instalados. Los elementos de escaneo y los complementos se actualizan con frecuencia y se pueden actualizar automáticamente. Nikto no está diseñado como una herramienta furtiva. (CIRT.NET, 2018)
- **ZAP:** El Zed Attack Proxy (ZAP) de OWASP es una herramientas de seguridad gratuita. Permite encontrar automáticamente vulnerabilidades de seguridad en las aplicaciones web mientras se desarrollan y prueba las aplicaciones. Es mayormente utilizada por personas con experiencia en pruebas manuales de seguridad. (OWASP-ZAP, 2018)

La detección de estas vulnerabilidades requieren de un sofisticado motor de detección. Pero más importante que el análisis de vulnerabilidades Web no es la cantidad de ataques que puede analizar, si no la complejidad y el rigor con el que el analizador lanza ataques de Inyección de SQL, Cross Site Scripting y otros ataques. (Nápolez, 2010)

[26] Del inglés. Structured Query Language(Lenguaje de Consulta Estructurada).

[27] Del inglés. Cross-Site Scripting(Ejecución de Código entre Sitios).

El laboratorio de pruebas perteneciente a la Dirección de Calidad de la UCI utiliza estas herramientas para realizar pruebas de seguridad a los sistemas. Entre estas la que posee mayores ventajas para su uso es el Acunetix, porque según (Nápolez, 2010), es una tecnología de seguridad única que posee un avanzado motor de detección de las vulnerabilidades, las detecta rápidamente con un bajo número de falsos positivos, indica dónde encuentra cada una de ellas en el código y realiza informes de depuración. Además es capaz de la ejecución de código, inclusión de archivos, vulnerabilidades de autenticación, etc.

Resultado obtenido al aplicar pruebas de seguridad con el software Acunetix en Xedro-Orbita desarrollado con Sauxe (Anexo 3):

El sistema Xedro-Orbita implementado con Sauxe (Sistema2) arrojó como resultado **47 alertas de amenaza**, las cuales se muestran a continuación:

- **16 alertas de nivel alto:** en donde se pone de manifiesto el **Cross Site Scripting**, con el cual se pueden insertar **Scripts** dentro del sistema; y de esta forma obtener información sensible y detallada para su uso malicioso.
- **17 amenazas de nivel medio:** que consisten en no utilizar el sistema desde una dirección segura (se utiliza http y no https); además de que se pueden enviar comandos no autorizados que son transferidos desde un usuario, en el cual el sitio web confía. Permite a un atacante que controle el sistema enviar enlaces (***links***) de sitios maliciosos, con la finalidad de permitir a los usuarios acceder a estos y robar su información.
- **8 de nivel bajo:** que consisten en archivos de documento, los cuales pueden contener información que ayude a un atacante a reconocer el sistema, saber cuál es su versión o qué tipo de sistema es; y con ello realizarle un ataque satisfactorio. Directorios que no están correctamente referenciados y el uso de registro de información (***cookies***) en una conexión http.
- **6 alertas de nivel Informacional**: consistiendo en habilitar por parte del navegador la opción de autocompletar la contraseña de un usuario. Enlaces "*rotos*", con los que el sistema no concuerda y hace referencia a un posible camino del servidor, de esta forma el atacante puede obtener la estructura del servidor web y utilizar esta información para ataques futuros.

Resultado obtenido al aplicar pruebas de seguridad con el software Acunetix en Xedro-Orbita desarrollado con Odoo (Anexo 4):

A continuación se muestran un total de **28 alertas** arrojadas por el Sistema Xedro-Orbita desarrollado con Odoo (Sistema2):

- **9 alertas de nivel medio:** por no utilizar el sistema desde una dirección segura (se usa http y no https); y se pueden enviar comandos no autorizados que son transferidos desde un usuario en el cual el sitio web confía.
- **4 alertas de nivel bajo:** permitiendo subir un archivo que pueda contener un Script malicioso para el sistema, que permita la extracción no autorizada de información. El uso de **cookies** en una conexión http y no https. Utilización de cookies sin la aplicación de banderas seguras proporcionadas por una conexión https.
- **15 alertas de tipo Informacional:** consistiendo en habilitar por parte del navegador la opción de autocompletar la contraseña de un usuario. Enlaces "*rotos*", con los que el sistema no concuerda y hace referencia a un posible camino del servidor, de esta forma el atacante puede obtener la estructura del servidor web y utilizar esta información para ataques futuros.
- **0 vulnerabilidades de nivel alto**.

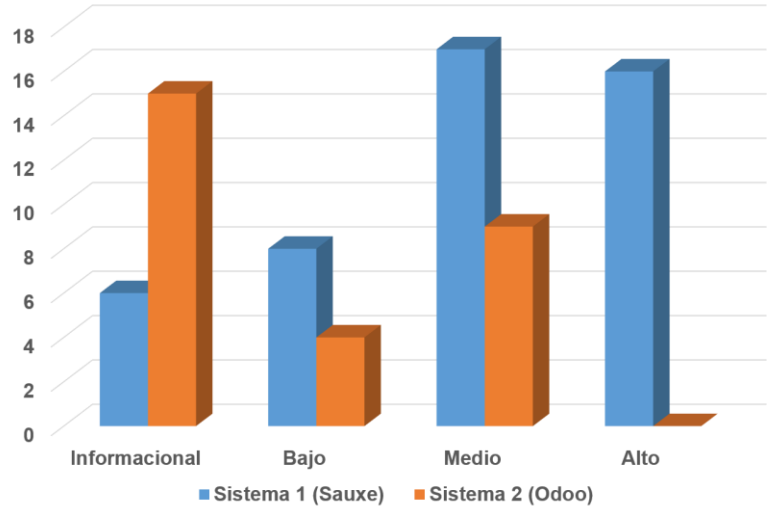


Figura 13. Gráfico comparativo entre el Sistema1 (con Sauxe) y el Sistema2 (con Odoo).

En la Figura 12 se muestra una comparación entre el Sistema1 (Xedro-Orbita con Sauxe) y el Sistema2 (Xedro-Orbita con Odoo). Arrojando primeramente 17 alertas de nivel medio el Sistema1, que disminuyeron a 9 en el Sistema2, mejorándolas gradualmente y brindando una mayor seguridad al Sistema2. Las 8 alertas de nivel bajo encontradas en el Sistema1 disminuyeron a 4 en el Sistema2, aunque . No siendo así con las 6 alertas de nivel informacional arrojas en el Sistema1, las cuales tuvieron un incremento de 9 hasta obtener 15 en el Sistema2, para este caso particular aunque el número aumenta en la nueva tecnología a utilizar, se considera poco significativo debido a la clasificación del tipo de alerta es solo de nivel informacional y los resultados no influyen en los niveles importantes, esto se evidencia y respalda al obtener una reducción a 0 alertas de nivel alto en Odoo de 16 alertas de nivel alto obtenidas en Sauxe.

Con estos resultados se puede apreciar que con las pruebas de seguridad realizadas, se lograron mejorar la mayoría de las alertas de seguridad presentes en Xedro-Orbita de grado alto, disminuyéndolas significativamente al ser migrado al marco de referencia Odoo, obteniendo de esta forma un sistema con mejor seguridad. A pesar de las diferencias arrojadas, las cuales no se consideran significativas de acuerdo con el resultado mostrado, se observa una mejora en los parámetros medidos en cuanto a la vulnerabilidad del sistema, por lo que se concluye que la seguridad de los módulos migrados fue mejorada. Los parámetros medidos muestran una mejora en las vulnerabilidades del sistema, pero es válido señalar que aunque se observa una mejora en la seguridad de estos módulos, se deben realizar pruebas de seguridad una vez que Xedro-Orbita esté completamente migrado, lo que permitirá obtener una mejor comparación de los resultados.

3.4 Conclusiones Parciales

- Se aplicaron las pruebas de caja blanca: técnica del Camino básico y la Prueba de condición y/o Cobertura de condición, comprobando que el flujo de trabajo de las funcionalidades fue correcto.
- Mediante la técnica de Partición equivalente se verificó en 3 iteraciones realizadas a partir de los casos de pruebas aplicados, que el sistema cumple con los requisitos definidos y aprobados.
- Se ejecutaron las pruebas internas y pruebas de aceptación, para el control de la calidad del sistema, que permitieron la corrección en tiempo y forma de las no conformidades detectadas.

- Con la realización de las pruebas de rendimiento y seguridad, empleando las herramientas Jmeter y Acunetix, se obtuvieron resultados satisfactorios, ya que las diferencias arrojadas no se consideran significativas de acuerdo con el resultado mostrado. Se obtiene además una mejora en los parámetros medidos en cuanto a la vulnerabilidad del sistema.

Conclusiones Generales

CONCLUSIONES GENERALES

La investigación realizada cumple los objetivos planteados, mediante la migración de los módulos Configuración, Clasificadores y el proceso Vehículo del sistema Xedro-Orbita a la arquitectura de referencia Odoo, permitiendo arribar a las siguientes conclusiones:

- Para la elaboración del marco teórico de la investigación se realizó el estudio del estado del arte, obteniendo como resultado las herramientas, tecnologías y la metodología utilizadas para realizar la migración desde el marco de trabajo Sauxe a la arquitectura de referencia Odoo del Sistema Xedro-Orbita.
- Con la redefinición de los requisitos de los módulos Configuración, Clasificadores y el proceso Vehículo del sistema Xedro-Orbita, se obtuvieron las descripciones de requisitos y salidas del sistema, que permitieron guiar el nuevo diseño para el proceso de migración.
- Para obtener un correcto rediseño de la estructura de componentes y clases, fueron aplicados los patrones de diseños GRASP y GOF, obteniendo resultados satisfactorios al demostrar que las clases poseen alta responsabilidad y coherencia. Además, el rediseño realizado fue validado utilizando las métricas TOC y RC, las cuales arrojaron buenos resultados al cumplir con todos los atributos de calidad medidos.
- La Disciplina de implementación permitió obtener la migración del sistema, realizado desde el marco de trabajo Sauxe a la arquitectura de referencia Odoo, a partir del diseño elaborado, cumpliendo con todos los requisitos redefinidos del sistema anterior.
- Con las pruebas internas y pruebas de aceptación fue validada la migración del sistema, obteniendo para ello las actas de aceptación que así lo acreditan. Además, fue demostrada y comprobada la mejoría en cuanto a rendimiento y seguridad, de la migración del sistema con respecto al anterior, a través de los resultados obtenidos al aplicar las herramientas Jmeter y Acunetix.

Recomendaciones

RECOMENDACIONES

Realizar la integración de los módulos Configuración, Clasificadores y el proceso Vehículo con el resto de los módulos del sistema Xedro-Orbita, implementados sobre la arquitectura de referencia Odoo, con el objetivo de que funcione como un todo y supla las necesidades del cliente.

Referencias Bibliográficas

REFERENCIAS BIBLIOGRÁFICAS

Alegsa, Leandro. 2010. DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA. *ALEGSA,Santa Fe-Argentina.* [En línea] 12 de octubre de 2010. [Citado: 30/05/2018] http://www.alegsa.com.ar/Dic/rendimiento.php

Almeida, Edson Saraiva de. 2014. Requisitos de Software. *Engenharia de Software . s.l. : Universidade Sao Judas Tadeus.* Universidad de las Ciencias Informáticas: s.n., 2014.

Apache Software Foundation. 2018. Apache JMeter™. *Apache JMeter™.* [En línea] The Apache Software Foundation. Montreal. Canada, 2018. https://jmeter.apache.org/

Baryolo Gómez , Oiner. 2012. *Marco de Trabajo Sauxe. Ficha Técnica.* s.l.: Brigadas Técnicas Juveniles, 2012.

Bascón Pantoja, Ernesto. 2004. El patrón de diseño modelo -vista - controlador (MVC) y su implementación el Java Swing. Universidad de las Ciencias Informáticas: s.n., 2004.

Blázquez Entonado, Florentino. 2001. Sociedad de la información y la educación. [aut. libro] Mérida : JAVIER FELIPE S.L. Universidad de las Ciencias Informáticas: s.n., 2001.

Buschmann, Frank. 1996. A System of Patterns. Universidad de las Ciencias Informáticas: s.n., 1996.

Cañete Pupo, Yanisleydi. 2010. *Libro de Ayuda del Marco de Trabajo Sauxe.* Universidad de las Ciencias Informáticas: s.n., 2010.

Cataldi, Zulma. 2000. Una metodología para el diseño, desarrollo y evaluación del software educativo [on line]. *Facultad de Informática.* Universidad de las Ciencias Informáticas: s.n., 2000.

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

CEIGE. 2015. *Informe del estudio realizado a Cedrux.* Centro de Informatización de Entidades. Universidad de las Ciencias Informáticas(UCI): s.n., 2015.

Referencias Bibliográficas

CIRT.NET. 2018. CIRT.NET. [En línea] 2018. https://cirt.net/Nikto2

CMMI, E. d. P. (2010). CMMI® para Desarrollo, Versión 1.3. Mejora de los procesos para el desarrollo de mejores productos y servicios (Vol. CMU/SEI-2010-TR-033 ESC-TR-2010-033).

CMS, C. f. M. M. S. (2008). Selecting a Development Approach.

Doctrine. 2009. [En línea] 2009. [Citado: 23/04/2018.] http://docs.doctrineproject.org/projects/doctrine1/en/latest/en/index.html

Donadello, Lic. Domingo. 2013. *Glosario de Ingenería de Software.* Buenos Aires: s.n., 2013.

Eclipse, OpenUP Copyright. 2011. Disciplina: Analysis & Design. [En línea] 2011. [Citado: 18/04/2018.] http://epf.eclipse.org/wikis/openupsp/openup_basic/disciplines/analysis_and_design,_0TX9AMlgEdmt

EcuRed. 2018. EcuRed. Enciclopedia colaborativa en la red. [En línea] 2018. [Citado: 07/05/2018.] https://www.ecured.cu/Est%C3%A1ndares_para_el_mantenimiento_del_software

Fajardo. 2011. El plan de migración a Código Abierto. Universidad de las Ciencias Informáticas: s.n., 2011, pág. 14.

Feal Delgado, William, Alvarez Acosta, Hugandy y Canosa Reyes, Rewer Miguel. 2014. Estrategia de migración al software libre en la Universidad de Cienfuegos. Universidad y Sociedad. Universidad de las Ciencias Informáticas: s.n., 2014, Vol. 6, págs. 75-81.

Fernández Pacheco, Juan Manuel. 2010. *Real Academia Española.* Madrid España: s.n., 2010.

Franco, J.A. 2005. *"UML en acción. Modelando Aplicaciones Web".* Universidad de las Ciencias Informáticas: s.n., 2005.

Gamma, Erich. 1995. *Design Patterns. Elements of Reusable.* Universidad de las Ciencias Informáticas: s.n., 1995.

Hernández Aguilar, MSc. Violena y Machado Peña, Ing. Yadira. 2010. *Describiendo requisitos verificables.* Perú: LACCEI, 2010.

Hernández Sampieri, DR Roberto, Fernández Collado, DR Carlos y del Pilar Baptista Lucio, Dra. María. 2010. *Metodología de la ivestigación.* 5ta edición. Universidad de las Ciencias Informáticas: s.n., 2010.

Referencias Bibliográficas

Hewlett, P. E. D. L. (2015). Requirements Management The key to successful IT projects. 4AA2-3219ENW, November 2015, Rev. 5

Ibrugor. 2014. Apache HTTP Server: ¿Qué es, cómo funciona y para qué sirve? [En línea] 2014. [Citado: 30/05/2018.] http://www.ibrugor.com/blog/apache-http-server-que-es-como-funciona-y-para-quesirve/

IEEE 610-1990. IEEE 610-1990 Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.

IngSoftwareII-CUFM. 2012. Desarrollo de Software. [En línea] noviembre de 2012. [Citado: 02/05/2018.] https://cufmingsoftware.wordpress.com/estandares-de-diseno

International Standard ISO/IEC 14764 IEEE. 2006. Universidad de las Ciencias Informáticas: Second Edition(2006), 2006, Software Engineering-Software Life Cycle Processes-Maintenance. Std 14764-2006.

ISO/12207-2. (2008). Systems and software engineering - Software life cycle processes. Institute of Electrical and Electronics Engineers.

ISTQB. 2014. PMOinformatica. [En línea] 29/01/2014. [Citado: 03/06/2018.] http://www.pmoinformatica.com/2014/01/tipos-de-pruebas-de-software-istqb.html

Jacobson, Booch Grady, Ivar y Rumbaugh, James. 1999. *The Inified Software.* s.n., 1999.

JasperSoft. 2014. JasperReports. [En línea] 6.0.0, 10/02/2014. [Citado: 02/05/2018.] http://community.jaspersoft.com/project/jasperreports-library

Kelly Naranjo, Juan Carlos. 2016. Personalización del módulo de Contabilidad de Odoo 8.0. Universidad de las Ciencias Informáticas: s.n., 2016.

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

Masadelante. 2018. ¿Qué es un navegador, explorador o buscador? [En línea] 2018. [Citado: 30/05/2018.] http://www.masadelante.com/faqs/que-es-un-navegador

Mastermagazine. 2015. [En línea] 15/01/2015. [Citado: 23/04/2018.]. http://www.mastermagazine.info/termino/5893.php

Microsoft. 2016. Revisiones de código y estándares de codificación. [En línea] 2016. [Citado: 02/05/2018.] https://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx

Referencias Bibliográficas

Mozilla. 2013. Mozilla Developers. [En línea] 2013. [Citado: 30/04/2018.]. http://www.mozilla.org

Nápolez. 2010. Acunetix Web Vulnerability Scanner. [En línea] 2010. [Citado: 04/06/2018.] https://www.acunetix.com

ODOO. 2016. [En línea] 20/01/2016. [Citado: 23/04/2018.]. http://www.odoo.com

ONE. 2007. *Tecnologías de la Información y las Comunicaciones(TIC)*, 2007.

OWASP-ZAP, Zed Attack Proxy. 2018. OWASP Zed Attack Proxy Project. [En línea] 2018. https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

Paradigm, VIsual. 2012. What is Visual Paradigm? [En línea] 07/052012. [Citado: 23/04/2018.]. http://www.visual-paradigm.com

PgAdmin. 2016. PgAdmin. [En línea] 2016. [Citado: 30/05/2018.]. https://www.pgadmin.org

PostgreSQL. 2016. PostgreSQL. [En línea] 2016. [Citado: 30/05/2018.]. https://www.postgresql.org

Potencier, Fabien. 2011. Librosweb. [En línea] 2011. [Citado: 02/05/2018.] http://librosweb.es/libro/symfony_1_4/capitulo_2/el_patron_mvc.html

Pressman, Roger. 2007. Ingenieria de Software. [aut. libro] Mc Graw Hill. *Un enfoque práctico. s.l.*, 2007.

Pressman, Roger. 2010. *Ingeniería de Software: Un enfoque práctico.* 2010.

Python. 2016. Python. [En línea] 03//02/2016. [Citado: 23/04/2018.]. http://www.python.org

Ramos, Isabel. 2007. Técnicas cuantitativas para la gestión en la Ingeniería del Software. Universidad de las Ciencias Informáticas: s.n., 2007.

Revista Cubana de Ciencias informáticas. **Pérez Guevara, Diosbel, y otros. 2014.** Especial UCIENCIA, Universidad de las Ciencias Informáticas: Ediciones Futuro, 23 de mayo de 2014, Plataforma Cubana de Migración a Código Abierto, Vol. 8, págs. 78-91. 2227-1899.

Rodríguez García, Juan David. 2012. Desarrollo de Aplicaciones web con symfony 1.4. [En línea] 2012. [Citado: 02/05/2018.]. http://juandarodriguez.es/cursosf14/unidad7.html

Rodríguez Sánchez, Tamara. 2014. Metodología de desarrollo para la Actividad productiva de la UCI. Universidad de las Ciencias Informáticas: s.n., 2014, Vol. 1.2.

Referencias Bibliográficas

Sencha. 2016. Sencha Inc. [En línea] 2016. [Citado: 23/04/2018.]. https://www.sencha.com

Domecq Babie, Dayana y Macías Hernández, Juan Darien. 2015. *Sistema para la gestión de información de Investigación, Desarrollo e Innovación Tecnológica del Centro de Informatización de Entidades.* Habana: UCI, 2015.

Sommerville, Ian. 2005. *Ingeniería de Software: 7ma edición.* 2005.

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

SUPERIOR, CORPORACIÓN UNIFICADA NACIONAL DE EDUCACIÓN. 2018. CORPORACIÓN UNIFICADA NACIONAL DE EDUCACIÓN SUPERIOR. *CORPORACIÓN UNIFICADA NACIONAL DE EDUCACIÓN SUPERIOR.* [En línea] 2018. http://clasescun.pbworks.com/f/pruebas_de_caja_blanca.pptx

Xedro-Orbita. 2014. *Xedro-Orbita. Sistema de Control de Flota y Mantenimiento.* Centro de Informatización de la Gestión de Entidades. Universidad de las Ciencias Informáticas: s.n., 2014.

XML. 2016. XML. [En línea] 06/02/2016. [Citado: 23/04/2018.]. http://www.xml.com

Zend, Technologies. 2016. Zend Technologies Ltd. [En línea] 25/01/2016. [Citado: 23/04/2018.] http://framework.zend.com

2009-2016. Renove Tecnología. Universidad de las Ciencias Informáticas, 2009-2016.

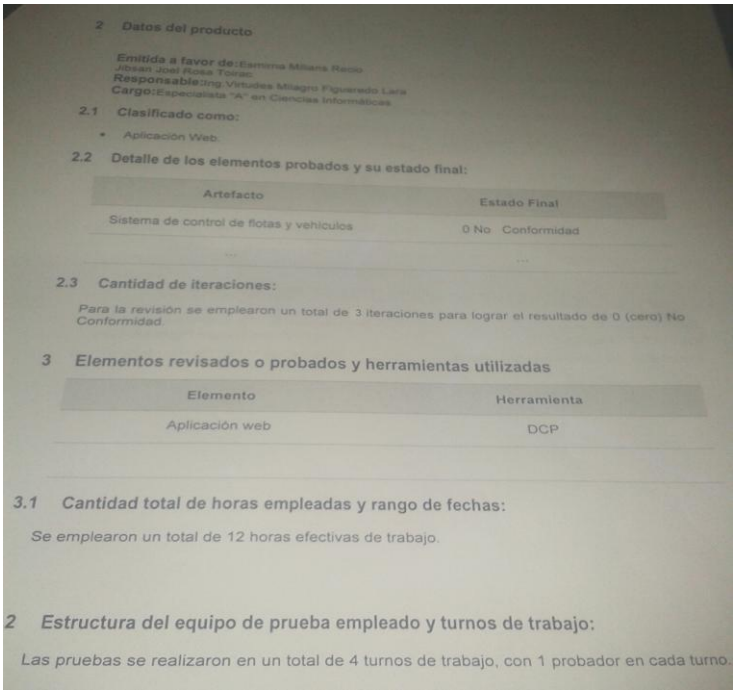
2018. EcuRed. Enciclopedia colaborativa en la red. [En línea] 2018. [Citado: 12/06/2018.] https://www.ecured.cu/Patrones_Gof#Patrones_Comportamiento

2018. Enciclopedia colaborativa en la red. [En línea] 2018. https://www.ecured.cu/Métrica_de_diseño#Tama.C3.B1o_operacional_de_clase_TOC

2000. *El Proceso Unificado de Desarrollo de Software. s.l :Rational Software Corporation*, 2000.

ANEXOS

Anexo 1. Acta de liberación de pruebas internas



2 Datos del producto

Emitida a favor de: Esmirna Milsara Recio
Jilsan Joel Rosa Toirac
Responsable: Ing. Virtudes Milagro Figueredo Lara
Cargo: Especialista "A" en Ciencias Informáticas

2.1 Clasificado como:

- Aplicación Web.

2.2 Detalle de los elementos probados y su estado final:

Artefacto	Estado Final
Sistema de control de flotas y vehículos	0 No Conformidad
...	...

2.3 Cantidad de iteraciones:

Para la revisión se emplearon un total de 3 iteraciones para lograr el resultado de 0 (cero) No Conformidad.

3 Elementos revisados o probados y herramientas utilizadas

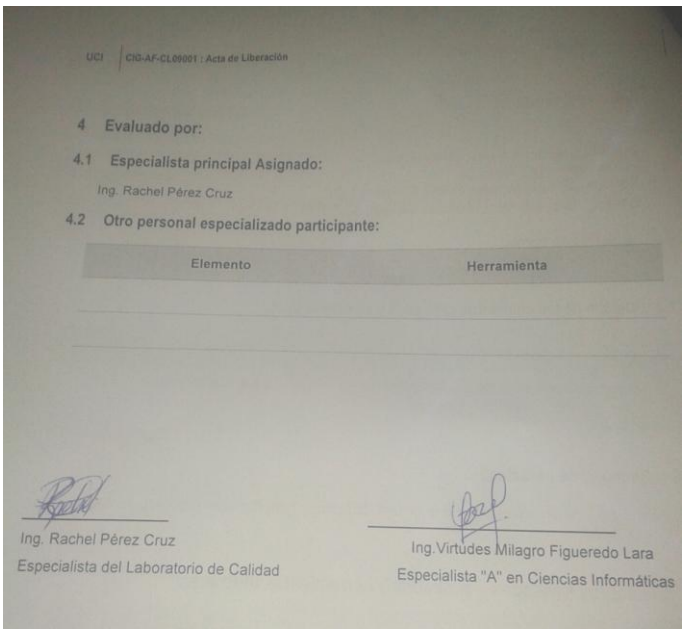
Elemento	Herramienta
Aplicación web	DCP

3.1 Cantidad total de horas empleadas y rango de fechas:

Se emplearon un total de 12 horas efectivas de trabajo.

2 Estructura del equipo de prueba empleado y turnos de trabajo:

Las pruebas se realizaron en un total de 4 turnos de trabajo, con 1 probador en cada turno.



UCI | CIG-AF-CL09001 : Acta de Liberación

4 Evaluado por:

4.1 Especialista principal Asignado:

Ing. Rachel Pérez Cruz

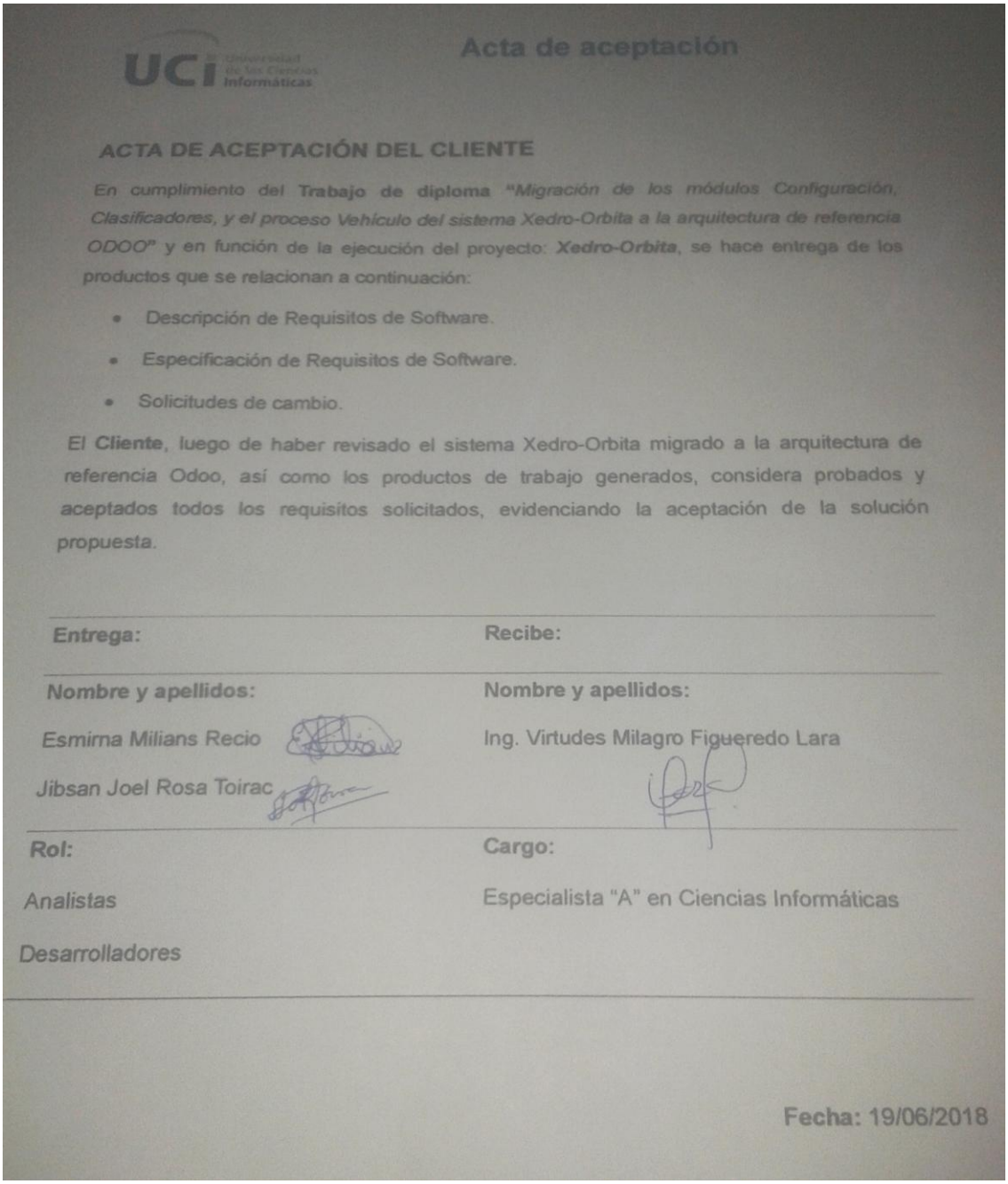
4.2 Otro personal especializado participante:

Elemento	Herramienta

Ing. Rachel Pérez Cruz
Especialista del Laboratorio de Calidad

Ing. Virtudes Milagro Figueredo Lara
Especialista "A" en Ciencias Informáticas

Anexo 2. Acta de liberación de pruebas de aceptación con el cliente



Acta de aceptación

UCI Universidad de las Ciencias Informáticas

ACTA DE ACEPTACIÓN DEL CLIENTE

En cumplimiento del **Trabajo de diploma** *"Migración de los módulos Configuración, Clasificadores, y el proceso Vehículo del sistema Xedro-Orbita a la arquitectura de referencia ODOO"* y en función de la ejecución del proyecto: ***Xedro-Orbita***, se hace entrega de los productos que se relacionan a continuación:

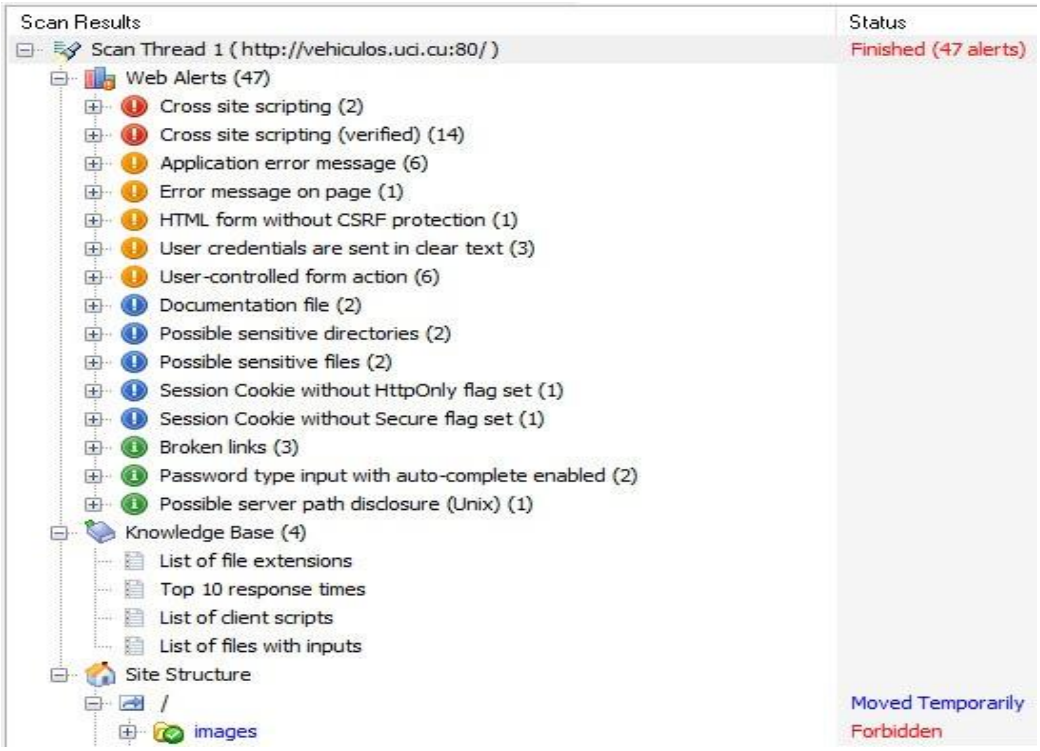
- Descripción de Requisitos de Software.
- Especificación de Requisitos de Software.
- Solicitudes de cambio.

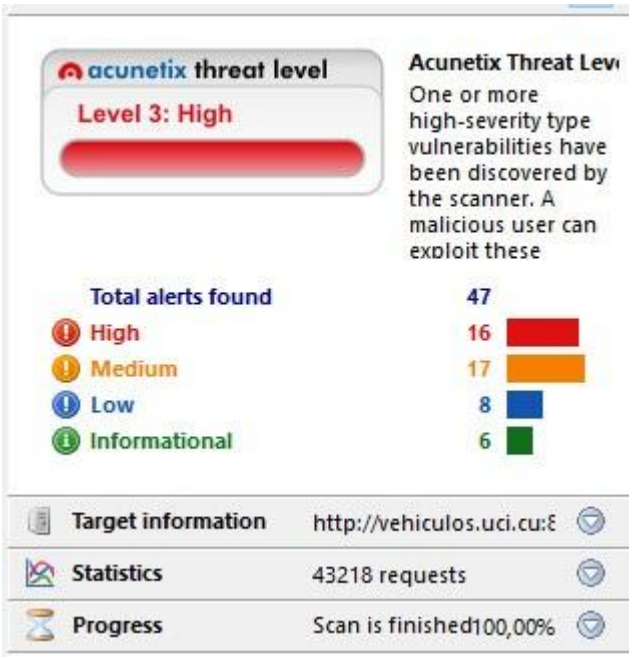
El **Cliente**, luego de haber revisado el sistema Xedro-Orbita migrado a la arquitectura de referencia Odoo, así como los productos de trabajo generados, considera probados y aceptados todos los requisitos solicitados, evidenciando la aceptación de la solución propuesta.

Entrega:	**Recibe:**
Nombre y apellidos:	**Nombre y apellidos:**
Esmirna Milians Recio	Ing. Virtudes Milagro Figueredo Lara
Jibsan Joel Rosa Toirac	
Rol:	**Cargo:**
Analistas	Especialista "A" en Ciencias Informáticas
Desarrolladores	

Fecha: 19/06/2018

Anexo 3. Pruebas de seguridad con herramienta Acunetix para Xedro-Orbita con Sauxe





Anexo 4. Pruebas de seguridad con herramienta Acunetix para Xedro-Orbita con Odoo