Universidad de las Ciencias Informáticas

Título: Sistema de gestión para los reportes de roturas de equipamiento tecnológico en la División de Servicios Técnicos Integrales UCI, Copextel S.A.

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

Autora: Jeydis Blanco Almora

Tutor: Ing. Gianny Martínez Perdomo

Co-tutora: M.sC. Vitalia Guía Valdés

Ciudad de La Habana, 2015

Año 57 de la Revolución

"Si buscas resultados distintos, no hagas siempre lo mismo"

Albert Einstein

Declaración de autoría

Declaro ser autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.
Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____
Autora

_____ _____
Tutor Co-tutor

Datos de contacto

Tutor:

Nombre y apellidos: Gianny Martínez Perdomo.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

e-mail: giannybsb@gmail.com

Co-tutor:

Nombre y apellidos: Vitalia Guía Valdés.

Institución: División de Servicios Técnicos Integrales UCI, Copextel S.A. (DSTI UCI).

Título: Master en Ciencias de la Educación Superior.

e-mail: vita@uci.cu

Autora:

Nombre y apellidos: Jeydis Blanco Almora.

Institución: Universidad de las Ciencias Informáticas (UCI).

e-mail: jeydis@uci.cu

Agradecimientos

Quiero agradecer a mi mamá, por educarme y quererme de la forma en que lo ha hecho, por estar siempre ahí cuando la necesité, por guiarme a través de todo y enseñarme como crecer, sin dudas, la razón por la cual hoy estoy aquí.

A mi hermana por el orgullo que me hace sentir y a Felipe que me ha impulsado y alentado en toda mi carrera profesional y personal.

A mi novio por todo el amor, la ayuda y comprensión que me ha brindado.

A mis tutores Gianny y Vitalia, por tanta atención y a todos los profesores que anónimamente han sido responsables de mi preparación como persona y futura ingeniera.

A mis compañeros de trabajo por su apoyo incondicional.

A mis amigos y mis compañeros de aula por estos inmejorables 6 años.

A todos aquellos que de una forma u otra han contribuido a la realización de este proyecto.

Dedicatoria

A mi pequeña familia, no hace falta decir porqué…

Resumen

La obtención de un sistema de reportes para la División de Servicios Técnicos Integrales UCI, Copextel S.A. (DSTI UCI) que logre satisfacer los requerimientos actuales, emerge como un problema necesitado de solución. Por tal motivo se realiza la modelación de un sistema informático que abarque el flujo de procesos necesarios para realizar un reporte de rotura asociado a equipamiento tecnológico, teniendo como premisa la mejora en los tiempos de respuesta y lograr uniformidad en el proceso de gestión de reporte por parte de los disímiles clientes, usando como entorno la web.

Durante la investigación se realizó un estudio del estado del arte referente a los sistemas automatizados vinculados al campo de acción, metodologías de desarrollo de software, herramientas y lenguajes de programación posibles a utilizar para la elaboración del producto seleccionándose las más apropiadas. La metodología de desarrollo XP fue la guía para el desarrollo del sistema, generando los artefactos propuestos en cada iteración.

El objetivo de este trabajo ha sido desarrollar un sistema informático interactivo, seguro y de interfaz amigable, que permita gestionar la información para el control y soporte a las afectaciones. A dicho sistema se le aplicaron diferentes pruebas de software permitiendo detectar y corregir la máxima cantidad de errores antes de su entrega al cliente.

Abstract

Obtaining a reporting system for the Division of Integrated Technical Services UCI, Copextel SA (DSTI UCI) that successfully solves the current requirements, emerge as a problem in need of solution. Therefore the modeling of a computer system that covers the workflow necessary to complete a report of rupture associated with technological equipment was made, with the premise of improving the response times and to obtain a unified management process from the different clients, using a web environment.

During the investigation a study of the state of the art was performed, about the automatized systems related to the field of action, software development methodologies, tools and possible programming languages used for making the product and in the end the most appropriate tools, methodologies and frameworks were selected. XP development methodology was the guide for the development of the system, generating artifacts proposed in each iteration.

The aim of this work was to develop an interactive, safe and friendly interface computer system that allows managing information for the control and support to the interruptions.

In such a system were applied different tests allowing to detect and correct the maximum amount of errors before delivery to the customer.

Índice

Índice de Figuras

Índice de Tablas

Introducción

La División de Servicios Técnicos Integrales UCI, Copextel S.A. (DSTI UCI) tiene como objeto social importar y exportar equipos, accesorios, partes, piezas, insumos, mobiliarios, relacionados con: sistemas de refrigeración, climatización, lavandería, gastronomía, recreación, audio y luces profesionales, computación, ofimática y copiadoras, redes informáticas, automatización industrial y de edificios, control de acceso en edificaciones, electrónica doméstica, telecomunicaciones, protección física y sistemas de tierra y pararrayos, seguridad y protección, hidrosanitarios y de piscinas (1). Además de brindar servicios de: montaje, puesta en marcha, garantía, post-garantía, diagnóstico, instalación, mantenimiento, reparación de equipos y demás suministros previstos en su objeto social a la Universidad de las Ciencias Informáticas (UCI) y a sus otros clientes como son:

1. Centro de Estudios Avanzados de Cuba (CEA).
2. Empresa de Tecnologías de la Información para la Defensa (XETID).
3. Centro Nacional de Calidad de Software (CALISOFT).
4. Empresa Productora de Software para la Técnica Electrónica (SOFTEL).
5. Empresa de Ingeniería y Sistemas S.A. (ALBET).
6. Empresa de Servicios de Ingeniería No.1 de La Habana (ESI).

El intenso trabajo al que se encuentra sometido el equipamiento tecnológico en la UCI y el resto de los clientes atendidos condiciona una gran cantidad de roturas y por consiguiente, reparaciones. Debido a esta situación es necesario que la recepción de las solicitudes y la solución de las mismas sean lo más rápidas y eficientes posible.

De los clientes a los que la DSTI UCI brinda servicios solamente la UCI cuenta con una aplicación de gestión de incidencias (GATServer) que da solución a la recepción de reportes y está limitada a los reportes de cómputo, redes y televisión por cable. El resto de los clientes realiza este procedimiento por distintas vías de forma manual.

Para la gestión de reportes de roturas o gestión de incidencias como es también llamada mundialmente, la DSTI UCI se divide por tipo de equipamiento a atender en tres talleres de Servicios Técnicos (ST):

1. Ofimática (cómputo y electrónica).
2. Telecomunicaciones (redes, televisión por cable y seguridad).

3. Electromecánica (clima, gastronomía y tierra física).

Cada taller de ST recibe los reportes de roturas de diferente forma por lo que no existe uniformidad entre los procedimientos actuales de gestión de los mismos. Esta situación es causada porque cada cliente define su propio procedimiento de acuerdo a la organización que tengan, los cuales difieren entre sí, por lo que el rendimiento en este proceso dista de los valores que se pudieran obtener. Los procedimientos empleados actualmente se encuentran descritos en el Anexo 1 y Anexo 2.

El procedimiento actual trae como consecuencias que:

1. No se conoce con claridad la cantidad de los reportes y los detalles de roturas de equipamiento.
2. Los reportes llegan por diferentes vías a los talleres de servicios técnicos de la DSTI UCI pues los mismos se realizan de forma manual y descentralizada provocando que exista duplicidad de solicitudes.
3. La búsqueda, análisis y actualización de la información se realiza de forma engorrosa, lenta y propensa a errores.
4. No se realiza un registro oportuno de las transacciones económicas, lo que atenta contra el flujo del trabajo y el control interno.
5. Dificultad para controlar el cumplimiento del plan de trabajo individual de los técnicos de la DSTI UCI así como su desempeño.
6. Los tiempos de respuesta a los reportes recibidos no cumplen las expectativas de los usuarios.

Todo lo antes expuesto le ha permitido a la autora determinar el siguiente **problema a resolver**: ¿Cómo contribuir a la gestión de los reportes de roturas de equipamiento tecnológico en la DSTI UCI?

Se define como **objeto de estudio** los sistemas de gestión de reportes de roturas. Donde el **campo de acción** está enmarcado en el sistema de gestión de reportes de roturas aplicado a la DSTI UCI.

Para dar solución a la problemática planteada se ha trazado como **objetivo general de la investigación**: Desarrollar un sistema informático que permita contribuir a la gestión de los reportes de roturas de equipamiento tecnológico en la DSTI UCI.

Objetivos específicos:

1. Elaborar la fundamentación teórica de la investigación.
2. Conformar el análisis y diseño de la solución de software propuesta para la DSTI UCI.

3. Realizar la implementación de la solución de software propuesta para la DSTI UCI.
4. Validar mediante pruebas los resultados obtenidos con la solución.

Para dar cumplimiento al objetivo general se han trazado las siguientes **tareas de investigación:**

1. Establecimiento de los fundamentos teórico-metodológicos para la gestión de incidencias o reportes de roturas.
2. Caracterización del proceso de gestión de los reportes de roturas de equipamiento tecnológico en la DSTI UCI.
3. Levantamiento de requerimientos, para conocer las capacidades y condiciones que debe poseer el sistema.
4. Definición del modelo de análisis del sistema para la gestión de los reportes de roturas de equipamiento tecnológico en la DSTI UCI.
5. Realización de la exploración y planificación del sistema a desarrollar.
6. Elaboración del diseño del sistema usando la metodología de desarrollo seleccionada.
7. Implementación del sistema informático “Le Atiendo” para dar solución a los requerimientos para la gestión de los reportes de roturas de equipamiento tecnológico en la DSTI UCI.
8. Validación del sistema implementado.

Métodos teóricos empleados:

- **Histórico – lógico:** Permitió realizar el recorrido de los antecedentes históricos de los sistemas de gestión de reportes de roturas y su utilización en las diferentes problemáticas, así como conocer las etapas evolutivas por las que han transitado, para lo cual se realizó un análisis lógico de los acontecimientos y sucesos que la condicionaron.

- **Analítico – sintético:** Se realizó un estudio con profundidad de la información acerca de las tecnologías, metodologías y herramientas posibles a ser utilizadas en el desarrollo del sistema de gestión propuesto, pudiendo definir con mayor certeza las mismas, sintetizando sus características y analizando la viabilidad de cada una.

Métodos empíricos:

- **Análisis documental:** Se obtuvo abundante información sobre el desarrollo, empleo e impacto de los sistemas de gestión de reportes de roturas en el mundo y en Cuba. Estas, una vez procesadas,

sirvieron de referencia para la construcción de la propuesta de solución a la problemática en la DSTI UCI.

- **Observación:** Se pudo conocer a través de este método la esencia de la problemática definida, lo que ayudó al planteamiento del problema a resolver, además de permitir conocer los sistemas de gestión de reportes de roturas, lo cual influye a la hora de tener un conocimiento más detallado de lo que se quiere, lo que hace falta hacer y cómo hay que hacerlo.

El desarrollo de este documento se encuentra compuesto por tres capítulos donde se refleja todo el trabajo investigativo, el diseño del sistema y la implementación de la solución propuesta, distribuido de la siguiente manera:

En el Capítulo I Fundamentación teórica: Se hace un análisis del estado del arte del objeto de estudio, se investiga acerca de los sistemas informáticos vinculados al campo de acción, se fundamentan las metodologías, tecnologías y herramientas utilizadas para el desarrollo del sistema de gestión.

En el Capítulo II Exploración, planificación y diseño del sistema: Se define el negocio y se describe la solución propuesta para la situación problemática. Se presentan las características y funcionalidades del sistema a partir de los requisitos funcionales y no funcionales capturados. Se realiza el análisis, la exploración, planificación y diseño del sistema.

En el Capítulo III Implementación y validación del sistema: Incluye la programación realizada a partir de los requerimientos y pruebas utilizadas para la validación de la misma.

Capítulo 1: Fundamentación teórica

Introducción

En este capítulo se hace un estudio sobre los conceptos asociados a los sistemas de gestión de incidencias o reportes de roturas los cuales se vinculan a la actividad económica - contable. Se describen brevemente las técnicas fundamentales de programación, también se hace un estudio de las tecnologías y herramientas utilizadas mundialmente en el desarrollo de software.

Se abordan los fundamentos teóricos asociados a la solución del problema descrito en la introducción, los antecedentes del trabajo y una descripción del flujo actual de los procesos que están involucrados en el campo de acción para llegar a la solución del problema planteado.

1.1. Marco conceptual

Gestión

Caridad Darromán Savigne y Reynerio Velázquez Leyva plantean que gestión se referirá a la acción y al efecto de administrar o gestionar un negocio. También y a la par de esto, en una gestión habrá que dirigir, gobernar, disponer, organizar y ordenar para lograr los objetivos propuestos (2).

Mientras que Fernando Fantova expresa que la gestión es la asunción y ejercicio de responsabilidades sobre un proceso. En la gestión se realizan un conjunto de trámites que se llevan a cabo para resolver un asunto o concretar un proyecto. Incluye para lograr los objetivos propuestos:

- La preocupación por la disposición de los recursos y estructuras necesarias para que tenga lugar.
- La coordinación de sus actividades (y correspondientes interacciones).
- La rendición de cuentas ante el abanico de agentes interesados por los efectos que se espera que el proceso desencadene (3).

La autora de esta investigación asume que el concepto más abarcador para gestión es el planteado por Fernando Fantova.

Sistema

Un sistema es un conjunto de elementos relacionados. Puede estructurarse de conceptos, objetos y sujetos, se compone de otros sistemas a los que llamamos subsistemas. También puede mencionarse la noción de sistema informático, muy común en las sociedades modernas. Este tipo de sistemas denominan al conjunto de hardware, software y soporte humano que forman parte de una empresa u organización. Incluyen

ordenadores con los programas necesarios para procesar datos y las personas encargadas de su manejo (4).

Sistema de gestión

Es la gestión de los recursos que tienen que ver con el apoyo a sistemas y servicios de la información para una empresa (5). También es considerado como un conjunto de etapas unidas en un proceso continuo, que permite trabajar ordenadamente hasta lograr mejoras. Establece cuatro etapas que hacen de este sistema un proceso circular, a medida que se repite el ciclo se logra obtener una mejora.

Las cuatro etapas del sistema de gestión son:

- Ideación
- Planeación
- Implementación
- Control (6)

Incidencia

Es aquello que acontece en el curso de un asunto y que cambia su devenir. Según la ITIL se refiere a cualquier evento que no forma parte del desarrollo habitual de un servicio y que causa, o puede causar una interrupción del mismo o una reducción de la calidad de dicho servicio (5).

También es conceptualizado por el Ministerio del Poder Popular para Ciencia, Tecnología e Innovación de Venezuela como un evento, anomalía o interrupción de un servicio, inesperado o no deseado, que tienen una probabilidad significativa de comprometer las operaciones de un sistema y de amenazar la seguridad de la información del mismo (7).

A partir de los conceptos anteriormente expuestos la autora conceptualiza una incidencia como cualquier evento que no forma parte del desarrollo habitual de un servicio que puede causar una interrupción del mismo o una reducción de la calidad de dicho servicio y tiene una probabilidad significativa de comprometer las operaciones de un sistema.

Gestión de incidencias

La gestión de incidencias es un área de procesos perteneciente a la gestión de servicios de tecnologías de la información. El primer objetivo de la gestión de incidencias es recuperar el nivel habitual de funcionamiento del servicio y minimizar en todo lo posible el impacto negativo en la organización de forma

que la calidad del servicio y la disponibilidad se mantengan. La resolución de las incidencias debe ser ejecutada lo antes posible para restaurar el servicio rápidamente (8).

1.2. Análisis de soluciones existentes

El desarrollo de la informática ha dado lugar a una creciente demanda de medios tecnológicos y otros artefactos relacionados a la ciencia de la informática. Para solucionar los desperfectos y diversos problemas que presentan a diario muchos de estos medios se han creado centros de servicios técnicos. Los centros de servicios técnicos también necesitan de un orden de trabajo y un sistema automatizado para gestionar cada incidencia recibida. A continuación se presentarán algunos ejemplos de sistemas informáticos para la gestión de reportes o también llamados sistemas gestores de incidencias empleados en diferentes centros de servicios técnicos para la gestión de su trabajo.

1.2.1. Ámbito internacional

Existen varios sistemas informáticos para la gestión de incidencias usados en diferentes talleres de reparación y centros de servicios técnicos en el mundo. Como resultado de una búsqueda en Internet se encontraron los siguientes:

GLPI (Gestión Libre du Parc Informatique)

Es un programa de Software Libre distribuido bajo licencia GPL, para la administración y gestión de un parque de recursos informáticos. También es definido como un administrador de recursos informáticos que posee una consola de administración web basada en PHP (9).

Los usos para GLPI son múltiples, entre los que se destacan:

- Permite crear una base de datos para mantener un inventario del equipamiento informático (computadores, impresoras, software).
- Presenta recursos para facilitar las labores administrativas; tales como programar, solicitar o darle seguimiento a las tareas, envío de correos para la notificación y comunicación de las tareas.
- Permite el registro y atención de solicitudes de servicio para el soporte técnico, con posibilidades de notificación por correo electrónico a usuarios y al mismo personal de soporte.
- Almacena historiales de las diferentes informaciones, labores de reparación y procedimientos relacionados llevados a cabo sobre los recursos informáticos.

Es una aplicación totalmente web que soluciona los principales problemas de la gestión del inventario informático, la administración de los recursos de hardware, software, usuarios, suministros e incidencias.

Aspectos negativos de GLPI:

- No permite obtener estadísticas e información sobre la disponibilidad de servicio de los puestos de trabajo.
- La estructura de GLPI está orientada a la administración de recursos informáticos.

Fig. 1: Interfaz de GLPI

KMKey Help Desk

KMKey Help Desk es un software de gestión de incidencias indicado para servicios de mantenimiento, ayuda al usuario y a la resolución de problemas en cualquier sector. Permite definir flujos de trabajo para abordar problemáticas derivadas de anomalías en servicios y maquinaria. La incidencia puede recibirse de forma automática (*e-mail*, entrada a través de una web o desde un dispositivo móvil) o bien ser abierta por el servicio de atención. Una vez en marcha seguirá el flujo diseñado por el cliente para su resolución. Además posibilita la resolución inmediata, escalado, consulta de información anterior, reparto de recursos y realizar los trabajos necesarios para atender el mantenimiento de sus instalaciones y disponer, desde cualquier acceso a Internet, de toda la información relevante:

Gestión de incidencias:

- **Recepción**: Vía *e-mail*, entradas por el propio usuario o desde un formulario web. Abre de forma automática un expediente y avisa mediante notificación (interna, correo electrónico o SMS) al responsable.
- **Resolución:** Atención y cerrado inmediato o planificación en el tiempo según tipo o urgencia. Asignación de recursos. Listas con el estado y los responsables. Retrasos. Avisos automáticos a quien sea necesario. Posibilidad de añadir notas, documentos, correos, fotos, manuales. Búsquedas en base de datos de conocimiento. Introducción de horas y material utilizado. Facturación.
- **Control:** Generación y envío automático del resumen de la incidencia. Generación de informes y estadísticas por varios criterios: urgencia, tipo, contrato, técnico. Informes de horas invertidas. Informe de elementos problemáticos. Filtros e informes por periodos y clientes

Mantenimientos preventivos:

- **Planificación:** Posibilidad de lanzar mantenimientos preventivos de elementos o contratos con diversas periodicidades.
- **Realización:** Avisos y listados de tareas a realizar. Gráficos GANTT. Reflejar tiempo y material utilizados. facturación y seguimiento administrativo. Adjuntar documentación y correos electrónicos
- **Control:** Informes de cada actuación realizada. Envío automático de listados y estadísticas de trabajos realizados.

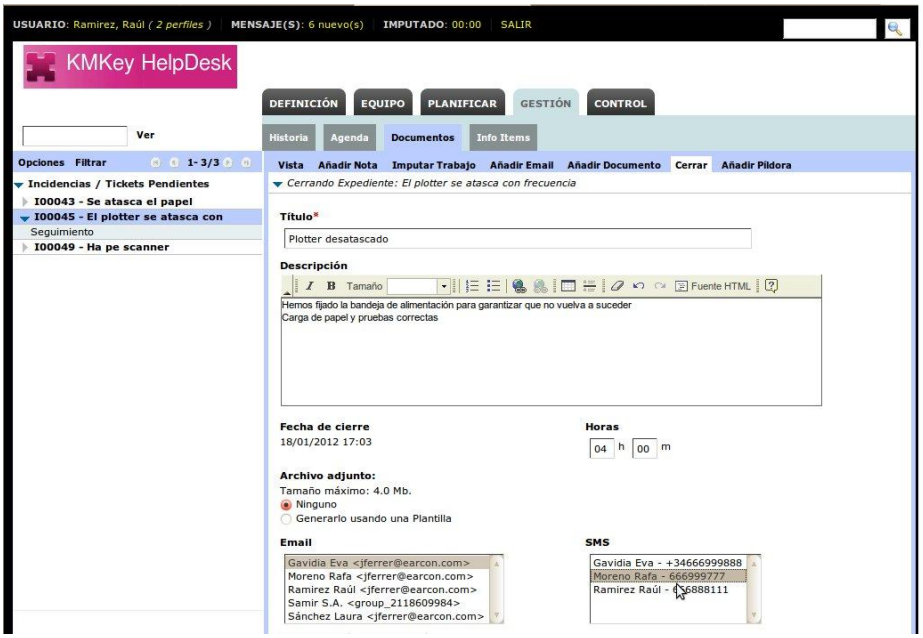

Fig. 2: Interfaz de KMKey Help Desk

1.2.2. Ámbito nacional

Centro de llamadas "Le Atiendo"

Copextel cuenta con el Centro de Llamadas de atención a clientes "Le Atiendo" mediante el cual la población puede contactar a través del teléfono 833-3333, en el caso de las provincias de La Habana, Mayabeque y Artemisa, en el resto de los territorios se podrá comunicar con cada una de las oficinas de Copextel en su provincia (10). Para la DSTI UCI esta solución es poco factible pues no se cuenta con personal disponible las 24 horas ni con la infraestructura requerida para la realización de este trabajo.

1.2.3. En la UCI

GATServer

Es una aplicación web desarrollada por el Grupo de Asistencia Técnica (GAT) de la UCI para la gestión de las afectaciones tecnológicas, que permite la gestión de reportes para las afectaciones de medios computacionales, redes y telefonía (11). La aplicación permite insertar reportes de roturas y generar las soluciones de los mismos pero al ser una aplicación web propia, adaptada a su función en el centro no es una solución óptima para la DSTI UCI pues en la misma se necesita una aplicación que se ajuste al flujo de sus procesos internos y que responda a las necesidades existentes, además debe ser una aplicación

escalable que permita la posible integración futura con el resto de los sistemas contables usados en la misma.

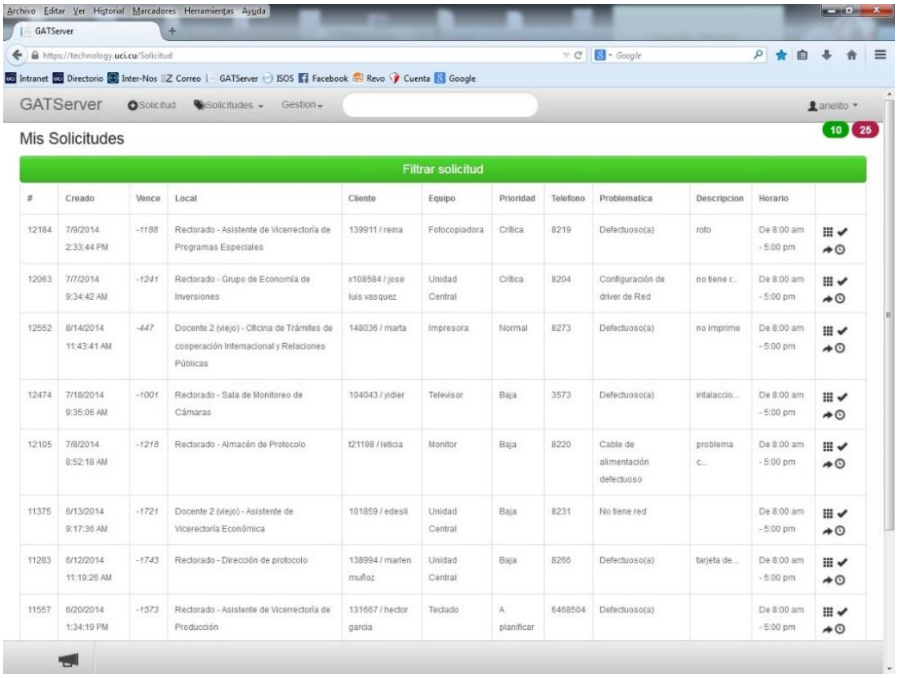

Fig. 3: Interfaz de GATServer

1.3. Metodologías a utilizar

El objetivo de un proceso de desarrollo es elevar la calidad del software (en todas las fases por las que pasa) a través de una mayor transparencia y control sobre el proceso. Es labor del proceso de desarrollo hacer que esas medidas para aumentar la calidad sean reproducibles en cada desarrollo.

En los últimos tiempos la cantidad y variedad de los procesos de desarrollo ha aumentado de forma impresionante, sobre todo teniendo en cuenta el tiempo que estuvo en vigor como ley única el famoso desarrollo en cascada. Se podría decir que en estos últimos años se han desarrollado dos corrientes en lo referente a los procesos de desarrollo, los llamados métodos pesados y ligeros. La diferencia fundamental entre ambos es que mientras los métodos pesados intentan conseguir el objetivo común por medio del orden y documentación, los métodos ligeros (también llamados métodos ágiles), tratan de mejorar la calidad del software por medio de una comunicación directa e inmediata entre las personas que intervienen en el proceso. Existen varias metodologías que se pueden utilizar para el desarrollo del presente trabajo de diploma, entre las que se encuentran las siguientes:

1.3.1. Proceso Unificado de Rational (RUP)

RUP es un proceso de desarrollo propuesto por Rational Software Corporation, resultado del esfuerzo de las tres últimas décadas en desarrollo de software y de la experiencia de sus creadores Ivar Jacobson, Grady Booch y James Rumbaugh. Este proceso proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en una organización de desarrollo.

Es una metodología basada en un pequeño grupo de principios claves: el equipo de un proyecto de software debe planificar el desarrollo, debe conocer hacia donde se dirige, debe documentar el proyecto de una manera perdurable y extensible.

RUP está definido por tres características fundamentales:

- **Dirigido por casos de uso**: Los casos de uso constituyen la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo incluyendo el diseño, la implementación y las pruebas del sistema.
- **Centrado en la arquitectura**: La arquitectura involucra los elementos más significativos del sistema y está influenciada por plataformas de software, sistemas operativos, manejadores de BD, protocolos, consideraciones de desarrollo como sistemas heredados y requisitos no funcionales.
- **Iterativo e incremental**: Para hacer más manejable un proyecto se recomienda dividirlo en ciclos. Para cada ciclo se establecen fases de referencia, cada una de las cuales debe ser considerada como un pequeño proyecto cuyo núcleo fundamental está constituido por una o más iteraciones de las actividades principales básicas de cualquier proceso de desarrollo.

La metodología RUP divide en 4 fases el desarrollo del software:

- **Inicio:** En la fase inicial se establecen los objetivos para el ciclo de vida del producto. Se establece el caso del negocio con el fin de delimitar el alcance del sistema.
- **Elaboración:** En elaboración se plantea la arquitectura para el ciclo de vida del producto. En esta fase se realiza la captura de la mayor parte de los requisitos funcionales, al manejar los riesgos que interfieran con los objetivos del sistema.
- **Construcción:** En construcción se alcanza la capacidad operacional del producto. En esta fase a través de sucesivas iteraciones e incrementos se desarrolla un producto software, listo para operar, el cual es frecuentemente llamado versión beta

- **Transición:** En transición se realiza la entrega del producto operado una vez realizadas las pruebas de aceptación por un grupo especial de usuarios al efectuarse los ajustes y correcciones que sean requeridos (12).

RUP incluye artefactos y roles, con dichas características ha hecho que junto con el UML, constituya la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos (13).

1.3.2. Programación Extrema XP

La metodología XP consiste básicamente en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lograr un producto de buena calidad en poco tiempo. La Programación Extrema es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software. Promueve el trabajo en equipo, preocupándose en todo momento del aprendizaje de los desarrolladores y estableciendo un buen clima de trabajo.

Este tipo de método se basa en una realimentación continuada entre el cliente y el equipo de desarrollo con una comunicación fluida entre todos los participantes, también busca simplificar las soluciones implementadas y coraje para los múltiples cambios. Este tipo de programación es la adecuada para los proyectos con requisitos imprecisos, muy cambiantes y con un riesgo técnico excesivo.

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck, el padre de XP, describe la filosofía de XP en dos: sin cubrir los detalles técnicos y de implantación de las prácticas.

Las prácticas fundamentales de la metodología XP son:

- **Entregas pequeñas:** La idea es producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad pretendida para el mismo.
- **Refactorización:** La refactorización es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. También mejora la estructura interna del código sin alterar su comportamiento externo.
- **Propiedad colectiva del código:** Cualquier programador puede cambiar una parte del código en un momento dado. Esta práctica motiva a todos a contribuir con nuevas ideas en todos los segmentos

del sistema, evitando a la vez que algún programador sea imprescindible para realizar cambios en alguna porción de código (14).

En la siguiente figura se muestra como esta metodología es iterativa y utiliza pequeños lapsos de tiempo para realizar las actividades.

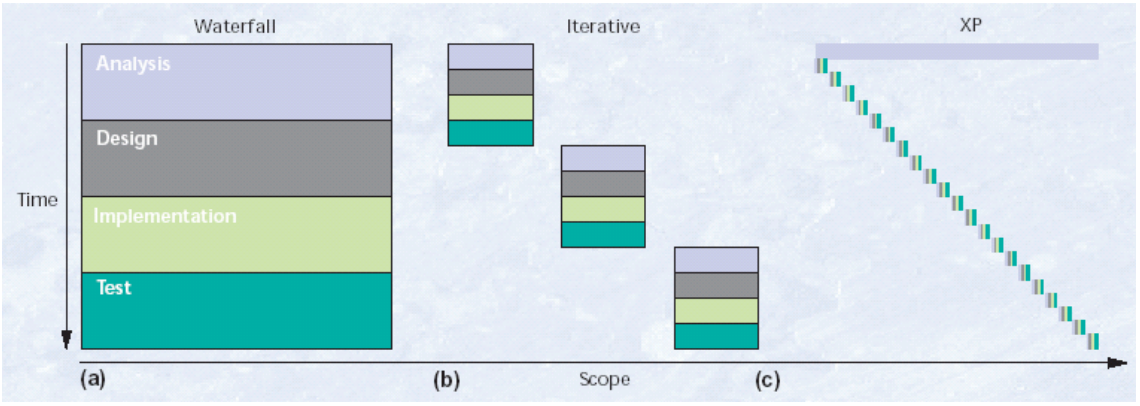

Fig. 4: Iteraciones de XP

¿Por qué XP?

Al realizar un estudio de las principales metodologías de desarrollo de software se decidió optar por XP al adaptarse a las características necesarias persiguiendo el objetivo de aumentar la productividad a la hora de desarrollar programas. Es una metodología que ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de software para el mejoramiento de la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo.

Está especialmente indicada para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad. Ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro, permitiendo además seguir de forma clara el avance de las tareas a realizar, de forma que los jefes pueden ver día a día cómo progresa el trabajo.

1.4. Lenguajes de modelado

Con el transcurso de los años se han ido desarrollando las técnicas de programación orientada a objetos (POO). Además se han desarrollado diferentes lenguajes de modelación orientada a objetos. Algunos ejemplos de estos lenguajes de modelado son I* y UML.

*1.4.1. Lenguaje de modelado OO i**

La notación i* fue creada en la primera mitad de la década de los 90. Permite expresar de forma clara y sencilla los objetivos de los actores que aparecen en los modelos y la dependencia entre ellos. Cuenta con una notación gráfica que permite tener una visión intuitiva y unificada del entorno modelado mostrando tales actores y dependencias. Tiene la desventaja de no tener una definición única del lenguaje. Además, las definiciones existentes no son tan claras como se desearía ya que contienen ambigüedades y contradicciones.

1.4.2. Lenguaje de modelado OO UML

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Además capta la información sobre la estructura estática y el comportamiento dinámico de un sistema.

Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo. La estructura estática define los tipos de objetos importantes para un sistema y para su implementación, así como las relaciones entre los objetos. El comportamiento dinámico define la historia de los objetos en el tiempo y la comunicación entre objetos para cumplir sus objetivos. El modelar un sistema desde varios puntos de vista, separados pero relacionados, permite entenderlo para diferentes propósitos (15) (16).

Las herramientas que provee UML para el desarrollo de sistemas orientados a objetos son:

- Diagrama de casos de uso
- Diagrama de clases
- Diagrama de estados
- Diagrama de secuencias
- Diagrama de actividades

- Diagrama de colaboraciones
- Diagrama de componentes
- Diagrama de distribución

Características de UML

- Permite adaptarse fácilmente a los usuarios, así como a otros usuarios de otros métodos.
- Permite modelar nuevas cosas en el proyecto que se desarrolla.
- UML provee una expresividad e integridad holística mejorada, respecto a otros lenguajes de moldeamiento visual.
- UML es fácil de aprender y usar, ya sea respecto a las técnicas más avanzadas, es decir, estereotipos y propiedades, así como algunos cambios en la anotación y semánticas.

¿Por qué UML?

Debido al estudio de las características de los lenguajes de modelado especificadas anteriormente se determinó como el lenguaje de modelado que será utilizado para dar solución a los objetivos planteados es UML ya que sus características se corresponden con lo que se quiere lograr del sistema y además es el lenguaje de modelado más usado en la UCI.

1.5. Herramienta CASE

Las herramientas CASE son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo tiempo y dinero. Estas herramientas ayudan en el 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 y documentación o detección de errores.

1.5.1. Rational Rose Data Modeler

Es una herramienta de modelado visual que posibilita que los diseñadores de bases de datos, analistas, arquitectos, desarrolladores y todos los demás miembros del equipo de desarrollo trabajen juntos, capturando y compartiendo los requerimientos de negocio y dándoles seguimiento a medida que cambian a través del proceso.

1.5.2. Visual Paradigm

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Además, 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.

¿Por qué Visual Paradigm?

Esta herramienta es estable en cuanto a su ejecución en diferentes sistemas operativos y la facilidad de abrir y trabajar con un modelo. Es una herramienta que guarda todo el modelo en un solo fichero por lo que basta con copiarse solo ese fichero para estar seguro de que tiene todo el trabajo encapsulado en él. Además el equipo de desarrollo presenta conocimientos básicos de la herramienta lo que posibilita un mejor desempeño con la misma. Finalmente, Visual Paradigm es una herramienta bastante ligera permitiendo su uso en ordenadores poco potentes.

1.6. Servidor web

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI.

1.6.1. Internet Information Services (IIS)

Internet Information Services o IIS es un servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows. IIS se proporciona con Windows, pero no se instala o habilita de forma predeterminada. Al habilitar IIS en el servidor se crea un servidor de sitio web para que los clientes puedan utilizar HTTP o HTTPS. Este servicio convierte a una computadora en un servidor web para Internet o una intranet, es decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente. Se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas. Incluye Active Server Pages (ASP) y ASP.NET. También pueden ser incluidos los de otros

fabricantes, como PHP o Perl. Una de las desventajas que presenta este servidor web es que solo se puede utilizar en computadoras con sistema operativo Windows (17).

1.6.2. Servidor web Apache 2.4.4

Apache es el servidor web más utilizado, líder con el mayor número de instalaciones a nivel mundial muy por delante de otras soluciones como el IIS de Microsoft. Apache es un proyecto de código abierto y uso gratuito, multiplataforma, muy robusto y que destaca por su seguridad y rendimiento.

El servidor Apache HTTP, también llamado simplemente Apache, es el estándar en la entrega de servicios web y ha abierto el camino para la mayor expansión de las capacidades de la web. Apache se basa en una plataforma de servicio web de fuente abierta originalmente desarrollada para servidores de Linux/Unix, pero se configuró posteriormente para que funcione con Windows y otros sistemas operativos. Aunque el servidor web Apache sea gratuito, no deja de ser más rápido en comparación con los servidores web más caros del mercado. Los servidores Apache pueden manejar más de un millón de visitas por día sin dificultades (18).

¿Por qué servidor web Apache?

Para el desarrollo del sistema informático “Le Atiendo” se utilizará el servidor web HTTP Apache por las características antes mencionadas las cuales han sido resumidas en la siguiente tabla comparativa, además de las ventajas que presenta, siendo el más indicado para satisfacer los requisitos actuales.

Tabla 1. Comparación entre servidores web

Internet Information Services (IIS)	Servidor web Apache
Es un producto comercial.	Es un producto libre.
Funciona solo sobre sistemas operativos de Microsoft Windows.	Funciona en múltiples sistemas operativos.
Fácil instalación.	La instalación requiere más trabajo y conocimiento.
Cuenta con soporte técnico de la corporación Microsoft.	Cuenta con el apoyo de las comunidades de programadores y usuarios.
Contiene un número determinado de funciones y capacidades.	Ofrece más flexibilidad, funcionalidad y un mejor rendimiento.
La administración se realiza mediante una interfaz gráfica de fácil uso.	Usualmente la administración no es gráfica y requiere más trabajo.

Ventajas

- Modular
- Código abierto
- Multiplataforma
- Extensible
- Popular (fácil conseguir ayuda/soporte)

1.7. Lenguajes de programación

Los lenguajes de programación intentan conservar una similitud con el lenguaje humano, con la finalidad de que sean más naturales a quienes los usan. Establecen un conjunto de reglas sintácticas y semánticas, las cuales rigen la estructura del programa de computación que se escribe o edita. De esta forma, permiten a los programadores o desarrolladores, poder especificar de forma precisa los datos sobre los que se va a actuar, su almacenamiento, transmisión y demás acciones a realizar bajo las distintas circunstancias consideradas.

1.7.1. ASP.Net

ASP es un entorno de secuencias de comandos en el lado del servidor que se puede utilizar para crear y ejecutar aplicaciones de servidor web dinámicas, interactivas y de alto rendimiento.ASP.Net es un ambiente de programación construido sobre el entorno NGWS (New Generation Windows Services, o sea, Servicios de la Nueva Generación de Windows), que permite crear poderosas aplicaciones de Internet.

Puede aprovechar las ventajas del enlace anticipado, la optimización nativa y los servicios de caché desde el primer momento por lo que aporta un mejor rendimiento. Debido a que permite el uso de una gran variedad de lenguajes de programación, tiene gran flexibilidad.

Este ambiente facilita la realización de tareas comunes y con servicio de código administrado como el recuento de referencia automático y el recolector de elementos no utilizados. Con la autenticación de Windows integrada a la configuración por aplicaciones, se puede tener la completa seguridad de que las aplicaciones están a salvo.

1.7.2. Hypertext Preprocessor (PHP) 5.4.7

PHP (acrónimo de Hypertext Preprocessor) es un lenguaje "del lado del servidor" (esto significa que PHP funciona en un servidor remoto que procesa la página web antes de que sea abierta por el navegador del

usuario) especialmente creado para el desarrollo de páginas web dinámicas. Puede ser incluido con facilidad dentro del código HTML.

PHP es un lenguaje de script interpretado en el lado del servidor que no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache o IIS (Internet Information Server) con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas. En la figura 4 se muestra su interacción en la web.

Fig. 5: Interpretación lenguaje PHP

Ventajas:

- Muy fácil de aprender.
- Se caracteriza por ser un lenguaje muy rápido.
- Soporta en cierta medida la orientación a objeto. clases y herencia.
- Es un lenguaje multiplataforma: Linux, Windows, entre otros.
- Capacidad de conexión con la mayoría de los manejadores de base de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otras.
- Capacidad de expandir su potencial utilizando módulos.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Incluye gran cantidad de funciones.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

¿Por qué PHP?

Se seleccionó PHP para el desarrollo de este trabajo de diploma por las características y ventajas antes mencionadas.

1.8. Framework de desarrollo

Un *framework* de aplicaciones web es un tipo de *framework* que permite el desarrollo de sitios web dinámicos, servicios web y aplicaciones web. El propósito de este tipo de *framework* es permitir a los desarrolladores construir aplicaciones web y centrarse en los aspectos interesantes, aliviando la típica tarea repetitiva asociada con patrones comunes de desarrollo web. La mayoría de los *frameworks* de aplicaciones web proporcionan los tipos de funcionalidad básica común, tales como sistemas de plantillas, manejo de sesiones de usuario, interfaces comunes con el disco o el almacenamiento en base de datos de contenido, y persistencia de datos. Normalmente, estos *frameworks* promueven la reutilización y conectividad de los componentes, así como la reutilización de código, y la implementación de bibliotecas para el acceso a base de datos (19).

1.8.1. Zend Framework

Zend Framework (ZF) es un *framework* de código abierto para desarrollar aplicaciones web y servicios web con PHP. ZF implementa el patrón MVC (Modelo, Vista, Controlador), es orientado a objetos y sus componentes tienen un bajo acoplamiento por lo que pueden ser usados en forma independiente. Un punto importante es que brinda un estándar de codificación a seguir en los proyectos.

Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de Zend Framework conforman un potente y extensible *framework* de aplicaciones web al combinarse. ZF ofrece un gran rendimiento y una robusta implementación MVC, una abstracción de base de datos fácil de usar, y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos (20).

1.8.2. Symfony 2.3

Symfony es un completo *framework* diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de dominio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador

dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de los gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, entre otras) como en plataformas Windows.

Características

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de los casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de las mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

Symfony incorpora varios entornos de desarrollo diferentes e incluye varias herramientas que permiten automatizar las tareas comunes de la ingeniería del software:

- Las herramientas que generan automáticamente código han sido diseñadas para hacer prototipos de aplicaciones y para crear fácilmente la parte de gestión de las aplicaciones.
- La barra de depuración web simplifica la depuración de las aplicaciones, ya que muestra toda la información que los programadores necesitan sobre la página en la que están trabajando.
- La interfaz de línea de comandos automatiza la instalación de las aplicaciones entre servidores.
- Es posible realizar cambios “en caliente” de la configuración (sin necesidad de reiniciar el servidor).

- El completo sistema de log permite a los administradores acceder hasta el último detalle de las actividades que realiza la aplicación.

¿Por qué Symfony?

Para la implementación del sistema a desarrollar se ha utilizado como *framework* de desarrollo Symfony, después de hacer una minuciosa y exhaustiva investigación. Los resultados de la misma y el por qué se seleccionó Symfony para el desarrollo de este trabajo de diploma se muestra en la tabla 15 en los anexos (Anexo 3).

1.9. Lenguajes de programación del lado del cliente

Un lenguaje del lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. Pero la página no se verá bien si el ordenador cliente no tiene instalados los *plugin* adecuados. El código, tanto del hipertexto como de los scripts, es accesible a cualquiera y ello puede afectar a la seguridad.

1.9.1. HTML

El HTML nació para producir todo tipo de documentos estructurados. En sus inicios el HTML creaba documentos muy básicos, pero muy pronto con el auge de Internet, comienza también su desarrollo. Así en sus comienzos los navegadores de Internet solo funcionaban en modo texto. Luego todos se fueron mejorando a un ritmo extraordinario debido al desarrollo de la red de redes (Internet).

Lenguaje de programación especializado en crear páginas web, para ello se utilizan una serie de etiquetas. Todo documento creado con HTML tiene una estructura claramente definida. Siempre se comienza con la etiqueta <HTML>, que es la que comprende a toda la página web. Tiene dos secciones básicas bien diferenciadas: la cabecera y el cuerpo, que se corresponden con las etiquetas <head> y <body> respectivamente. La cabecera puede contener información, siempre lleva el título del documento HTML encerrado por la etiqueta <title>, mientras que en el cuerpo se localiza todo el contenido de la página web, sea texto, imágenes, sonido, hipervínculos, video, entre otros.

1.9.2. JavaScript

El JavaScript es un lenguaje interpretado, lo que significa que no necesita ser compilado. Proviene del Java y se utiliza principalmente para la creación de páginas web. Es una mezcla entre Java y HTML. Es un lenguaje muy diferente del Java porque es orientado a objetos, no tiene herencia, sino que es más bien un lenguaje orientado a eventos. Otra diferencia entre ambos lenguajes es que mientras con Java se puede

crear aplicaciones autónomas, el JavaScript es un lenguaje que se incorpora dentro de la página web, formando parte del código HTML, sin el que no puede existir.

Se puede incluir el código del JavaScript en cualquier página web o documento HTML, desde el punto de vista cliente como servidor en documentos PHP, ASP, entre otros. Este código va incluido dentro de las etiquetas de HTML, de esta manera: <SCRIPT> </ SCRIPT>. El Java Script es un lenguaje que diferencia entre mayúsculas o minúsculas y los espacios en blanco también los tiene en cuenta.

Características:

- JavaScript es manejado por eventos: JavaScript puede responder a eventos como el movimiento del mouse y la carga de una página web.
- JavaScript es independiente de cualquier plataforma. Los programas de JavaScript están diseñados para ejecutarse dentro de documentos HTML. Son independientes de cualquier plataforma o sistema operativo.
- JavaScript permite desarrollo rápido. El navegador web y el código HTML manejan la mayoría de las características como formas, cuadros, y otros elementos de interfaz gráfica del usuario (GUI). Esto hace que los programadores de JavaScript no tienen que preocuparse de crear o manejar estos elementos en sus aplicaciones.
- JavaScript es fácil de aprender: No incluye complejas reglas sintácticas.

¿Por qué JavaScript?

Para el desarrollo del sistema informático “Le Atiendo” se ha seleccionado el lenguaje de programación web del lado del cliente JavaScript pues es el más extendido. Con este lenguaje script basado en objetos se pueden generar páginas dinámicamente en función de las preferencias del usuario, validar los datos introducidos en un formulario o modificar dinámicamente el contenido de la página.

1.10. Frameworks en JavaScript

Actualmente en el mundo existen una gran cantidad de *frameworks* JavaScript los cuales tienen como objetivo mejorar y amenizar la experiencia del usuario. En la tabla siguiente se muestra una comparación entre algunos de ellos de los cuales los más usados son jQuery y ExtJS entre los que se realizará la selección para el uso en el desarrollo del sistema informático “Le Atiendo”.

Tabla 2. Comparación de *frameworks* en JavaScript

Framework	Navegadores compatibles	Ajax	Efectos	Eventos	CSS	Extensibilidad	licencia
jQuery	IE6+, FF2+, SF3+, OP9+, CR1+	x	x	x	x	x	MIT & GLP
Moo Tools	IE6+, FF1.5+, SF2+, OP9+	x	x	x	x	x	MIT
Prototype	IE6+, FF1.5+, SF2+, OP9.25+, CR1+	x	x	x	x	x	MIT
Script aculo us	IE6+, FF1.5+, SF2+, OP9.25+, CR1+	x	x	x	x	x	MIT
Dojo	IE6+, FF2+, SF3.2+, OP9.6+	x	x	x	x	x	BSD & AFL
Yahoo! UI Library	IE6+, FF3+, SF3+, OP9+	x	x	x	x	x	BSD
Ext	IE6+, FF1.5+, SF3+, OP9+	x	x	x	x	x	GLP, Comercial
qooxdoo	IE6+, FF1.5+, SF3+, OP9+, CR1+	x	x	x	x	x	LGLP & EPL
$fx()	IE7+, FF2+, SF3+, OP9+, CR1+	-	x	-	x	x	MIT & GLP
Scripty2	IE6+, FF2+, SF3+, OP9+, CR1+	-	x	x	x	x	MIT

1.10.1. jQuery

jQuery es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privados. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

Es una librería liviana que enfatiza la interacción entre JavaScript y HTML. Contiene selectores de elementos DOM usando el motor Sizzle y permite la modificación de DOM (incluyendo soporte para CSS 3 y Xpath).

1.10.2. ExtJS 4.0

ExtJS es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. Fue desarrollada por Sencha.

Originalmente construida como una extensión de la biblioteca YUI por Jack Slocum, en la actualidad puede usarse como extensión para las bibliotecas jQuery y Prototype. Posee controles para campos de textos, incluyendo áreas de texto. Controladores selectores de fecha, campos numéricos, para *radiobox* y

checktbox. También componentes para crear y manipular *datagrids* donde goza de cierta ventaja sobre otros *frameworks*. Es posible crear "ventanas" con barras de herramientas y menús con estilo de aplicaciones de escritorio, diálogos modales y eventos.

ExtJS puede ser adquirido bajo licencias libres y comerciales. La compañía detrás de este *framework* ofrece cursos de capacitación y un extenso soporte.

¿Por qué ExtJS?

Para el desarrollo del sistema informático "Le Atiendo" se ha seleccionado ExtJS que además de flexibilizar el manejo de componentes de la página como el DOM, Peticiones AJAX, DHTML, tiene la capacidad de crear interfaces de usuario muy funcionales.

Esta librería incluye:

- Componentes UI del alto performance y personalizables.
- Modelo de componentes extensibles.
- Un API fácil de usar.
- Licencias *open source* (GPL) y comerciales.

1.11. IDEs de desarrollo

Una decisión importante a la hora de desarrollar en PHP que es el lenguaje de programación a utilizar, es el IDE a usar, ya que el entorno de desarrollo que puede suponer una verdadera diferencia en el tiempo de trabajo invertido.

1.11.1. Netbeans

NetBeans es un entorno de desarrollo muy completo y profesional. Es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento. Contiene muchas funcionalidades, para distintos tipos de aplicaciones y para facilitar al máximo la programación, la prueba y la depuración de las aplicaciones que se desarrollan. También incorpora un editor propio. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. Es multiplataforma y se puede utilizar para programar en otros lenguajes además de PHP.

Además de las funciones más básicas con las que debería contar cualquier IDE, como resaltado de sintaxis, autocompletado, formateo de código o depurador, también cuenta con otras funcionalidades menos

comunes como la integración con PHPUnit para las pruebas unitarias y con CVS, Subversion y Mercurial para el control de versiones.

1.11.2. Sublime Text 3.05

Sublime Text es un editor de texto y editor de código fuente escrito en C++. Se distribuye de forma gratuita, sin embargo no es software libre o de código abierto, se puede obtener una licencia para su uso ilimitado, pero el no disponer de ésta no genera ninguna limitación más allá de una alerta cada cierto tiempo.

Características principales

- **Minimapa:** Consiste en una visualización previa de la estructura del código, es muy útil para desplazarse por el archivo cuando se conoce bien la estructura de este.
- **Multi selección:** Hace una selección múltiple de un término por diferentes partes del archivo.
- **Multi cursor:** Crea cursores con los que es posible escribir texto de forma arbitraria en diferentes posiciones del archivo.
- **Multi *layout*:** Trae siete configuraciones de plantilla y permite elegir editar en una sola ventana o hacer una división de hasta cuatro ventanas verticales o cuatro ventanas en cuadrícula.
- **Soporte nativo para infinidad de lenguajes:** Soporta de forma nativa 43 lenguajes de programación y texto plano.
- ***Syntax highlight* configurable:** El remarcado de sintaxis es completamente configurable a través de archivos de configuración del usuario.
- **Búsqueda dinámica:** Se puede hacer búsqueda de expresiones regulares o por archivos, proyectos, directorios, una conjunción de ellos o todo a la vez.
- **Auto completado y marcado de llaves:** Se puede ir a la llave que cierra o abre un bloque de una forma sencilla.
- **Soporte de *snippets* y *plugins*:** Los *snippets* son similares a las macros o los *bundles* además de la existencia de multitud de *plugins*.
- **Configuración total de *keybindings*:** Todas las teclas pueden ser sobrescritas a nuestro gusto.

- **Acceso rápido a línea o archivo:** Se puede abrir un archivo utilizando el conjunto de teclas Ctrl+P y escribiendo el nombre del mismo o navegando por una lista. También se puede ir a una línea utilizando los dos puntos ":" y el número de línea.
- **Paleta de comandos:** Un intérprete de Python diseñado solo para el programa, con el cual se puede realizar infinidad de tareas.
- **Coloreado y envoltura de sintaxis:** Si se escribe en un lenguaje de programación o marcado, resalta las expresiones propias de la sintaxis de ese lenguaje para facilitar su lectura.
- **Pestañas:** Se pueden abrir varios documentos y organizarlos en pestañas.
- **Resaltado de paréntesis:** Cuando el usuario coloca el cursor en un paréntesis, corchete o llave, resalta esta y el paréntesis, corchete o llave de cierre o apertura correspondiente.

¿Por qué Sublime Text?

Para el desarrollo del sistema informático “Le Atiendo” se ha utilizado el sofisticado editor de código multiplataforma Sublime Text, al ser una herramienta concebida para programar sin distracciones. Su interfaz de color oscuro y la riqueza de coloreado de la sintaxis, centra nuestra atención completamente. Su interfaz es limpia e intuitiva, es un editor ligero, rápido y fácilmente configurable además de tener la capacidad de ampliar sus funcionalidades a través de *plugins*.

1.12. Sistema gestor de base de datos

Un sistema gestor de base de datos (SGBD) es un conjunto de programas que permiten crear y mantener una base de datos (BD) asegurando su integridad, confidencialidad y seguridad.

Por tanto debe permitir:

- Definir una base de datos: especificar tipos, estructuras y restricciones de datos.
- Construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD.
- Manipular la base de datos: realizar consultas, actualizarla, generar informes.

1.12.1. PostgreSQL

Es un sistema de gestión de base de datos objeto relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es un SGBD extensible y multiplataforma. 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.

A continuación se exponen algunas de las características más importantes de PostgreSQL:

- Es una base de datos segura.
- Tiene integridad referencial.
- Tiene múltiples métodos de autentificación.
- Acceso encriptado vía SSL.
- Provee actualización integrada momentánea (pg_upgrade).
- Posee documentación completa.
- Interfaz con diversos lenguajes (C, C++, Java, Delphi, Python, Perl, PHP, Bash entre otros).

1.12.2. MySQL 5.6.12

MySQL (My Structured Query Language o Lenguaje de Consulta Estructurado) es un sistema de gestión de base de datos relacional, multihilo y multiusuario que por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Frente a sus adversarios presenta frecuentes ventajas, que bien pueden ser la razón por la cual es la base de datos de código fuente abierto más usada del mundo.

Ventajas

- MySQL es *open source.*
- Velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento.
- Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema.
- Facilidad de configuración e instalación.
- Soporta gran variedad de sistemas operativos.
- Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está.
- Su conectividad, velocidad, y seguridad hacen de MySQL Server altamente apropiado para acceder bases de datos en Internet.
- El software MySQL usa la licencia GPL.

¿Por qué MySQL?

Para la gestión de la base de datos del sistema informático “Le Atiendo” se selecciona MySQL al ser uno de los gestores de código abierto más potentes del mercado, además por las características de su fácil manejo y las ventajas anteriormente expuestas. Igualmente se tuvo en cuenta que es el gestor que utilizan otros sistemas heredados de la DSTI.

Coclusiones parciales

En el presente capítulo se han dado a conocer los conceptos fundamentales relacionados con la compresión de la investigación, se realizó un breve estudio sobre los sistemas gestores de reportes, su desarrollo a nivel mundial y en el ámbito nacional. A partir de los resultados del mismo, se llegó a la conclusión de que los sistemas analizados no proporcionan las funcionalidades requeridas por lo que no representan una solución a las necesidades existentes de la gestión de reportes de roturas o gestión de incidencias, como es también llamada mundialmente, adaptada a la función de la DSTI UCI. Se realizó además un análisis de las técnicas actuales de programación, tendencias de la programación web, algunas técnicas y metodologías utilizadas para la confección del sistema, así como de los *frameworks* PHP y se definieron algunas de las ventajas que brinda el *framework* seleccionado para la implementación.

Capítulo 2: Exploración, planificación y diseño del sistema

Introducción

En el presente capítulo, se realizará todo el modelamiento de la información del sistema para la gestión de reportes de roturas en la DSTI UCI, este consiste en la definición de los objetivos del sitio, la organización de los contenidos y los servicios que se brindarán, especificándose los requisitos funcionales y no funcionales que debe cumplir, así como la estructura y diseño del mismo. También se hará alusión a las fases de exploración, planificación y desarrollo, las primeras de la metodología de desarrollo XP. El objetivo de estas es conocer el alcance del producto a desarrollar, así como estimar los tiempos de entrega de cada versión. En este capítulo se exponen, además, los artefactos que se generan a partir de los requerimientos expuestos por el cliente.

2.1. Modelo conceptual

Para complementar el entendimiento del problema se hace uso de un modelo conceptual, aunque la metodología XP no propone artefactos en este sentido, es flexible y puede contrastarse con algunos que simplifiquen y apoyen el proceso, que esclarezcan los conceptos referentes al problema antes mencionado.

Un modelo conceptual captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que trabaja el sistema.

Los principales conceptos identificados en este dominio fueron:

- **reportador:** Persona encargada de realizar un reporte.
- **procesador:** Persona encargada de insertar los reportes de roturas y procesar sus datos en los sistemas contables de la DSTI UCI.
- **técnico:** Persona que recibe los reportes que le son asignados por el jefe de taller y les da solución a los mismos.
- **jefe_taller:** Jefe de un taller de ST, encargado de asignar a los técnicos los reportes recibidos por su taller.
- **taller:** Hace referencia a los diferentes talleres de ST existentes en la DSTI UCI.
- **brigada:** Hace referencia a las diferentes brigadas existentes en los talleres de ST.
- **equipo:** Se refiere a equipos que son atendidos por la DSTI UCI.

- **tipo_equipo:** Se refiere a los tipos de equipos que son atendidos por la DSTI UCI.
- **marca:** Se refiere a las marcas de los equipos que son atendidos por la DSTI UCI.
- **cliente:** Son los clientes de la DSTI UCI a los cuales pertenecen los reportadores.
- **dirección:** Hace referencia a los diferentes locales pertenecientes a cada cliente.
- **reporte:** Es un reporte realizado por un reportador sobre cualquier situación fuera de lo normal en un equipo determinado. Permite informar el estado técnico de un equipo en específico.

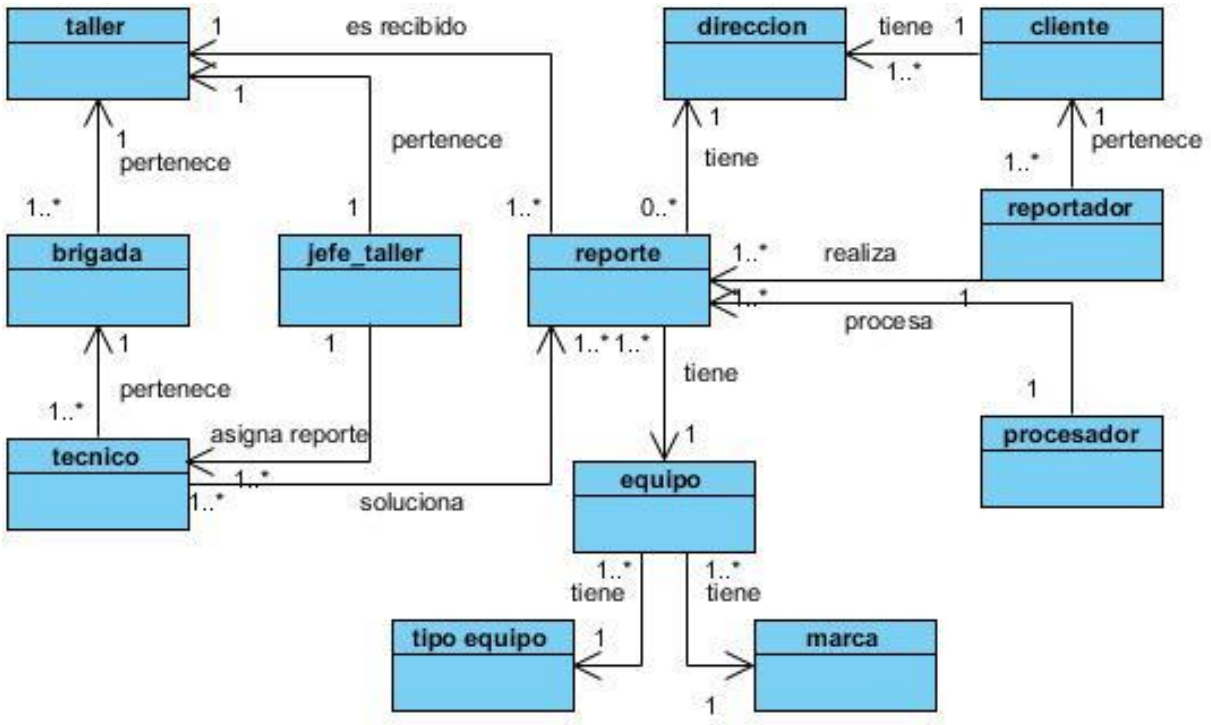

Fig. 6: Modelo conceptual

La figura 6 muestra un modelo conceptual del sistema donde se refleja el funcionamiento del proceso de gestión de reportes de roturas. A través de las relaciones entre las clases se define el flujo de este proceso. Los números en las relaciones indican la cantidad de instancias de una clase a otra siguiendo la dirección de las relaciones.

2.2. Propuesta del sistema

Teniendo en cuenta los requerimientos planteados y como medio de cumplimiento a los objetivos propuestos inicialmente se ha concebido como propuesta de solución a la problemática la implementación de un sistema

informático que garantice la gestión de los reportes de roturas en la DSTI UCI, el mismo se encontrará brindando servicios sobre las redes de la UCI y la DSTI UCI, a través de las cuales todos los nodos tendrán la posibilidad de acceder al mismo.

El sistema proporcionará una interfaz web como forma de presentación al usuario, la cual tendrá una página de bienvenida que ofrecerá información general sobre el sistema y desde la cual los usuarios se podrán autenticar. El usuario autenticado puede ser: de tipo usuario, jefe de taller, técnico, procesador, administrativo y administrador del sistema.

Una vez autenticado el usuario, el sistema le proporcionará de acuerdo a su rol ciertos privilegios y permisos que le permitirán tener acceso a las diferentes funcionalidades del sistema. Para ello el sistema contará con la existencia de 10 roles, los cuales son: usuario, reportador, reportador ofimática, reportador telecomunicaciones, reportador electromecánica, técnico, jefe de taller, procesador, directivo y administrador del sistema.

El sistema incluirá todo el proceso de gestión de reportes de roturas de equipamiento tecnológico así como las intervenciones de usuarios en el mismo y el manejo de la información pertinente. Cada usuario podrá generar informes donde se recoja información en dependencia del nivel de acceso que posea. De manera general el sistema permitirá al administrador la gestión de equipos, tipos de equipos, marcas, talleres, clientes, direcciones, estados, roles, brigadas y usuarios.

En resumen, con esta propuesta de sistema se facilitará la centralización, manejo y control de la información del proceso gestión de reportes de roturas de la DSTI UCI de una forma estándar y organizada.

2.2.1. Personas relacionadas con el sistema

Como elemento indispensable a tener en cuenta cuando se comienza el desarrollo de un sistema informático es delimitar la audiencia a la cual va dirigido el mismo. Teniendo en cuenta que la misma a su vez puede ser dividida en grupos atendiendo a sus necesidades.

Dentro de la audiencia a interactuar con el sistema a desarrollar se incluyen todas aquellas personas que obtienen un resultado de valor de al menos uno de los procesos que se ejecutarán en el mismo.

Tabla 3. Personas relacionadas con el sistema

Personas relacionadas con el sistema	Justificación
reportador	Es la persona que tiene la posibilidad de crear un reporte de rotura llenando los datos requeridos así como consultar información y estadísticas de sus reportes realizados.
jefe de taller	Es la persona que podrá acceder a los reportes realizados concernientes a su taller así como asignarlos a las distintas brigadas o técnicos que los vayan a atender. Además podrá consultar información y estadísticas de los reportes referentes a su taller.
técnico	Es la persona que podrá acceder a los reportes que le son asignados por su jefe de taller así como cambiarle el estado a los mismos una vez solucionados. Además podrá consultar información y estadísticas de los reportes que le han sido asignados.
procesador	Es la persona facultada para visualizar los reportes una vez que han sido asignados a un técnico para insertarlos en los sistemas contables de la DSTI UCI y se encarga de cambiarle el estado a “en proceso”. Una vez terminada la reparación cambia el estado del reporte a “cerrado”.
administrativo	Es la persona que tiene permisos para visualizar los reportes que son realizados por el usuario. Además de consultar información y estadísticas de los reportes recibidos.
administrador	Es la persona facultada para la gestión del sistema en general. Podrá gestionar la administración de usuarios, equipos y demás elementos del sistema.

2.3. Especificación de los requisitos de software

Los requisitos de software en general, son condiciones o capacidades que necesitan los usuarios para resolver un problema o alcanzar un objetivo. Los mismos se pueden clasificar en requisitos funcionales y no funcionales. A continuación se muestra la captura de requisitos realizada para desarrollar la solución propuesta.

2.3.1. Requerimientos funcionales

Los requerimientos funcionales especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física, de manera que especifican el comportamiento de entrada y salida del sistema y surgen de la razón fundamental de la existencia del producto. A partir de los procesos de dominio estudiados y las actividades a automatizar identificadas se pueden definir los siguientes requisitos funcionales:

RF 1. Autenticar usuario
RF 2. Crear usuario
RF 3. Buscar usuario
RF 4. Modificar usuario
RF 5. Eliminar usuario
RF 6. Crear rol
RF 7. Buscar rol
RF 8. Modificar rol
RF 9. Eliminar rol
RF 10. Crear brigada
RF 11. Buscar brigada
RF 12. Modificar brigada
RF 13. Eliminar brigada
RF 14. Crear técnico
RF 15. Buscar técnico
RF 16. Modificar técnico
RF 17. Eliminar técnico
RF 18. Crear cliente
RF 19. Buscar cliente
RF 20. Modificar cliente
RF 21. Eliminar cliente
RF 22. Crear dirección
RF 23. Buscar dirección
RF 24. Modificar dirección
RF 25. Eliminar dirección
RF 26. Crear reporte
RF 27. Buscar reporte
RF 28. Modificar reporte
RF 29. Eliminar reporte
RF 30. Crear estado

RF 31. Buscar estado
RF 32. Modificar estado
RF 33. Eliminar estado
RF 34. Crear taller
RF 35. Buscar taller
RF 36. Modificar taller
RF 37. Eliminar taller
RF 38. Crear equipo
RF 39. Buscar equipo
RF 40. Modificar equipo
RF 41. Eliminar equipo
RF 42. Crear tipo de equipo
RF 43. Buscar tipo de equipo
RF 44. Modificar tipo de equipo
RF 45. Eliminar tipo de equipo
RF 46. Crear marca
RF 47. Buscar marca
RF 48. Modificar marca
RF 49. Eliminar marca
RF 50. Cambiar estado de reporte
RF 51. Asignar reporte a técnico
RF 52. Mostrar historial de equipo
RF 53. Mostrar reportes realizados por dirección
RF 54. Mostrar reportes realizados por técnico
RF 55. Mostrar reportes realizados por taller
RF 56. Mostrar reportes realizados por usuario
RF 57. Mostrar reportes realizados por procesador
RF 58. Mostrar reportes realizados por brigadas
RF 59. Mostrar reportes realizados por fecha
RF 60. Mostrar reportes por estados

RF 61. Mostrar reportes por tipo de equipo

RF 62. Imprimir informe

RF 63. Exportar informe

2.3.2. Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Existen múltiples categorías para clasificarlos, siendo las siguientes representativas de un conjunto de aspectos que se deben tener en cuenta, aunque no limitan a la definición de otros.

Requerimiento de hardware en servidores

La solución estará conformada, fundamentalmente, por servidores de alta capacidad de procesamiento y redundancia, que permitan garantizar movilidad y residencia de la información y las aplicaciones bajo esquemas seguros y confiables.

RnF 1. El servidor de base de datos debe tener como mínimo: un procesador a 1,80 GHZ *DualCore*, 2GB de memoria RAM y 1TB de disco duro.

RnF 2. El servidor para la instalación del sistema informático “Le Atiendo” debe tener como mínimo: un procesador a 1,80 GHZ *DualCore* y 2GB de memoria RAM.

Requerimiento de hardware en estaciones de trabajo

RnF 3. Las estaciones de trabajo cliente deben tener como mínimo 256Mb de memoria RAM y un microprocesador de 2.0Hz.

Usabilidad

RnF 4. El sistema deberá estar diseñado de manera que los usuarios adquieran las habilidades necesarias para explotarlo en un tiempo reducido.

RnF 5. Facilidad de uso de parte de los desarrolladores de software.

RnF 6. El sistema debe ser fácil de extender por parte de los desarrolladores.

Seguridad

RnF 7. La información manejada por el sistema contará con protección ante intrusos y accesos no autorizados, será vista únicamente por aquellos usuarios que tengan derecho a verla.

RnF 8. El sistema controlará los diferentes niveles de acceso y funcionalidad de usuarios al sitio, o sea, prioriza la identificación del usuario antes de que sea capaz de realizar cualquier acción sobre el sistema.

Rendimiento

RnF 9. El sistema deberá dar respuesta de menos de 3000 milisegundos para las operaciones que impliquen reportes y de menos de 1500 milisegundos para las operaciones de almacenamiento o eliminación.

Software

RnF 10. El servidor debe correr en sistemas operativos Windows, Unix y Linux.

RnF 11. Los clientes deberán disponer de un navegador web, estos pueden ser IE 7 o superior, Opera 9 superior, Google Chrome 1 o superior y Firefox 2 o superior.

Interfaz de usuario

RnF 12. El diseño de las interfaces que se propone para el sistema a implementar está basado en la premisa de organización y sencillez de la información a mostrar.

2.4. Flujos de trabajo para el desarrollo del sistema

2.4.1. Exploración

El ciclo de vida ideal de un proyecto realizado con XP se inicia con la fase de Exploración, al final de la cual el equipo de desarrollo cuenta con suficiente material de trabajo traducido en historias de usuario, que se recopilan en esta etapa, como para producir una primera entrega. Además se produce el contacto necesario con las herramientas y tecnologías que se emplearán para construir el sistema y algunas ideas experimentales en cuanto a su arquitectura son consideradas. Se pueden explorar soluciones puntuales, también, cuando no se tenga una concepción transparente de alguna funcionalidad.

Historias de usuario

Una de las mejores prácticas adoptadas en el desarrollo de software es la administración de requerimientos. XP propone en este sentido hacer uso de las historias de usuario (HU) como técnica para especificar las funcionalidades que brindará el sistema y constituye una manera muy dinámica de realizar esta actividad. Como su nombre lo indica son especificadas por los propios usuarios y por tanto redactadas en su lenguaje; de manera sencilla y breve, evitando tecnicismos innecesarios que puedan crear confusión, aunque los programadores pueden contribuir en la tarea. Además son la base para realizar las pruebas de aceptación,

así como la estimación y planificación del proyecto. Una vez identificadas las historias de usuario necesarias para liberar una primera versión operativa, los programadores proceden a descomponer cada una en tareas específicas, las denominadas tareas de programación que están escritas técnicamente, que darán solución a la historia correspondiente. A continuación se muestra la historia de usuario Gestionar reporte:

El resto de las HU identificadas se pueden ver en los Anexos (Anexo 4).

Tabla 4. HU Gestionar reporte

Historia de usuario	
Número: 8	**Nombre:** Gestionar reporte
Usuario: reportador y administrador	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Alto	
Descripción: La HU se inicia cuando se selecciona la opción reportes. Se muestra un listado de todos los reportes existentes, si el usuario selecciona la opción: • Crear reporte: el sistema muestra una vista con los campos que deben ser llenados, luego de introducir los datos, se muestra un mensaje de confirmación "El reporte ha sido insertado satisfactoriamente" quedándose la vista activada brindando la posibilidad de crear otro reporte. • Modificar reporte: luego de seleccionar el reporte a modificar, el sistema muestra una vista con los datos del reporte, permitiendo su modificación, después de modificar los datos deseados el sistema muestra un mensaje de confirmación "El reporte ha sido modificado satisfactoriamente" y muestra el listado de los reportes actualizado. • Eliminar reporte: después de seleccionar el reporte a eliminar, se elige en la opción eliminar y seguido de esto se muestra un mensaje de confirmación "El reporte ha sido eliminado satisfactoriamente".	
Observaciones: Existe un administrador general del sitio. Para modificar o eliminar un reporte primero se debe seleccionar éste en el listado de reportes. Cuando se realiza un reporte este obtiene el estado "abierto". Solo puede modificar o eliminar un reporte el usuario que lo crea, si el reporte no está "asignado" aun.	

2.4.2. Planificación y entrega

A lo largo de esta fase los clientes establecen la prioridad de las historias de usuario de acuerdo a sus necesidades más inmediatas para luego asignarlas, por orden de relevancia, a las iteraciones planificadas. El resultado de esta fase es un plan de entregas. En dependencia de la prioridad establecida por el cliente,

los programadores realizan una estimación del esfuerzo necesario para la posterior implementación de cada una de las HU. El método escogido para realizar la estimación tiene como elemento el punto, el cual equivale a una semana perfecta de trabajo, esto se refiere a que solamente el equipo se dedica a labores relacionadas con la construcción del sistema sin la influencia o el retraso provocado por otros factores, lo que en la práctica es complejo lograr.

Plan de entrega

Una vez terminada la elaboración de las HU, se prosigue con la creación del plan de entregas. Este plan se realiza en base a las estimaciones de esfuerzos de desarrollo realizadas por los programadores. En este se recogen las historias de usuario que serán agrupadas por prioridad de acuerdo a sus necesidades más inmediatas para conformar una entrega.

Tabla 5. Plan de entrega

Historias de usuario	Final 1ra Iteración 28/11/2014	Final 2da Iteración 29/01/2015
Gestionar brigada Gestionar técnico Gestionar cliente Gestionar dirección Gestionar reporte Gestionar estado Gestionar taller Gestionar equipo Gestionar tipo de equipo Gestionar marca	10 semanas	Finalizado
Cambiar estado de reporte Asignar reporte a técnico Visualizar informe Imprimir informe Exportar informe Autenticar usuario Gestionar usuario Gestionar rol	No empezado	8 semanas

Estimación de esfuerzos por historias de usuario

Las estimaciones del esfuerzo para implementar las historias de usuario permiten tener una medida bastante real de la velocidad de progreso del proyecto y brindan una guía razonable a la cual ajustarse. Los resultados estimados se exhiben seguidamente:

Tabla 6. Estimación de esfuerzos por historias de usuario

No.	Historias de usuario	Puntos de estimación
1	Autenticar usuario	1
2	Gestionar usuario	1
3	Gestionar rol	1
4	Gestionar brigada	1
5	Gestionar técnico	1
6	Gestionar cliente	1
7	Gestionar dirección	1
8	Gestionar reporte	1
9	Gestionar estado	1
10	Gestionar taller	1
11	Gestionar equipo	1
12	Gestionar tipo de equipo	1
13	Gestionar marca	1
14	Cambiar estado de reporte	1
15	Asignar reporte a técnico	1
16	Visualizar informe	1
17	Imprimir informe	1
18	Exportar informe	1

Plan de iteraciones

En este plan se agrupan las historias de usuario que serán implementadas y probadas en cada iteración, de acuerdo al orden establecido.

Iteración 1

El propósito de esta primera iteración es implementar las historias de usuario de mayor importancia (prioridad alta) para el cliente, necesarias para comenzar la implementación de la iteración posterior. Estas

HU son: Gestionar brigada, Gestionar técnico, Gestionar cliente, Gestionar dirección, Gestionar reporte, Gestionar estado, Gestionar taller, Gestionar equipo, Gestionar tipo de equipo y Gestionar marca.

Iteración 2

Esta iteración tiene como objetivo implementar las historias de usuario de menor importancia (en este caso las de prioridad media) para el cliente, las cuales son: Cambiar estado de reporte, Asignar reporte a técnico, Visualizar informe, Autenticar usuario, Gestionar usuario y Gestionar rol. Las implementaciones anteriores junto con la versión de prueba de esta iteración, serán mostradas al cliente para realizar posibles cambios según criterios del mismo. Terminada esta iteración, se dispondrá de la versión 1.0 del producto final con todas las funcionalidades descritas por el cliente. Luego el sistema es puesto en funcionamiento durante un período de tiempo para evaluar su desempeño.

Plan de duración de las iteraciones

Como parte del ciclo de vida de desarrollo de un producto de software guiado por la metodología XP, se crea un plan de duración de cada una de las iteraciones, así como el orden en que serán implementadas las HU en cada una de las mismas. Este plan ayuda a tener una noción aproximada del tiempo que durará el desarrollo del sistema en su totalidad. A modo de resumen se presenta la siguiente tabla que muestra las 2 iteraciones analizadas previamente con las historias de usuario que incluyen y su duración:

Tabla 7. Plan de duración de las iteraciones

Iteraciones	Historias de usuario a implementar	Duración
Iteración 1	Gestionar brigada Gestionar técnico Gestionar cliente Gestionar dirección Gestionar reporte Gestionar estado Gestionar taller Gestionar equipo Gestionar tipo de equipo Gestionar marca	10 semanas

Iteración 2	Cambiar estado de reporte Asignar reporte a técnico Visualizar informe Imprimir informe Exportar informe Autenticar usuario Gestionar usuario Gestionar rol	8 semanas

2.5. Diseño del sistema

XP establece prácticas especializadas que inciden directamente en la realización del diseño para lograr un sistema robusto y reutilizable tratando de mantener su simplicidad, es decir, crear un diseño evolutivo que se va mejorando incrementalmente y que permite hacer entregas pequeñas y frecuentes de valor para el cliente. A la hora de darle cumplimiento a la actividad de diseñar, XP no especifica ninguna técnica de modelado, puede utilizarse indistintamente sencillos esquemas en una pizarra, diagramas de clases utilizando UML o tarjetas CRC (Clase, Responsabilidad y Colaboración) siempre que sean útiles, tributen a la comprensión y no requieran mucho tiempo en su creación.

2.5.1. Tarjetas CRC

Las características más sobresalientes de las tarjetas CRC son su simpleza y ductilidad. Una tarjeta CRC no es más que una ficha de papel o cartón que representa a una entidad del sistema (21).

Tabla 8. Modelo de tarjeta CRC

clase	
Responsabilidades	Colaboradores

A continuación se muestra la tarjeta CRC reporte, el resto de las tarjetas CRC identificadas se pueden ver en los Anexos (Anexo 5).

Tabla 9. Tarjeta CRC Reporte

Reporte	
Crear reporte Buscar reporte Listar reporte	Reporte Equipo TipoEquipo

Modificar reporte Eliminar reporte	Marca Tecnico Estado ReporteDireccion

2.5.2. Descripción de arquitectura

Model-view-controler o modelo-vista-controlador (MVC) es una arquitectura que integra los tres niveles de diseño siguientes:

- El modelo representa la información con la que trabaja el sistema, es decir, su lógica de dominio.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

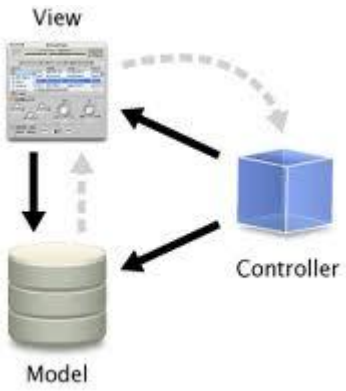

Fig. 7: Modelo Vista Controlador

Como se aprecia en la figura 7 la arquitectura MVC separa la lógica de dominio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, entre otros). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación (22).

2.5.3. Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño. La misma tiene la característica principal de ser capaz de resolver problemas similares. Debe ser reutilizable, o sea que es aplicable a diferentes problemas de diseño en distintos contextos. Su uso es fundamental para un correcto diseño del software. Ayudan a la reutilización de diseño gráfico, identificando aspectos claves de la estructura de un diseño que puede ser aplicado en una gran cantidad de situaciones. Lo que supone una gran ventaja ya que disminuye los esfuerzos de desarrollo y mantenimiento, mejora la seguridad informática, eficiencia y consistencia del diseño, y proporciona un inmenso ahorro en la inversión. También ayudan a tener un software más flexible, modular y extensible (23). A continuación se describen los patrones utilizados:

Patrón Decorador

Existen algunas formas de evitar la repetición de elementos comunes de las páginas. Entre estos se encuentran la cabecera y el pie, ya que estos elementos no contienen código HTML válido. Para darle solución a este problema se pueden emplear patrones de diseño entre los que se encuentra el patrón decorador.

En el *framework* Symfony el patrón Decorador está implementado de forma tal que todas las páginas mostradas al usuario se componen de un *Layout*, el cual agrupa los elementos comunes, como los banner, menús, *footers*, entre otros, y una página conocida como plantilla que es la encargada de mostrar o recoger la información en dependencia de la acción del usuario. En la figura 8 se muestra un ejemplo de cómo se utiliza el patrón Decorador en el sistema propuesto.

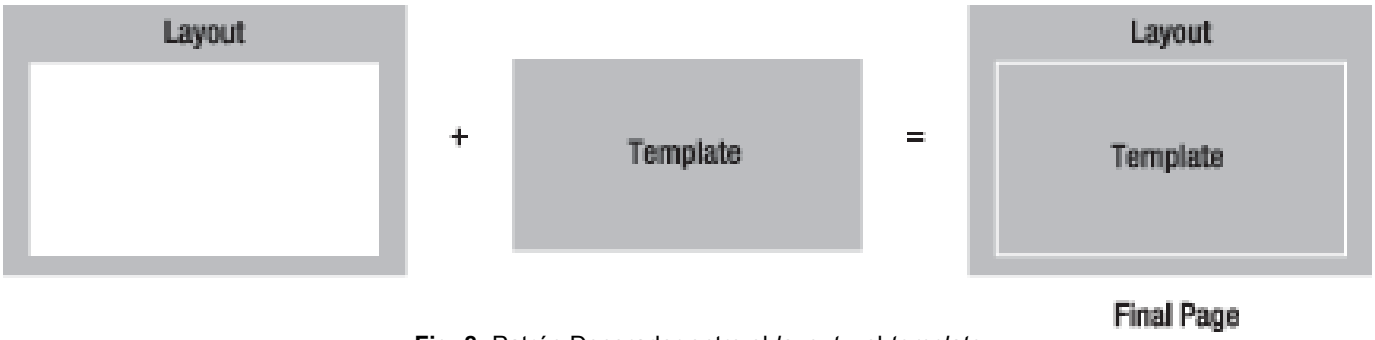

Fig. 8: Patrón Decorador entre el *layout* y el *template*

Patrón Cadena de Responsabilidad

La ejecución de la secuencia de filtros se basa en el patrón *Chain of Responsibility* (Cadena de Responsabilidad) que permite establecer una cadena de objetos receptores a través de los cuales se pasa

una petición formulada por un objeto emisor. Cualquiera de los objetos receptores puede responder a la petición en función de un criterio establecido, posibilitando que más de un objeto pueda manejar una petición. En la figura 9 se muestra un ejemplo de su funcionamiento.

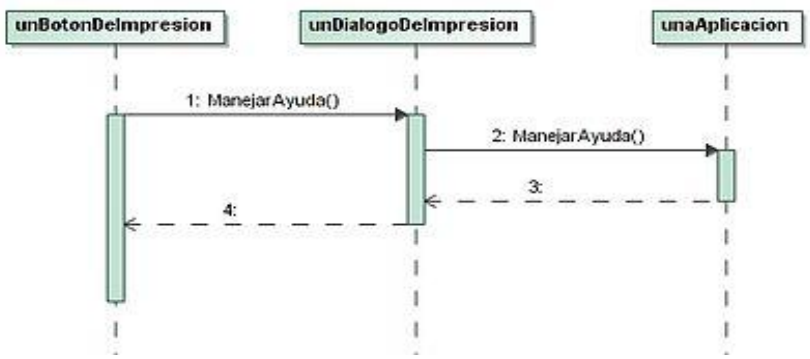

Fig. 9. Secuencia de filtros en Symfony

Patrón Controlador Frontal

Un patrón que es utilizado implícitamente en las aplicaciones desarrolladas con el *framework* de desarrollo Symfony es el patrón Controlador Frontal. Consiste en la existencia de un único controlador en el sistema que soluciona el problema de la descentralización presente en el patrón MVC. Es el único punto de entrada al sistema y realiza las siguientes operaciones:

1. Define las constantes del núcleo.
2. Localiza las librerías de Symfony.
3. Carga e inicializa las clases del núcleo del *framework*.
4. Carga la configuración.
5. Decodifica la URL de la petición para determinar la acción a ejecutar y los parámetros de la petición.
6. Si la acción no existe, redirecciona a la acción error 404.
7. Activa los filtros.
8. Ejecuta los filtros, primera pasada.
9. Ejecuta la acción y produce la vista.
10. Ejecuta los filtros, segunda pasada.
11. Muestra la respuesta.

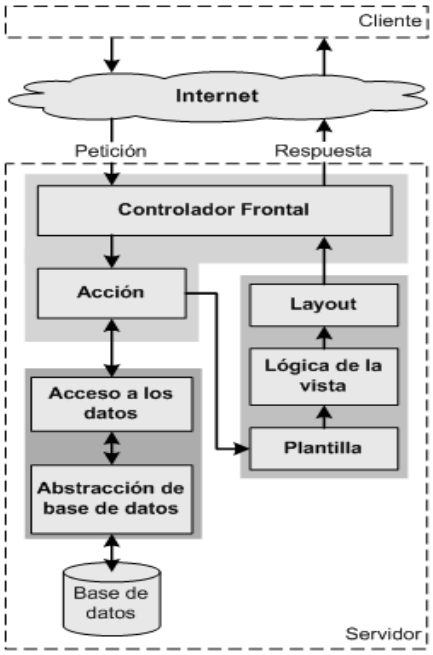

Fig. 10: Flujo básico de una petición a Symfony

2.5.4. Diseño de base de datos

En cualquier sistema en la cual se gestione información la base de datos desempeña un papel fundamental, y sobre todo para las aplicaciones web como es el caso de esta, por lo que se hace necesario que los datos se almacenen de forma coherente y organizada, para evitar que dicha información se pierda. Es por esto que se realiza el diseño de la base de datos en cuestión, para lograr la persistencia de los datos de los usuarios, así como la información de los recursos que se gestionan por los mismos. Mediante un diagrama Entidad-Relación se modela todo lo referente a la disposición de la base de datos que se ha creado. En el Anexo 6 se muestran las descripciones de las clases del sistema.

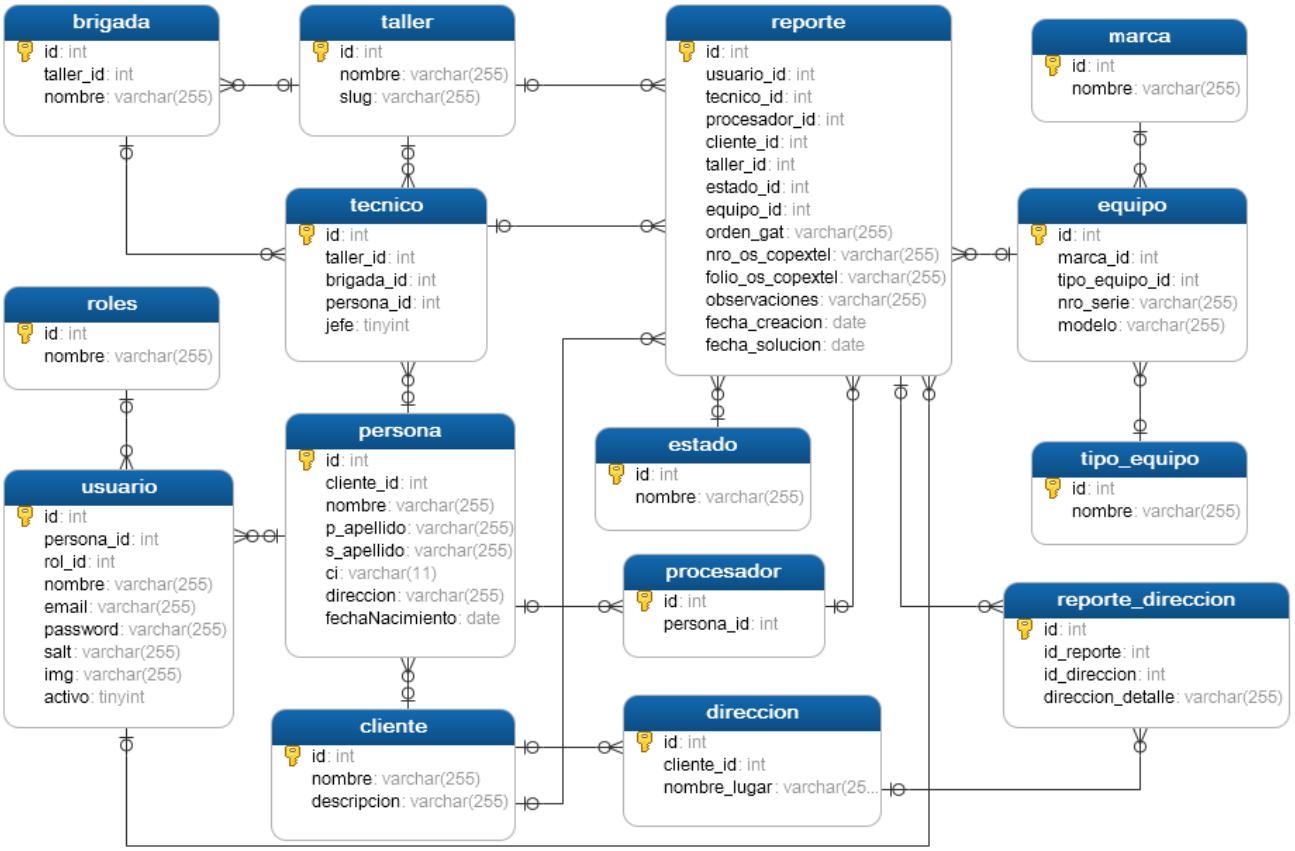

Fig. 11: Diagrama Entidad-Relación

2.5.5. Diagrama de despliegue

El diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación que muestra las relaciones físicas entre los componentes de hardware que forman la topología sobre la que se ejecuta el sistema y la distribución de sus partes. Este refleja tanto la distribución del sistema con los nodos físicos (ordenadores) como la correspondencia que tienen los componentes con los nodos (24). En el sistema a desarrollar se pueden identificar como procesadores uno para la base de datos, otro para que funcione como servidor web y uno que representa en general todos los clientes que puedan conectarse al sistema. Los nodos que representan el servidor de base de datos y el servidor web deben ser capaces de brindar una serie de prestaciones de hardware para cumplir con los requerimientos.

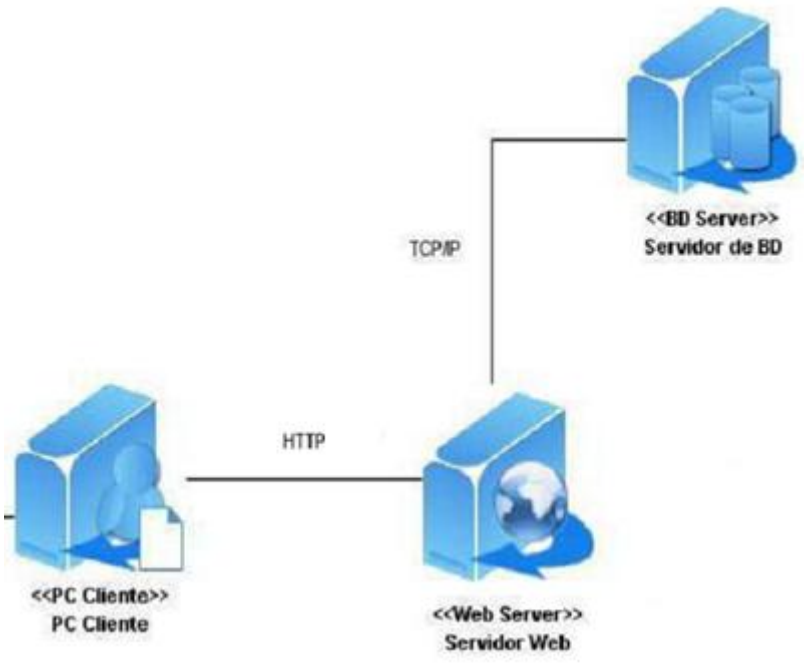

Fig. 12: Diagrama de despliegue

Coclusiones parciales

En el capítulo se describió la propuesta del sistema que se desea implementar para una mejor comprensión del mismo, definiéndose los requisitos funcionales y no funcionales y obteniendo el plan de entrega como parte de la fase de planificación. Quedaron plasmadas las iteraciones por las que transitará el sistema, señalando las historias de usuario que serán implementadas en cada una de ellas. Como parte del diseño, se realizó la descripción de las principales clases a través de las tarjetas CRC garantizando posteriormente una mejor calidad del código en la fase de implementación y se definieron los patrones arquitectónicos y de diseño a utilizar. Además se definió el diseño de la base de datos y se muestra la vista de despliegue en un ambiente real, indicando los elementos fundamentales y el protocolo de comunicación que se utiliza entre ellos.

Capítulo 3: Implementación y validación del sistema

Introducción

En la etapa de implementación de un software, se transforman las clases y objetos en ficheros fuente, binarios y ejecutables. El resultado final, es un sistema ejecutable que en la etapa de pruebas, es evaluada su calidad y desempeño como producto de software. En esta etapa se detectan y corrigen errores para la posterior aceptación del producto.

El presente capítulo abordará los temas relacionados con la implementación y pruebas realizadas al sistema. Se definen los estándares de codificación, que posibilitan la organización, limpieza y claridad en la escritura del código en el proceso de desarrollo del software y se describen los artefactos generados en estas fases.

3.1 Implementación del sistema

En esta fase se genera todo el código fuente necesario para satisfacer las historias de usuario definidas para la solución y se describen todas las tareas realizadas en cada iteración.

3.1.1. Tareas de ingeniería

Para el desarrollo de las iteraciones las historias de usuario son divididas en tareas de ingeniería. Estas tareas son escritas en fichas, en un lenguaje técnico para el uso de los programadores, donde cada una debe ser estimada para un período de uno a tres días de desarrollo (25).

Tabla 10. Tareas genéricas

Tareas
• Diseñar base de datos. • Generar las clases modelo. • Implementar los métodos en las clases controladoras. • Sincronizar con la base de datos. • Definir sistema de enrutamiento. • Definir componentes para la interfaz. • Exportar datos mostrados. • Imprimir datos mostrados.

Tabla 11. Tareas según HU

Historia de usuario	Tareas
Visualizar informe	• Consultar en BD según parámetros Introducidos por el usuario. • Crear plantilla para informes.
Autenticar usuario	• Crear plantilla para formulario de autenticación. • Validar los campos del formulario. • Verificar los datos insertados. • Definir los privilegios del usuario

Las tareas de ingeniería están descritas en el Anexo 7.

3.1.2. Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código (26).

Seguir un determinado estilo de codificación permite a los programadores revisar, mantener y actualizar el una manera sencilla y ordenada, evitando que incurran en errores y malas prácticas que dificultan la comprensión de las líneas de código. A continuación se muestran los estándares definidos para la implementación del sistema informático “Le Atiendo”.

Declaraciones: Los nombres utilizados en las declaraciones deben estar acorde con su significado en el sistema.

Atributos: Deben comenzar con letra minúscula, si son atributos compuestos comienzan con minúscula y la segunda palabra con mayúscula.

Ejemplos:
atributo simple: string reporte
atributo compuesto: string reporteDireccion.

Métodos: Comienzan con letra minúscula, si son compuestos comienzan con minúscula y la segunda palabra con mayúscula.

Ejemplos:
simple: public function Adicionar($id)

compuesto: public function existeEquipo($nombre)

Clases: Deben comenzar con letra mayúscula.

Comentarios: Para comentar una línea se utilla //, si es un bloque de texto se comienza con /* y se termina con */.

Instrucciones: Las llaves que se utilizan para abrir y cerrar un bloque de instrucciones deben estar al mismo nivel.

Variables de control de ciclos: Las variables utilizadas para los ciclos serán i, j, k.

3.1.3. Estilo de codificación

Existen varios estilos reconocidos por equipos que se dedican al desarrollo de software, durante la implementación se utilizó el estilo K&R ya que es el más usado en el lenguaje C y PHP. Se trata de abrir la llave en la misma línea de declaración de la orden, indentando los siguientes pasos al mismo nivel que la llave y cerrando la llave en el mismo nivel que la declaración (27).

```
function saludar($val){
    if ($val == 1) {
        echo "Hola!";
    } else {
        echo "Chao!";
    }
}
```

3.2. Pruebas de software

Las pruebas son un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente, por esta razón se deben definir en el proceso de ingeniería del software. Estas contribuyen a elevar la calidad de los productos desarrollados y a la seguridad de los programadores a la hora de introducir modificaciones (28).

La metodología XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores, encargadas de verificar el código de forma automática y las pruebas de aceptación,

destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente.

El método de prueba que se aplica es el de caja negra; permitiendo examinar los aspectos funcionales del sistema, haciendo mínimo enfoque en la estructura lógica interna del software. El sistema recibe un conjunto de entradas y produce salidas, a partir de las cuales se comprobará si el comportamiento es el esperado o no. Esta técnica aplica pruebas de aceptación para verificar que el software está listo y que puede ser usado por los usuarios finales para ejecutar las funciones para el cual fue concebido. Este tipo de prueba es recomendado por la metodología seleccionada XP, donde en colaboración con el cliente se diseñan para cada historia de usuario (29).

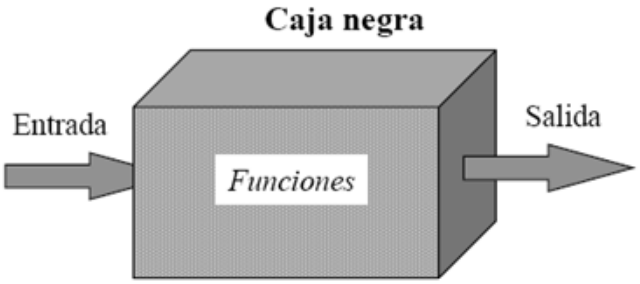

Fig. 13: Prueba de caja negra

3.2.1. Diseño de casos de pruebas

Los casos de prueba se diseñaron a partir de las historias de usuario de la propuesta de solución, el principal objetivo es comprobar que las HU han sido correctamente implementadas. Una historia de usuario puede tener tantos casos de prueba como sean necesarios para evaluar su funcionamiento. A continuación se muestran las iteraciones de pruebas elaboradas para el sistema informático “Le Atiendo”:

Iteración 1: La presente iteración de pruebas tiene como objetivo comprobar la correcta implementación de varias HU y detectar posibles no conformidades, la relación de HU involucradas en el proceso es la siguiente: Gestionar brigada, Gestionar técnico, Gestionar cliente, Gestionar dirección, Gestionar reporte, Gestionar estado, Gestionar taller, Gestionar equipo, Gestionar tipo de equipo y Gestionar marca.

Tabla 12. Prueba 1 de la HU 8

Caso de prueba de aceptación: 27
Historia de usuario: Gestionar reporte
Nombre: Insertar reporte

Responsable: Jeydis Blanco Almora
Descripción: Prueba de funcionalidad para la inserción de un reporte en el sistema.
Condiciones de ejecución: El usuario debe estar registrado previamente en el sistema con su permiso pertinente.
Entradas/ Pasos de ejecución: El usuario procede a la inserción de los datos necesarios de un reporte y se ejecuta la acción.
Resultado esperado: Los reportes son insertados sin generar error. En caso que el usuario omita algún dato o inserte algún dato no válido al crear un reporte el sistema informará al usuario que existe un dato erróneo especificándole el mismo.
Evaluación de la prueba: Prueba satisfactoria.

El resto de los casos de prueba de aceptación elaborados se muestran en el Anexo 8.

Iteración 2: La presente iteración de pruebas tiene como objetivo comprobar la correcta implementación de varias HU y detectar posibles no conformidades, la relación de HU involucradas en el proceso es la siguiente: Cambiar estado de reporte, Asignar reporte a técnico, Visualizar informe, Imprimir informe, Exportar informe, Autenticar usuario, Gestionar usuario y Gestionar rol.

Tabla 13. Prueba 1 de la HU 14

Caso de prueba de aceptación: 51
Historia de usuario: Cambiar estado de reporte
Nombre: Cambiar estado de reporte
Responsable: Jeydis Blanco Almora
Descripción: Probar la funcionalidad cambiar el estado a un reporte.
Condiciones de ejecución: El usuario debe estar autenticado y tener permisos para realizar estas operaciones.
Entradas/ Pasos de ejecución: Se intenta modificar el estado a un reporte pasando los datos correctamente.
Resultado esperado: Se muestra el reporte con el cambio realizado en el listado de reportes.
Evaluación de la prueba: Prueba satisfactoria.

El resto de los casos de prueba de aceptación elaborados se muestran en el Anexo 8.

Iteración 3: La presente iteración de pruebas tiene como objetivo comprobar la correcta implementación de todas las HU y detectar posibles no conformidades.

3.2.2. Ejecución de los casos de pruebas de aceptación

El proceso de pruebas a cualquier software se realiza a través de iteraciones, donde, a medida que se procede con una nueva iteración deben haberse erradicado los defectos encontrados en la anterior, para garantizar que al final del proceso el producto quede libre de la mayor cantidad de errores posible y listo para entregar al cliente.

Durante la ejecución de los casos de pruebas de aceptación algunas no arrojaron los resultados esperados. Se aplicaron un total de 55 casos de prueba que se distribuyeron en 3 iteraciones. A continuación se refleja en la Figura 12, la cantidad de casos de pruebas ejecutados y las no conformidades obtenidas por cada iteración.

Fig. 14: Totales de casos de prueba ejecutados y no conformidades detectadas por iteración

Iteración 1. Se ejecutaron un total de 46 casos de pruebas, detectándose un total de 5 no conformidades. A continuación se listan los casos de prueba de aceptación que arrojaron alguna no conformidad durante la ejecución de la primera iteración de pruebas:

1. Insertar reporte.
2. Modificar reporte.
3. Modificar estado.
4. Insertar equipo.

5. Modificar equipo.

Iteración 2. Se ejecutaron un total de 9 casos de pruebas, detectándose un total de 3 no conformidades. A continuación se listan los casos de prueba de aceptación que arrojaron alguna no conformidad durante la ejecución de la segunda iteración de pruebas:

1. Visualizar informe.
2. Insertar usuario.
3. Modificar usuario.

La plantilla de no conformidades constituye un registro de los defectos y fallos encontrados en el transcurso de las pruebas, cuyo principal objetivo es verificar en un futuro que estos errores fueron erradicados en posteriores iteraciones. La siguiente tabla muestra un resumen del registro de defectos y dificultades encontradas durante la ejecución de las iteraciones de pruebas:

Tabla 14. Resumen de defectos y dificultades

Elemento	No.	No conformidad	Aspecto correspondiente	Etapa de detección	Respuesta del desarrollador
Interfaz de “Le Atiendo”.	1	La creación permitió dejar el campo vacío de número de serie del equipo en el reporte.	Tabla de equipos, campo “nro_serie”.	Primera iteración de pruebas de aceptación.	Resuelto una vez terminada la primera iteración de pruebas.
Interfaz de “Le Atiendo”.	2	La modificación permitió dejar el campo vacío de número de serie del equipo en el reporte.	Tabla de equipos, campo “nro_serie”.	Primera iteración de pruebas de aceptación.	Resuelto una vez terminada la primera iteración de pruebas.
Interfaz de “Le Atiendo”.	3	La modificación permitió dejar el campo vacío de nombre del estado.	Tabla de estados, campo “nombre”.	Primera iteración de pruebas de aceptación.	Resuelto una vez terminada la primera iteración de pruebas.
Interfaz de “Le Atiendo”.	4	La creación permitió dejar el campo vacío de la marca del equipo.	Tabla de marcas, campo “nombre”.	Primera iteración de pruebas de aceptación.	Resuelto una vez terminada la primera iteración de pruebas.

Interfaz de “Le Atiendo”.	5	La modificación permitió dejar el campo vacío de la marca del equipo.	Tabla de marcas, campo “nombre”.	Primera iteración de pruebas de aceptación.	Resuelto una vez terminada la primera iteración de pruebas.
Interfaz de “Le Atiendo”.	6	Se intentó visualizar un informe de reportes realizados a un número de serie de equipo no válido y no mostró el mensaje de error deseado.	Tabla de equipos, campo “nro_serie”.	Segunda iteración de pruebas de aceptación.	Resuelto una vez terminada la segunda iteración de pruebas.
Interfaz de “Le Atiendo”.	7	La creación permitió dejar el campo vacío del rol en el usuario.	Tabla de usuario, campo “rol_id”.	Segunda iteración de pruebas de aceptación.	Resuelto una vez terminada la segunda iteración de pruebas.
Interfaz de “Le Atiendo”.	8	La modificación permitió dejar el campo vacío del rol en el usuario.	Tabla de usuario, campo “rol_id”.	Segunda iteración de pruebas de aceptación.	Resuelto una vez terminada la segunda iteración de pruebas.

Pruebas de rendimiento

Se le realizaron pruebas al sistema haciendo uso de la página de administración del *framework* Symfony para determinar si el mismo cumplía los tiempos de respuesta establecidos como requisitos no funcionales. Como se muestra en la figura 13 el tiempo de respuesta para la petición de un informe es de 543 milisegundos siendo menos que los 3000 milisegundos previstos como requerimientos y para la creación de un reporte como se muestra en la figura 14 se necesitaron 505 milisegundos de los 1500 previstos, siendo ambos resultados satisfactorios.

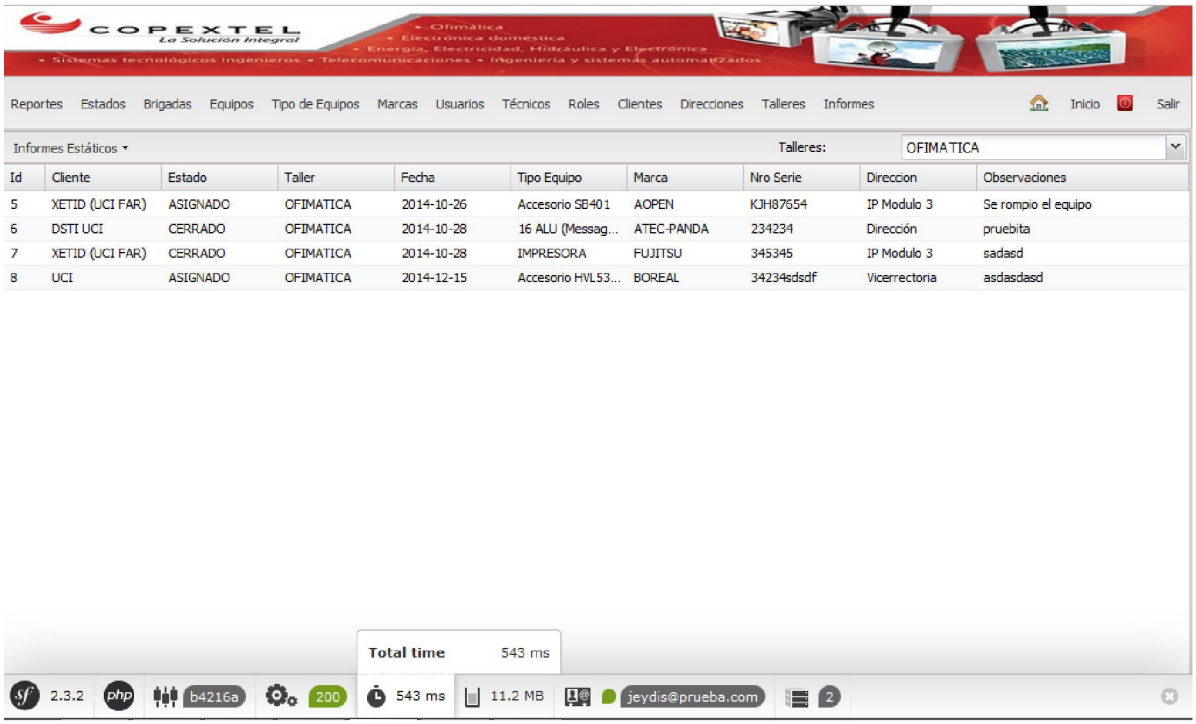

Fig. 15: Tiempo de respuesta para la petición de un informe

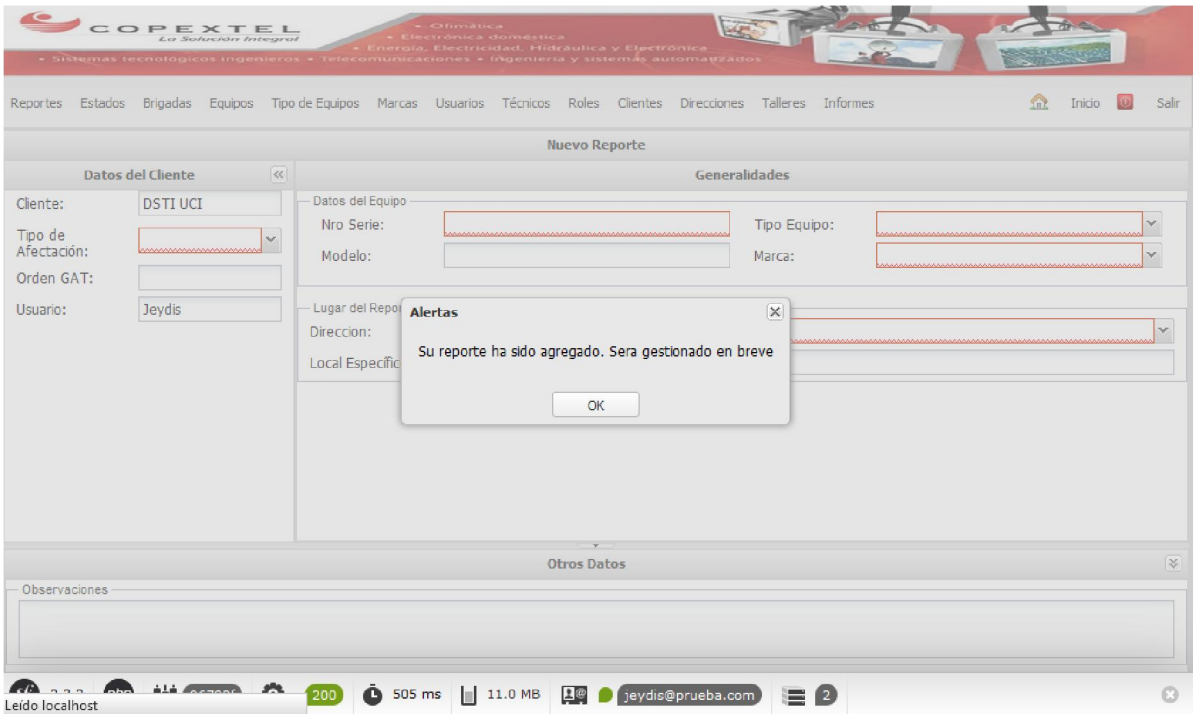

Fig. 16: Tiempo de respuesta para la creación de un reporte

3.2.3. Resultados

Para que un proceso de pruebas tenga éxito se requiere de un análisis final de los resultados arrojados, es decir, la evaluación del producto que se está probando de acuerdo a todos los defectos y fallos del sistema encontrados a lo largo del proceso.

Las pruebas de aceptación se realizaron en tres iteraciones y se utilizaron los 55 casos de pruebas diseñados, para verificar que las funcionalidades implementadas fueron las acordadas en las historias de usuario y que respondían correctamente a sus necesidades. A continuación se muestra un cuadro resumen que muestra los resultados obtenidos en las pruebas ejecutadas durante las tres iteraciones realizadas:

	55 47	55
Casos de prueba diseñados	55	55
Casos de prueba ejecutados	55	55
Casos de prueba exitosos	47	55

Fig. 17: Resumen de los casos de pruebas y sus resultados

Como se muestra en la Figura 13, de los 55 casos de pruebas diseñados, 47 resultaron exitosos, existiendo 8 que lanzaron una no conformidad. Por tal motivo, una vez resueltos los defectos detectados se pasó a una tercera iteración de pruebas, donde se volvieron a realizar todos los casos de prueba diseñados, para verificar que fueron resueltas las no conformidades de la primera y segunda iteración y que además no habían sufrido cambios los casos de pruebas exitosos de las mismas. Al concluir la tercera iteración, todos los casos de prueba arrojaron resultados satisfactorios.

Los resultados de las pruebas de rendimiento arrojaron resultados satisfactorios cumpliendo con los tiempos de respuesta requeridos para el sistema.

Coclusiones parciales

En este capítulo se descompusieron las historias de usuarios en tareas de ingeniería, a las cuales se le asignaron tiempos de implementación que fueron cumplidos cabalmente, garantizando el objetivo principal de su realización. Fue definido el estándar de codificación a utilizar, con el objetivo de evitar errores y lograr una mejor comprensión del código generado. Además se realizó la descripción de las pruebas de aceptación a las que fue sometido el sistema con la intención de verificar el funcionamiento de la misma, probando individualmente la implementación de cada historia de usuario en su correspondiente iteración, además de la realización de las pruebas de rendimiento lo que brindará al cliente conformidad y seguridad ante las funcionalidades del sistema.

Conclusiones

Durante el desarrollo del presente trabajo se llevaron a cabo los diferentes flujos de trabajo que propone la metodología utilizada para darle cumplimiento a los objetivos propuestos.

- Se realizó un estudio de los sistemas de gestión de reportes para confeccionar los fundamentos teóricos y se analizó que no era posible utilizar ninguna solución anterior.
- Se estudiaron y seleccionaron las tendencias, tecnologías y herramientas más idóneas para el desarrollo del sistema y se realizó el diseño del mismo.
- Se desarrolló el sistema informático “Le Atiendo”, lo cual facilitará la gestión de los reportes en la DSTI UCI.
- Se ejecutaron las pruebas de aceptación y rendimiento las cuales demostraron que el sistema está listo para su uso, dándole cumplimiento de esta forma al problema planteado en la investigación.

Recomendaciones

Luego de terminada la investigación, la autora recomienda:

- Dotar al sistema con la capacidad de definir prioridades a los reportes para garantizar que sean atendidos en el orden necesario.
- Estudiar la factibilidad de integrar el sistema “Le Atiendo” al resto de los sistemas contables existentes en la DSTI UCI.
- Desarrollar un manual de usuario para uso de los clientes, con el objetivo de lograr un mayor entendimiento del sistema y su funcionamiento.
- Realizar pruebas de penetración como ataques xss e inyecciones sql al sistema para comprobar los mecanismos de seguridad.
- Extender el uso del sistema a las demás divisiones de Copextel, para que sirva como herramienta de gestión de reportes e informes estadísticos.

Referencias

1. **Copextel S.A.** Portal corporativo. [En línea] 2012. [Citado el: 25 de 06 de 2014.] http://192.168.11.11/portalcorporativo/.

2. **Darromán Savigne, Caridad y Velázquez Leyva, Reynerio.** Observatorio de la Economía Latinoamericana. *El proceso de gestión y la gestión económica en las empresas.* [En línea] 2011. [Citado el: 25 de 08 de 2014.] http://www.eumed.net/cursecon/ecolat/cu/2011/dsvl.htm.

3. **Fantova, Fernando.** *Manual para la gestión de la intervención social. Políticas, organizaciones y sistemas para la acción.* Madrid : CCS, 2005.

4. **Oscar.** INGENIERIA INDUSTRIAL. *UNIDAD 2: Sistemas y diseño de sistemas.* [En línea] 2011. [Citado el: 26 de 08 de 2014.] http://oscaring.bligoo.com.mx/unidad-2-sistemas-y-diseno-de-sistemas.

5. **The Information Technology Infrastructure Library (ITIL) Open Guide.** ITIL Open Guide. *ITIL Incident Management.* [En línea] [Citado el: 04 de 09 de 2014.] http://www.itlibrary.org/index.php?page=Incident_Management.

6. **Vergara, Gonzalo.** Mejora Tu Gestión. *¿Qué es un Sistema de Gestion?* [En línea] 2009. [Citado el: 27 de 08 de 2014.] http://mejoratugestion.com/mejora-tu-gestion/que-es-un-sistema-de-gestion/comment-page-1/#comment-1215.

7. **Ministerio del Poder Popular para Ciencia, Tecnología e Innovación.** VenCERT. *Sistema Nacional de Gestión de Incidentes Telemáticos.* [En línea] Gobierno Bolivariano de Venezuela, 2008. [Citado el: 18 de 09 de 2014.] http://vencert.gob.ve/index.php/vencert/faqs/90-que-es-un-incidente.

8. **OSIATIS S.A.** ITIL® v3. *Gestión de servicios TI.* [En línea] [Citado el: 18 de 09 de 2014.] http://itilv3.osiatis.es/operacion_servicios_TI/gestion_incidencias.php.

9. **GLPI, Grupo.** GLPI Gestion Libre de Parc Informatique. [En línea] 2010. [Citado el: 04 de 09 de 2014.] http://www.glpi-project.org/.

10. **Valle, Amaury E. del.** Juventud Revelde. [En línea] 2014. [Citado el: 04 de 09 de 2014.] http://www.juventudrebelde.cu/cuba/2014-05-22/abren-nuevos-servicios-informaticos.

11. **Universidad de las Ciencias Informáticas (UCI).** GATServer. [En línea] Universidad de las Ciencias Informáticas (UCI). [Citado el: 18 de 11 de 2014.] https://technology.uci.cu/.

12. **Electrónica, Laboratorios.** Geocities.ws. Anotaciones de RUP. [En línea] 2014. http://www.geocities.ws/gustsucc/Archivos/An otacionesRUP.pdf.

13. **Martínez, Alejandro Martínez y Raúl.** Escuela Politécnica Superior de Albacete – Universidad de Castilla la Mancha. *Guía a Rational Unified.* [En línea] 2014. http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/TrabajoGuia%20RUP.pdf.

14. **Penadés, Patricio Letelier y Mª Carmen.** Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea] 2006. http://www.cyta.com.ar/ta0502/v5n2a1.htm.

15. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : Prentice Hal, 1999. ISBN: 970-1 7-0261-1.

16. **Martin fowler, kendall sccott.** *"UML Gota a Gota".* 1999.

17. **Microsoft.** microsoft. *Habilitar los Servicios de Internet Information Server (IIS).* [En línea] 2014. [Citado el: 18 de 09 de 2014.] http://msdn.microsoft.com/es-es/library/ms181052%28vs.80%29.aspx.

18. **Digital Learning.** Formación online en Nuevas Tecnologías. *¿Qué hace un Servidor Web como Apache?. Configuración .* [En línea] [Citado el: 18 de 09 de 2014.] http://www.digitallearning.es/blog/apache-servidor-web-configuracion-apache2-conf/.

19. **Pontificia Universidad Católica del Perú (PUCP).** TUXPUX. [En línea] [Citado el: 18 de 08 de 2014.] http://tuxpuc.pucp.edu.pe/articulo/comparativa-de-frameworks-en-php-cakephp-symfony-y-zend-framework.

20. **Powered by WordPress and zBench.** PHPLand y otros pensamientos. [En línea] 2014. [Citado el: 18 de 09 de 2014.] http://www.ricardclau.com/2011/06/elegir-un-framework-de-php-para-un-nuevo-proyecto/.

21. **Casas, Sandra y Reinaga, Héctor.** Identificación y Modelado de Aspectos Tempranos dirigido por Tarjetas de Responsabilidades y Colaboraciones. [En línea] 2008. [Citado el: 24 de 11 de 2014.] http://www.oocities.org/espanol/profeprog2/INVPAPER25.pdf.

22. **Zaninotto, François y Potencier, Fabien.** *Symfony, la guía definitiva.*

23. **Saavedra, Jorge.** El mundo informático. [En línea] [Citado el: 27 de 11 de 2014.] http://jorgesaavedra.wordpress.com/category/patrones-grasp/...

24. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HAL. ISBN: 970-1 7-0261-1.

25. **Lorenzo, Alberto Marturelo.** *Desarrollo de middleware de comunicación para el sistema Video Vigilancia Xilema Suria.* La Habana : Universidad de las Ciencias Informáticas, 2014.

26. **Piël, Nicholas.** ZeroMQ an introduction. *ZeroMQ.* [En línea] 2010. [Citado el: 8 de 15 de 2014.] http://nichol.as/zeromq-an-introduction.

27. **Perdomo, Gianny Martínez.** *Módulo Fantasy Béisbol del sitio de béisbol de la UCI.* La Habana : Universidad de las Ciencias Informáticas, 2009.

28. **Quintero, Naychel Pérez de Medina.** *Sistema Informático para el registro contable de los Útiles y Herramientas de la División Territorial Copextel Matanzas.* Matanzas : Universidad de Matanzas “Camilo Cienfuegos”, 2013.

29. **PRUEBASDESOFTWARE.** Pruebas de Software. *Gestión de Calidad y Pruebas de Software.* [En línea] [Citado el: 8 de 12 de 2014.] http://pruebasdesoftware.com/pruebadeaceptacion.htm.

30. **Pressman, Roger S.** *Ingenieria del Software Un enfoque práctico.* 6. s.l. : McGraw-Hill Interamericana. 970-10-5473-3.

31. **Somerville, Ian.** *Ingeniería del Software.* Madrid : Pearson Educación S.A., 2005. 84-7829-074-5.

32. **Vaquero Santiago, David y Díaz Valencia, Néstor.** Morfeo FORMACION. [En línea] Centro de Competencia Morfeo, 2009. [Citado el: 20 de 09 de 2014.] http://formacion.morfeo-project.org/wiki/index.php/PT3:GPSL:UF6.

33. **Real Academia Española.** Real Academia Española. [En línea] 2014. [Citado el: 15 de 10 de 2014.] http://lema.rae.es/drae/?val=rendimiento.