Facultad 6

“Sistema de Gestión de Información de las Órdenes de Servicios en Copextel UCI”

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

Autor: Yamil Martínez Mengual

Tutor: Ing. Angel Luis González Martínez

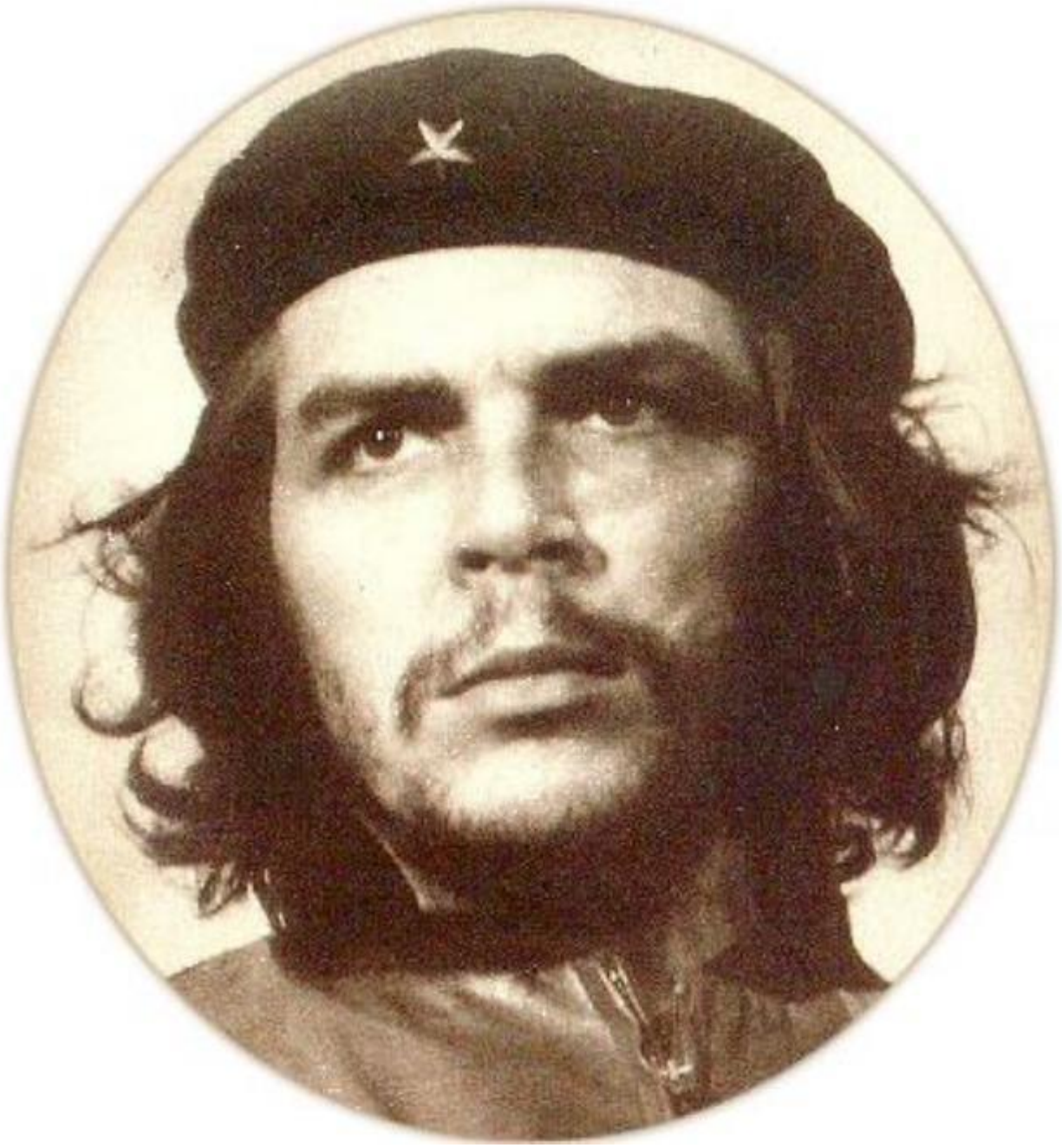
Co-tutor: Ing. Yadira Beatriz Reyes García

Consultante: Ing. Osvaldo Pérez González

Asesor: Ing. Yaneisy López Marrero

“Año 58 de la Revolución”

La Habana, 2016



"No se vive celebrando victorias sino superando derrotas"

Ernesto Guevara

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 6 de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____

Yamil Martínez Mengual

_____

Firma del Autor

Ing. Angel Luis González Martínez

_____

Firma del Tutor

Ing. Yadira Beatriz Reyes García

_____

Firma del Tutor

Agradecimientos

A mis Padres por no dejar de apoyarme y siempre estar presente, Kiko gracias, Ina gracias por ser padre y madre, gracias, gracias, a mis hermanos, mi familia, Lisi, Arami, lo más grande que tengo en este mundo, gracias a mis abuelos Emilio Hugo y Roberto Valdés siempre los tengo presente, gracias a mi tutor Ángel Luis, muchas gracias por el apoyo incondicional, por el tiempo dedicado, por sus consejos y ayuda brindada, a mi tutora Yadira Beatriz, a Osvaldo Pérez, a Alexis de Almas, por el tiempo dado y dedicado, gracias por su apoyo, gracias a Yaneisy López, gracias a los profesores que me enseñaron en toda mi vida de estudiante, gracias a mi profesora de secundaria Lazara, para quien fuimos sus hijos y con amor nos guio y aconsejo, gracias a ellos que sembraron las bases de la rectitud y perseverancia en mi, gracias a todos por su paciencia y amor, gracias al tribunal, Castaño, Elennis, Yinet, gracias al oponente, Willian, gracias a ellos que tuvieron la calma y supieron guiarme, gracias a mis compañeros de aula, decir, que estos años de estudios la pasamos bien aun en los momentos más difíciles de la carrera es poco, gracias por su ayuda año tras año, gracias a mis amigos, Gleder Noa, Yosbel García, Yandro, a todas mis amistades, a mis compañeros de trabajo, a todos aquellos que no mencione y que de una forma u otra me dieron aliento y fuerza para seguir adelante, y darle frente a todos los problemas.

Gracias.

Dedicatoria

A mi Madre

Miguelina Mengual León

A mis Hermanos

Lizandra y Aramis

RESUMEN

En la Universidad de las Ciencias Informáticas la división Copextel Servicios Técnicos Integral de la empresa Copextel es la encargada de atender las solicitudes relacionadas al soporte técnico y mantenimiento de televisión, telefonía, redes, climatización, energéticos y sistemas de alarma contra intruso e incendio. Por el gran número de equipamientos que tiene la universidad, constantemente se producen reportes por roturas en dichos equipos. Los técnicos de Copextel, una vez que atienden estos reportes realizan una orden de servicio la cual deja descrito el trabajo realizado. Actualmente, debido a la demora desde que se genera una orden de servicio hasta su facturación, se imposibilita llevar un control actualizado del cumplimiento del plan de trabajo asignado a la división. Por tal motivo se desarrolló un sistema para gestionar las órdenes de servicio y su información, el cual confiere a los trabajadores de la empresa una herramienta útil para trazar mejores estrategias mediante los datos almacenados, gráficos y estadísticas que se obtienen a partir de la inserción de las órdenes de servicio en el sistema. Para el desarrollo de la solución se utilizó Symfony en su versión 2.7 como *framework* de desarrollo y PHP 5.4 como lenguaje de programación. Como Sistema Gestor de Base de Datos MySQL 5.6.12, como servidor web Apache 2.4, así como HTML 5 para el maquetado del contenido y CSS 3 para los estilos. Fue utilizado PhpStorm como entorno de desarrollo y Visual Paradigm *for* UML 8.0 para el modelado de los diagramas de la investigación. Para la validación del sistema desarrollado se utilizaron las pruebas de caja negra basándose en los requisitos funcionales del sistema y las pruebas de validación, utilizadas para comprobar el valor de los datos introducidos en el sistema por los usuarios.

Índice

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

Cuba ha identificado desde muy temprano la conveniencia y necesidad de dominar e introducir en la práctica social las Tecnologías de la Información y las Comunicaciones (TIC); y lograr una cultura digital como una de las características imprescindibles del hombre nuevo, lo que facilitaría a la sociedad cubana acercarse más hacia el objetivo de un desarrollo sostenible. (SITEAL/TIC, 2014)

El uso correcto de las TIC brinda un entorno de soluciones amplias que incluyen la recuperación, el almacenamiento, el envío de datos de un sitio a otro y el procesamiento de la información para poder definir el alcance de los resultados y elaborar informes. Convertir las TIC en un sector de desarrollo estratégico para el país constituye uno de los principales retos en que se hallan enfrascados especialistas de esa y otras áreas del conocimiento, junto a la máxima dirección del Gobierno, que en aras de potenciar este campo ha creado instituciones para el desarrollo de esta ciencia como los Joven Club de Computación y la Universidad de las Ciencias Informáticas (UCI).

Con la creación de la UCI en el 2002, surge un centro para cubrir la demanda del sector informático en el país. Esta institución ha sido de vital importancia para el desarrollo de la industria del *software* debido a sus métodos de formación docente. En este centro se vincula el estudio con la producción de *software*, siendo la misma, un eje conductor para llevar a cabo procesos de cambios y transformaciones en todos los sectores del ámbito social.

Desde su fundación, en la UCI se encuentran prestando servicios varios grupos externos al centro, entre los que se encuentra la empresa Copextel SA. Esta entidad es la encargada de atender las solicitudes relacionadas al soporte técnico y mantenimiento de televisión, telefonía, redes, climatización, energéticos y sistemas de alarma contra intruso e incendio; siendo notables los ingresos a la economía del país.

Por el gran número de equipamientos que tiene la universidad, la misma está propensa a que existan roturas diarias, generando cada una de ellas solicitud de arreglo por parte del centro para el grupo de Copextel SA. Cada solicitud genera una orden de servicio la cual necesita ser archivada con datos específicos del trabajo realizado. Esta información es de vital importancia para la empresa, debido a que a partir de las órdenes archivadas se tiene conocimiento del trabajo realizado.

Actualmente Copextel SA utiliza el sistema Hércules para gestionar la información de órdenes de servicio. Dicho sistema, a pesar de las prestaciones que hoy brinda presenta las limitantes que a continuación se mencionan.

Existe un desfasaje de tiempo entre los datos reales del trabajo realizado y los que se encuentran en el sistema, debido a que desde que se genera una orden de servicio, hasta que se realice su facturación, pueden transcurrir varios días. El acceso solo está permitido para los trabajadores del departamento de economía y niveles superiores a los técnicos. Solo es accesible desde escritorio remoto y desde computadoras con un IP[1] específico, designado por el departamento de seguridad informática. No es multiplataforma ya que solo funciona en Sistemas Operativos Windows.

Estas limitantes representan un problema para los Jefes de áreas y los técnicos, ya que es de gran importancia para ellos contar con la información actualizada sobre los trabajos realizados. De esa manera es posible controlar el cumplimiento del plan asignado, alertar a los trabajadores que se encuentran atrasados e influir en la toma de decisiones.

El principal problema al que se enfrentan los trabajadores de la empresa actualmente es: la disponibilidad de la información sobre los trabajos realizados, lo cual atenta contra el cumplimiento de las planificaciones e influye en la toma de decisiones. Para contrarrestar esta situación los Jefes de áreas y técnicos utilizan documentos Excel y Access para registrar la información referente a las órdenes de servicios, la cual se actualiza cada vez que se atiende un reporte. Esta es una forma alternativa de realizar la gestión de las órdenes de servicio y que, además permite realizar planificaciones de nuevos trabajos y llevar el control de los realizados.

Sin embargo, esta no es una solución óptima, ya que puede traer consigo errores en la digitalización de los datos y existe el riesgo de pérdida de información. Además, cuando se cuenta con un volumen muy elevado, la búsqueda de información es ineficiente y en muchas ocasiones se hace tediosa. La demora es un factor que trae como resultado pérdida de dinero para la empresa; que los empleados tengan sobrecarga de trabajo; doble costo de tiempo o incluso, en el peor de los casos, que no se pueda dar respuesta a un cliente de su solicitud en el mismo día que la realizó.

A raíz de lo anteriormente expuesto, se plantea como **problema de la investigación**: ¿Cómo viabilizar la información de las órdenes de servicios generada por los técnicos de Copextel STI UCI, para mejorar la planificación de su trabajo diario?

Teniendo como **objeto de estudio:** Sistemas de Gestión de Información de las Órdenes de Servicio. Enmarcando la investigación en el **campo de acción:** Sistemas de Gestión de Información de las Órdenes de Servicio en Copextel STI UCI.

[1] **IP:** (Protocolo de internet), estándar que se emplea para el envío y recepción de información mediante una red.

Introducción

Dada la problemática planteada se propone como **objetivo general:** Desarrollar una aplicación web que permita gestionar la información de las órdenes de servicio generadas por los técnicos de Copextel STI UCI, para mejorar la planificación de su trabajo diario.

Para darle cumplimiento al objetivo general se desglosan los siguientes **objetivos específicos:**

1. Fundamentar la selección de la metodología, herramientas y tecnologías a utilizar en el desarrollo del sistema para la gestión de la información de las órdenes de servicio de Copextel STI UCI.
2. Diseñar las funcionalidades del Sistema de Gestión de Órdenes de Servicio para los técnicos de Copextel STI UCI.
3. Implementar las funcionalidades del Sistema de Gestión de Órdenes de Servicio para los técnicos de Copextel STI UCI.
4. Validar el sistema obtenido mediante la aplicación de pruebas funcionales, de rendimiento y de seguridad.

Para cumplir con los objetivos específicos antes descritos se definieron las siguientes **tareas de la investigación:**

1. Realización de un estudio de sistemas homólogos para analizar las tendencias de los mismos.
2. Selección de las tecnologías, herramientas y estándares que se necesitan para implementar la propuesta de solución.
3. Selección de la metodología de desarrollo a utilizar.
4. Definición de los requisitos funcionales y no funcionales de la propuesta de solución.
5. Modelación de Diseño y de Datos.
6. Aplicación y documentación de las pruebas funcionales, de rendimiento y de seguridad.

Los **métodos de trabajo científico** utilizados en esta investigación son los siguientes:

- **Histórico-lógico:** Este método permite realizar una investigación que da inicio con los orígenes de las áreas de procesos de gestión de órdenes y las tendencias que actualmente existen.
- **Analítico- sintético:** Para seleccionar los elementos de mayor importancia, se descompone el problema de la investigación con el objetivo de comprender el área de acción.
- **Modelación:** Utilizado para representar toda la información obtenida hasta el momento mediante diagramas, los cuales permiten reflejar las relaciones y cualidades de la aplicación web que se desea desarrollar.

El presente documento está estructurado de la siguiente forma: introducción, cuatro capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía, glosario de términos y anexos.

En el **Capítulo 1** se realiza la fundamentación teórica de la investigación. Se describe todo el contenido relacionado con los conceptos y definiciones fundamentales respecto a los sistemas de gestión de órdenes. Se realiza un análisis de sistemas existentes en el mundo y en Cuba orientados a la gestión de órdenes de servicios. Se explican y justifican las tendencias, tecnologías y herramientas en las que se apoya la solución al problema. En el **Capítulo 2**, "Propuesta de solución", se exponen y describen los procesos del negocio; se especifican los requisitos funcionales y no funcionales que cumple la solución. Se muestra el diagrama de Casos de Uso del Sistema y se describen los actores, detallándose los Casos de Uso del Sistema. En el **Capítulo 3** se define la estructura del diseño de la aplicación desarrollada, y para ello se representa el diagrama de clases del diseño y el modelo Entidad-Relación, así como el modelo de datos del sistema. Por último, en el **Capítulo 4,** "Implementación y pruebas", se hace una descripción de los estilos de codificación utilizados; se presenta el diagrama de componentes que permite comprender la estructura del sistema y los resultados de las principales pruebas realizadas a la aplicación para verificar que responda a un correcto funcionamiento de acuerdo a los requerimientos establecidos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo serán abordados conceptos importantes para la comprensión y el entendimiento del problema en cuestión, se realiza un breve análisis de la situación actual de algunos de los sistemas de gestión de órdenes existentes a nivel mundial y nacional. Así mismo, se exponen las características de las herramientas, metodologías y tecnologías que serán empleadas para dar cumplimiento a los objetivos de este trabajo.

1.1. Definiciones de interés

Se hace necesario partir de un conjunto de definiciones que conformarán el sustento teórico de la presente investigación. De esta manera se facilitará la comprensión de los elementos que componen la propuesta del diseño de un sistema para la gestión de información de órdenes de servicio para Copextel STI UCI.

Gestión

El término gestión se asocia al conjunto de trámites que se llevan a cabo para resolver un asunto o concretar un proyecto. La gestión es también la dirección o administración de una empresa o de un negocio. La gestión de información es el proceso que se encarga de suministrar los recursos necesarios para la toma de decisiones, así como para mejorar los procesos, productos y servicios de la organización. Esta última permite trabajar en función de la erradicación de las dificultades. (Ponjuan, s.f)

Sistema de gestión

Un sistema de gestión es una estructura probada para la gestión y mejora continua de las políticas, los procedimientos y procesos de la organización. Los sistemas de gestión ayudan a lograr las metas y objetivos de una organización mediante una serie de estrategias, que incluyen la optimización de procesos, el enfoque centrado en la gestión y el pensamiento disciplinado. Por tanto, el Sistema de Gestión es un conjunto de etapas unidas en un proceso continuo, que deja trabajar ordenadamente una idea hasta lograr mejoras y su continuidad. Hace que las empresas funcionen como unidades completas con una visión compartida. Ello engloba la información compartida, evaluaciones comparativas, trabajo en

equipo y un funcionamiento acorde con los más rigurosos principios de calidad. (Implementación SIG, 2015)

Orden de Servicio

Las órdenes de servicio o de trabajo como también se les conoce son utilizadas para registrar los distintos pedidos que llegan, ya sean: pedido de presupuesto, confirmación de un presupuesto, realización del servicio, etc., y determinar qué técnico realizará el servicio, qué día y en qué hora según la prioridad que se le asigne, especificando también en caso de que se realice el servicio, los materiales utilizados en el mismo. (AyudaAlma, 2012)

1.2. Estudio de los sistemas homólogos

Para lograr una mejor comprensión de las características del sistema a desarrollar, se hace necesario realizar el análisis de algunos sistemas con similitudes al propuesto en la presente investigación. A continuación, se expone el análisis desarrollado de los sistemas homólogos.

KRAMA GOT

Se trata de un producto para la gestión de grupos de órdenes de trabajo con agentes móviles: redes de comerciales, técnicos de mantenimiento, agentes logísticos, repartidores, reponedores y peritos de seguros. Es una solución que se puede adaptar a diferentes industrias y permite la integración con los sistemas *backend* de cada empresa.

Haciendo uso de su consola web se permite la administración de campañas, de órdenes de trabajo y de usuarios y operarios. Permite configurar, por cada grupo órdenes de trabajo, diferentes fases y cuestionarios a medida que proporcionan el flujo de trabajo de cada orden, permitiendo incluir preguntas, fotos y firmas. Este *software* fue desarrollado por el grupo KRAMA de Madrid, España. Es *software* propietario, por lo que para su uso es necesario pagar una licencia. (Krama GOT, 2013)

Moyex Órdenes de trabajo

Programa de gestión de órdenes de trabajo para servicio de reparaciones. Es una herramienta de gestión que permite organizar el trabajo de un servicio técnico y optimizar la productividad de sus operarios. Es un sistema de control de órdenes de trabajo que permite la elaboración de listados e informes basados en los distintos módulos: clientes, facturación, horas de trabajo y operarios. Es posible registrar:

Fechas relacionadas (recepción y entrega.)

- Cliente
- Técnico u operario.
- Precio y costo
- Detalles del trabajo pedido
- Descripciones de trabajo, solicitado, realizado y comentarios.

Fue desarrollado por Sistemas de información Paez una pequeña empresa americana. Es una aplicación de escritorio, por lo que su principal desventaja es que tiene que estar instalada en cada una de las PC de trabajo de los operarios, la licencia de este sistema, así como la aplicación deben de comprarse para su uso. (Moyex, 2011)

GATServer

Es una aplicación Web desarrollada por la Dirección de Gestión Tecnológica de la UCI para la gestión de las afectaciones tecnológicas (medios computacionales, redes y telefonía), que permite la gestión de reportes para dichas afectaciones. Este sistema tiene entre sus principales funcionalidades, gestionar todos los datos referentes a las incidencias técnicas que ocurren en la universidad. Su utilización permite almacenar de manera clara, sencilla y ordenada los datos referentes a las incidencias, así como una mayor velocidad en el proceso de reporte. Además, la aplicación cuenta con un equipo capacitado para darle solución a dichos incidentes. Fue desarrollada con la tecnología privativa ASP.NET (*Active Server Page*) y Microsoft .NET Framework.

Hércules

Hércules es el sistema establecido oficialmente por la empresa Copextel S.A. para la gestión de los servicios técnicos. Su funcionamiento está enmarcado en dos tipos de actividades: la creación y modificación de los datos, la obtención de información mediante búsquedas y la ejecución de informes específicos y generales. Además, es el sistema mediante el cual se procesan las órdenes de servicio. Permite gestionar, seguir el curso de esos servicios técnicos en toda su trayectoria para controlar desde el vale de salida de una pieza del almacén, hasta la reparación solicitada por el cliente final.

Hércules facilita la gestión de los servicios técnicos en su totalidad; mediante este se pueden hacer los vales de salida, órdenes de servicio, facturación para el cliente final y además ofrece toda la información

acerca de las condiciones en que se encuentra el proceso; por ejemplo, cuántos equipos están rotos, donde se encuentran y qué piezas presentan roturas. (Copextel, 2008)

Esta herramienta fue desarrollada sobre la plataforma .Net, utilizando para ello el lenguaje de programación C# y el *framework* .NET Framework.

Resultado del estudio de los sistemas homólogos.

Para el estudio de los sistemas homólogos, se tuvieron en cuenta un grupo de indicadores definidos por el autor del presente trabajo, para caracterizar dichos sistemas. A continuación, se muestra el resultado del análisis:

Tabla 1: Resultado del estudio de los sistemas homólogos

Indicadores / **Sistema**	**Multiplataforma**	**Pagar por su uso**	**Similitud de información**	**Características similares**
KRAMA GOT	SI	SI	NO	SI
Moyex	SI	SI	NO	SI
GATServer	SI	NO	NO	NO
Hércules	NO	NO	SI	SI

Luego de este análisis se puede concluir que, los sistemas KRAMA GOT, Moyex y GATServer, aunque se encargan de gestionar reportes, no se ajustan a la situación problémica planteada en la presente investigación, debido a que no gestionan la misma información con la que se trabaja en Copextel. Además, Moyex y KRAMA GOT son aplicaciones privativas por lo que es necesario pagar por su utilización. Por otra parte, el sistema Hércules tiene como principal desventaja, que existe un desfasaje de tiempo entre los datos reales del trabajo realizado y los que se encuentran en él, lo que provoca desconocimiento por parte de los técnicos y jefes de talleres acerca del cumplimiento del plan de producción asignado. Además, es un sistema al que solo tienen acceso los directivos y los trabajadores del departamento económico. No obstante, a todas las características señaladas, se tendrán en cuenta algunas de las funcionalidades de estos sistemas para la implementación de la propuesta de solución.

Por lo anteriormente planteado se propone desarrollar un sistema para gestionar la información de las órdenes de servicio en Copextel STI UCI.

1.3. Metodología de desarrollo de software

El empleo de una metodología durante el desarrollo de un *software*, le confiere a este y a la investigación que se le asocia, transparencia y calidad, elementos muy perseguidos por el cliente y necesarios para el producto. La aplicación correcta de una metodología ayuda a mejorar el tiempo de desarrollo del *software*; definir con exactitud el personal que requiere el proyecto, las herramientas a utilizar, conocimientos concretos sobre el problema a resolver, entre otros aspectos que deben ser chequeados y controlados.

Actualmente el desarrollo de *software* dentro de la actividad productiva de la UCI se caracteriza por el uso de diferentes metodologías de desarrollo entre robustas y ágiles, específicamente las nueve metodologías siguientes: XP, OPEN UP, RUP, BPM, DAC, KIMBALL, SXP, SCRUM, NOVA OPEN UP. (Rodríguez, 2015)

Al no existir una metodología de *software* universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable, se decide hacer una variación de la metodología AUP (*Agile Unified Process*), de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI.

AUP-UCI

El Proceso Unificado Ágil de Scott Ambler es una versión simplificada del Proceso Unificado de *Rational* (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de *software* de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo:

- Desarrollo Dirigido por Pruebas (*Test Driven Development* - TDD en inglés)
- Modelado ágil
- Gestión de Cambios ágil
- Refactorización de Base de Datos para mejorar la productividad.

Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva (Inicio, Elaboración, Construcción, Transición). Se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, llamada Ejecución y se agrega la fase de Cierre. Para una mayor comprensión se muestra la siguiente Tabla. (Rodríguez, 2015)

Tabla 2: Variación de las Fases de AUP adaptadas a la UCI

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

Además, se adaptan las disciplinas propuestas por AUP quedando de la siguiente manera:

1. Modelado de negocio
2. Requisitos
3. Análisis y diseño
4. Modelado de despliegue
5. Implementación
6. Pruebas internas

Se utilizará la metodología AUP UCI para el desarrollo de la aplicación, ya que esta se ajusta al grupo de desarrollo del proyecto, el cual está compuesto por un solo integrante. También el empleo de técnicas ágiles, que facilitan el trabajo del desarrollador del mismo, además, es una metodología adaptada al ciclo de vida productivo de la UCI. Se hacen uso de algunas disciplinas como, Modelado de negocio, Modelado de despliegue e Implementación, para el desarrollo de esta.

1.4. Lenguaje de Modelado

Lenguaje Unificado de Modelado (UML por sus siglas en inglés) es el lenguaje escogido para especificar, visualizar, construir y documentar los artefactos de sistemas de *software*. UML es gratuito, accesible a todos, y conforma la colección de las mejores técnicas de ingeniería que han probado ser un éxito en el modelamiento de sistemas grandes y complejos. (González, 2009)

1.5. Notación utilizada para modelar los procesos del negocio

La notación para el modelado de procesos de negocio (*Business Process Model And Notation* – BPMN por sus siglas en ingles), es una forma estándar y gráfica de modelar procesos de negocios. La meta fundamental de BPMN es proporcionar una notación estándar que sea fácilmente comprensible por todos los involucrados. Provee una notación simple para los flujos, independiente del entorno de implementación. En los últimos años, BPMN ha sido ampliamente adoptado por los productos relacionados a la Gestión de Procesos de Negocios (BPM - *Business Process Management*), tanto para los fabricantes de herramientas de Análisis de Procesos de Negocios (BPA - *Business Process Analysis*), como por los de herramientas de Modelado y *Suites* completas de BPM. (White, 2009)

1.6. Herramienta CASE

Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de *Software* Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software* reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del *software* en tareas como, el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores. (Hernández, 2011)

Visual Paradigm para UML v8.0

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

Se caracteriza por:

- Disponibilidad en múltiples plataformas
- Diseño centrado en casos de uso y enfocado al negocio que genera un *software* de mayor calidad.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.

También, la herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto y permite control de versiones. Cabe destacar igualmente su robustez, usabilidad y portabilidad. (Pressman, 2002)

1.7. *Framework* de desarrollo

Un *framework* (plataforma, entorno, marco de trabajo o marco de desarrollo) es una estructura de soporte definida, en la cual otro proyecto de *software* puede ser organizado y desarrollado. Los *frameworks* suelen incluir soporte de programas, bibliotecas, *software* para desarrollar y unir diferentes componentes de un proyecto de desarrollo de programas. Permiten, además facilitar el desarrollo de *software* y evitar los detalles de bajo nivel, permitiendo concentrar más esfuerzo y tiempo en identificar los requerimientos de *software*. (Alegsa, 2012)

JQuery

JQuery es una biblioteca del lenguaje JavaScript que permite a los diseñadores web agregar funcionalidades extra a sus aplicaciones. Es de código abierto y distribuido bajo la licencia MIT. Al ejecutarse del lado del cliente, puede actualizar información en la página web en tiempo real sin necesidad de actualizar la página. (Alegsa, 2015)

1.7.1. Framework de desarrollo PHP

CodeIgniter

CodeIgniter es un sencillo *framework* escrito para que funcione sobre PHP 4 y PHP 5. A continuación se describen un conjunto de características:

- Implementa el patrón MVC (Modelo-Vista-Controlador).

- Implementa el patrón *Active Record*[2] como capa de abstracción de base de datos, aunque no posee capa de ORM.
- Es compatible con varios motores de base de datos entre los que se encuentran MySQL, PostgreSQL, SQL Server, SQLite y Oracle.
- El mecanismo de control de sesiones es a través de *cookies*[3] y puede usar base de datos para el control de las mismas.

Symfony

Symfony es un *framework* diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que desde la propia instalación del *framework*, ya se tiene una estructura definida, así como la implementación de muchas funcionalidades, por lo que es innecesario cada vez que se crea una nueva aplicación web, preocuparse por la estructura que deba tener esta.

Symfony está desarrollado completamente con PHP 5[4]. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas basadas en Unix como en plataformas Windows. Además, cumple con las siguientes características:

1. Fácil de instalar y configurar en la mayoría de plataformas.
2. Independiente del sistema gestor de bases de datos seleccionado para el desarrollo de las aplicaciones.

[2] **Active Record:** Es un patrón de software utilizado en aplicaciones robustas, que permite trabajar los registros de una tabla en una base de datos como instancias de una clase, en los cuales se pueden aplicar métodos Buscar, Guardar y Borrar sin necesidad de utilizar sentencias SQL.

[3] **Cookies:** Información que se almacena en el disco duro del visitante de una página web a través de su navegador, a petición del servidor de la página.

[4] **PHP:** acrónimo recursivo de PHP: Hypertext Preprocessor.

3. Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
4. Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador solo debe configurar aquello que no es convencional.
5. Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además, es lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
6. Fácil de extender, lo que permite su integración con las librerías de otros fabricantes. (Eguiluz, 2012)

Se elige Symfony en su versión 2.7 ya que éste integra muchas librerías y herramientas que aportan fortaleza al producto final. Usa, además Doctrine o Propel como ORM *(Object Relational Mapping)* a opción del desarrollador, para el mapeo de objetos relacional. Unido a lo plasmado anteriormente las herramientas de generación de código, la interfaz de línea de comando para la instalación del sistema desarrollado y otras tareas comunes automatizadas, convierten a Symfony en un potente *framework* para PHP. Además, de seguir una política LTS *(Long Term Support)*, lo que significa que las versiones estables, como la escogida en esta investigación, se mantiene durante un período de 3 años con una continua corrección de errores.

1.8. Lenguajes de desarrollo

Actualmente existen diferentes lenguajes de programación para desarrollar en la web, agrupados en: lenguajes del lado del cliente y lenguajes de programación del lado del servidor. Los lenguajes de programación del lado del cliente son aquellos que solamente pueden ser interpretados por una aplicación cliente como el navegador Web. Por otro lado, los lenguajes del lado servidor son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él.

PHP 5.4

PHP (*PHP Hypertext Pre-processor*) es un lenguaje de programación usado generalmente en la creación de contenidos para sitios web. Es un lenguaje interpretado especialmente usado para crear contenido dinámico web y aplicaciones para servidores. Generalmente los *scripts* se embeben en otros códigos como HTML, ampliando las posibilidades del diseñador de páginas web enormemente. La interpretación y ejecución de los *scripts* PHP se hacen en el servidor, el cliente (un navegador que pide una página web)

solo recibe el resultado de la ejecución y jamás ve el código PHP. Permite la conexión a diferentes servidores de base de datos como son MYSQL, PostgreSQL y Oracle. (Alegsa, 2010)

Se decidió el uso de PHP en su versión 5.4 como lenguaje de desarrollo del lado del servidor. Este lenguaje aparte de las características mencionadas, está diseñado especialmente para la web. Como la aplicación a desarrollar es un sistema de gestión, la mejor forma de interactuar con los usuarios y recoger los datos necesarios es mediante la publicación de una aplicación web a la que se pueda acceder desde cualquier lugar, en el momento que lo deseen.

HTML 5

HTML (*Hyper Text Mark-up Language* o Lenguajes de Marcas de Hipertexto) es la base para la creación de páginas web tradicionales. El texto se modela a partir del uso de etiquetas. Es un estándar reconocido en todo el mundo, multiplataforma, soportado por muchos navegadores. Con él, se puede obtener un producto atractivo y rápido sin mucho esfuerzo por parte del desarrollador. Permite establecer enlaces entre diferentes documentos y la introducción de referencias a otras páginas por medio de enlaces de hipertexto. (Musciano y Kemmedy, 1999)

JavaScript

Lenguaje de programación interpretado, o sea, no requiere compilación. Es utilizado especialmente en páginas web embebido en el código HTML o similares. La mayoría de los navegadores pueden interpretar los códigos JavaScript incluidos en las páginas web. JavaScript es un lenguaje basado en prototipos, pues las nuevas clases se generan clonando las clases base (prototipos) y extendiendo sus funcionalidades. (Alegsa, 2010)

1.9. Entorno de Desarrollo Integrado

Un IDE (*Integrated Development Environment* - Entorno integrado de desarrollo) es una aplicación compuesta por un conjunto de herramientas útiles para un programador. Un entorno IDE puede ser exclusivo para un lenguaje de programación o bien, poder utilizarse para varios. Suele consistir de un editor de código, un compilador y un constructor de interfaz gráfica GUI. (Alegsa, 2010)

PhpStorm

Es un IDE de programación desarrollado por JetBrains[5]. Es uno de los entornos de programación más completos de la actualidad, permite editar código no sólo del lenguaje de programación PHP como lo indica su nombre. Actualmente es compatible con Sistemas Operativos como Windows, Linux y Mac OS X. Algo que destaca en PhpStorm es la ejecución del código en la misma interfaz del IDE. Así como también la interpretación y visualización inmediata de código PHP hasta en 5 de los navegadores web más populares.

Con PhpStorm se pueden crear nuevos proyectos sobre la base de algún *framework* de diseño web y CMS[6] como Bootstrap, HTML Boilerplate, Drupal Module, Foundation y Symfony. Se caracteriza por permitir la gestión de proyectos fácilmente, proporcionar un fácil autocompletado de código, soporta el trabajo con PHP 5.5 y sintaxis abreviada. (Editores de Código, 2014)

1.10. Sistema Gestor de Base de Datos

Sistema de gestión de base de datos o en inglés *Database Management System* (DBMS), es una agrupación de programas que sirven para definir, construir y manipular una base de datos. En la manipulación de una base de datos, los SGBD deben incluir un control de concurrencia, o sea, deben permitir a varios usuarios tener acceso "simultáneo" a la base de datos. Controlar la concurrencia implica que, si varios usuarios acceden a la base de datos, la actualización de los datos se haga de forma controlada para que no haya problemas. Un SGBD también debe encargase de cumplir las reglas de integridad y redundancias. Otra función importante en un SGBD es su capacidad de realizar copias de seguridad y de recuperación de datos. (Alegsa, 2016)

Oracle

El sistema gestor de Base de Datos de ORACLE puede ser configurado para dar el servicio de manera más eficiente según sea la configuración que éste tome. Una Base de Datos ORACLE es una colección de datos tratados todos ellos como una unidad. Una Base de Datos que está formada por diversos tipos de ficheros dentro de un sistema operativo. Físicamente, trataremos la Base de Datos como un conjunto de ficheros de base de datos y ficheros de traza. Lógicamente, la veremos como un conjunto de

[5] https://www.jetbrains.com/

[6] Content Managment System o Sistema de Gestión de Contenidos

diccionarios, tablas de usuarios y ficheros de traza conteniendo datos de recuperación de errores. Adicionalmente, una Base de datos requiere uno o más ficheros de control. Ellos contienen aquella información que identifica y describe el resto de la Base de Datos. (Bartomeu, 2007)

MySQL

Se puede decir que es un sistema de gestión de base de datos relacional, multihilo y multiusuario. Fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca. Por su sencillez y sus características es usado por muchas personas ya que consume muy pocos recursos, es usado tanto en aplicaciones sencillas como complejas. Es utilizado también en aplicaciones Web como Drupal, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla, además, sus conexiones generalmente son muy seguras.

Las principales características de este gestor de bases de datos son las siguientes:

Aprovecha la potencia de sistemas multiprocesador a la hora de realiza las búsquedas de datos.

- ✓ Soporta gran cantidad de tipos de datos para las columnas.
- ✓ Gran portabilidad entre sistemas.
- ✓ Soporta hasta 32 índices por tabla.
- ✓ Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos. (Castell, 2014)

PostgreSQL

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

Después de analizar las características y prestaciones de los SGBD mencionados anteriormente, se aprecian las ventajas que brinda MYSQL. Este sistema provee de gran capacidad de almacenamiento, consistencia, escalabilidad y soporta gran cantidad de datos. Es un sistema que funciona sobre múltiples

plataformas, incluyendo algunos como Windows, Mac OS X, BSD y Solaris. Por lo tanto, se decide desarrollar solución propuesta en esta investigación haciendo uso del SGBD MYSQL en su versión 5.6.

SQL Manager for MySQL

Ha sido seleccionado SQL Manager for MySQL dado que es una aplicación de diseño y manejo de bases de datos para su uso con MySQL. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL[7] simples hasta desarrollar bases de datos complejas, ofrece todas las herramientas necesarias para que un usuario experto pueda administrar bases de datos MySQL y posee una interfaz sencilla con asistentes que facilitarán el manejo a los usuarios más inexpertos.

1.11. Servidor web Apache

Apache constituye una tecnología gratuita de código abierto. Es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos. Permite la creación de sitios web dinámicos mediante el uso de *Server Side Includes* (SSI por sus siglas en inglés), de lenguajes de *scripting* como PHP, JavaScript y Python. Se ejecuta en varios sistemas operativos. Posee una arquitectura modular que admite ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API[8] de programación de módulos, para el desarrollo de módulos específicos. Tiene una alta configurabilidad en la creación y gestión de log; y soporta personalizar la respuesta ante los posibles errores que se puedan generar en el servidor.

Fue escogido el servidor web Apache en su versión 2.4, debido a su configurabilidad, robustez y estabilidad. La licencia Apache es una descendiente de las licencias BSD. La selección de esta tecnología como servidor web, está justificada por las siguientes características que posee Apache:

- Es posible su ejecución en una multitud de sistemas operativos, lo que lo hace prácticamente universal.
- Apache es una tecnología gratuita de código abierto.

[7] **SQL** (por sus siglas en inglés *Structured Query Language*) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

[8] **Interfaz de Programación de Aplicaciones** (del inglés *Application Programming Interface*) es el conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

- Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que sean instalados cuando sea necesario.
- Apache trabaja con gran cantidad de lenguajes como Perl, PHP y otros lenguajes de *script*.
- Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor.
- Es posible configurarlo para que ejecute un determinado *script* cuando ocurra un error en concreto.
- Permite la creación de ficheros de registro a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor. (Ciberaula, 2014)

Conclusiones del capítulo

En este capítulo se abordaron los elementos teóricos que sustentan la solución del problema llegando a las siguientes conclusiones:

1. Debido a que ninguno de los sistemas encontrados da una solución total que solucione lo planteado en la situación problemática, es necesario la creación de una aplicación que permita gestionar las órdenes de servicio de Copextel STI UCI.
2. El estudio realizado de las posibles tecnologías y lenguajes a utilizar, permitió determinar la base tecnológica necesaria para desarrollar el sistema propuesto.
3. La selección de la metodología de desarrollo AUP-UCI permitirá estructurar, planificar y controlar el proceso de desarrollo de la propuesta de solución y garantizará la calidad de dicho proceso.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

Introducción

La creación de sistemas informáticos es un proceso en el que se desarrollan artefactos que, luego de ser integrados, posibilitan responder a las necesidades del cliente. En este proceso se identifican varias etapas, que van desde la declaración del problema y los requisitos del sistema, hasta las pruebas y la liberación del mismo. En el presente capítulo se define la propuesta de solución a desarrollar, con el propósito de satisfacer el objetivo de la investigación. Para esto se partirá de la descripción del problema. Para optimizar la comprensión de la solución propuesta se realiza el análisis utilizando la metodología ágil AUP-UCI, donde describe el modelo conceptual y los principales procesos del sistema. Son definidos los requisitos funcionales y no funcionales. Por último, se presenta el diagrama de caso de uso del sistema y se describe uno de estos casos de usos.

2.1. Flujo actual del proceso de negocio

La División Copextel STI UCI es la encargada de las instalaciones, mantenimiento y soporte técnico en los sistemas de telecomunicaciones, ofimática, electromecánica, Sistema de Alarma de Detección de Incendio (SADI) y Sistema de Alarma Contra Intruso (SACI) de la UCI. Esta se encarga de atender todos los reportes generados por el centro universitario. A continuación, se describe el proceso de gestión de órdenes de servicio en el centro Copextel STI UCI.

Inicialmente las direcciones tecnológicas del centro universitario generan un reporte de cualquiera de los sistemas tecnológicos que atiende Copextel STI UCI, el cual puede ser de carácter de instalación, mantenimiento o soporte técnico. A su vez el taller correspondiente de la empresa se encarga de velar porque la solicitud sea atendida en el menor tiempo posible; una vez atendido el reporte se genera una orden de servicio con los datos del trabajo efectuado y los materiales utilizados en el mismo. Luego el técnico de Copextel entrega la orden de servicio en el departamento de Economía de la empresa en donde, luego de registrar los datos de dicha orden en el sistema Hércules, se procede a generar la factura correspondiente.

2.2. Propuesta del sistema

Con la implementación de la solución planteada se pretende informatizar el proceso de gestión de la información de las órdenes de servicio. Se desea, además que la aplicación permita la obtención de

informes de los datos almacenados en la misma, con el objetivo de controlar el estado de dichas órdenes de servicio. El sistema mostrará gráficas con los datos almacenados sobre el estado de cumplimiento del plan anual, mensual y cantidad de órdenes de servicio por taller, además de, permitir a los directivos y trabajadores en general conocer los atrasos o sobrecumplimiento de la división Copextel STI UCI.

Tendrá acceso a la aplicación todo el personal de Copextel STI UCI. Para ello, se permitirá la autenticación en el sistema luego de que se le registre como plantilla de Copextel STI UCI y obtenga su usuario, por lo que cualquier trabajador de Copextel STI UCI podrá acceder a la aplicación. Una vez que el usuario se autentique se le otorgarán privilegios para gestionar la información de las órdenes de servicio. Además, dichos usuarios solo podrán modificar las órdenes de servicio que ellos previamente hayan insertado y no la de otra persona.

Según el privilegio del usuario podrá llevar un control de los blogs de órdenes de servicio y los blogs de sellos entregados a cada trabajador, obteniendo de esta manera, la información de la cantidad de sellos y órdenes de servicio entregadas, las pérdidas, las canceladas, rotas y las que faltan por ser utilizadas por cada técnico. Por otra parte, todos los usuarios podrán descargar la información de las órdenes de servicio de su taller guardada en la base datos en un archivo Excel. Los cuadros y directivos de Copextel STI UCI serán administradores del sistema por lo que tendrán acceso total a sus funcionalidades. Así como la gestión de cualquier orden de servicio existente en el sistema.

2.3. Modelo de negocio

La modelación de proceso de negocio permite realizar una exploración del dominio del problema, con el fin de lograr comprensión por parte del equipo de desarrollo de los procesos que se realizan actualmente en la entidad y la relación que existe entre estos. De esta forma, se van determinando necesidades operacionales, así como restricciones que presenta la entidad, obteniéndose finalmente un entendimiento del negocio para dar paso a la fase inicial del sistema. Permite comprender las características del negocio a través de la descripción de los procesos.

Para la realización del modelo de negocio se utilizó la notación BPMN, teniendo en cuenta que la misma aporta una mayor visibilidad de las actividades que se realizan y ayuda a lograr un mejor entendimiento del flujo de trabajo existente entre las áreas.

2.3.1. Mapa de proceso del negocio

El mapa de proceso del negocio ofrece una visión general del sistema de gestión de órdenes de servicio. En él se representa el proceso que compone el sistema, así como sus relaciones principales. Dichas relaciones se indican mediante flechas y registros que representan los flujos de información. El proceso describe cómo se realiza el trabajo en la organización caracterizándose por ser observables, medibles, mejorables y repetitivos. (Samespinosa, 2009)

El mapa de proceso para el sistema de gestión de órdenes de servicios abarca un (1) proceso:

1. Generar orden de servicio.

Figura 1: Mapa del proceso del negocio

2.3.2. Actores del negocio

Cliente: persona que realiza el reporte.

Trabajador de Copextel: persona que atiende el reporte y realiza el servicio.

Almacenero: persona encargada de proveer los materiales a los técnicos de Copextel.

2.3.3. Descripción del proceso de negocio: Generar orden de servicio

Tabla 3: Descripción del proceso de negocio Generar orden de servicio.

Objetivo	Generar una orden de servicio después de atender satisfactoriamente un reporte.
Evento(s) que lo genera(n)	Reporte
Pre condiciones	Existencia de un reporte
Marco legal	N/A
Clientes internos	Técnico de Copextel, Cliente, Almacenero
Entradas	Reporte
Flujo de eventos	
Flujo básico Instalación y mantenimiento de equipos	
1.	Realizar reporte
2.	Atender reporte
3.	Solicitar materiales
4.	Entregar materiales
5.	Realizar servicio

6. Crear orden de servicio
Pos-condiciones
N/A
Salidas
1. Orden de servicio
Flujos paralelos
N/A
Flujos alternos
1 Realizar mantenimiento
2 Realizar instalación

2.3.4. Diagrama del proceso de negocio: Generar orden de servicio.

En el siguiente diagrama se muestra cómo se lleva a cabo el proceso de generar orden de servicio.

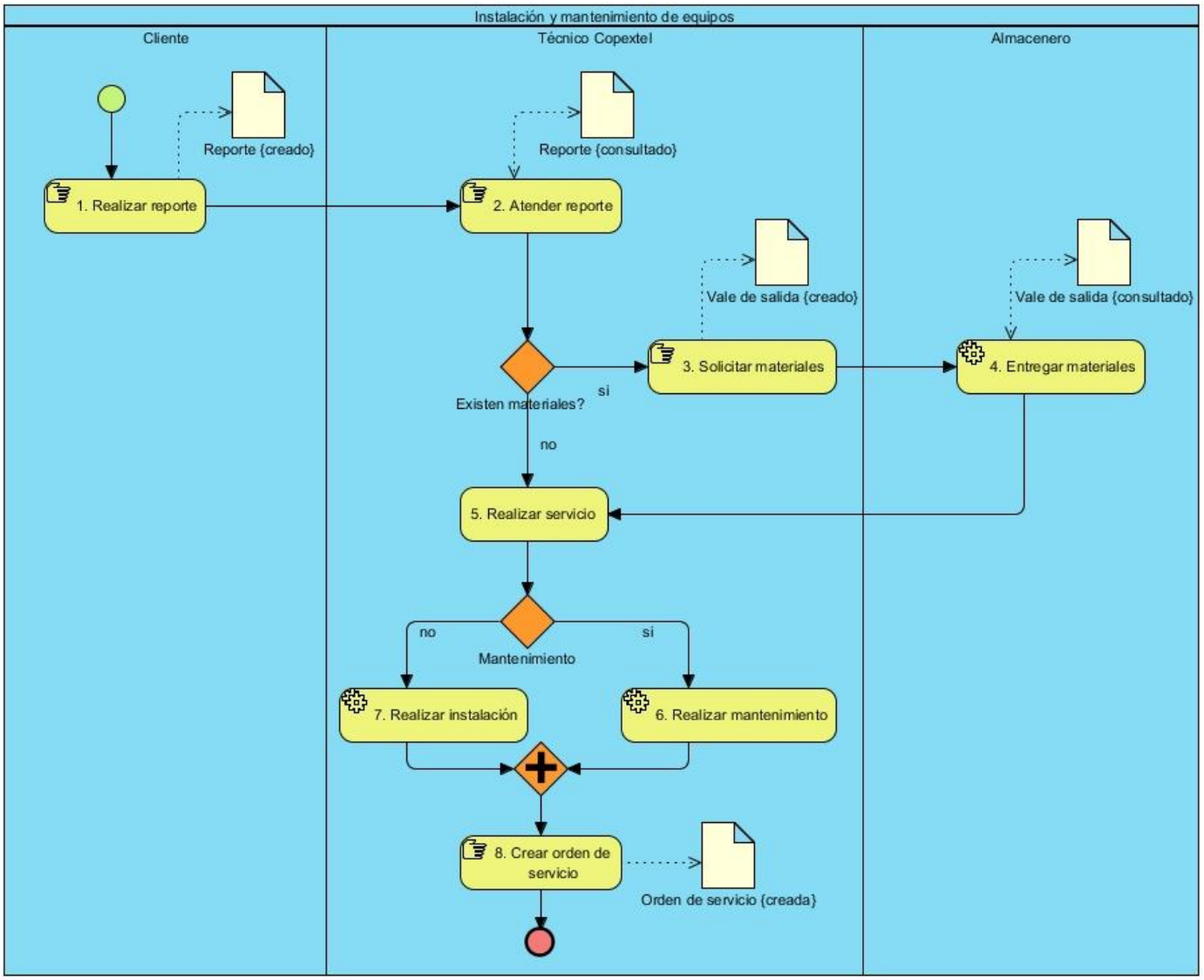


Figura 2: Diagrama de procesos de negocio: Generar orden de servicio

2.4. Modelo conceptual

El modelo conceptual es una representación de conceptos en un dominio del problema. Este modelo muestra asociaciones entre conceptos y atributos de conceptos. Se puede ver cómo un modelo que comunica los términos importantes y cómo se relacionan entre sí. En el siguiente diagrama se representan los conceptos que intervienen en el negocio actual de la presente investigación, así como las relaciones entre ellos.

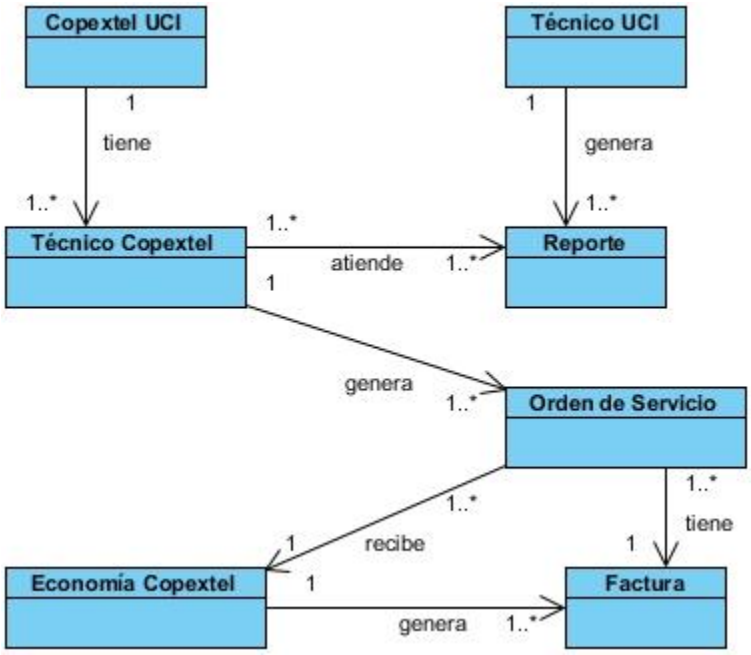


Figura 3: Modelo conceptual

2.4.1. Descripción de los conceptos que intervienen en el dominio del problema

Se muestra a continuación una serie de conceptos para el entendimiento del contexto de la presente investigación.

Copextel UCI: La división de Copextel dentro de la UCI.

Técnico Copextel: Persona encargada de atender los reportes de la universidad llegados a Copextel UCI.

Técnico UCI: Trabajador de la UCI encargado de hacer los reportes de rotura, mantenimiento o instalación.

Reporte: Solicitud hecha por un Técnico de la UCI por rotura, mantenimiento o instalación de equipos.

Orden de Servicio: Documento generado luego de atender un reporte, con los datos del trabajo realizado.

Economía Copextel: Dirección encargada de recibir las órdenes de servicio y realizar las facturas.

2.5. Modelo del sistema

La especificación de requisitos en el proceso de desarrollo del sistema es de vital importancia. Tener los requisitos bien claros y definidos permite comprender desde un inicio la línea a seguir en el desarrollo y así garantizar la eficiencia y calidad del *software*.

2.5.1. Requisitos del Sistema

Los requisitos del sistema son declaraciones que identifican atributos, capacidades, características y/o cualidades que necesita cumplir un sistema (o un sistema de *software*) para que tenga valor y utilidad para el usuario. En otras palabras, los requerimientos muestran qué elementos y funciones son necesarias para un proyecto. (Alegsa, 2009)

- **Requisitos funcionales**

Para poder identificar qué debe hacer el sistema y entender su funcionamiento, es fundamental conocer los requisitos funcionales que el sistema debe cumplir. A continuación, se muestran los requisitos funcionales identificados:

RF 1: Autenticar usuario
RF 2: Gestionar Órdenes de servicio
- **RF 2.1:** Listar órdenes de servicio
- **RF 2.2:** Insertar orden de servicio
- **RF 2.3:** Editar orden de servicio
- **RF 2.4:** Eliminar orden de servicio
- **RF 2.5:** Mostrar orden de servicio
- **RF 2.6:** Cancelar orden de servicio

RF 3: Gestionar usuarios
- **RF 3.1:** Listar usuarios
- **RF 3.2:** Insertar usuario
- **RF 3.3:** Editar usuario
- **RF 3.4:** Eliminar usuario
- **RF 3.5:** Mostrar usuario

RF 4: Gestionar Material
- **RF 4.1:** Listar materiales
- **RF 4.2:** Insertar material
- **RF 4.3:** Editar material
- **RF 4.4:** Eliminar material
- **RF 4.5:** Mostrar material

RF 5: Gestionar Tarifa
- **RF 5.1:** Listar tarifas
- **RF 5.2:** Insertar tarifa
- **RF 5.3:** Editar tarifa
- **RF 5.4:** Eliminar tarifa
- **RF 5.5:** Mostrar tarifa

RF 6: Gestionar Blog de Órdenes
- **RF 6.1:** Listar Blog de órdenes
- **RF 6.2:** Insertar Blog de órdenes
- **RF 6.3:** Editar Blog de órdenes
- **RF 6.4:** Eliminar Blog de órdenes
- **RF 6.5:** Mostrar Blog de órdenes

RF 7: Gestionar Blog de sellos

RF 7.1: Listar Blogs de sellos
RF 7.2: Insertar Blog de sellos
RF 7.3: Editar Blog de sellos
RF 7.4: Eliminar Blog de sellos
RF 7.5: Mostrar Blog de órdenes
RF 8: Mostrar Gráficas estadísticas
RF 8.1: Gráfica Plan Anual General
RF 8.2: Gráfica Plan Mensual General
RF 8.3: Gráfica Órdenes General
RF 8.4: Gráfica Plan Anual Taller
RF 8.5: Gráfica Plan Mensual Taller
RF 8.6: Gráfica Órdenes Taller
RF 9: Exportar reporte de órdenes de servicio en documento Excel
RF 10: Mostrar materiales más usados
RF 11: Mostrar materiales más vendidos

- **Requisitos no funcionales**

Los requisitos no funcionales detallan las propiedades o cualidades que el producto debe tener, aumentándole funcionalidad al sistema, pues hacen al producto atractivo, fácil de usar, rápido y confiable, los cuales se encuentran separados por categorías que ahora se mencionarán.

Usabilidad

- En el sistema podrán trabajar, usuarios con el mínimo de conocimiento en informática; esto se logrará a partir de una correcta estructura de la información, con el empleo de menús, que proporcionan una navegación sencilla, la cual no sobrepasa dos (2) niveles de profundidad.
- En el sistema se debe mostrar información al usuario de las acciones a realizar, esto se logra a partir de mensajes en los iconos que representan dichas acciones.

Eficiencia

- El sistema deberá tener un nivel de respuesta menor a cinco (5) segundos, tanto para los accesos a la base de datos, como para el proceso de administración de tareas.
- El sistema debe permitir la navegación de varios usuarios simultáneamente sin que el rendimiento del sitio se vea afectado drásticamente.

Software

- Se debe emplear un sistema gestor de bases de datos MySQL para el almacenamiento de los datos que manejará el sistema.

Seguridad

- **Confidencialidad:** en el sistema solo podrá gestionar información el personal autorizado, con los permisos correspondientes, por lo que será necesaria una autenticación previa.
- **Integridad:** el *software* deberá estar protegido contra acciones no autorizadas o que puedan afectar la integridad de los datos.
- **Disponibilidad:** el sistema deberá estar disponible las 24 horas del día para todos los usuarios, exceptuándose los días que se encuentre en actualización, corrección de errores, o soporte.

Hardware mínimo para el servidor web

- La computadora que funciona como servidor debe tener al menos 1 GB de memoria RAM DDR3.
- Para el almacenamiento de datos debe contar como mínimo con 80 GB de disco duro.
- Como procesador debe tener un Intel Pentium 4 o de una mayor generación.

Requisitos de soporte

- Para que los usuarios finales sean productivos en el manejo del sistema, se les dará una capacitación donde adquieran las habilidades necesarias para que trabajen eficientemente con el sistema.

2.5.2. Definición de los actores del sistema

Tabla 4: Definición de los actores del sistema.

Actor	Objetivos
Administrador	Usuario con todos los permisos sobre la aplicación. Se encarga de gestionar los usuarios y sus roles, así como modificar la información mostrada en el sistema.
Jefe de taller	Usuario con permiso de autenticarse y navegar por la aplicación. Puede gestionar las órdenes de servicios generadas en su taller, así como los blogs de sellos y de órdenes entregadas a sus técnicos.
Técnico	Usuario con permiso de autenticarse y navegar por la aplicación; así como agregar órdenes de servicio.

2.5.3. Diagrama de Casos de uso del sistema

El diagrama de casos de uso del sistema siguiente ayuda a comprender gráficamente los procesos del sistema y su interacción con los actores. En el cual se utilizaron los patrones de diseño de casos de uso Múltiples Actores y CRUD Completo.

CRUD Completo: El patrón CRUD Completo consiste en un caso de uso para administrar la información, nos permite modelar las diferentes operaciones para administrar una entidad de información, tales como crear, leer, cambiar y eliminar o dar de baja.

Múltiples Actores: A un Caso de Uso ingresan más de dos actores y estos tienen un rol común.

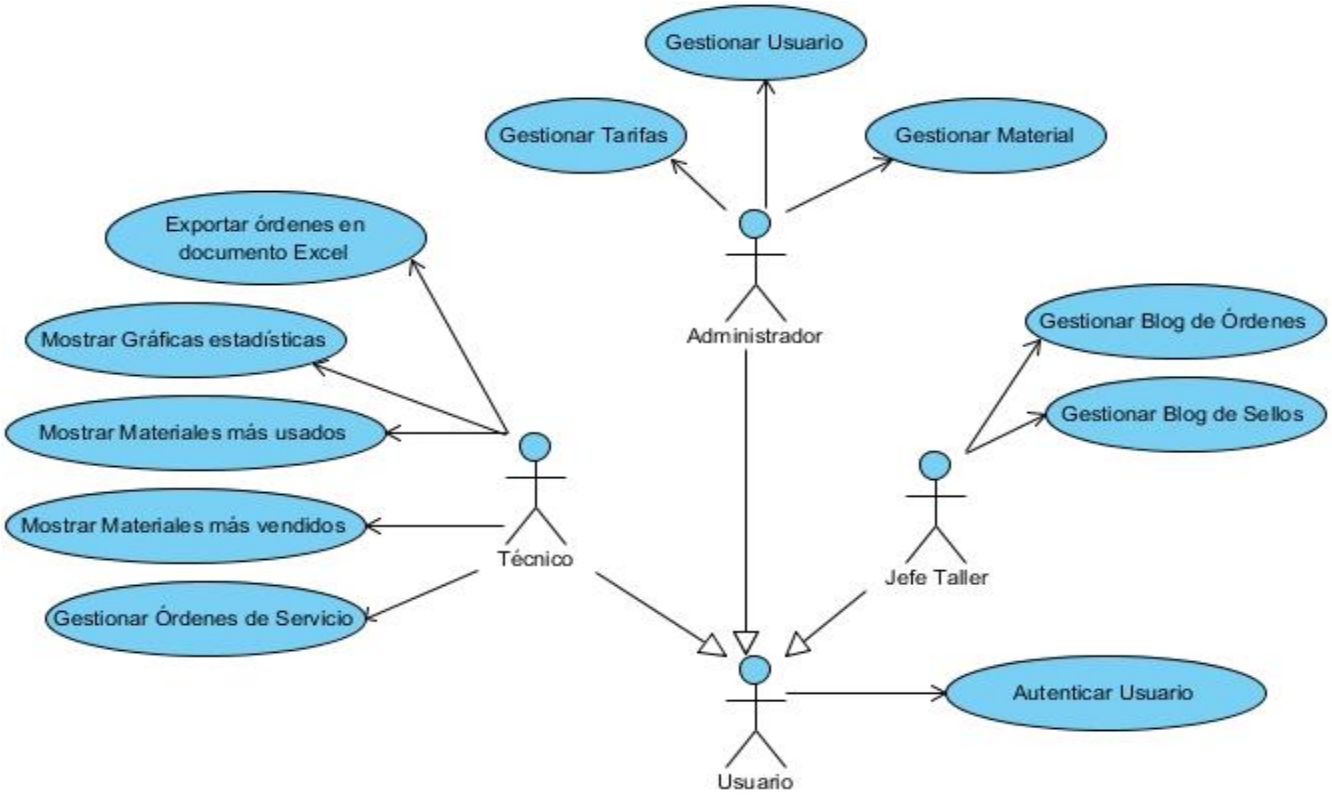


Figura 4: Diagrama de Casos de uso del sistema

2.5.4. Descripción de los Casos de uso del sistema

A continuación, se describe uno de los Casos de Uso del Sistema Gestionar Órdenes de Servicio. El resto de las descripciones se puede consultar en el Anexo 1.

Tabla 5: Descripción del Caso de Uso Gestionar Órdenes de Servicio

Caso de Uso	Gestionar Órdenes de Servicio	
Objetivo	Permite insertar, modificar, eliminar y mostrar las órdenes en el sistema.	
Actores	Técnico (Inicia)	
Resumen	El caso de uso inicia cuando el técnico selecciona la opción “Órdenes” del menú lateral derecho. La aplicación muestra el listado de órdenes almacenadas de su taller y brinda la opción de insertar una nueva orden; editar o eliminar las suyas que ya existan en el sistema.	
Complejidad	Media	
Prioridad	Alta	
Precondiciones	El usuario debe estar previamente autenticado en el sistema.	
Pos condiciones	Se inserta, edita o elimina una orden de servicio.	
Flujo de eventos		
Flujo básico: Gestionar órdenes de servicio		
	Actor	**Sistema**
1	Selecciona la opción “Órdenes” del menú lateral derecho.	
2		Muestra el listado de órdenes de servicios almacenadas del taller del usuario actual. Permite realizar varias acciones: - Insertar orden. Ver Sección 1: Insertar orden. - Editar orden. Ver Sección 2: Editar orden - Eliminar orden. Ver Sección 3: Eliminar orden. - Cancelar orden. Ver Sección 4: Cancelar orden.
3		Termina el caso de uso
Flujos alternos		
N/A		

Sección 1: "Insertar orden"		
Flujo básico: Insertar orden		
	Actor	**Sistema**
1.	Selecciona la opción "Crear orden de Servicio" de la página principal de las Órdenes.	
2.		Muestra una vista con un formulario de registro con los siguientes campos: No orden, vale de salida, sello, persona, No Hércules, estado, observación, dirección, fecha inicio, fecha fin, tarifas, materiales.
3.	Introduce los datos correspondientes y da clic en el botón "Crear".	
4.		Crea una nueva orden y almacena en la base de datos la información de dicha orden. Muestra los datos de la evidencia creada.
5.		Termina el caso de uso
Flujos alternos		
3a. Deja campos vacíos		
	Actor	**Sistema**
1		No se inserta la orden y señala los campos que aún estén vacíos, para que el usuario los llene.
3b. Inserta datos inválidos		
	Actor	**Sistema**
1		No inserta la orden y señala los campos inválidos.
Sección 2: "Editar orden"		
Flujo básico: Editar orden		
	Actor	**Sistema**
1	Selecciona la opción "Editar orden de Servicio" de la	

	página principal de las Órdenes.	
2		Muestra una vista con un formulario con los siguientes campos para editar: No. Orden, vale de salida, sello, persona, No. Hércules, estado, observación, dirección, fecha inicio, fecha fin.
3	Modifica los datos que desee y da clic en el botón “Actualizar”.	
4		Modifica la orden y guarda la información modificada en la base de datos.
5		Termina el caso de uso
Flujos alternos		
1a. Intenta editar una orden de la cual no es su creador insertando en el navegador la dirección *URL* de la orden correspondiente.		
	Actor	**Sistema**
1		Retorna al usuario a la vista principal de las órdenes de servicio y muestra un mensaje de error.
3a. Deja campos vacíos		
	Actor	**Sistema**
1		No se actualiza la orden y señala los campos que aún estén vacíos, para que el usuario los llene.
3b. Inserta datos inválidos		
	Actor	**Sistema**
1		No actualiza la orden y señala los campos inválidos.
Sección 3: “Eliminar orden”		
Flujo básico: Eliminar orden		
	Actor	**Sistema**
1	Selecciona la opción “Eliminar Orden de Servicio” de la	

	página principal de las Órdenes.	
2		Muestra un mensaje para que el usuario confirme la eliminación de la orden seleccionada.
3	Selecciona la opción “Eliminar”.	
4		Elimina la orden seleccionada de la base de datos.
5		Termina el caso de uso.
Sección 4: “Cancelar orden”		
Flujo básico: Cancelar orden		
	Actor	**Sistema**
1	Selecciona la opción “Cancelar Orden de Servicio” de la página principal de las Órdenes.	
2		Elimina los datos de las tarifas y los materiales usados en la orden de servicio y modifica su estado a “cancelada”.
3		Termina el caso de uso.

Conclusiones del capítulo

A través de la modelación del negocio se identificaron los procesos que intervienen actualmente en la entidad y ayudó a comprender el dominio del problema. Mediante el análisis de los requerimientos del sistema se identificaron 41 requisitos funcionales y 12 no funcionales. Con la obtención de los requisitos funcionales se pudo identificar lo que debe hacer el sistema y entender su funcionamiento. Además, los requisitos no funcionales permitieron definir los atributos que debe exhibir el sistema. Mediante la descripción detallada de los casos de usos se pudieron definir los detalles internos sobre qué debe hacer el sistema, como respuesta a las acciones de los actores.

Al concluir el presente capítulo se han creado las condiciones para efectuar el diseño y la implementación del Sistema de Gestión de Información de las Órdenes de Servicio.

CAPÍTULO 3: DISEÑO DEL SISTEMA

Introducción

En el presente capítulo se describen los patrones de diseños y arquitectónicos que se evidencian en el sistema. Además, se desarrollan los diagramas de clases de diseño, que constituyen los artefactos principales generados para esta etapa. Se presenta también el modelo de datos, así como el diagrama de despliegue.

3.1. Modelo de Diseño

El Modelo de Diseño es una disciplina que no se puede obviar en el proceso de desarrollo del software. Es imprescindible para comprender la forma en que va a funcionar el sistema en conjunto con los requisitos, lenguajes de programación, componentes reutilizables y tecnologías de interfaz de usuario que se eligieron para el desarrollo del mismo. Es una representación gráfica, mediante varios diagramas muy explícitos, de la implementación del sistema.

3.1.1. Descripción de los Patrones arquitectónicos y de diseño

Los patrones ayudan al arquitecto a definir la composición y el comportamiento del sistema de *software*, y una combinación adecuada de ellos permite alcanzar los requerimientos de calidad. A continuación, se describen los utilizados en el desarrollo la propuesta de solución.

- **Patrones arquitectónicos**

Los patrones arquitectónicos expresan el esquema de organización estructural fundamental para sistemas de *software*. Provee un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos. Propone que son plantillas para arquitecturas de *software* concretas, que especifican las propiedades estructurales de una y tienen un impacto en la arquitectura de subsistemas. La selección de un patrón arquitectónico es, por lo tanto, una decisión fundamental de diseño en el desarrollo de un sistema de *software*. (Buschmann, 1996)

Los patrones arquitectónicos:

- ✓ Definen la estructura básica de una aplicación.
- ✓ Pueden contener o estar contenidos en otros patrones.

- ✓ Proveen un subconjunto de subsistemas predefinidos, incluyendo reglas y pautas para su organización.
- ✓ Son una plantilla de construcción. (Cardoso, 2004)

Patrón arquitectónico Modelo – Vista – Controlador

El patrón arquitectónico Modelo – Vista – Controlador (MVC) divide una aplicación interactiva en tres componentes. El "modelo" contiene la información central y los datos. Las "vistas" despliegan información al usuario. Los "controladores" capturan la entrada del usuario.

Es el patrón arquitectónico utilizado por Symfony, separando la vista de las aplicaciones de la lógica del negocio y del controlador.

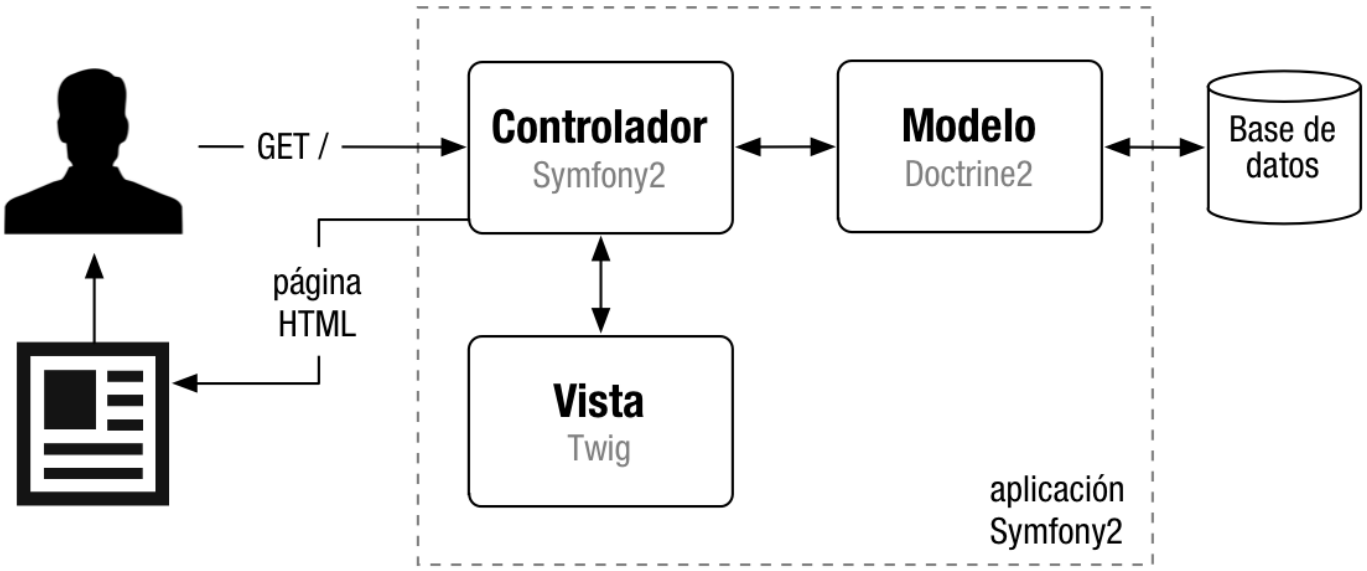


Figura 5: Diagrama del Modelo –Vista – Controlador de Symfony

➢ **Patrones de diseño**

Un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de *software*, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular. (Buschmann, 1996)

Patrones GRASP

Los Patrones de Principios Generales para Asignar Responsabilidades (GRASP por sus siglas en inglés) describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones.

Experto: Las responsabilidades deben ser asignadas a las clases que poseen la información para realizar dicha responsabilidad. El SGOS hace uso de este patrón y se evidencia cuando se desea mostrar todas las órdenes de servicio almacenadas en el sistema, ya que la única clase con la responsabilidad de conocer esta información es Orden. De la misma manera sucede con las demás entidades del SGOS. Solo la clase Usuario conoce quienes son los usuarios registrados en el sistema.

Creador: Asignarle a una clase la responsabilidad de crear una instancia de otra. Dentro del sistema este patrón se evidencia en las acciones de los controladores, las cuales crean objetos del modelo o los formularios que representan las entidades.

Alta Cohesión: Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Significa que las clases del sistema tienen asignadas solo las responsabilidades que les corresponde y mantienen una estrecha relación con el resto de las clases. Un ejemplo de este patrón en el SGOS se evidencia cuando se desea mostrar todos los materiales existentes en almacén. La generación de esta vista es responsabilidad del MaterialController, pero quien tiene los datos a mostrar es la clase Materiales. De esta manera se evidencia la relación que debe existir entre ambas clases ya que la primera solo se encarga de mostrar los materiales del almacén, utilizando para ello, los datos que devuelven los métodos de Materiales.

Bajo Acoplamiento: Determina el nivel de dependencia de una clase con respecto a otras. Una clase con bajo acoplamiento no depende de muchas otras. Este patrón es utilizado por el *framework* Symfony, y por ende en el sistema, al no asociar las clases del modelo con las de la vista o el controlador, la dependencia entre las clases, en este caso, se mantiene baja.

Controlador: Es el encargado de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones. Se evidencia el uso de este patrón en el SGOS, ya que para cada petición o evento que se genere en el mismo, existe un controlador con la responsabilidad de obtenerla y devolver una respuesta. La respuesta puede ser mostrar una vista, ejecutar un método y devolver un mensaje.

- **Patrones GoF**

Los patrones GoF (*Gang of Four* o “Pandilla de los Cuatro” en español), describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación clases y la formación de estructuras de mayor complejidad. Nos permiten crear grupos de objetos para ayudarnos a realizar tareas complejas.

Observador (*Observer*): es un patrón de comportamiento. Se utiliza para mantener “informados” a objetos desacoplados entre sí y que presentan dependencia de un objeto, notificando a los primeros ante cualquier cambio que ocurra en el último. Este patrón es utilizado tradicionalmente en la capa de presentación en el diseño de componentes para las interfaces de usuario de las aplicaciones.

Decorador (*Decorator*): Patrón de tipo estructura, a nivel de objetos. Añade responsabilidades adicionales a un objeto dinámicamente. Se utiliza este patrón para la vista y el *layout* o plantilla global que decora el contenido de la misma.

3.1.2. Diagrama de Clases de Diseño

Los diagramas de clases ayudan a coordinar todos los requerimientos que se imponen a una clase, así como contribuyen a detallar las operaciones, atributos y asociaciones que esta debe aportar. Tienen en cuenta las características del entorno de programación concebidas para satisfacer los requisitos funcionales y no funcionales. Seguidamente se muestra este diagrama. El resto de los diagramas pueden consultarse en el Anexo 2.

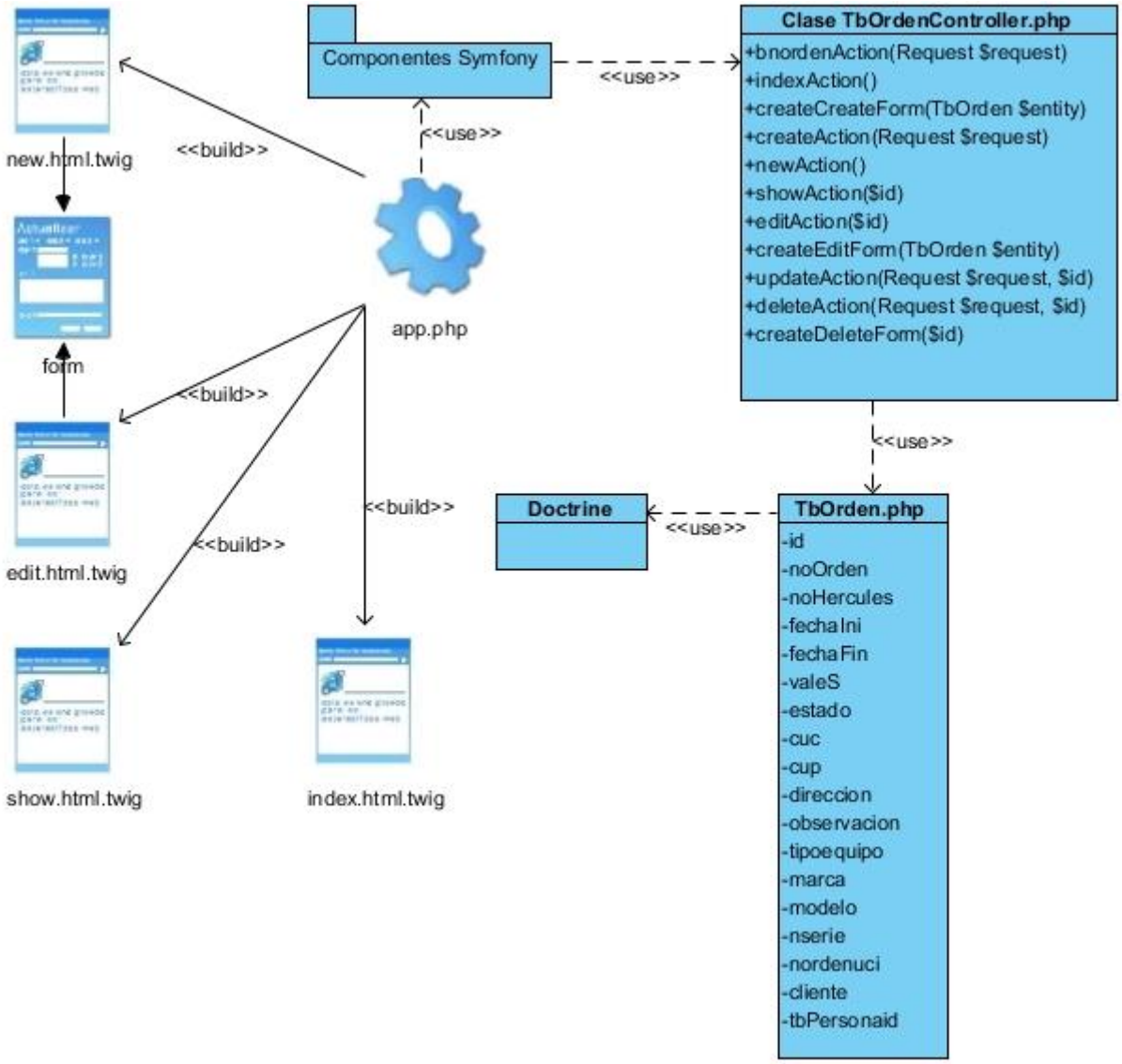


Figura 6: Diagrama de Clase del Diseño del Caso de uso Gestionar Orden

3.2. Modelo de Datos

Un modelo de datos es la descripción de una base de datos. Típicamente un modelo de datos permite describir las estructuras de datos de la base, su tipo, descripción y la forma en que se relacionan, restricciones de integridad entre otros. Es factible pensar que un modelo de datos permite describir los elementos de la realidad que intervienen en un problema dado y la forma en que se relacionan esos elementos entre sí. A continuación, se presenta el modelo de Base de Datos del SGOS.

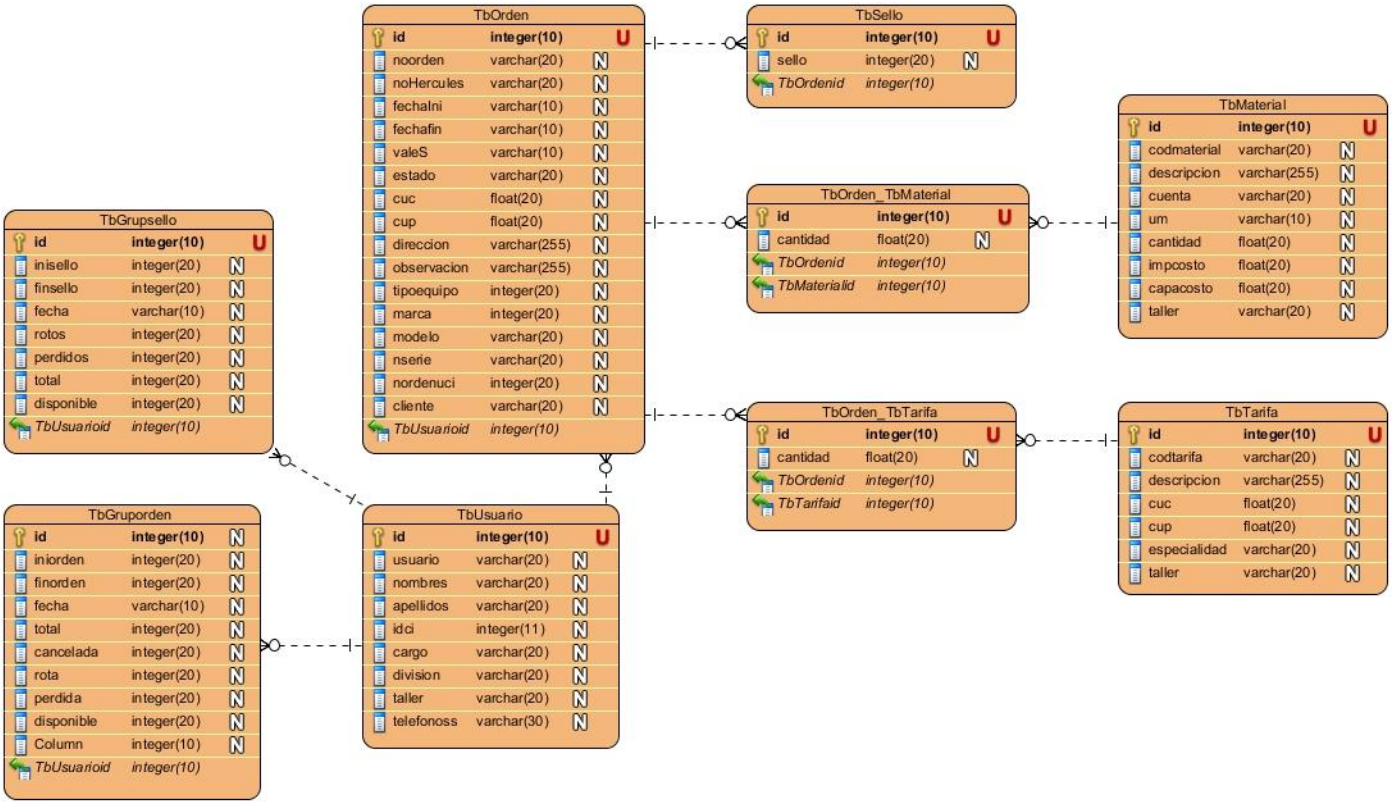


Figura 7: Modelo de Datos del SGOS

3.3. Modelo de Despliegue

El diagrama de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los vínculos de comunicación entre ellos, las instancias de los componentes y objetos que residen en ellos. Está compuesto por nodos, dispositivos y conectores. El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento y las conexiones entre estos elementos en el sistema.

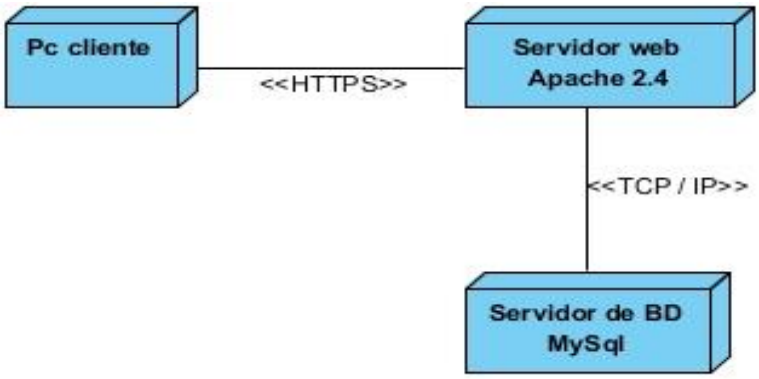


Figura 8: Modelo de despliegue

PC Cliente: Representa las computadoras clientes que se conectan al servidor de aplicaciones, las mismas se comunican con el servidor a través del protocolo seguro HTTP.

Servidor web: Representa el servidor donde se encuentra instalada la aplicación web. Este accede al servidor de Base de Datos para el manejo de la información mediante el protocolo TCP/IP.

Servidor de Base de datos: Es donde se almacena toda la información de la aplicación.

Conclusiones del capítulo

La generación de los artefactos relacionados con el flujo de análisis y diseño, teniendo en cuenta la arquitectura MVC que Symfony establece, permitió obtener una mayor comprensión de la aplicación y definir los principios que guiarán la implementación y organización de la misma. A través del modelado de análisis se obtuvo una visión general conceptual del sistema y mediante el modelado del diseño se obtuvo una abstracción de la implementación del sistema.

A partir de aquí están creadas las condiciones para efectuar la implementación y validación del sistema para la gestión de órdenes de servicio

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS

Introducción

La fase de implementación en el desarrollo de un producto de *software*, es el mecanismo donde se ponen en práctica todas las descripciones y arquitecturas propuestas en las fases de análisis y diseño, es el complemento del trabajo de las fases que lo preceden dentro del proceso de desarrollo de *software*. La implementación ofrece una materialización precisa de los requisitos.

Una de las últimas fases del ciclo de vida antes de entregar un *software* para su explotación es la fase de pruebas, cuyo objetivo es comprobar si este cumple sus requisitos. Dentro de ella pueden desarrollarse varios tipos de pruebas en función de los objetivos de las mismas.

4.1. Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Se realizan con el objetivo de poseer una vista de forma general del sistema a partir de las dependencias e integraciones de los componentes y módulos.
A continuación, se muestra el diagrama de componentes del *framework* Symfony2, donde se integra el sistema propuesto:

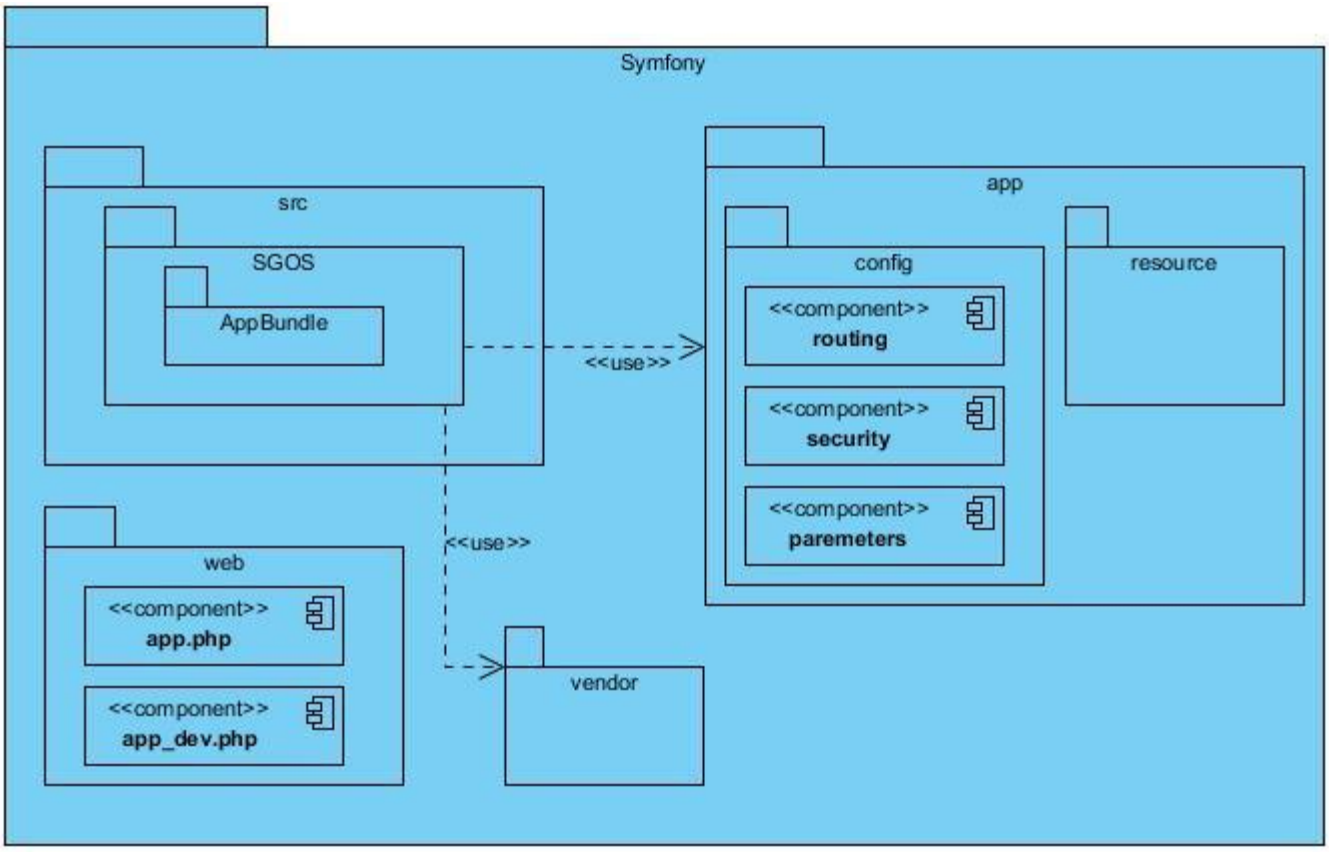


Figura 9: Diagrama de componentes

El sistema desarrollado se encuentra empaquetado en un *bundle*[9], donde se encuentran distribuidos tres paquetes: modelo, vista y controlador, de tal forma que exista correspondencia con el patrón arquitectónico MVC. En el paquete controlador se encuentran las clases controladoras, encargadas de manejar las peticiones de los usuarios a través de los métodos que tienen implementados. El paquete modelo agrupa las entidades del sistema, a través de las cuales se realiza el acceso a la base de datos y el paquete vista, agrupa los archivos que permiten visualizar las respuestas que devuelven los controladores al usuario.

A continuación, se muestra el diagrama de componentes para el AppBundle, donde se encuentran los principales componentes del sistema a desarrollar.

[9] Un **bundle** es un directorio que contiene un conjunto de archivos (archivos PHP, JavaScript, hojas de estilo, imágenes, etc.) que implementan una sola característica (un blog, un foro, etc.). En Symfony (casi) todo está dentro de un bundle.

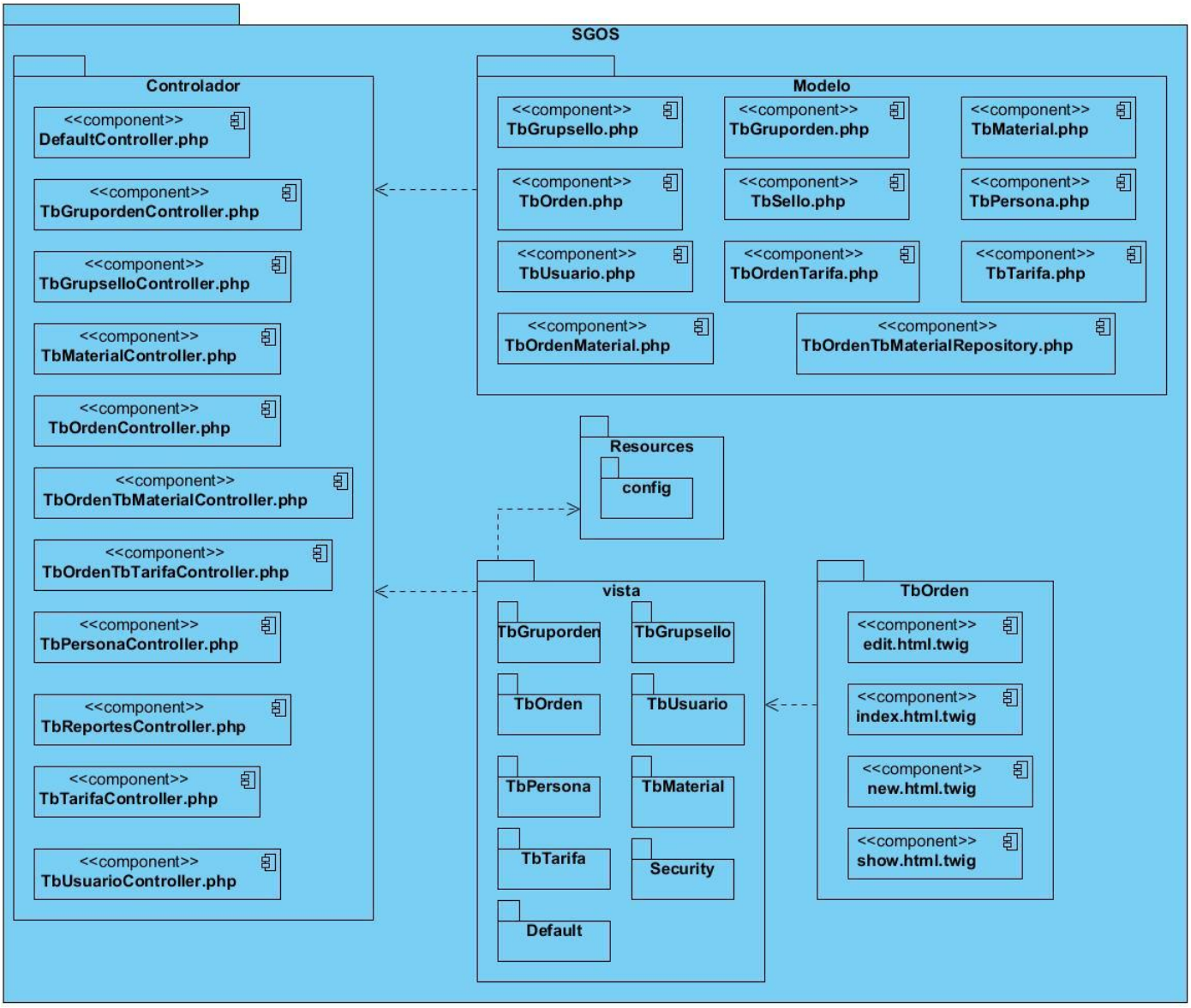


Figura 10: Diagrama de componentes del SGOS

4.2. Código fuente

Para obtener una versión funcional de la aplicación se deben implementar los componentes que se han definido, como resultado se obtienen archivos que contienen el código fuente de la aplicación. El código fuente de un *software* es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. Por tanto, en el código fuente de un programa está escrito su funcionamiento. Estas instrucciones son escritas en un lenguaje de programación que consiste en un

conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

4.2.1. Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez.

Para facilitar el entendimiento del código y fijar un modelo a seguir, se establecieron estándares de codificación. A continuación, se muestran algunos de estos estándares para el lenguaje PHP utilizados en Symfony2.

Estructura

- El código debe usar cuatro espacios para la identación en vez de usar el tabulado. Esto minimiza problemas con otras herramientas de desarrollo.
- Las líneas podrían tener 80 caracteres o menos, evitando tener más de 120 caracteres.
- Las llaves de apertura deben ir en la siguiente línea y la llave de cierre debe ir en la siguiente línea después del cuerpo.
- Las llaves de apertura en las estructuras de control deben ir en la misma línea y las llaves de cierre deben de ir después del cuerpo.
- Los paréntesis en las estructuras de control no deben usar espacios antes o después.
- Añadir un solo espacio después de cada limitador de coma.
- Añadir un solo espacio alrededor de los operadores (==, &&, …).
- Usa llaves para indicar el control de la estructura sin tener en cuenta el número de declaraciones que el grupo pueda contener.
- Definir una clase por fichero.
- Declarar las propiedades de clase antes que los propios métodos de clase.

Nombres de clases y métodos

Para la definición de las clases y métodos en el código de la aplicación fue utilizado el estándar CamelCase. Este es la práctica de escribir frases o palabras compuestas eliminando los espacios y poniendo en mayúscula la primera letra de cada palabra. El nombre viene del parecido de estas

mayúsculas, entre las demás letras, con las jorobas de los camellos. Existen dos tipos de estándares de CamelCase:

- UpperCamelCase, cuando la primera letra de cada una de las palabras es mayúscula.
  Ejemplo: EjemploDeUpperCamelCase.
- lowerCamelCase, igual que la anterior con la excepción de que la primera letra es minúscula.
  Ejemplo: ejemploDeLowerCamelCase. (Diclib**,** 2015)

En la definición del nombre de las clases fue utilizado UpperCamelCase y en la nomenclatura de los métodos lowerCamelCase:

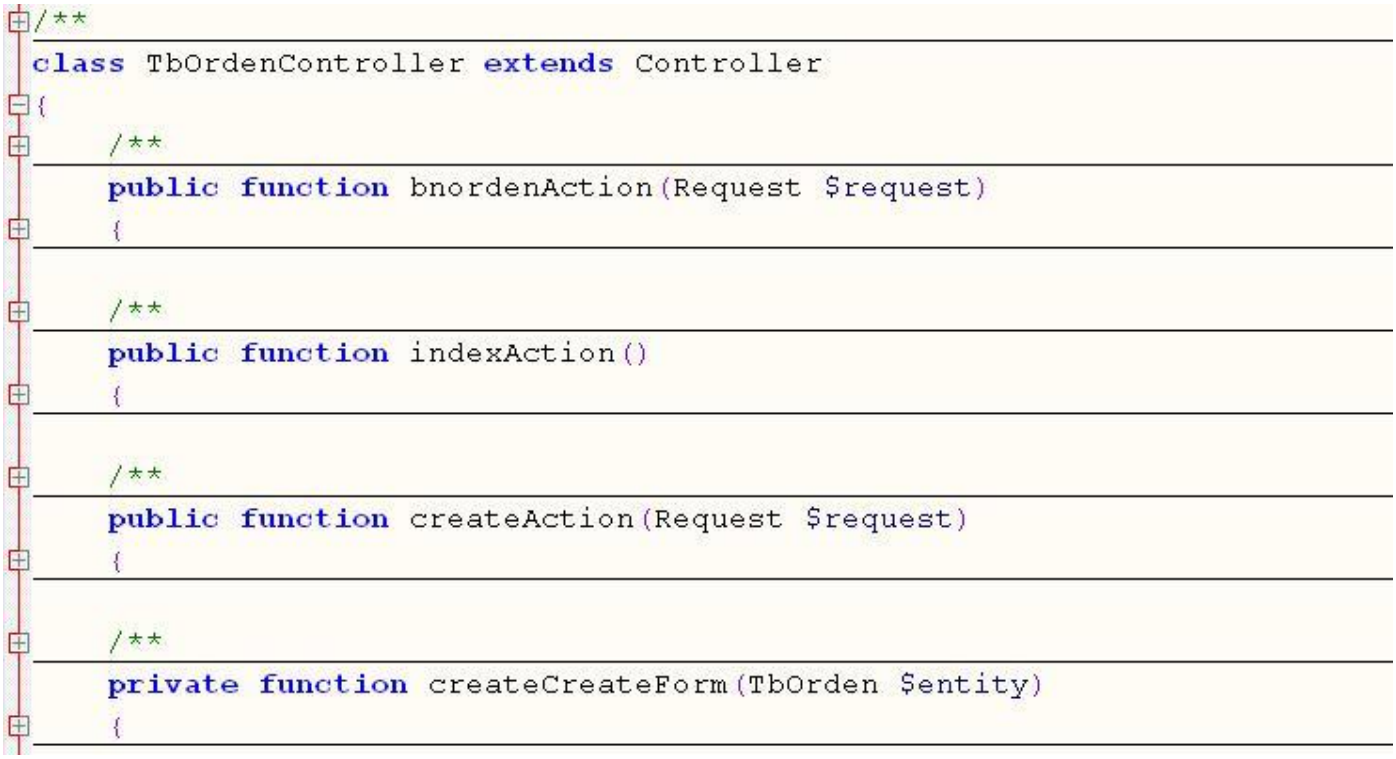


Figura 11: Uso del estándar de codificación CamelCase

4.2.2. Pantallas principales de la aplicación

La interfaz de una aplicación permite el flujo de información entre el usuario y el sistema. A continuación, se muestran algunos ejemplos de las interfaces del sistema desarrollado:

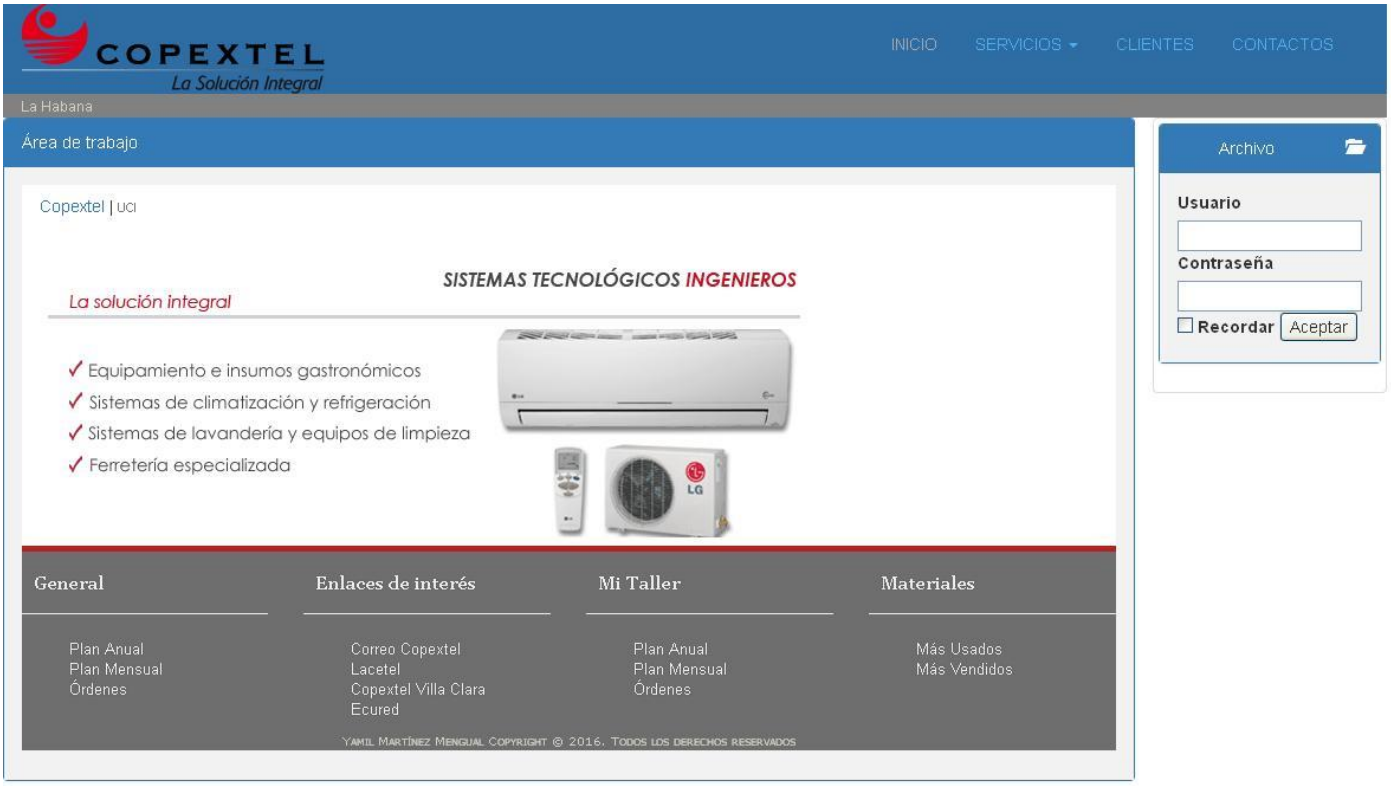


Figura 12: Pantalla de la portada del sistema

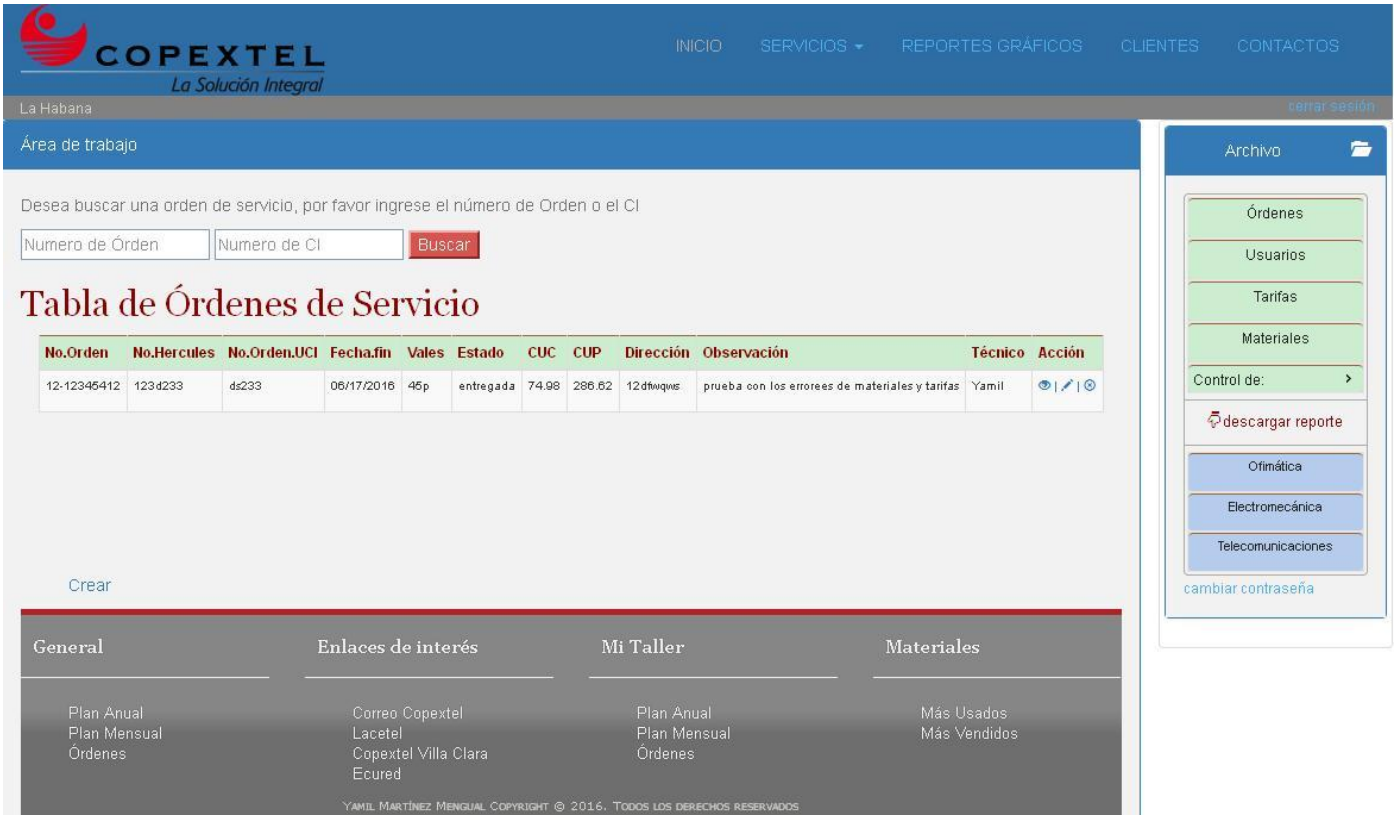


Figura 13: Pantalla de la página Listar Órdenes de Servicio

4.3. Validación del sistema

Una vez terminada la implementación del producto que se requiere es necesario realizarle pruebas con el objetivo de detectar errores en la aplicación y la documentación; este proceso resulta de gran importancia ya que da una medida de la calidad del mismo siempre que se lleve a cabo de la forma correcta. A continuación, se muestran las pruebas realizadas al sistema y los resultados obtenidos por cada una.

4.3.1. Pruebas funcionales o de caja negra

Las pruebas de caja negra, también denominadas pruebas funcionales 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.

En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro.
Las pruebas funcionales que se realizarán a la solución, estarán enfocadas o dirigidas a los casos de uso del sistema para verificar su correcto funcionamiento. En este tipo de pruebas se ejecutarán los distintos servicios prestados con datos correctos e incorrectos. En caso de que los datos sean incorrectos se verificará que los mensajes de error sean los deseados y en el caso opuesto que los resultados sean los esperados. (Pressman, 2002)

4.3.2. Casos de prueba

Las pruebas son una fase de gran importancia en el desarrollo de *software*. Permite verificar la calidad de un producto de *software*. Durante el diseño e implementación de la aplicación se pueden cometer errores, los cuales son descubiertos a través de las pruebas de *software*. La aplicación de pruebas permite identificar dichos errores con el fin de corregirlos y así garantizar la calidad del producto

Los tipos de pruebas aplicadas para la verificación del funcionamiento del sistema son:

- **Prueba funcional:** Son las que se realizan sobre el sistema mientras está funcionando, para comprobar que este cumple con la especificación. La prueba está basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para la aplicación.
- **Prueba de validación:** Son utilizadas para comprobar el valor de los datos introducidos en el sistema por los usuarios. Estas pruebas las realiza el desarrollador.
- **Prueba de caja negra:** Conociendo una funcionalidad específica para la que fue diseñado el producto, es posible realizar pruebas que demuestren que dicha funcionalidad está correcta. Las pruebas de caja negra fueron realizadas sobre la interfaz del sistema proporcionando valores de entradas y analizando los valores que se obtuvieron como salida para comprobar si eran los esperados. Esto se hizo sin tener en cuenta el funcionamiento interno de la aplicación.
- **Prueba de aceptación:** Este tipo de pruebas son realizadas por el usuario para evaluar la calidad del sistema. Las pruebas realizadas fueron específicamente pruebas alfa, ya que fueron hechas en presencia del personal del desarrollo del proyecto

A continuación, se presentan los casos de prueba correspondientes al caso de uso “Gestionar Órdenes”.

El resto de los casos de prueba pueden ser consultados en el Anexo 3.

- **Caso de prueba para el Caso de Uso "Gestionar Órdenes"**

Tabla 6: Caso de prueba para el Caso de Uso Gestionar Órdenes: Sección "Insertar Orden"

Sección: Insertar Orden									
Escenario	**Descripción**	**No. Orden**	**Tarifa**	**Materiales**	**Técnico**	**Fecha**	**Dirección**	**Respuesta del Sistema**	**Flujo central**
EC 1.1 Insertar datos de la Orden Correctamente	Se adiciona la orden	V	V	V	V	V	V	Se inserta una nueva orden en el sistema	1. Selecciona la opción "Crear" de la página principal de las Órdenes 2. Se llenan y seleccionan los campos 3. Se selecciona la opción "Aceptar"
EC 1.2 insertar campos inválidos	Intenta insertar tarifa o materiales inexistentes	V	F	F	V	V	V	Muestra un mensaje de error, mostrando si la tarifa o el material es inexistente	1. Selecciona la opción "Crear" de la página principal de las Órdenes 2. Se llenan y seleccionan los campos 3. Se selecciona la opción "Aceptar"

									4. El sistema muestra mensajes de error
EC 1.3 Dejar campos vacíos	Se intenta insertar una orden dejando campos vacíos	V	I	I	I	V	I	Muestra un mensaje de error, mostrando el campo vació	1. Selecciona la opción “Crear” de la página principal de las Órdenes 2. Se llenan y seleccionan los campos 3. Se selecciona la opción “Aceptar” 4. El sistema muestra mensajes de error

Tabla 7: Caso de prueba para el Caso de Uso Gestionar Órdenes: Sección “Editar Orden”

Sección: Editar Orden									
Escenario	**Descripción**	**No. Orden**	**Tarifa**	**Materiales**	**Técnico**	**Fecha**	**Dirección**	**Respuesta del Sistema**	**Flujo central**
EC 2.1 Insertar datos de la Orden Correctamente	Se adiciona la orden	V	V	V	V	V	V	Se guarda la nueva orden en el sistema	1. Selecciona la opción “Editar” de la página principal de las Órdenes 2. Se llenan y seleccionan los campos

									3. Se selecciona la opción “Modificar”
EC 2.2 Editar una orden sin ser su creador	El usuario intenta editar una orden sin haber sido el su creador	V	V	V	V	V	V	Muestra un mensaje de error: no tiene permiso para editar esta orden	1. Selecciona la opción “Editar” de la página principal de las Órdenes 2. Se muestra el mensaje de error
EC 2.3 Dejar campos vacíos	Se intenta editar una orden dejando campos vacíos	V	I	I	I	V	I	Muestra un mensaje de error, mostrando el campo vació	1. Selecciona la opción “Editar” de la página principal de las Órdenes 2. Se llenan y seleccionan los campos 3. Se selecciona la opción “Modificar” 4. El sistema muestra mensajes de error

Tabla 8: Caso de prueba para el Caso de Uso Gestionar Órdenes: Sección “Mostrar Orden”

Sección: Mostrar Orden			
Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo central**
EC 3.1	Se selecciona la orden	Muestra los datos de la orden	1. Selecciona la opción “Mostrar” de la página

Mostrar orden	para mostrar sus datos		principal de las Órdenes.
EC 3.2 mostrar una orden que no existe	Se intenta mostrar una orden a través de la URL que no existe en el sistema.	Muestra un mensaje de error, la orden no existe.	1. Se accede mediante la URL a la orden 2. Se redirige hacia una página de error. 3. Se muestra el mensaje de error.

Tabla 9: Caso de prueba para el Caso de Uso Gestionar Órdenes: Sección "Eliminar Orden"

Sección: Eliminar Orden			
Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo central**
EC 4.1 Eliminar correctamente una orden	Se elimina una orden del sistema	Elimina la orden de la base de datos.	1. Selecciona la opción "Mostrar" de la página principal de las Órdenes. 2. Se selecciona la opción eliminar.
EC 4.2 eliminar una orden sin ser su creador	El usuario intenta editar una orden sin haber sido el su creador	Muestra un mensaje de error: no tiene permiso para eliminar esta orden	1. Se accede mediante la URL a la orden 2. Se redirige hacia una página de error. 3. Se muestra el mensaje de error.

Tabla 10: Caso de prueba para el Caso de Uso Gestionar Órdenes: Sección "Cancelar Orden"

Sección: Cancelar Orden			
Escenario	**Descripción**	**Respuesta del Sistema**	**Flujo central**
EC 4.1 Cancelar correctamente	Se cancela una orden del sistema	Elimina los datos de las tarifas y los materiales usados en la orden de servicio y modifica su	1. Se redirige hacia una página donde puede modificar la fecha en que se cancela la orden.

una orden		estado a “cancelada”.	

A partir del diseño y ejecución de los casos de prueba, introduciendo juegos de datos, tanto correctos como incorrectos, se detectaron errores tales como: funcionalidades ausentes, errores de idioma, validación de los datos de entradas y funcionalidades incorrectas, para un total de 29 no conformidades.

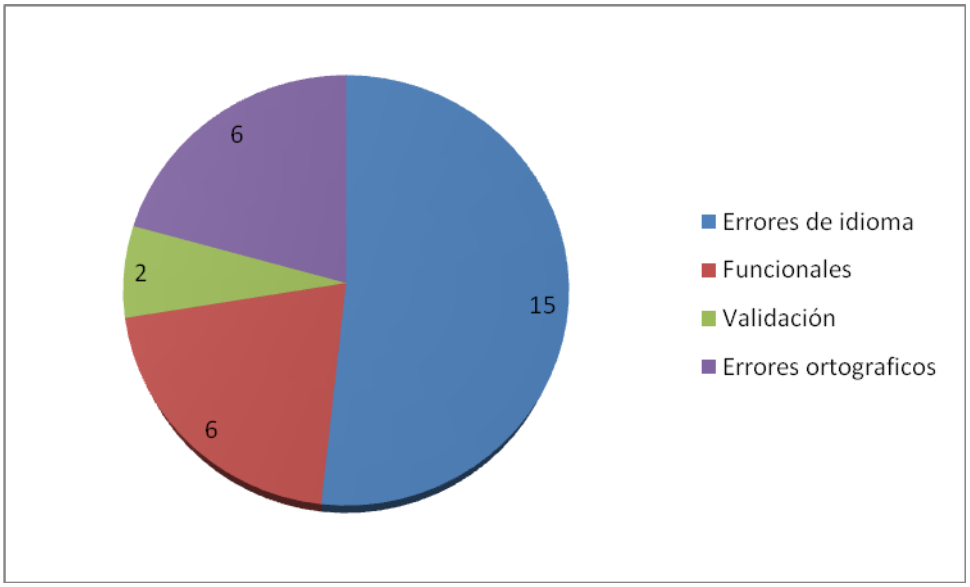


Figura 14: Gráfica de no conformidades de las pruebas funcionales.

La gráfica anterior representa los resultados alcanzados en las pruebas funcionales. Solamente se detectaron 2 errores en las validaciones de datos, 6 de las funcionalidades incorrectas, otros 6 de los errores ortográficos y 15 de los errores del idioma. Al realizar un análisis de los resultados, se concluye que la mayor cantidad de no conformidades se refiere a los errores del idioma.

Las no conformidades de la primera iteración fueron corregidas, dándole paso a una segunda iteración de pruebas, en la cual se evaluaron nuevamente las funcionalidades, obteniendo resultados satisfactorios.

4.3.3. Pruebas de rendimiento

Son realizadas para determinar qué tan rápido un sistema realiza una tarea. También son utilizadas para validar y verificar diferentes aspectos de la calidad de *software*, como por ejemplo el buen uso de los recursos. Para su diagnóstico, se utilizan herramientas que permiten monitorear y obtener información sobre el rendimiento del sistema según la ejecución de determinadas funcionalidades.

La realización de estas pruebas está enfocada en determinar la capacidad del sistema de recibir múltiples peticiones sin que se vea afectado, así como, la velocidad de respuesta del mismo. Se decidió utiliza JMeter, que es una herramienta de *software* libre, la cual permite medir la capacidad de carga de una

aplicación. Además, con ella es posible conocer los tiempos de respuesta de un sistema dado un número de usuarios determinados y un número real de transacciones procesadas por unidad de tiempo.

Es preciso realizar las pruebas de carga y estrés pues resulta necesario comprobar el rendimiento del sistema soportando una cantidad máxima de usuarios interactuando con él y su comportamiento al aumentar esta carga con los mismos recursos disponibles.

El entorno de trabajo en el que fueron realizadas las pruebas de rendimiento cumple con las siguientes características: 1 PC cliente con un procesador Intel Pentium 4 y 1 GB de memoria RAM DDR3, con el Sistema Operativo Windows 7.

En el diseño del plan de pruebas de rendimiento para el sistema se tuvo en cuenta las acciones que el usuario puede realizar al conectarse a la aplicación. En una muestra de 200 usuarios conectados concurrentemente, con un período de subida de 1 segundo (tiempo de espera de cada usuario para realizar una petición) la herramienta JMeter generó los siguientes reportes.

Summary Report

Nombre: Summary Report

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Display Only: Escribir en Log Sólo Errores | Successes | Configurar

Label	# Muestras	Media	Mín	Máx	Std. Dev.	% Error	Rendimiento	Kb/sec	Avg. Bytes
/sgost/web/app...	200	1466	1031	2583	242,07	0,00%	11,0/min	0,85	4721,0
/sgost/web/app...	200	1431	984	2623	263,25	0,00%	11,0/min	0,85	4721,0
/sgost/web/app...	200	1421	984	2890	271,94	0,00%	11,0/min	0,85	4721,0
/sgost/web/app...	200	1442	969	2438	243,10	0,00%	11,0/min	0,85	4721,0
/sgost/web/app...	200	1448	938	3937	298,74	0,00%	11,0/min	0,85	4721,0
/sgost/web/app...	200	1547	1000	19284	1279,62	0,00%	11,0/min	0,85	4721,0
/sgost/web/app...	200	1439	895	2281	231,22	0,00%	11,0/min	0,85	4721,0
/sgost/web/app...	200	1479	611	8188	539,40	0,00%	11,0/min	0,85	4721,0
/sgost/web/app...	200	1483	594	5531	411,63	0,00%	11,0/min	0,85	4721,0
/sgost/web/app...	200	1457	589	2650	296,49	0,00%	11,0/min	0,85	4721,0
/sgost/web/app...	200	1445	579	2219	252,84	0,00%	11,0/min	0,85	4721,0
/sgost/web/app...	200	1396	588	2016	230,40	0,00%	11,0/min	0,85	4721,0
/sgost/web/app...	200	1393	500	5251	424,95	0,00%	11,0/min	0,85	4721,0
/sgost/web/app...	200	1109	500	2705	376,87	0,00%	11,0/min	0,85	4721,0
TOTAL	2800	1425	500	19284	474,80	0,00%	2,5/sec	11,70	4721,0

Figura 15: Reporte generado por el JMeter como resultado de las pruebas al sistema.

Como se puede observar en el análisis del resumen arrojado por la herramienta JMeter para un total de 200 muestras que se le realizaron al sistema se alcanzó un rendimiento de 2.5 peticiones por segundo, con un porcentaje de 0% de errores para cada petición realizada.

La gráfica siguiente muestra el tiempo máximo tomado por una petición para cada una de los recursos del sistema.

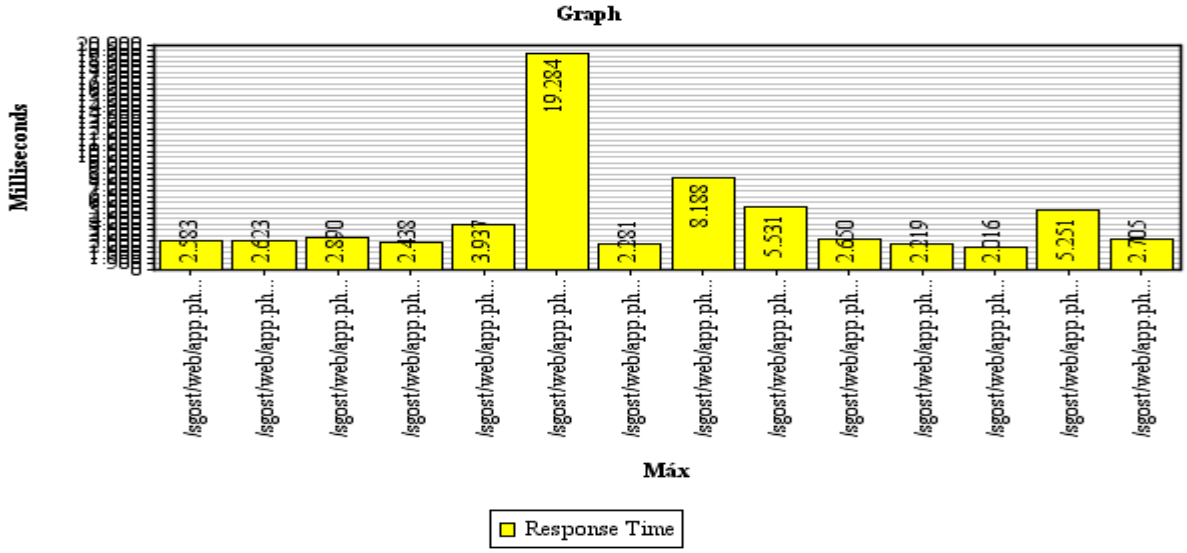


Figura 16: Tiempos máximos de respuesta.

A continuación, se muestra el tiempo mínimo tomado por una petición para cada una de los recursos del sistema.

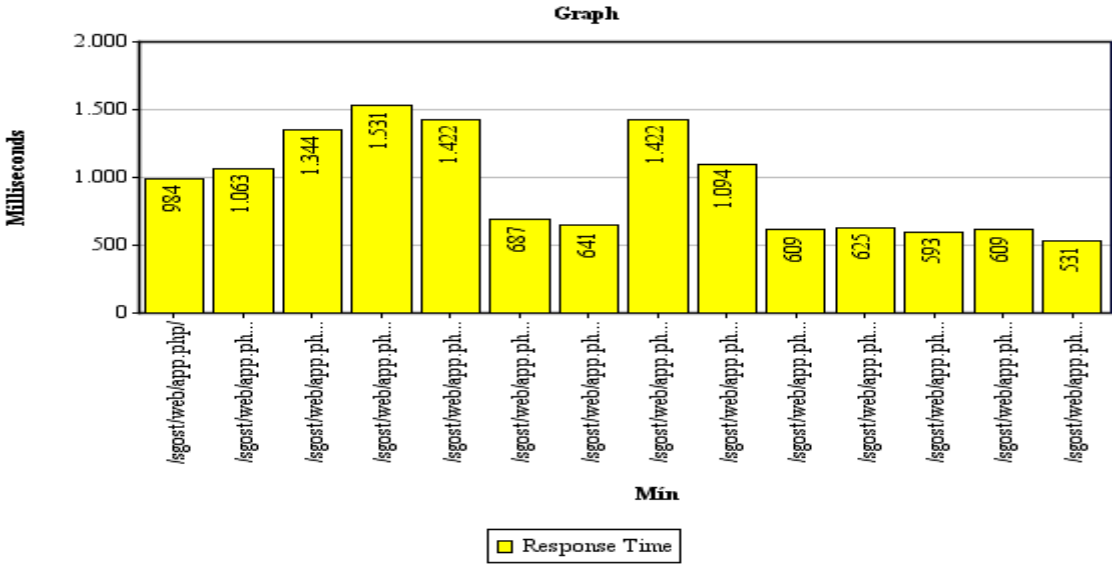


Figura 17: Tiempos mínimos de respuesta.

A partir de los resultados arrojados por la herramienta JMeter, se considera que el sistema responde correctamente ante situaciones de carga y estrés, debido a que las pruebas se realizaron con una cantidad de 200 usuarios, mayor a la cantidad de trabajadores de la división Copextel STI UCI. Aun así, los resultados fueron satisfactorios.

Conclusiones del capítulo

Tras el flujo de implementación y pruebas el sistema quedó desarrollado. En los diagramas generados pudo ilustrarse la relación entre los principales componentes del sistema y cómo estará distribuido este. Las pruebas realizadas permitieron identificar algunos defectos los que fueron corregidos y permitieron aumentar la calidad final de la solución.

CONCLUSIONES

Con la realización de un sistema para la gestión de información de las órdenes de servicio en Copextel STI UCI se da por cumplido el objetivo general planteado. De esta manera se puede arribar a las siguientes conclusiones:

1. El estudio de los sistemas homólogos permitió identificar un conjunto de funcionalidades que se incluyeron en la nueva solución.
2. La fundamentación teórica realizada en la investigación posibilitó justificar la selección de la metodología de desarrollo, las herramientas y tecnología a utilizar para el desarrollo del sistema.
3. La aplicación de pruebas permitió validar el correcto funcionamiento del sistema desarrollado simulando ambientes reales.
4. La implementación del sistema significó un aporte importante para Copextel STI UCI, lo cual ayuda en la toma de decisiones y a trazar estrategias, esto permite mantener el control sobre el cumplimiento del plan de trabajo asignado.

RECOMENDACIONES

Presentar esta investigación, así como el sistema desarrollado a los máximos directivos de Copextel SA, para su utilización en todas las divisiones de la empresa en el país.

REFERENCIAS BIBLIOGRÁFICAS

ALEGSA. *Definición de PHP*, [Citado el: 20 de diciembre del 2015] [En línea] [Disponible en: http://www.alegsa.com.ar/Dic/*framework*.php], 2010.

ALEGSA. *Definición de Framework*, [Citado el: 20 de diciembre del 2015] [En línea] [Disponible en: http://www.alegsa.com.ar/Dic/*framework*.php], 2012.

ALEGSA. *Definición de JQuery*, [Citado el: 20 de diciembre del 2015] [En línea] [Disponible en: http://www.alegsa.com.ar/Dic/*framework*.php], 2015.

ALEGSA. *Definición de Requisitos del Sistema*, [Citado el: 20 de diciembre del 2015] [En línea] [Disponible en: http://www.alegsa.com.ar/Dic/requerimientos.php], 2009.

ALEGSA. *Definición de JavaScript*, [Citado el: 20 de diciembre del 2015] [En línea] [Disponible en: http://www.alegsa.com.ar/Dic/javascript.php], 2010.

ALEGSA. *Definición de IDE*, [Citado el: 20 de diciembre del 2015] [En línea] [Disponible en: http://www.alegsa.com.ar/Dic/ide.php], 2010.

ALEGSA. *Definición Sistema Gestor de Base de Datos*. [Citado el: 29 de mayo del 2016] [En línea] [Disponible en: http://www.alegsa.com.ar/Dic/sgbd.php], 2016.

AYUDAALMA. *Orden de Servicio* [Citado el: 29 de mayo del 2016] [En línea] [Disponible en: http://drrsystemas.no-ip.info/helpalma/Ordenes_de_Servicio.html] 2012

BARTOMEU VIVES SANSÓ. *Definición de Oracle.* [Citado el: 29 de mayo del 2016] [En línea] [Disponible en: http://dmi.uib.es/~labsoft/Labsg/4003_Labsg_Tema2-1.pdf], 2007.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P, y Stal, M*, Pattern – Oriented Software Architecture. A System of Patterns. Inglaterra: John Wiley & Sons, 1996*

Cardoso, E, Camacho, F, y Nuñez, G*, ARQUITECTURAS DE SOFTWARE. GUÍA DE ESTUDIO. 2004*

CASTELL, HEBER SIMEI DELGADO. *Definición de MYSQL*, [Citado el: 20 de diciembre del 2015] [En línea] [Disponible en: https://prezi.com/923yydsinkww/concepto-caracteristicas-ventajas-y-desventajas-de-mysql-y-workbench/], 2014.

CIBERAULA. *Apache*, [Citado el: 26 de noviembre del 2015] [En línea] [Disponible en: http://linux.ciberaula.com/articulo/linux_apache_intro], 2014.

COPEXTEL. [Citado el: 6 de diciembre del 2015] [En línea] [Disponible en:

http://www.opciones.cu/turismo/2008-12-12/copextel-se-prepara-para-informatica-2009/], 2008.

DICLIB*, CamelCase*, [Citado el: 26 de noviembre del 2015] [En línea] [Disponible en: http://www.diclib.com/CamelCase/show/es/es_wiki_10/30673], 2015.

EDITORES DE CODIGO. *Definición de PhpStorm*, [Citado el: 20 de diciembre del 2015] [En línea] [Disponible en: http://www.editoresdecodigo.com/2014/06/descargar-phpstorm-full-ide-para-php-y-mas.html], 2014.

EGUILUZ, J. *Desarrollo Web Ágil con Symfony2.* [Citado el: 15 de enero del 2016] [En línea] [Disponible en: http://n3wton.net:2020/get/pdf/1092], 2012

GONZÁLEZ, Z. *Introducción a UML*. [Citado el: 11 de diciembre del 2015] [En línea] [Disponible en: http://www.slideshare.net/zamanthag/introduccin-uml], 2009.

HERNÁNDEZ ROJAS ALEJANDRO. *Herramienta case* [Citado el: 29 de mayo del 2016] [En línea] [Disponible en: http://it.10-multa.com/ekonomika/490/index.html] 2012

IMPELMENTACIÓN SIG. *Definición de Sistema de gestión*, [Citado el: 26 de noviembre del 2015] [En línea] [Disponible en: http://www.implementacionsig.com/index.php/23-noticiac/28-que-es-un-sistema-de-gestion], 2015.

KRAMA GOT. *Gestión de trabajo en movilidad.* [Citado el: 5 de diciembre del 2015] [En linea] [Disponible en: http://www.kramagot.com/es/], 2013.

MARTÍNEZ, RAFAEL. *Definición de PostgreSql*, [Citado el: 20 de diciembre del 2015] [En línea] [Disponible en: http://www.postgresql.org.es/sobre_postgresql], 2013.

MOYEX. *Sistema de Gestión de órdenes de trabajo*, [Citado el: 26 de noviembre del 2015] [En línea] [Disponible en: http://sistemaspaez.com/ordenes-de-trabajo], 2011.

MUSCIANO, CH Y KEMEDY, B. *HTML la guía completa.* 2da edición. México: McGRAW-HILL INTERAMERICANA EDITORES, 1999.

PONJUÁN DANTE, GLORIA. *La Era de la Información. Gestión de información en las organizaciones: Principios, conceptos y aplicaciones.* Empresa Gráfica Haydee Santamaría: Palma Soriano, SA.

PRESSMAN, ROGER S. *Ingeniería de Software, un enfoque práctico*. Quinta edición. S.l: McGraw-Hill Companies, 2002. ISBN: 8448132149.

RODRÍGUEZ SÁNCHEZ, TAMARA. *Metodología de desarrollo para la Actividad productiva de la UCI*. La Lisa, Habana: Universidad de las Ciencias Informáticas, 2015. 1.2.

Referencias bibliográficas

SAMESPINOSA. *Mapa de procesos. Business & Mgmt. S.l*. [Citado el: 29 de mayo del 2016] [En línea] [Disponible en: http://www.slideshare.net/samespinosa/mapa-de-procesos] 2009

SITEAL/TIC. *TIC* [Citado el: 29 de mayo del 2016] [En línea] [Disponible en: http://www.tic.siteal.org/politicas/1102/programa-rector-de-la-informatizacion-de-la-sociedad-cubana] 2014

White, Stephen A.; MIERS, Phd-Derek. *Guía de Referencia y Modelado BPMN. Comprendiendo y utilizando BPMN. Future Strategies Inc*., USA, 2009

BIBLIOGRAFÍA

Balliauw, Maarten. *PHPExcel Developer Documentation versión 1.7.9,* 2013.

Hogan, Brian P. *HTML5 and CSS3 Develop with Tomorrow's Standards Today,* 2011. ISBN-10:1-934356-68-9

CANÓS, José H.; LETELIER, Patricio; PENADÉS, Mª Carmen. *Metodologías ágiles en el desarrollo de software.* Universidad Politécnica de Valencia, Valencia, 2003. [Disponible en: http://ima.udg.edu/Docencia/07-08/3105200728/TodoAgil.pdf]

De la Caridad, Isyed Rodríguez Trujillo, Ceballos, Yasmani Izquierdo y Terry, Yasirys González. *JavaScript y la programación orientada a objetos,* 2011

Eguiluz, Javier. *Desarrollo web ágil con Symfony2*, 2013. [Disponible en: http://symfony.com]

Sensio Labs. *Symfony2 The Reference Book Versión: 2.7*, 2015

Ghezzi, Carlo, Jazayeri, Mehdi y Wiley, John. 1997. *Programming language concepts. Third edition*. 1997.

Laman, Craig. 1999. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* [trad.] Luz Maria Henhndez Rodriguez y Humberto Cárdenas Anaya. México: PRENTICE HALL, 1999.

Pressman, Roger. 2005. *Ingeniería de Software: Un Enfoque Práctico. 6ta. Edición. s.l.*: McGraw-Hill, 2005

PALACIO, Juan. *Flexibilidad con Scrum. Principios de diseño e implantación de campos de Scrum. SafeCreative. Edición Octubre*, 2007. [Disponible en: http://www.navegapolis.net/files/Flexibilidad_con_Scrum.pdf]

Sommerville, Ian. 2005. *Ingeniería de Software. [ed.] Miguel Martín Romo. 7ma. Edición.* Madrid: PEARSON EDUCASION, S.A., 2005

Romer, Michael. *PHP Data Persistence with Doctrine 2 ORM Concepts, Techniques and Practical Solutions,* 2013

GLOSARIO DE TÉRMINOS

URL: Acrónimo de *Universal Resource Locator* (localizador universal de recursos), método de identificación de documentos o lugares en Internet, que se utiliza principalmente en *World Wide Web* (WWW).

CSS: Siglas en inglés de *Cascading Style Sheets* (Hojas de Estilo en Cascada).

XML: Es el acrónimo de eXtensible *Markup Language* (lenguaje de marcado ampliable o extensible) desarrollado por el *World Wide Web Consortium* (W3C).

UML: Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modelling Language*) es el lenguaje de modelado de sistemas de software más conocido en la actualidad.

IDE: del inglés *Integrated Development Environment*, en español Entorno de Desarrollo Integrado, es un programa que ofrece una serie de herramientas que facilitan el trabajo de los desarrolladores de software para programar sus programas.

Herramienta CASE: es un *software* que permite a los desarrolladores modelar parte o todos los componentes de una aplicación. Generalmente a través de diagramas. Las siglas CASE provienen de *Computer Aided Software Engineering*, que en castellano significan Ingeniería de *Software* asistida por computadora.

Framework: es un conjunto de bibliotecas o clases que pueden ser empleadas por los programadores para reutilizar su código en sus programas.

SQL: (*Structured Query Language*): Es un lenguaje declarativo de acceso a bases de datos que permite especificar diversos tipos de operaciones sobre las mismas. Las características del álgebra y del cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos.

TCP/IP: Es un conjunto de protocolo de red en la que se basa internet y que permiten la transmisión de datos entre redes de computadoras.

XP: (*Extreme Programming*): Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de *software*.

SGBD: (Sistemas Gestores de Bases de Datos): Son un tipo de *software* que sirven de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Glosario de términos

HTML: (*HyperText Markup Language*): Lenguaje de Marcado de Hipertexto. Lenguaje en el que se escriben las páginas a las que se accede a través de navegadores WWW. Admite componentes de hipertexto y multimedia.

MVC: (Modelo Vista Controlador): Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

PHP: Lenguaje procesador de Scripts servidor. Libre y de código abierto.

Bundle: Estructura de directorios generado en los marcos de trabajo.

Doctrine: Es un Mapeador de Objeto Relacional (ORM) creado para PHP.

TIC: Tecnología de la Información y las Comunicaciones.

IP: (Protocolo de internet), estándar que se emplea para el envío y recepción de información mediante una red