Universidad de las Ciencias Informáticas

Facultad 1

Sistema para la gestión de reportes de incidencias de mantenimiento en la Universidad de las Ciencias Informáticas

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

Autor:

Reynaldo Columbie Mamalova

Tutores:

Ing. Yordanka Fuentes Castillo

Ing. Carlos Yordan González Herrera

Lic. Luis Enriquez Benet

La Habana, junio 2019

La comprensión de que la vida es absurda no puede ser un fin, sino un comienzo.

Albert Camus.

Declaración de autoría

Declaro por este medio que yo: Reynaldo Columbie Mamalova, con carné de identidad 94041528502, soy el autor principal del trabajo final de tesis de pregrado que se titula: “Sistema para la gestión de reportes de incidencias de mantenimiento en la Universidad de las Ciencias Informáticas”. El cual ha sido desarrollado como parte del trabajo en el Departamento de Ingeniería y Gestión de Software, específicamente en la asignatura Sistemas de Bases de Datos de la Facultad 1. Autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, se firma la presente declaración jurada de autoría en La Habana, a los _____ días del mes de _____ del año 2019.

_____

Autor

Reynaldo Columbie Mamalova

_____

Tutor

Ing. Yordanka Fuentes Castillo

_____

Tutor

Ing. Carlos Yordan González Herrera

_____

Tutor

Lic. Luis Enriquez Benet

Dedicatoria

A mi abuela, que siempre la llevaré en mi corazón hasta el final de mis días, por iniciarme en el camino del saber.
A mis padres por confiar en mí, por su esfuerzo y dedicación.

Agradecimientos

Primeramente, quisiera agradecer a mi abuela, que, aunque no esté aquí, la llevaré para siempre en mi corazón y mente, por su esfuerzo y dedicación diaria desde que era muy pequeño, por enseñarme a ser humilde y tener buen corazón, por enseñarme a ver el mundo desde otro punto de vista, por iniciarme en los caminos del saber y por enseñarme a amar los libros, pasión que aún mantengo gracias a ella; soy la persona que soy al día de hoy gracias a ella; a ti, te dedico este título.

Agradecer a mi padre, que, aunque no esté aquí en este momento tan especial de mi vida, sé que estará muy contento de que su hijo sea ingeniero, gracias por todos los sacrificios que tuviste que hacer para que yo pudiera llegar hasta aquí, por toda la confianza que tienes en mí, por los consejos, por guiarme durante toda mi vida por el buen camino, por tu amistad y por saber escucharme.

Agradecer a mi madre por darme todo su cariño y atención durante mi carrera y durante toda mi vida, por los sacrificios hechos y toda su dedicación, por cuidarme y ser parte de mi vida.

Agradecer a mi hermana por estar a mi lado y confabular a mi lado todo este tiempo, por portarnos mal y ser diferentes; a mi abuelo por todo su apoyo y por confiar siempre en mí, a mi tío por todos los consejos y apoyo, a mi prima Leyanis por quererme tanto y a mi tía por todo su cariño y amistad.

Quisiera agradecer a todas las personas que he conocido durante estos cinco años, es sabido que soy una persona muy solitaria, pero siempre estaré agradecido de haber conocido a mi hermano y amigo del alma Luis Miguel, por todo lo que vivimos juntos y viviremos, por todo lo que compartimos y las locuras que hicimos, pues nadie sabe entenderme como tú, sabes que nunca voy a olvidarte y sé que nuestra amistad durará para siempre; gracias Leo por ser esa persona que siempre eres, y por apoyarme en momentos difíciles cuando no sabía qué hacer, siempre te llevaré en mi corazón.

También quisiera agradecerte Garnache por toda tu ayuda y tus consejos, y por aguantarme todo este tiempo; gracias también a Tamara por ofrecerme su ayuda cuando la necesité y a la profe Madelin; gracias a toda la gente del "voly" por todos los buenos momentos que pasamos. Gracias a mis amigos, a Eduardo por tu compañía y tu música, a Leysdanis por estar siempre inventando, a Christopher por tu ayuda, a Ariel por tus repasos, a todos mis compañeros de aula, a Ewelyn por tu amistad y cariño, a Javier y Alfredo por los momentos que compartimos, en fin, a todas las personas que formaron parte de esta experiencia; gracias a mi hermano pequeño Arian por estar siempre a mi lado.

Pero especialmente quisiera agradecer a mis tutores Yordanka, Carlos Yordan y Luis, sin ellos no hubiese podido llegar hasta aquí. Muchas gracias Carlos por todas esas horas ya tarde apoyándome en todo momento, esas largas horas con

sueño, gracias por toda tu dedicación, por tu tiempo, por tu sacrificio y tu ayuda, agradezco mucho, pero mucho, todo el apoyo que me has dado durante todo el desarrollo de mi tesis. Muchas gracias Yordanka por toda tu comprensión, por toda tu atención y por soportarme todo este tiempo, por ayudarme a salir del bache en donde estaba y por aconsejarme, sin ti, no sé qué hubiera sido de mi este quinto año donde todo se me tornaba muy difícil, gracias por toda tu voluntad, tus consejos y por encaminarme. Gracias Luis Enrique por formar parte de este equipo, por tu ayuda y por toda tu preocupación; sin ustedes no hubiera podido alcanzar esta meta de ser ingeniero, muchas gracias de todo corazón.

Resumen

La Universidad de las Ciencias Informáticas (UCI) es un centro de estudios superiores, el cual dispone de más de 150 instalaciones constructivas, debido a la cantidad de personal que en ella reside. Estas instalaciones, además de la gran estructura constructiva, poseen un número elevado de recursos y medios que hacen posible su uso. La UCI cuenta, además, con una Dirección de Mantenimiento (DM), encargada de dar solución a las incidencias que son reportadas motivo del desgaste que provocan más de 15 años de explotación por parte de la comunidad universitaria. La DM cuenta con un sistema para la gestión de los reportes de incidencias. Este sistema no permite al autor del reporte de la incidencia editarla, por lo que una vez guardada la incidencia, si desea rectificar algún campo, debe volver a insertar otra incidencia, incurriendo en que la incidencia se encuentre duplicada en el sistema. Además, no genera reportes estadísticos que permitan llevar un control sobre el estado de resolución de las incidencias. La presente investigación tiene como objetivo desarrollar un sistema para la gestión de reportes de incidencias de mantenimiento para la Universidad de las Ciencias Informáticas, que permite disminuir el esfuerzo que se invierte en la gestión de los reportes de incidencias. Para el desarrollo de la solución se empleó el gestor de contenidos Drupal, el lenguaje PHP, como gestor de base de datos Mysql y Apache como servidor de aplicaciones web. Para validar la solución se aplicaron pruebas funcionales, de rendimiento, de seguridad y de usabilidad.

Palabras clave: gestión de incidencias, incidencia, mantenimiento, reporte, toma de decisiones

Índice de contenidos

Índice de figuras

Índice de tablas

Introducción

La Universidad de las Ciencias Informáticas (UCI) es un centro de estudios superiores, que abrió sus puertas al primer curso escolar el 23 de septiembre de 2002, como fruto de la visión futurista del Comandante en Jefe Fidel Castro Ruz (MES, 2019). La remodelación de antiguos edificios existentes en la zona donde está actualmente situada la UCI y la construcción de otros, permitió que, en solo 106 días, surgiera lo que hoy es llamada “La Universidad del Futuro”. Esta institución no fue concebida de la misma manera que lo fueron otras universidades. Esta, se convertiría en una de las universidades más grandes del país y con mayor cantidad de instalaciones constructivas, más de 150 hasta el día de hoy; debido a la cantidad de personal, entre estudiantes, profesores, trabajadores no docentes y otros, que formarían parte de la comunidad universitaria y que actualmente asciende a más de ocho mil (8000) personas (UCI, 2018).

Dentro de las áreas e instalaciones que forman parte del campus universitario se encuentran los edificios docentes, los edificios de la residencia, del rectorado, los complejos comedores, entre otros. Estos, además de la gran estructura constructiva, poseen una gran cantidad de recursos y medios que los complementan y hacen posible su uso (UCI, 2018). Motivo del desgaste que provocan más de 15 años de explotación por parte de una numerosa comunidad universitaria, así como el lógico deterioro de los medios que se ponen a su disposición, existen diariamente diversos problemas de mantenimiento constructivo. Por esta razón, la Universidad cuenta con una Dirección de Mantenimiento (DM), encargada de atender y dar solución a las incidencias que son reportadas a diario. Estas pueden estar relacionadas con la albañilería, carpintería, plomería, electricidad y alumbrado y equipos de clima y refrigeración.

El trabajo del personal de mantenimiento es dar solución a las incidencias que se reporten. En caso de que la solución no pueda ser inmediata, se debe dar una respuesta al afectado informándole de esta situación y explicándole los motivos por los cuales su solicitud no ha sido solucionada hasta ese momento. Se debe tener en cuenta que hay soluciones que pueden ser a largo plazo e incidencias que requieren de recursos que pueden no estar disponibles al momento del reporte. En cualquier caso, la persona afectada necesita tener una retroalimentación del estado en que se encuentra su reporte, lo cual debe ser controlado también por el personal de mantenimiento. Es posible que una persona, con el afán de encontrar una solución inmediata, pueda realizar varios reportes sobre una misma incidencia, y si a esto se suma la variedad de incidencias que pueden ser reportadas, al ser diversas las categorías con las que pueden estar relacionadas como se menciona con anterioridad, el número de reportes aumenta considerablemente.

Por las razones antes mencionadas, la DM cuenta con un sistema para la gestión de los reportes de incidencias, que ha permitido realizar un trabajo más eficiente para solucionarlas. Sin embargo, el objetivo es garantizar una mayor satisfacción del personal al que se atiende y realizar un trabajo lo más eficaz posible. En ese sentido, el sistema utilizado actualmente, carece de funcionalidades que podrían permitirle al personal de la DM optimizar la gestión de los reportes que reciben mediante este sistema. Para una mejor comprensión de lo planteado, se relacionan a continuación las deficiencias relacionadas a dichas funcionalidades:

- Se notifica al usuario autor del reporte de la incidencia solo cuando esta se crea, por lo que luego de creada, el usuario no es notificado durante toda la gestión del reporte hasta la resolución de la incidencia. Esto provoca que el usuario desconozca en muchos casos el estado de su reporte y no sepa si su solicitud ha sido atendida o no.
- Una vez que la incidencia está creada, el sistema no permite al usuario autor editarla en caso de necesidad, lo cual provoca que, en caso de cometer algún error en la creación de la incidencia, este deba crearla nuevamente para corregir dicho error. Tampoco pueden ser eliminadas las incidencias creadas, ni antes ni después de su solución, por lo cual, al sumar las que no se pueden editar que deben ser repetidas más todas las que un usuario pueda crear, sería una lista bastante amplia que puede resultar incómoda para el usuario.
- No genera reportes que permitan a los directivos de las áreas que brindan los servicios relacionados en las incidencias, tomar decisiones o llevar un control en cuanto a cuál ha sido el área que más incidencias ha reportado, cuál ha sido el área que más incidencias ha resuelto, cuántos y cuáles recursos se han destinado para resolver las incidencias, entre otros.

Además de estas carencias, otra de las dificultades que presenta el sistema actual viene dada porque, aun cuando se trata de una aplicación web, esta presenta un diseño poco atractivo y no adaptable a dispositivos móviles. Esto último, trae como consecuencia que los usuarios se vean obligados a acceder al sistema solo por computadoras y, sin embargo, hoy en la UCI es común el acceso de los usuarios a los servicios telemáticos a través de dispositivos móviles.

La problemática existente y descrita anteriormente permite identificar como **problema de investigación**: ¿Cómo contribuir al proceso de gestión de los reportes de incidencias de mantenimiento en la Universidad de las Ciencias Informáticas?

El **Objeto de estudio** de la presente investigación va dirigido al proceso de gestión de reportes de incidencias de mantenimiento, enmarcando como **campo de acción**: el proceso de gestión de reportes de incidencias de mantenimiento en la Universidad de las Ciencias Informáticas.

Para dar solución al problema planteado se define como **objetivo general**: Desarrollar un sistema web, que contribuya al proceso de gestión de reportes de incidencias de mantenimiento en la Universidad de las Ciencias Informáticas.

En función de cumplir con el objetivo planteado, se formulan las siguientes **preguntas científicas**:

1. ¿Cuáles son los referentes teóricos que sustentan la investigación, relacionados con el uso de las herramientas informáticas, en función de la gestión de reportes de incidencias de mantenimiento?
2. ¿Cuál es el estado actual de las herramientas informáticas para la gestión de reportes de incidencias de mantenimiento?
3. ¿Qué elementos deben tenerse en cuenta para llevar a cabo el análisis y diseño del sistema para la gestión de reportes de incidencias de mantenimiento en la Universidad de las Ciencias Informáticas?
4. ¿Cómo materializar, en términos de componentes y código fuente, los diseños especificados para el sistema para la gestión de reportes de incidencias de mantenimiento en la Universidad de las Ciencias Informáticas?
5. ¿Qué resultados se obtendrán al validar, a través de una estrategia de pruebas de software, el sistema para la gestión de reportes de incidencias de mantenimiento en la Universidad de las Ciencias Informáticas?

Para responder estas preguntas científicas, es necesario dar cumplimiento a las siguientes **tareas de investigación**:

1. Sistematización de los referentes teóricos que sustentan la investigación, relacionados con el uso de las herramientas informáticas en función de la gestión de reportes de incidencias de mantenimiento.
2. Análisis del estado actual de las herramientas informáticas para la gestión de reportes de incidencias de mantenimiento.
3. Análisis y diseño de las funcionalidades del sistema para la gestión de reportes de incidencias de mantenimiento en la Universidad de las Ciencias Informáticas.
4. Implementación de las funcionalidades del sistema para la gestión de reportes de incidencias de mantenimiento en la Universidad de las Ciencias Informáticas.
5. Validación, a través de una estrategia de pruebas, del sistema para la gestión de reportes de incidencias de mantenimiento en la Universidad de las Ciencias Informáticas.

Para obtener los conocimientos necesarios que hagan posible el cumplimiento del objetivo trazado, se lleva a cabo una investigación en las que se utilizan algunos de los métodos científicos existentes, tanto teóricos como empíricos.

Teóricos:

Histórico-Lógico: Permite una mayor comprensión de la evolución de las herramientas informáticas para la gestión de reportes de incidencias de mantenimiento

Analítico-Sintético: Se utiliza con el objetivo de realizar un análisis bibliográfico para establecer las bases teóricas en relación al desarrollo de los sistemas para la gestión de reportes de incidencias de mantenimiento.

Modelación: Es empleada en la representación mediante diagramas de las características, procesos y componentes de la solución propuesta, así como la relación existente entre ellos.

Empíricos:

Entrevista: Se emplea en encuentros con el cliente para conocer la necesidad del desarrollo de la propuesta de solución, definir sus funcionalidades e identificar a la vez particulares de cada usuario y las restricciones que se imponen (ver Anexo 1).

Observación: Posibilita obtener conocimiento acerca del funcionamiento de los sistemas existentes en la actualidad para la gestión de reportes de incidencias de mantenimiento.

La presente investigación está estructurada en tres (3) capítulos, los cuales se explican brevemente a continuación:

Capítulo 1: “Fundamentación teórica”:

Se hace una descripción de los conceptos relacionados con los sistemas de incidencias de mantenimiento, las tendencias, técnicas, tecnologías, metodologías y el análisis de algunas soluciones existentes a nivel internacional y nacional que son de interés por tener estrechos lazos con el alcance de esta investigación. Se define, además, el entorno de desarrollo para la solución propuesta.

Capítulo2. “Análisis y diseño de la propuesta de solución”:

Se analizan las características del negocio, explicando la propuesta de solución al problema de investigación, se definen los principales requisitos a partir de la metodología seleccionada y se describen los diagramas de clases del diseño, diagramas de comportamiento (diagramas de secuencia), lo referente al estilo arquitectónico del sistema, así como los patrones de diseño utilizados y el entorno de despliegue propuesto para desplegarlo.

Capítulo3: “Implementación y pruebas de la propuesta de solución”:

En este capítulo se detalla la propuesta de solución al problema planteado. Se define el entorno de despliegue necesario para el sistema y se describe la organización del mismo en un diagrama de componentes. Se especifican los estándares de codificación que se utilizan en base al lenguaje de programación propuesto para el desarrollo del sistema. Además, se realiza la estrategia de pruebas definida para validar la propuesta de solución verificando el cumplimiento de los requisitos funcionales y no funcionales descritos en el capítulo anterior y se muestran imágenes de interfaces del sistema como parte del resultado final.

Con la realización de la presente investigación, se pretende obtener un sistema para la gestión de reportes de incidencias que permitirá, a la Dirección de Mantenimiento (DM) de la UCI, ejecutar los procesos que se desarrollan durante la gestión de los reportes de incidencias que se realicen desde cada área del campus universitario, además de proveer a los usuarios de una herramienta para registrar y supervisar sus reportes sobre cualquier incidencia de mantenimiento, con el fin de que estas sean solucionadas eficientemente. El sistema facilitará el manejo de toda la información referente a las incidencias y mostrará el estado real del trabajo que se realiza por parte de la DM.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

Para comprender los antecedentes de la investigación, es necesario describir los aspectos teóricos relacionados con la gestión de reportes de incidencias de mantenimiento y las bases técnicas a tener en cuenta para el desarrollo del sistema. Con el objetivo de lograr una mayor comprensión del alcance de la investigación y esclarecer su objeto de estudio, se exponen en el presente capítulo, los conceptos asociados al dominio de la investigación y se realiza un análisis del estado del arte que la precede. Además, se presenta el entorno de desarrollo a utilizar para dar solución al problema planteado.

1.2 Conceptos asociados al dominio de la investigación

A continuación, se relacionan los principales conceptos que ayudan a entender el desarrollo de la investigación.

Los sistemas web o también conocidos como "aplicaciones web" son aquellos que están creados e instalados no sobre una plataforma o sistemas operativos, sino que se alojan en un servidor en Internet o sobre una intranet (red local). Su aspecto es muy similar a páginas web que pueden ser vistas normalmente, pero en realidad los sistemas web tienen funcionalidades potentes que brindan respuestas a casos particulares, entre los que se encuentras los sistemas de gestión (Knowdo, 2018).

Un sistema de gestión es una serie de procesos, acciones y tareas que se llevan a cabo sobre un conjunto de elementos (personas, procedimientos, estrategias, planes, recursos, productos) para lograr el éxito sostenido de una organización, es decir, disponer de los recursos necesarios para satisfacer las necesidades y las expectativas de sus clientes o beneficiarios, trabajadores y de otras partes interesadas a largo plazo y de un modo equilibrado y sostenible (Naranjo, 2015). Dentro de la gestión de procedimientos, destacan los sistemas de gestión para incidencias.

Según Samaniego (2013), la gestión de reportes 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 reportes de incidencias es recuperar el nivel habitual de funcionamiento del servicio y minimizar en la medida de lo posible el impacto negativo en la organización, de forma que la calidad del servicio y la disponibilidad se mantengan. Existen incidencias de distinta naturaleza, lo cual depende del lugar y de la actividad a la que esté relacionada, pueden manifestarse incidencias de tipo laborales, informáticas, de personal, de software y de mantenimiento, entre otras. Estas últimas, se presentan

cuando los servicios que se necesitan están enmarcados en el área de mantenimiento de una organización.

Entiéndase por incidencia, cualquier acontecimiento que tenga una influencia negativa sobre una organización, incluidos su personal, el producto de la organización, los equipos o el entorno en el que opera (LQMS, 2014). Por su parte, un reporte, según Martínez (2015) es un documento que a partir de los objetivos de quien lo crea, pretende transmitir información. Estos pueden ser de diversos tipos o estar soportados en diversos formatos (documentos impresos, digitales, audiovisual). En el ámbito de la informática, los reportes son informes que se organizan y muestran información a partir de fuentes de datos digitales diversas (textos, bases de datos). Su función es aplicar un formato determinado a los datos para mostrarlos por medio de un diseño atractivo y que sea fácil de interpretar por los usuarios.

Por otra parte, el mantenimiento**,** es la realización de tareas o actividades que permitan reparar una unidad funcional para que ésta pueda continuar cumpliendo sus objetivos, e incluyen acciones de inspección, comprobaciones, clasificación, reparación, entre otras (Torres, 2016).

Procesos de gestión de reportes de incidencias

Con el objetivo de sentar las bases de las funcionalidades de la propuesta de solución, se describe a continuación cómo funciona el proceso de gestión de reportes de incidencias en el sistema que se emplea actualmente en la UCI (disponible en https://incidencias.uci.cu/):

La dirección de mantenimiento recibe el reporte (ticket) cuando los usuarios lo registran en el sistema, o cuando lo hacen llegar vía teléfono, correo o por defectación. En este momento el reporte pasa a estar en estado Abierto.

Luego se revisa el reporte para conocer la naturaleza de la incidencia y se determinan los recursos necesarios para su solución. El reporte pasa a estar en estado de Ejecución.

A partir de la necesidad de recursos, se hace el pedido de estos al almacén.

Posteriormente, se envía el personal de mantenimiento (operarios) para dar solución a la incidencia, para lo cual hacen uso de los recursos asignados. En este momento, el reporte se encuentra Ejecutado.

El ticket se cierra (Cerrado) cuando el cliente (usuario o persona afectada) firma el destino final u orden de trabajo en conformidad con la calidad del trabajo y los recursos utilizados.

1.3 Resultados del análisis de sistemas informáticos para la gestión de reportes de incidencias de mantenimiento

En la actualidad existen diversas aplicaciones para la gestión de reportes de incidencias de mantenimiento. Con el objetivo de identificar ventajas y desventajas de estas, se realiza el siguiente análisis haciendo énfasis en las características y funcionalidades de este tipo de herramientas.

1.3.1 Sistemas a nivel internacional

OTRS, es un sistema online de gestión de tickets y procesos, moderno y flexible, que permite a los profesionales de la gestión de servicios, de cualquier industria, mantenerse al día con el entorno empresarial actual orientado a resultados. OTRS está implantado en empresas de todos los tamaños. Desde departamentos de IT de empresas de tamaño medio a grandísimas compañías y organizaciones como el Instituto Nacional de Estadística, Boeing, Porsche, Lufthansa, Nokia o la NASA.

Entre sus principales características, y lo que la hace una herramienta muy útil, se encuentran la gestión y monitorización automática de incidencias, la existencia de una aplicación de *OTRS* para *iPhone*, y la generación de reportes para la toma de decisiones. Sin embargo, es un sistema privativo que no permite el acceso a su código fuente (Exevi, 2019).

Protecnus, es un software de mantenimiento dirigido a todas aquellas empresas con servicio técnico de campo, donde podrán gestionar y planificar de manera fácil y online toda la actividad de la empresa, desde sus técnicos hasta el análisis de datos y gestión empresarial. Posee un módulo de escritorio y un módulo móvil. Permite conocer el estado de los trabajos e incidencias en tiempo real. Localiza a sus técnicos en el momento. Al final del día la información es procesada automáticamente. A pesar de estas ventajas, no permite la obtención de reportes que necesitan los directivos y, además, es un software privativo (Protecnus, 2018).

Suite CRM **(Módulo de gestión de incidencias)** es una solución con la que se puede trabajar eficientemente la recogida y tratamiento de quejas. Normalmente, la entrada de las quejas se realiza por vía telemática o por centralita. En este sentido, cada operador accede a la plataforma con su cuenta de usuario. En caso de recibir una incidencia a través de un correo electrónico o formulario de la web, toda esta información se traspasa de forma automática.

Una vez registrada la incidencia, éste deberá pasar por todos los estados y validadores antes de considerarse como cerrada. El siguiente paso a la validación del informe es la emisión del informe de respuesta al cliente. Esta comunicación, puede realizarse por diferentes canales (correo electrónico, teléfono, o por el canal que utilice la empresa). Se engloba dentro de los softwares *Open Source*, lo

cual significa que el código fuente del programa es editable. A pesar de estas ventajas, no permite la obtención de reportes estadísticos que necesitan los directivos para la toma de decisiones (Activalink, 2018).

KMKey Help Desk es un software de gestión de incidencias indicado para servicios de mantenimiento, ayuda al usuario y 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, 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. Permite: resolución inmediata, escalado, consulta de información anterior, reparto de recursos. No permite la generación de reportes estadísticos que necesitan los directivos para la toma de decisiones. Otra desventaja es su licencia privativa, la cual no permite tener acceso al código fuente de esta herramienta ni su uso libre (HDKey, 2018).

Mantis es una aplicación de software libre multiplataforma que permite gestionar las incidencias en una empresa, en un sistema o en un proyecto. Es un sistema fácil de usar y adaptable a varios escenarios. Además, cuenta con diferentes *plugins* que aumentarán la capacidad de trabajo con la herramienta. Cuenta con una gran variedad de funcionalidades que permitirán que todos los objetivos queden cubiertos completamente (Gomez, 2013).

Entre todas ellas cabe destacar (Gomez, 2013): El reporte de incidencias, que permite a los distintos usuarios realizar reportes de cualquier tipo, ya sean incidencias técnicas, peticiones de soporte o bugs de un sistema. El usuario puede añadir un breve título a la incidencia, especificar una descripción detallada del mismo y algunos detalles técnicos si fueran necesarios.

1.3.2 Sistemas a nivel nacional

SRMC-UCI es una aplicación web desarrollada en la Universidad de las Ciencias Informáticas, la cual está diseñada para la dirección de mantenimiento de la UCI y permite gestionar las incidencias de reportes. El sistema proporciona una interfaz como forma de presentación al usuario desde la cual se puede autenticar. Una vez autenticado, la aplicación ofrece información general sobre el sistema y le brinda de acuerdo a su rol, una serie de funciones. El usuario autenticado puede realizar reportes mediante una interfaz que cuenta con un formulario a llenar, una vez reportado recibe una notificación al correo con los datos de dicho reporte. El sistema también brinda al técnico general la opción de realizar reportes y llevar un control de los reportes realizados según una serie de parámetros como son: especialidad, local, área, instalación, afectación, entre otros (Pérez, y otros, 2015).

Sistema para la gestión de reportes de la Dirección de Mantenimiento de la Universidad de Ciencias Informáticas (SRMC-UCI v2.0) es una nueva versión desarrollada en la Universidad de las Ciencias Informáticas, para implementar nuevas funcionalidades que no estaban presentes en la versión anterior. Este sistema se centra en la asignación de recursos a los reportes de acuerdo a la disposición de los almacenes, la asignación, seguimiento y evaluación de brigadas y los informes de estado de resolución que se emiten a la Dirección de la Universidad (Rafael, 2015). Por motivos de seguridad no se tiene acceso a su código fuente.

El análisis realizado sobre los sistemas de gestión de reportes de incidencias de mantenimiento, permitió realizar un resumen (ver Tabla 1) teniendo en cuenta los siguientes parámetros:

- **Licencia:** se refiere al estado jurídico de la aplicación en cuanto a su uso, modificación y distribución, esta puede ser pública, en aquellas que no necesitan un pago para ser utilizadas; o privativa, en aquellas que si lo requieren.

- ***Responsive*:** especifica si la herramienta es capaz de adaptarse a las resoluciones de pantalla de cualquier dispositivo donde se visualice.

- **Reportes:** evalúa si el sistema genera reportes generales que permitan a los directivos de las áreas que brindan los servicios relacionados a las incidencias, tomar decisiones o llevar un control estadístico determinado.

Tabla 1. Análisis de sistemas homólogos (elaboración propia)

Herramienta	Licencia	Adaptable	Reportes
OTRS	Privativa	Si	Si
Protecnus	Privativa	Si	No
Suite CRM	Pública	No	Si
KMKey Help Desk	Privativa	No	No
Mantis	Pública	Si	No
SRMC-UCI	Pública	No	No
SRMC-UCI v2.0	Pública	No	No
Sistema de Gestión de Incidencias	Privativa	No	No

A partir del análisis realizado, y del resumen antes presentado, el autor arriba a las siguientes conclusiones:

- Los sistemas SRMC-UCI y SRMC-UCI v2.0, son públicos, pero ninguno es adaptable a dispositivos móviles, ni genera reportes.

- El sistema Suite CRM es público y genera reportes, sin embargo, no es adaptable a dispositivos móviles.

- El sistema Protecnus es adaptable a dispositivos móviles, pero es privativo y no genera reportes.

- El sistema Mantis, a pesar de ser público y adaptable a dispositivos móviles, no genera reportes.

- El sistema OTRS es adaptable a dispositivos móviles y genera reportes, pero no se tiene acceso a su código fuente.

Por todo lo antes expuesto, se puede resumir que los sistemas analizados no cumplen con el objetivo de la presente investigación, pues ninguno reúne todos los requisitos de ser una herramienta adaptable a cualquier tipo de resolución de pantalla, gratuita y que permita generar reportes para la toma de decisiones. Sin embargo, el análisis realizado posibilitó la identificación de funcionalidades y tecnologías que pueden contribuir al desarrollo de la propuesta de solución que se propone en la investigación.

1.4 Entorno de desarrollo de la propuesta de solución

Metodología de desarrollo

Actualmente no se puede decir que existe una metodología única para garantizar el éxito de cualquier proyecto de desarrollo de software. Debido a que estas deben ajustarse al tipo de proyecto que se desarrolle.

La metodología ágil *AUP*, es una versión simplificada del Proceso Unificado de *Rational* (*RUP*). Esta describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en *RUP*. El *AUP* aplica técnicas ágiles incluyendo (Sánchez, 2015):

- Desarrollo Dirigido por Pruebas (*test driven development* - TDD en inglés)
- Modelado ágil
- Gestión de Cambios ágil
- Refactorización de Base de Datos para mejorar la productividad

Según Sánchez (2015), al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos), exigiéndose así que el proceso sea configurable, se decide hacer una variación de la metodología *AUP*, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Esta variación está unida al modelo CMMI-DEV v1.3[1], para garantizar las buenas prácticas en función de un software de calidad. Para el desarrollo de la propuesta de solución, y al tener en cuenta todo lo antes expuesto y las características asociadas a este desarrollo, se decide usar la metodología *AUP*, en su variación para la UCI. Para modelar el sistema y el trabajo con los requisitos, se emplea, de la metodología antes mencionada, el escenario número dos (2), que describe el negocio mediante el modelo conceptual y modela el sistema mediante el diagrama de casos de uso del sistema.

Lenguaje de modelado

El lenguaje de modelado es cualquier lenguaje informático gráfico o textual que provee el diseño y construcción de estructuras y modelos siguiendo un conjunto sistemático de reglas y marcos (Techopedia, 2017). Como lenguaje de modelado se utiliza el Lenguaje Unificado de Modelado (*UML* por sus siglas en inglés) en su versión 2.1, que consiste en un lenguaje diseñado para visualizar, especificar, construir y documentar software orientados a objetos (Martínez, y otros, 2014).

Herramienta de modelado

Dentro de las principales herramientas para el modelado se encuentran herramientas *CASE* definidas por Araujo y otros (2013) como herramientas informáticas que asisten al diseñador en algunas de las actividades relacionadas con el desarrollo de un sistema (requerimientos, análisis, diseño, codificación y pruebas).

Visual Paradigm para *UML* es una herramienta para desarrollo de aplicaciones utilizando modelado *UML*, ideal para ingenieros de software, analistas de sistemas y arquitectos de sistemas, que están interesados en la construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Para la presente investigación, se utiliza esta herramienta en su versión 8.0.

Herramienta de desarrollo

[1] Colección de buenas prácticas de desarrollo procedentes de la industria y del gobierno. Es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software (SEI, 2010).

Al tener en cuenta que la propuesta de solución es un sistema web, se tienen en cuenta las principales herramientas para el desarrollo web, dentro de las que se encuentran los marcos de trabajo (*frameworks*, en inglés) y los gestores de contenido. A continuación, se hace un análisis de estos dos tipos de herramientas en cuanto a sus características, ventajas y desventajas, lo cual permitirá hacer la selección de la más adecuada para el sistema en cuestión:

Sistemas de gestión de contenido:

Un sistema de gestión de contenidos (*CMS*, por sus siglas en inglés) hace referencia al software que permite la creación de una estructura de soporte (*framework*) para crear y administrar contenidos (generalmente páginas web) mediante una interfaz de usuario. Es una interfaz que puede controlar una o varias bases de datos en las que se encuentra el contenido del sitio web. Se puede administrar de forma independiente el contenido del diseño por lo que en cualquier momento es posible, con el mismo contenido, visualizarlo de forma distinta cambiando el estilo sin tener que cambiar el formato del contenido (Revistadigital, 2018).

Ventajas de los *CMS*:

- Existe una gran comunidad de desarrolladores, la cual es una gran ventaja porque se puede encontrar una gran documentación para aclarar las dudas que puedan surgir durante el desarrollo.
- Permiten personalizar el diseño web de la página, así como individualizar las funcionalidades de la plataforma del gestor de contenidos. De esta manera, se puede implementar rápidamente funcionalidades que de otra manera implicarían mucho más trabajo si no se cuenta con habilidades técnicas.
- Se pueden añadir nuevas funcionalidades al sitio web en cualquier momento, ya sea a través de *plugins* o módulos.

Desventajas de los *CMS*

- Se hace necesario realizar mantenimiento y actualizaciones, sobre todo de los problemas de seguridad ya que los *CMS* son más propensos a sufrir ataques sino están actualizados.

- Los *CMS* de código abierto suponen una estructura más o menos rígida, donde se puede hacer lo que se quiera siempre que se realice correctamente la configuración propia de la plataforma.

- Los usuarios deben de aprender nociones mínimas para administrar el sitio web o de lo contrario necesitarán recurrir a personal con experiencia para que el *CMS* esté perfectamente actualizado.

Marcos de trabajo:

Se define como a una serie de herramientas que nos permiten y facilitan el desarrollo de un programa, aplicación o página web. Puede tener soporte de programas y bibliotecas, además de otras herramientas para facilitar el desarrollo y la unión de las diferentes partes de un proyecto (Revista digital, 2018).

Ventajas de los *frameworks*:

- Tiene mayor flexibilidad a la hora del desarrollo.
- Se reducen los tiempos con respecto a si no se utilizara un *framework*.

Desventajas de los *frameworks*:

- Hay que desarrollar todos los módulos para poder personalizar los sitios ya que los que vienen por defecto son muy básicos.

Luego de lo antes expuesto, teniendo en cuenta las características que debe tener la propuesta de solución y la experiencia del autor con estas herramientas de desarrollo web, se decide utilizar, para la solución propuesta, un gestor de contenido. A continuación, se realiza un análisis de los gestores de contenido más utilizados actualmente según Mening (2017), en función de seleccionar el más adecuado para la propuesta de solución. Existen varios parámetros para realizar la comparación, en este caso, se tienen en cuenta aquellos que están relacionados a las características de la investigación (CMSmatrix, 2018):

Tabla 2. Comparación de CMS (elaboración propia)

Características	Drupal	Joomla!	WordPress
Licencia libre	Si	Si	Si
¿El sistema hace un seguimiento de quién hizo las adiciones, actualizaciones y eliminaciones? (seguridad)	Si	No	Si
Opción que tiene el usuario para crear áreas de contenido predeterminadas durante la instalación del sitio o en la	Si	No	Si

configuración del sistema (facilidad de uso)			
Opción para generar estilos, temas, colores y logos de forma sencilla para el usuario teniendo conocimiento o no acerca de *HTML* o *CSS*.	Si	No	No
Tomar ventaja de la replicación de base de datos para una mejor escalabilidad(rendimiento)	Si	No	No
Certificación profesional (soporte)	Si	No	Si
Flexibilidad	Si	Si	Si

A partir de la anterior comparación, el autor decide utilizar Drupal como gestor de contenido, el cual, teniendo en cuenta los módulos que deben utilizarse en la propuesta de solución, se empleará en su versión 7.66.

Lenguajes de programación

Según Peralta y otros (2014), un lenguaje de programación es un lenguaje formal creado para describir el conjunto de acciones que un equipo debe ejecutar. El lenguaje de programación está compuesto de una serie de reglas sintácticas y semánticas que permiten expresar instrucciones que posteriormente serán interpretadas por el equipo.

Para el desarrollo de la propuesta de solución, al tener en cuenta que se utiliza el *CMS* Drupal, se emplea el lenguaje de programación *PHP* en su versión 7, lenguaje de uso general de código del lado del servidor, originalmente diseñado para el desarrollo web de contenido dinámico, contiene soporte para bases de datos específicamente *MySql*. Otra de sus características es que es un lenguaje de programación multiplataforma, lo cual hace su uso extensivo a multitud de sistemas operativos como *Windows*, *Mac* o *Linux*. Además, existen protocolos y servicios con los que se puede comunicar *PHP*, como correo con POP3 o SNMP, autenticación de dominio con LDAP e IMAP, y es un lenguaje de programación libre, por lo que se presenta como una alternativa fácil para todo desarrollador (PHP, 2018).

Como lenguajes para el lado del cliente se emplean *JavaScript* en su versión 1.8.5, *HTML* en su versión 5 y *CSS* en la versión 3. *JavaScript* es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase, más conocido como el lenguaje de *script* para páginas web, pero también usado en muchos entornos sin navegador, tales como *node.js*, *Apache CouchDB* y *Adobe Acrobat*. Es un lenguaje *script* multi-paradigma, basado en prototipos, dinámico, soporta estilos de

programación funcional, orientada a objetos e imperativa. Permite incluir macros en páginas web. Estas macros se ejecutan en el ordenador del visitante de dichas páginas, y no en el servidor, algo muy interesante, pues los servidores web suelen estar sobrecargados, mientras que las computadoras de los usuarios no suelen estarlo (MDN, 2019).

HTML5, por su parte, es un estándar que se considera versátil, pues permite realizar una interacción poderosa y simple, mejorando la experiencia de uso por parte del usuario y facilitando la depuración del código web. La utilización de HTML5 permite una mejor experiencia de usuario, un código más amplio, optimización para móviles y compatibilidad múltiple entre navegadores (Nixon, 2014).

El uso de CSS3 permite definir el estilo visual de documentos HTML. Pueden mostrarse distintas hojas de estilo según el dispositivo que se esté utilizando, ya sea versión impresa o versión móvil. Las novedades de CSS3 permiten el ahorro de tiempo y trabajo al poder seguir varias técnicas (bordes redondeados, sombra en el texto, sombra en las cajas, etc.) sin necesidad de usar un editor gráfico (Nixon, 2014).

Biblioteca de código

Con la aparición, hace algunos años, de la web 2.0, Internet ha cambiado y se ha transformado para dar acogida a todas las necesidades de sus usuarios, y por esa razón los sitios web también han tenido que cambiar mucho. A mediados de 2011 se empezó a hablar de los sitios web *responsive* o adaptables a todo tipo de pantallas y dispositivos, fuese cual fuese su tamaño; esta capacidad de adaptación de los sitios web se consiguió utilizando técnicas *CSS* avanzadas para su desarrollo o utilizando *frameworks CSS* como es el caso de *Bootstrap*.

Para el desarrollo del sistema se emplea *Bootstrap 4*, biblioteca multiplataforma de código que tiene como objetivo facilitar el diseño web. Permite utilizar muchos elementos web combinando *HTML5*, *CSS* y *Javascript***.** Permite crear de forma sencilla webs de diseño adaptable, es decir, que se ajusten a cualquier dispositivo y tamaño de pantalla y siempre se vean igual de bien. Es una herramienta de código abierto. Cuenta, además, con implementaciones externas para *WordPress*, Drupal, entre otros gestores de contenido (Puntoabierto, 2016).

Servidor de bases de datos

Un Sistema Gestor de Bases de Datos (SGBD) se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos (Alvarez, 2018). Para la selección del SGBD a utilizar fueron considerados *PostgreSQL* y *MySQL*.

PostgreSQL: Sistema Gestor de Bases de Datos objeto-relacional; basado en el proyecto POSTGRES, de la Universidad de Berkeley. *PostgreSQL* es un sistema objeto-relacional, ya que

incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones y restricciones. Soporta los tipos base y otros de tipo como fecha, monetarios, elementos gráficos y cadenas de bits. También permite la creación de tipos propios. Incluye herencia entre tablas, por lo que se incluye entre los gestores de objetos-relacionales (PostgreSQL, 2014).

PostgreSQL se desempeña mejor en ambientes con altas cargas de usuario y consultas complejas y donde la integridad de los datos es muy importante. Una vez almacenados los datos, proporciona un sistema de respaldos en línea donde se puede ver una tabla en el estado en el que se encontraba en cierta fecha, proporcionando así un método flexible para la rápida recuperación de datos. Otra ventaja importante de este SGBD es la capacidad de su arquitectura para soportar módulos agregados.

MySQL: Sistema gestor de base de datos relacional, disponible para múltiples plataformas. Posee un licenciamiento dual, por un lado, se ofrece bajo la Licencia Pública General de GNU (GNU-GPL por sus siglas en inglés) 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. MySQL posee alta velocidad al realizar las operaciones, lo que le hace que posea un buen rendimiento (Portilla, y otros, 2017). Es considerado como de mediana escala y es adecuado para aplicaciones web de tamaño medio. MySQL permite que varios usuarios a través de múltiples hilos puedan acceder a datos relacionales y soporta múltiples herramientas para la gestión (Kemppainen, 2015).

Selección del sistema gestor de base de datos

Luego de valorar las opciones anteriores se decide usar MySQL en su versión 5.7.21 como sistema gestor de base de datos en la propuesta de solución, ya que su velocidad a la hora de realizar las operaciones, lo hace uno de los gestores que ofrecen mayor rendimiento, además su bajo consumo lo hacen apto para ser ejecutado en una máquina con escasos recursos sin ningún problema. Es, además, el SGBD que se utiliza por defecto con el *CMS* Drupal.

Servidor web

Para el alojamiento del sistema se emplea el servidor web *Apache* 2.4, recomendado para el desarrollo con el *CMS* Drupal (Drupal, 2018). Según Mifsuf (2013), *Apache* es un servidor web de software libre desarrollado por *Apache Software Foundation (ASF).* Desde 1996 es el servidor más utilizado en Internet y es el utilizado en sistemas *GNU/Linux*. Es un servidor robusto y con un ciclo de desarrollo muy rápido, gracias a la gran cantidad de colaboradores con los que dispone.

Herramientas para pruebas de software

Las pruebas de software son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada (Pressman,

2010). En el caso de la propuesta de solución de la investigación, este será validado a través de varias tipos y técnicas de pruebas, algunas de ellas de forma manual y otras mediante el uso de herramientas que permiten la realización de dicha tarea. Entre las herramientas existentes, se pretende utilizar las herramientas *Acunetix Web Vulnerability Scanner* y *Apache JMeter*, para las pruebas de seguridad y de carga y estrés, respectivamente.

Acunetix realiza automáticamente auditorías a aplicaciones web comprobando vulnerabilidades de Inyección SQL, *Cross site scripting* y otras vulnerabilidades que puedan ser explotadas por atacantes.

Apache JMeter es una herramienta diseñada para pruebas de estrés en aplicaciones web simulando las funcionalidades de un navegador o de cualquier otro cliente. Muestra los resultados de las pruebas en una amplia variedad de informes y gráficas, con gran cantidad de variables que permiten interpretar los resultados desde diferentes puntos de vista.

1.5 Conclusiones del capítulo

En este capítulo se han abordado los elementos teóricos que dan sustento a la propuesta de solución del problema planteado, en tal sentido se puede arribar a las siguientes conclusiones:

1. La definición de los principales conceptos asociados al dominio de la presente investigación y las relaciones entre estos, permite alcanzar una mayor comprensión de la propuesta de solución.

2. El análisis de los procesos de gestión de los reportes de incidencias de mantenimiento permite sentar las bases para las funcionalidades que debe brindar la propuesta de solución.

3. El análisis de los sistemas homólogos permite identificar las tendencias en cuanto al desarrollo de herramientas informáticas para la gestión de reportes de incidencias de mantenimiento.

4. La definición del entorno de desarrollo permite especificar las versiones actuales de las tecnologías seleccionadas, que guiadas por la metodología AUP-UCI satisfacen las necesidades de la investigación.

Capítulo 2: Análisis y diseño de la propuesta de solución

2.1 Introducción

Una de las prioridades cuando se desea desarrollar un software, es establecer un entendimiento entre el cliente y el equipo de trabajo en relación con los objetivos a lograr (Labrada, y otros, 2013), realizando un correcto análisis y diseño de dicho sistema. El objetivo de este capítulo es presentar los resultados que se obtuvieron una vez cumplidas las fases de Análisis y Diseño que propone la metodología *AUP*-UCI. Se detallan las características del sistema se especifican los requisitos funcionales y no funcionales del mismo y se presentan los artefactos generados para dar solución al problema planteado en la investigación.

2.2 Modelado conceptual

Un modelo conceptual tiene como objetivo identificar y explicar los conceptos significativos en un dominio de problema, identificando los atributos y las asociaciones existentes entre ellos. Puede ser visto, también como una representación de las cosas, entidades, ideas, conceptos u objetos del "mundo real" o dominio de interés (Sommerville, 2011).

Para lograr un mejor entendimiento de los procesos que requieren informatización, se realizó un modelo conceptual (ver Figura 1), con un total de veintitrés (23) clases y veintitrés (23) relaciones, el cual recoge y describe los conceptos más importantes dentro del contexto del sistema, así como las relaciones entre ellos.

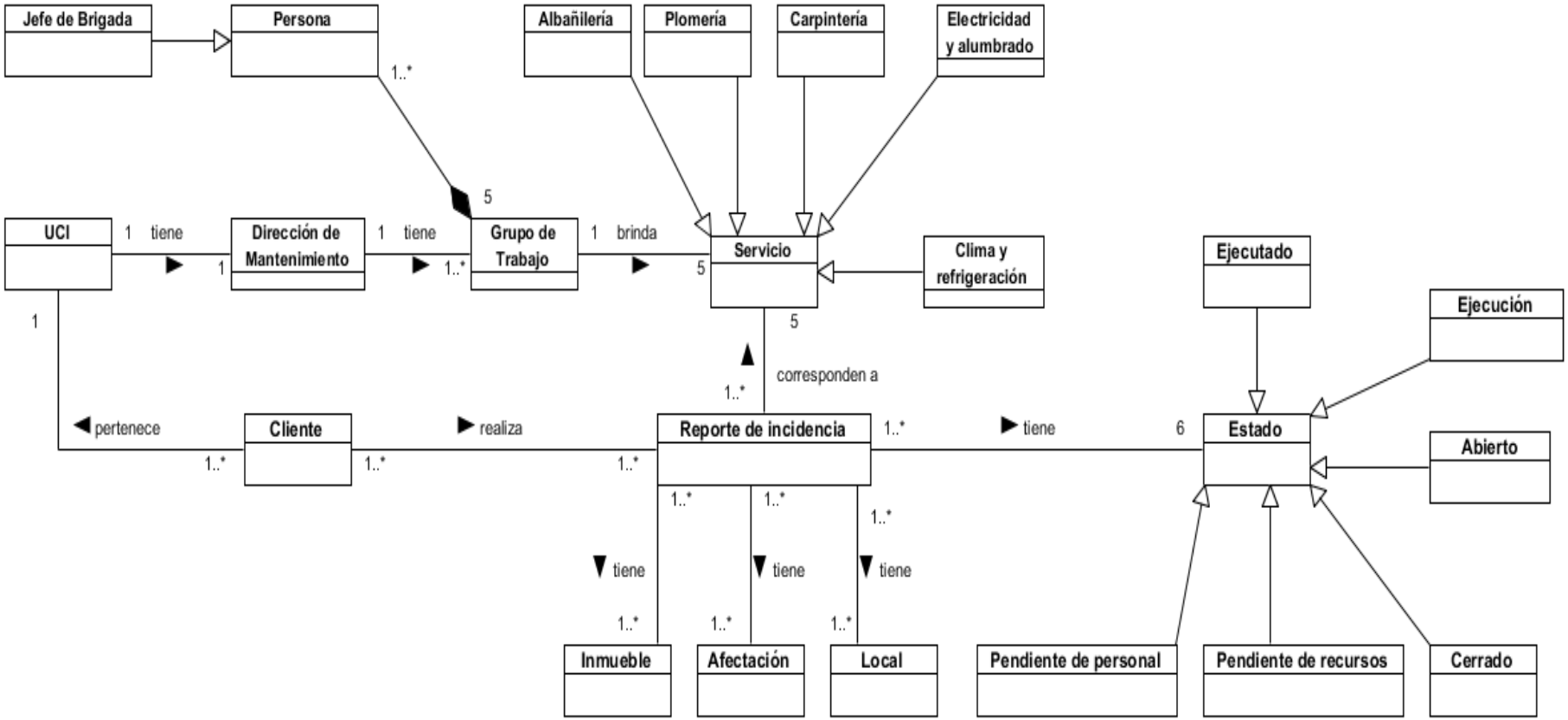

Figura 1. Modelo Conceptual (elaboración propia)

2.3 Requisitos funcionales

Los requisitos funcionales son declaraciones de las funcionalidades que debe cumplir el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares (Pressman, 2010).

Tabla 3. Relación de requisitos funcionales del sistema (elaboración propia)

Requisito	Prioridad	Requisito	Prioridad
RF1: Añadir incidencia	Alta	**RF2:** Editar incidencia	Media
RF3: Eliminar incidencia	Media	**RF4:** Listar incidencia	Media
RF5: Mostrar incidencia	Alta	**RF6:** Filtrar incidencia	Alta
RF7: Cambiar estado de la incidencia	Media	**RF8:** Añadir usuario	Alta
RF9: Editar usuario	Alta	**RF10:**Eliminar usuario	Media
RF11: Mostrar usuario	Baja	**RF12:** Filtrar usuario	Media
RF13: Cambiar estado del usuario	Media	**RF14:** Listar usuario	Alta
RF15: Autenticar usuario		**RF16:** Añadir rol	
RF17: Mostrar rol	Alta	**RF18:** Editar rol	Media
RF19: Eliminar rol	Media	**RF20:** Añadir término de taxonomía	Baja
RF21: Mostrar término de taxonomía	Alta	**RF22:** Editar término de taxonomía	Media
RF23: Eliminar término de taxonomía	Media	**RF24:** Listar Brigada	Baja
RF25: Añadir Brigada	Alta	**RF26:** Editar Brigada	Alta
RF27: Eliminar Brigada	Media	**RF28:** Mostrar Brigada	Media
RF29: Añadir Trabajador	Alta	**RF30:** Editar Trabajador	Alta
RF31: Eliminar Trabajador	Media	**RF32:** Mostrar Trabajador	Media

RF33: Listar trabajador		**RF34:** Importar datos de los trabajadores del servicio web del Sistema de Gestión Universitaria	Media
RF35: Cambiar asignación de permisos	Media	**RF36:** Generar reporte servicio con más incidencias	Alta
RF37: Generar reporte afectación con más incidencias	Alta	**RF38:** Generar reporte apartamento con más incidencias	Alta
RF39: Generar reporte inmueble con más incidencias	Alta	**RF40:** Generar reporte local con más incidencias	Alta
RF41: Enviar notificación al correo	Media		

2.4 Requisitos no funcionales

Los requisitos no funcionales representan características generales y restricciones de la aplicación o sistema que se esté desarrollando. Son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. Suelen expresarse de una manera general y sin hacer referencia a algún modulo, transacción o característica del sistema (PMOinformatica, 2015).

RnF 1. Usabilidad:

RnF 1.1: El sistema para la gestión de reportes de incidencias deberá ser una aplicación web.

RnF 1.2: La aplicación debe presentar una interfaz agradable e intuitiva para el usuario.

RnF 2. Confiabilidad:

RnF 2.1: El sistema debe ser tolerante a fallos, y mostrar solo la información necesaria para orientar al usuario.

RnF 3. Eficiencia:

RnF 3.1: El sistema debe permitir que los usuarios interactúen con él de manera concurrente.

RnF 3.1: El tiempo de demora de una petición al servidor debe ser menor de cinco (5) segundos aproximadamente.

RnF 4. Soporte:

RnF 4.1: El sistema contará con toda la documentación definida en el expediente de proyecto asociada a su proceso de desarrollo para las actividades de soporte.

RnF 5. Restricciones de diseño e implementación:

RnF 5.1: El componente deberá ser desarrollado en su totalidad con tecnologías y componentes de código abierto.

RnF 6. Software:

RnF 6.1: Para el despliegue del sistema se debe contar en el servidor de aplicaciones web con: PHP versión 7 y Apache 2.4.

RnF 6.2: Para el despliegue del sistema se debe contar en el servidor de bases de datos con MySQL 5.7.21.

RnF 6.3: La comunicación entre el cliente y el servidor de aplicaciones se realiza a través del protocolo HTTPS.

RnF 7. Hardware:

RnF 7.1: Para la ejecución del componente se requiere que la PC cliente tenga los siguientes componentes de hardware: Pentium 4 o superior, 512 MB RAM y 200 MB disco duro disponible como mínimo.

2.5 Diagrama de Casos de uso del sistema:

Un caso de uso captura un contrato que describe el comportamiento del sistema en diferentes condiciones mientras este responde a la petición de uno de sus usuarios (Pointeau, 2018). A continuación, se muestra el Diagrama de Casos de Uso del Sistema (DCUS) para la propuesta de solución, el cual está compuesto por siete (7) actores, veintiséis (26) relaciones y veinte (20) casos de uso.

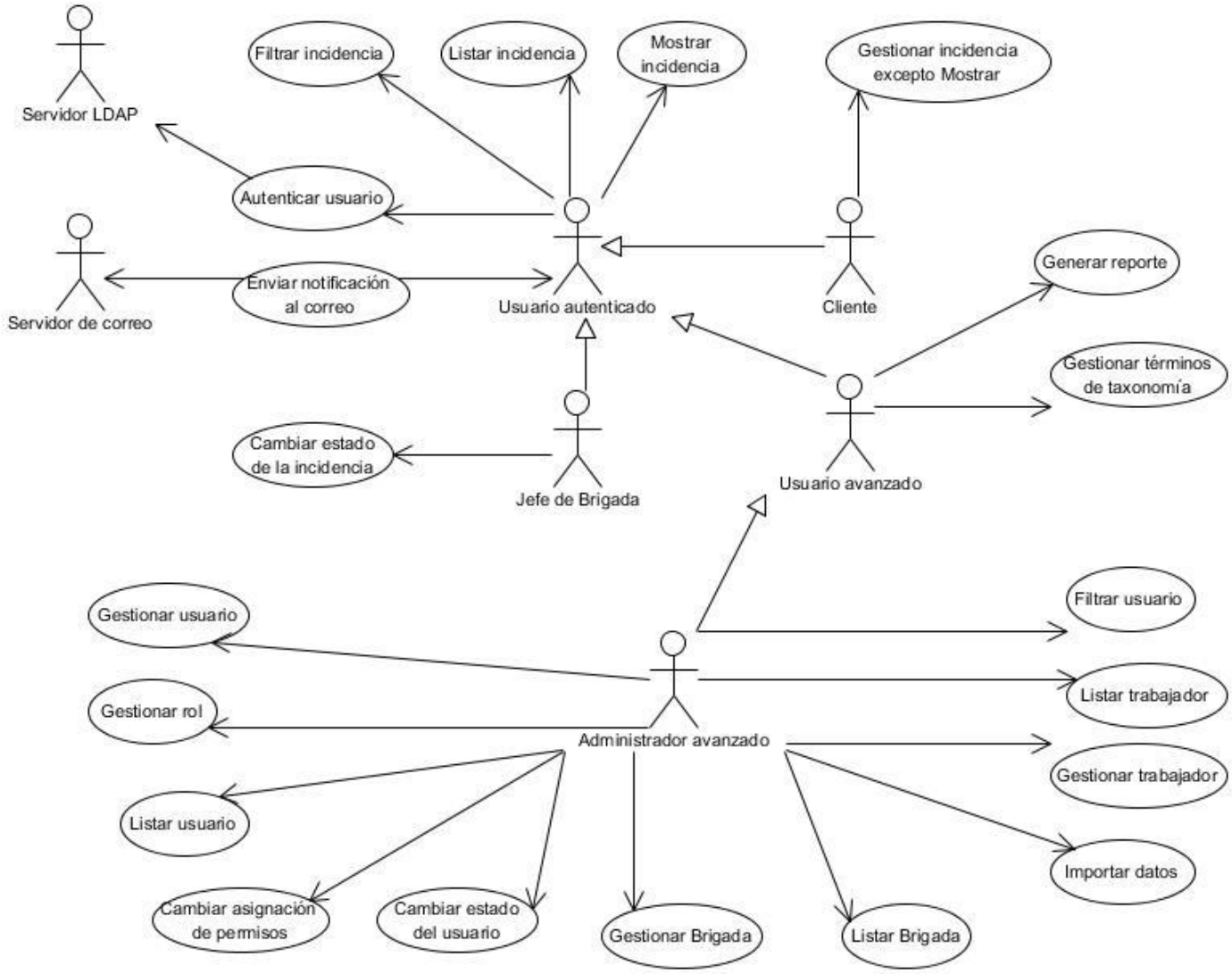

Figura 2. DCUS (elaboración propia)

2.6 Descripción textual de los Casos de uso:

Tabla 4. Descripción textual del CU Gestionar incidencia (elaboración propia)

Objetivo	Permitir añadir, editar, listar y eliminar incidencias, además de publicarlas y retirarlas de publicación.
Actores	Administrador avanzado
Resumen	El caso de uso se inicia cuando el administrador avanzado decide añadir, editar, listar y eliminar incidencia.
Complejidad	Alta.
Prioridad	Alto.
Precondiciones	Administrador ya autenticado.
Postcondiciones	Se registró, modificó o se eliminó incidencia(s).
Flujo de eventos	
Flujo básico "Gestionar incidencia"	
Actor	**Sistema**

1.	Selecciona de la página principal la opción “Gestionar incidencia”.	
2.		Muestra una vista con un listado de las incidencias y permite Añadir, Editar, Listar o Eliminar incidencia(s).
3.	Desea añadir, editar, listar o eliminar incidencias(s).	
4.		Da la posibilidad de realizar alguna de las siguientes acciones: Si decide añadir una incidencia, ir a la sección “Añadir incidencia”. Si decide editar los datos de una incidencia, ir a la sección “Editar incidencia”. Si decide eliminar incidencia(s), ir a la sección “Listar incidencia”. Si decide eliminar uno(s) incidencia(s), ir a la sección “Eliminar incidencia”.
Prototipo elemental de interfaz gráfica		
No aplica		
Flujos Alternos		
2a. Listado de incidencias vacío		
Actor		**Sistema**
1.		Carga una pantalla en blanco sin incidencias registradas.
3a. Desea realizar una búsqueda		
Actor		**Sistema**
1.		Ver el CU Buscar incidencia.
Sección 1: “Añadir incidencia”		
Flujo básico Añadir incidencia		
Actor		**Sistema**
1.	Presiona el botón “Añadir incidencia”.	
2.		Muestra el listado de los tipos de incidencias existentes, el usuario selecciona el deseado y el sistema muestra un formulario en el que se introducen los datos del mismo. Servicio Afectación Inmueble Local Teléfono Asunto Descripción Y el botón “Guardar”.

3.	Introduce los datos y presiona el botón “Guardar”.	
4.		Verifica que todos los campos estén llenos.
5.		Verifica que los datos introducidos estén correctos.
6.		Verifica que esta incidencia no exista.
7.		Almacena los datos de la(s) incidencia(s) y muestra el mensaje “Registro satisfactorio”. *Finalizando así el caso de uso.*
Prototipo elemental de interfaz gráfica		
No aplica		
Flujos Alternos		
4a. Campos vacíos		
Actor		**Sistema**
1.		Muestra el mensaje “Campos vacíos”.
5a. Datos incorrectos		
Actor		**Sistema**
1.		Muestra el mensaje “Datos incorrectos”.
6a. Incidencia existente		
Actor		**Sistema**
1.		Muestra el mensaje “Incidencia existente”.
Sección 2: “Editar Incidencia”		
Flujo básico Editar Incidencia		
Actor		**Sistema**
1.	Marca la opción “Editar” de uno de las incidencias mostradas y presiona el botón “Editar”.	
2.		Muestra un formulario en el que se modifican los datos: Servicio Afectación Inmueble Local Teléfono Asunto Descripción Y el botón “Guardar”.
3.	Realiza las actualizaciones deseadas y presiona el botón “Editar Incidencia”.	
4.		Verifica que todos los campos estén llenos.
5.		Verifica que los datos introducidos estén

		correctos.
6.		Actualiza la información incorporada a la incidencia y se emite un mensaje "Modificación realizada satisfactoriamente". *Finalizando así el caso de uso.*
Prototipo elemental de interfaz gráfica		
No aplica		
Flujos Alternos		
4a. Campos vacíos		
Actor		**Sistema**
1.		Muestra el mensaje "Campos vacíos".
5a. Datos incorrectos		
Actor		**Sistema**
1.		Muestra el mensaje "Datos incorrectos".
Sección 3: "Eliminar Incidencia"		
Flujo básico Eliminar Incidencia		
Actor		**Sistema**
1.	Marca la opción "Eliminar" de una(s) incidencia(s) mostrada(s) y presiona la opción "Aceptar".	
		Muestra el mensaje de confirmación "¿Está seguro que desea eliminar la(s) incidencia(s) seleccionada(s)?". Y los botones "Aceptar" y "Cancelar".
2.	Presiona el botón "Aceptar".	
		Elimina la(s) incidencia(s) seleccionada(s) y emite el mensaje "Incidencia eliminada satisfactoriamente". *Finalizando así el caso de uso.*
Prototipo elemental de interfaz gráfica		
No aplica		
Flujo Alterno		
2a. Cancelar eliminación de incidencia		
Actor		**Sistema**
1.	Presiona el botón "Cancelar".	
2.		Vuelve al paso 2 del flujo básico "Gestionar incidencia".
Sección 3: "Listar Incidencia"		
Flujo básico Listar Incidencia		
Actor		**Sistema**
1.	Marca la opción "Listar" incidencia y presiona	

	la opción “Aceptar”.		
2.			Muestra la(s) incidencia(s) registrada(s) en el sistema.
Relaciones	**CU Incluidos**	No Aplica.	
	CU Extendidos	No Aplica.	
Requisitos no funcionales	No Aplica.		
Asuntos Pendientes	No Aplica.		

2.7 Descripción de la arquitectura de software y los patrones de diseño:

2.7.1. Arquitectura de software:

Al utilizarse el *CMS* Drupal para el desarrollo de la propuesta de solución, la arquitectura de software a utilizar es la que éste define, la cual es una arquitectura n-capas o dividida específicamente en 5 capas que son descritas a continuación (Drupal, 2018):

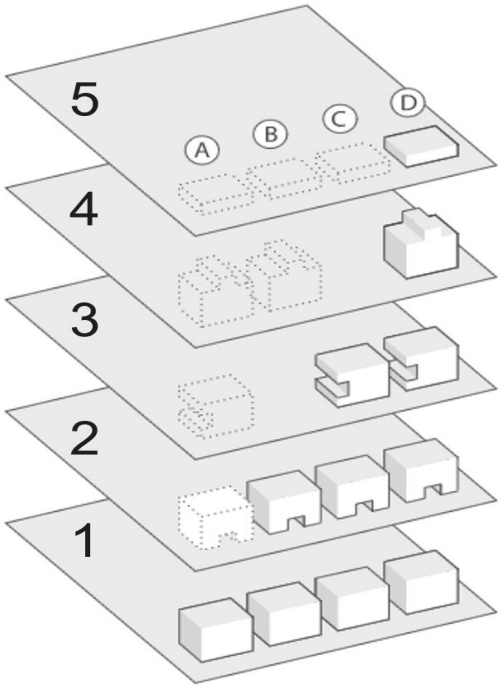

Figura 3. Arquitectura de Drupal *(Drupal, 2018)*

5. Plantillas: Esta capa establece la apariencia gráfica que se le muestra al usuario. Esta separación entre la información y los estilos permite cambiar la apariencia del portal web sin necesidad de modificar los contenidos. En la propuesta de solución esta capa contiene el tema **Incidencias** aplicado al sistema, así como su codificación en el lenguaje CSS.

4. **Permisos de usuario:** Determina lo que pueden ver y hacer los usuarios. Los permisos son asignados (o no) a varios roles, los cuales son asignados a los usuarios.

3. **Bloques y menús:** Los bloques a menudo proporcionan la salida de un módulo o se pueden crear para mostrar lo que se desee, y luego se pueden colocar en varios puntos (regiones) en el diseño de su plantilla (tema). Los bloques se pueden configurar para generar resultados de varias maneras, así como solo para mostrar en ciertas páginas definidas, o solo para ciertos usuarios definidos. Por su parte, los menús, son navegadores en Drupal, constituyen elementos fundamentales que proporcionan enlaces a todas las páginas creadas en Drupal.

2. Módulos: Engloba los elementos que operan sobre los nodos otorgando funcionalidades a Drupal. Permiten incrementar sus capacidades o adaptarlas a las necesidades de cada portal web. Dentro de los módulos usados en la propuesta de solución, se encuentran gestionar incidencia, gestionar brigadas, los cuales fueron implementados para lograr una mejor gestión de los reportes de incidencias.

1. Base de Datos: Esta capa es la encargada de gestionar el acceso a la información almacenada referente al funcionamiento del sistema y a los contenidos que serán mostrados a través del tema activo. La base de datos de la propuesta de solución tiene por nombre “incidencia”.

2.7.2. Patrones de diseño:

Los diseñadores expertos en orientación a objetos (y también otros diseñadores de software), van formando un amplio repertorio de principios generales y de expresiones que los guían a crear el software. A unos y a otras podemos asignarles el nombre de patrones, si se codifican en un formato estructurado que describe el problema y su solución, y si se les asigna un nombre (Larman, 2004). A continuación, se presentan los patrones utilizados en el desarrollo del Sistema para la gestión de reportes de incidencias de mantenimiento en la Universidad de las Ciencias Informáticas.

Patrones *Gang of Four (GOF)*

El catálogo de patrones más famoso es el contenido en el libro “*Design Patterns: Elements of Reusable Object-Oriented Software*”, también conocido como: El libro GOF (*Gang-Of-Four Book*). Según este documento (Gamma, y otros, 1997), estos patrones se clasifican por su propósito en creacionales, estructurales y de composición, mientras que respecto a su ámbito se clasifican en clases y objetos. Los patrones GOF utilizado en la investigación son los siguientes:

Decorador: Permite añadir responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. Se evidencia en el sistema en el módulo que permite la gestión de reportes de incidencia de mantenimiento.

Instancia única: Este patrón está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un objeto único. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. En de Drupal se utiliza este patrón de diseño en diversas tareas como la gestión de manejo de conexiones con la base de datos y pensando en los módulos y temas de Drupal como objetos para llevar a cabo la gestión de dichos elementos. Un ejemplo que evidencia este patrón en el sistema de gestión de reportes de incidencia de mantenimiento es el proceso de creación de tipos de contenidos, al cual se le asigna un identificador propio que evita la existencia en la base datos de elementos repetidos constituyendo así una instancia única.

Cadena de responsabilidades: El sistema de menús de Drupal sigue este patrón. En cada solicitud de la página, el menú del sistema determina si hay un módulo para gestionar la solicitud y si el usuario tiene acceso a los recursos solicitados. Para ello, el mensaje se pasa a la opción del menú correspondiente a la vía de la solicitud. Si el elemento de menú no puede manejar la petición, se pasa a otro. Esto continúa hasta que un módulo se encarga de la petición, un módulo niega el acceso para el usuario, o la cadena se ha agotado (Drupal, 2018). La implementación del hook_menu es un ejemplo claro que genera un menú de configuración del módulo.

2.8 Diagramas de clases del diseño:

Los diagramas de clase (DC) pueden usarse cuando se desarrolla un modelo de sistema orientado a objetos para mostrar las clases en un sistema y las asociaciones entre dichas clases (Sommerville, 2011). Estas representaciones contienen información acerca de las clases, asociaciones, atributos, métodos y dependencias. A continuación, se muestra el DCD para el CU gestionar incidencia:

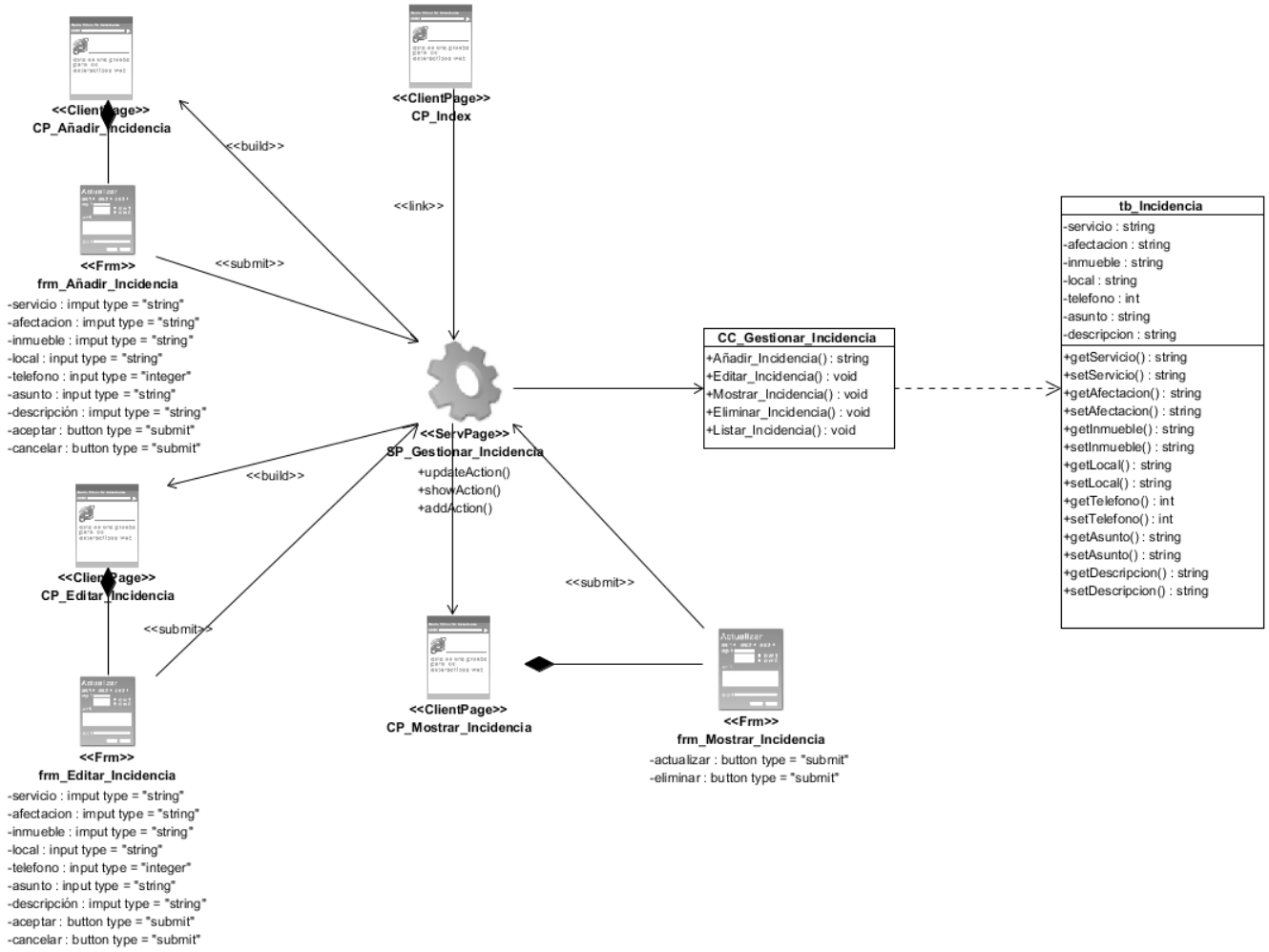

Figura 4. DCD CU Gestionar incidencia (elaboración propia)

2.9 Diagramas de secuencia

Los diagramas de secuencia (DS) en el UML se usan principalmente para modelar las interacciones entre los actores y los objetos en un sistema, así como las interacciones entre los objetos en sí *(Sommerville, 2011)*. Para la presente investigación se generaron los DS correspondientes a los casos de uso del sistema, a continuación, se muestra el DS para el RF Añadir Incidencia, en los anexos, se muestran los DS correspondientes a los RF Actualizar Incidencia y Eliminar Incidencia (Anexo 2 y Anexo 3):

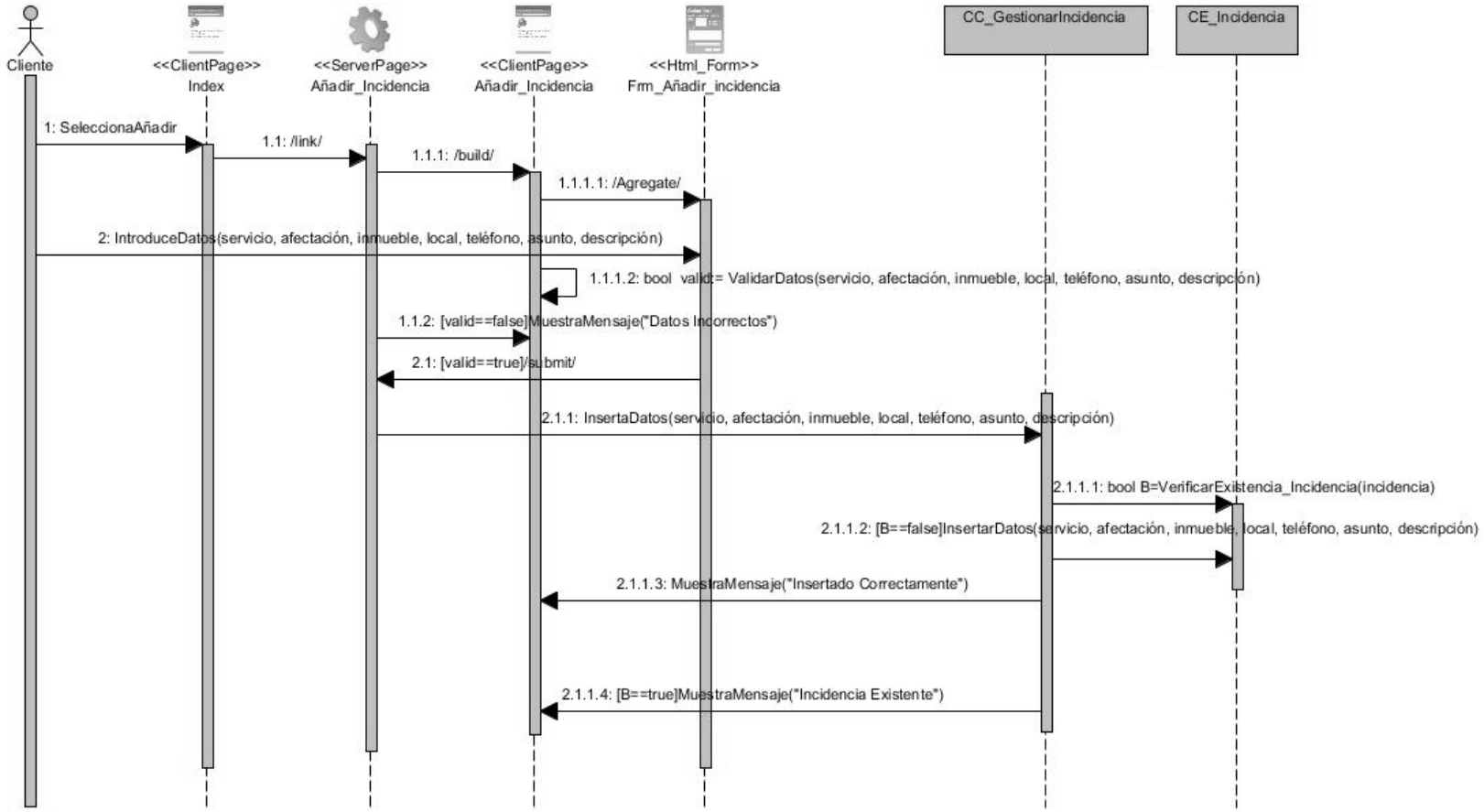

Figura 5.. DS CU Añadir Incidencia (elaboración propia)

2.10 Conclusiones del capítulo

Después de realizado el análisis y diseño de la propuesta de solución y haber generado los diferentes artefactos que dispone la metodología AUP, se puede concluir lo siguiente:

1. Los requisitos funcionales y no funcionales identificados a partir del proceso de determinación de requisitos permiten desarrollar las distintas funcionalidades que debe cumplir el sistema para solucionar las necesidades detectadas.
2. Las actividades contempladas en el análisis y diseño permiten dar una mayor descripción de los elementos del sistema, permitiendo una mejor comprensión para su implementación.
3. El diseño de los diagramas de clases y de secuencia, facilita la visión en cuanto a composición física y lógica del módulo.
4. La generación de todos los artefactos requeridos por el modelo de desarrollo, documentan la solución propuesta, lo cual facilita su posterior mantenimiento (actualización o adición de funcionalidades).

Capítulo 3. Implementación y pruebas de la propuesta de solución

3.1 Introducción

Antes de escribir una sola línea de código, es fundamental haber comprendido bien el problema que se pretende resolver y haber aplicado principios básicos de diseño que permitan construir un sistema de calidad. Una vez que se sabe qué funciones debe desempeñar el sistema (análisis) y se ha decidido cómo organizar sus distintos componentes (diseño), es el momento de pasar a la etapa de implementación. En esta fase, según Russo (2011), se toma como punto de partida el modelo de la fase anterior y se procede a programar los diseños especificados, los cuales son implementados en términos de componentes, ficheros de código fuente y ejecutables.

Por otra parte, el desarrollo de un software es algo complejo y son innumerables las posibilidades de cometer errores. Por esta razón todo proceso de implementación debe ir acompañado de alguna actividad que garantice la calidad. Las pruebas de validación constituyen una base para garantizar la aceptación favorable de una aplicación informática por parte del usuario. Con la realización de las mismas se pretende encontrar y documentar los errores que tiene un sistema, validar los requisitos y comprobar que estos fueron implementados correctamente. Este capítulo tiene como objetivo, documentar los resultados de las fases de implementación del módulo y de la estrategia de pruebas desarrollada.

3.2 Modelo de Despliegue

Representa de forma visual las relaciones físicas que existen entre los componentes de software y hardware en el sistema. Los nodos son elementos de hardware sobre los cuales pueden ejecutarse los elementos de software**.** Se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño.

Descripción de elementos e interfaces de comunicación:

Dispositivo PC_Cliente: La estación de trabajo necesita un navegador web para conectarse al sistema hospedado en el servidor de aplicaciones utilizando el protocolo de comunicación HTTPS.

Servidor de aplicaciones: Es la estación de trabajo que hospeda el código fuente de la aplicación y que le brinda al usuario las interfaces para realizar los procesos del sistema. Es un servidor de aplicaciones web *Apache* y se comunica con el servidor de base de datos donde se almacenan los datos de la aplicación realizando la comunicación mediante el protocolo TCP.

Servidor de BD: Servidor de bases de datos *MySQL*, es el encargado del almacenamiento de los datos del sistema. Se comunica con el servidor de aplicaciones del sistema, posibilitando el acceso mediante el usuario con privilegios para las operaciones determinadas a realizarse en el mismo.

Servidor de correo: Este servidor es el encargado del envío de correos electrónicos comunicándose a través del protocolo SMTP.

LDAP: (Protocolo Ligero/Simplificado de Acceso a Directorio) es un protocolo de tipo cliente-servidor para acceder a un servicio de directorio. Los servicios de directorio facilitan el acceso a información organizada a una gran variedad de aplicaciones. Estos servicios le permiten a los usuarios y aplicaciones autorizadas buscar información de personas, computadoras, aplicaciones y dispositivos red.

A continuación, se presenta al modelo de despliegue correspondiente a la propuesta de solución:

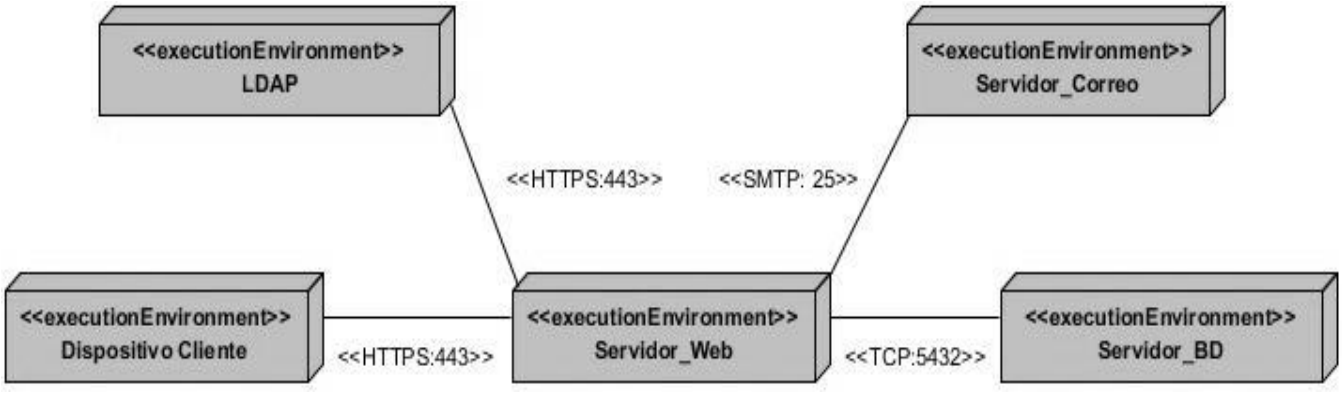

Figura 6. Diagrama de despliegue del sistema (elaboración propia)

3.3 Diagrama de componentes

El diagrama de componentes muestra los componentes de un sistema de software conectados por las relaciones de dependencias lógicas entre cada uno de ellos. Provee una vista arquitectónica de alto nivel del sistema, ayudando a los desarrolladores a visualizar el camino de la implementación. Cada componente representa una unidad del código (fuente, binario o ejecutable), que permite mostrar las dependencias en tiempo de compilación y ejecución. La realización del diagrama posibilita tomar decisiones respecto a las tareas de implementación y los requisitos.

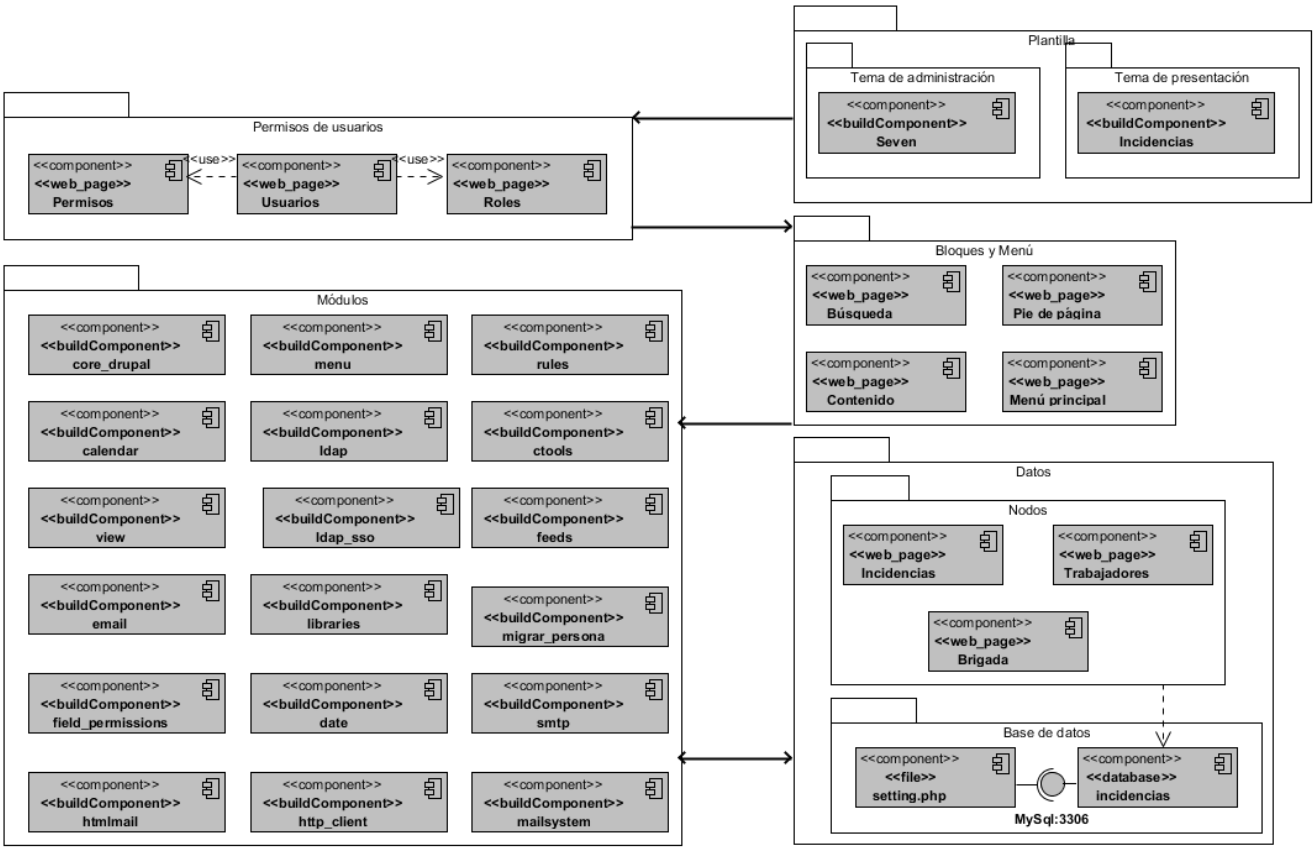

Figura 7. Diagrama de componentes (elaboración propia)

3.4 Estándares de codificación

Los estándares de codificación constituyen un principio esencial en el desarrollo de software. Garantizan que el código obtenido sea fácil de leer, entendido y modificado independientemente de quien haya sido el desarrollador del producto. Son una guía para el equipo de desarrollo, permiten asegurar que el código presente calidad y no contenga errores (Muñoz, 2016). Drupal proporciona a sus desarrolladores un conjunto de normas para fomentar el código de una forma uniforme para todos. A continuación, se detallan los estándares de codificación utilizados en la implementación de la solución propuesta.

Cuando se escribe en PHP, siempre se deben utilizar las etiquetas **<?php** y **?>**, y en ningún caso la versión corta **<?** y **?>**. En general se omite la etiqueta de cierre de PHP (**?>**) al final de los archivos .module y .inc. Esta convención evita que se puedan quedar olvidados espacios no deseados al final del archivo (después de la etiqueta de cierre ?**>**), que serían identificados como salida HTML y podrían provocar un error muy típico, “Cannot modify header information-headers alrealy sent by…”. Por tanto, la etiqueta de cierre final del archivo (**?>**) es opcional en Drupal.

Indentación

La indentación consiste en insertar espacios en blanco o tabuladores en determinadas líneas de código para facilitar su comprensión. En programación se emplea indentación para anidar elementos. En Drupal se debe indentar con 2 espacios, nunca con tabuladores. Además, no se debe dejar espacios en blanco al final de cada línea (Solucionex, 2016).

Operadores

Los operadores binarios, que se utilizan entre dos valores, deben separarse de estos valores, a ambos lados del operador, por un espacio. Por ejemplo, $max_age = 452, en el lugar de $max_age=452. Esto se aplica a operadores como +,-,*,/,=,==,!=,>,<,. (Concatenación de cadenas), .=, +=, -=, etc. Los operadores unarios como ++, -- no deben tener separación (Solucionex, 2016).

Uso de comillas

Se utilizan tanto comillas simples como la (‘cadena’) como las comillas dobles (“cadena”) para delimitar las cadenas de caracteres. Las comillas dobles son necesarias si se desean incluir variables dentro de las cadenas de texto (Solucionex, 2016).

Uso de punto y coma (;) en código PHP

Aunque PHP permite escribir líneas de código individuales sin el terminador de línea (;), como por ejemplo <?php print $title ?>. En Drupal es siempre obligatorio: <?php print $title; ?>. Es importante señalar que el cierre de la etiqueta php es opcional (Torres, 2012).

Estructuras de control

Con respecto a las estructuras de control, hay que tener en cuenta las siguientes normas:

Debe haber un espacio entre el comando que define la estructura (if, while, for, etc) y el paréntesis de apertura. Esto es así para no confundir las estructuras de control con la nomenclatura de las funciones.

La llave de apertura ({) se situará en la misma línea que la definición de la estructura, separada por un espacio.

Se recomienda usar siempre las llaves {} aun en los casos en que no sea obligatorio su uso (una sola “línea” de código dentro de la estructura de control).

Las estructuras *else* y *elseif* se escribirán en la línea siguiente al cierre de la sentencia anterior (Solucionex, 2016).

Funciones

Los nombres de las funciones deben estar escritos en minúsculas y las palabras separadas por guion bajo. Además, se debe incluir siempre como prefijo el nombre del módulo o tema, para evitar así duplicidad de funciones. En su declaración, después del nombre de la función, el paréntesis de inicio de los argumentos debe ir sin espacio. Cada argumento debe ir separado por un espacio, después de la coma del argumento anterior (Solucionex, 2016).

Arreglos (*Arrays*)

Los valores dentro de un array (o matriz) se deben separar por un espacio (después de la coma que los separa). El operador => debe separarse por un espacio a ambos lados. Cuando la línea de declaración del array supera los 80 caracteres, cada elemento se debe escribir en una única línea. En este último caso, la coma de separación del último elemento también se escribirá, aunque no existan más elementos. De esta forma se evitan errores al añadir nuevos elementos al vector.

Nombres de archivos

Los nombres de archivos deben escribirse siempre en minúscula. La única excepción son los archivos de documentación, que tendrán extensión .txt y el nombre en mayúscula. Por ejemplo README.txt.

Comentar el código

Para la realización de comentarios suelen emplear /*/ para comentarios en varias líneas y // para comentarios de una única línea. Se deben escribir frases completas, comenzándolas con mayúscula y terminándolas con un punto. En caso de que en el comentario se haga referencia a una constante, esta deberá escribirse en mayúscula (por ejemplo: TRUE o FALSE).

3.5 Validación del Sistema de gestión de reportes de incidencias de mantenimiento de la Universidad de las Ciencias Informáticas

3.5.1 Pruebas de rendimiento (carga y estrés)

Las pruebas de rendimiento se diseñan para asegurar que el sistema pueda procesar su carga esperada. Éstas se ocupan tanto de demostrar que el sistema satisface sus requerimientos, como de descubrir problemas y defectos en el sistema (Sommerville, 2011).

Las pruebas de carga consisten en simular una carga de trabajo similar y superior a la que tendrá cuando el sistema esté funcionando, con el fin de detectar si el software instalado (programas y aplicaciones) cumple con los requerimientos de muchos usuarios simultáneos y también si el hardware (servidor y el equipamiento computacional de redes y enlace que lo conecta a Internet) es capaz de soportar la cantidad de visitas esperadas (Pressman, 2010).

Las pruebas de estrés evalúan la robustez y la confiabilidad del software sometiéndolo a condiciones de uso extremas. Entre estas condiciones se incluyen el envío excesivo de peticiones y la ejecución en condiciones de hardware limitadas. El objetivo es saturar el programa hasta un punto de quiebre donde aparezcan defectos potencialmente peligrosos (Pressman, 2010).

Resultados de las pruebas de rendimiento

Para las pruebas de rendimiento se utiliza el software Apache Jmeter v2.10. Para ello se definen las propiedades de las PC implicadas.

Hardware de prueba (PC cliente):

Sistema Operativo: Windows v.10

Microprocesador: Intel(R) Core(TM) i3-4100U CPU @2.40GHz

Memoria RAM: 4.00 GB

Disco Duro: 1024 GB

Hardware de prueba (PC servidor):

Sistema Operativo: Linux Mint v.18

Microprocesador: Intel(R) Core(TM) i3-7100U CPU @2.40GHz

Memoria RAM: 8.00 GB

Disco Duro: 1024 GB

Software instalado en ambas PC:

Tipo de servidor web: Apache 2.4.

Plataforma: SO Linux (PC servidor) y SO Windows (PC cliente).

Servidor de BD: MySQL 5.7.24

Luego de definido el hardware se configuran los parámetros del Apache JMeter logrando un ambiente de simulación con un total de 50 usuarios conectados concurrentemente, se realizan peticiones a diferentes páginas del Sistema de gestión de reportes de incidencias de mantenimiento de la Universidad de las Ciencias Informáticas. En la (6) se puede observar los resultados obtenidos por el sistema.

Análisis de los resultados de las pruebas de rendimiento

Para un mejor entendimiento de las pruebas de Rendimiento, se explica cada parámetro que la compone a continuación:

Usuarios: Total de usuarios.

 # Muestras: El número de peticiones.

Media: El tiempo medio transcurrido en milisegundos para un conjunto de resultados.

Mín: El mínimo tiempo transcurrido en milisegundos para las muestras de la URL dada.

Máx: El máximo tiempo transcurrido en un milisegundo para las muestras de la URL dada.

% Error: Porcentaje de las peticiones con errores.

Rendimiento: Rendimiento medido en base a peticiones por segundo/minuto/hora.

Kb/s Recibidos: Rendimiento medido en Kbytes por segundos.

Tabla 5. Resultados de las pruebas de rendimiento (elaboración propia)

Usuarios	# Muestras	Media	Mín	Máx	% Error	Rendimiento (peticiones/segundos)	Kb/s Recibidos
50	200	2963	1904	3869	0.4 %	1.6	1
100	400	3896	2675	5142	1.7%	2.3	3.6
1000	4000	5632	3589	6752	2.9%	5.7	6.9

Las pruebas realizadas muestran que el sistema es capaz de responder a 200 peticiones de 50 usuarios conectados simultáneamente en un tiempo promedio de 2963 milisegundos (2.9 segundos aproximadamente) con 0.4 % de error, esto evidencia que el sistema puede procesar la carga esperada.

Por otra parte, se realizaron 400 peticiones iniciadas por 100 usuarios y en este caso el sistema respondió en 3896 milisegundos (3.8 segundos aproximadamente) como tiempo promedio. Esto demuestra que el sistema puede procesar la carga esperada, aunque no fue capaz de responder correctamente el 1.7% de las peticiones realizadas.

Por último, y con el objetivo de analizar el comportamiento del sistema en condiciones extremas, se realizó una prueba de estrés para un conjunto de 1000 usuarios conectados simultáneamente. En este caso, el módulo responde a las 4000 peticiones en un tiempo promedio de 5632 milisegundos (5.6 segundos aproximadamente), con un porciento de error de 2.9. Este resultado está estrechamente relacionado al entorno donde se realizó la prueba, el cual no es un servidor dedicado sino un cliente habilitado.

3.5.2 Pruebas funcionales

Las pruebas funcionales utilizan la especificación del producto para diseñar los casos de prueba (entrada–salida esperada), existiendo técnicas como partición de equivalencia, análisis del valor límite, grafo causa-efecto y conjetura de errores; la prueba funcional debe mostrar errores y disconformidades con la especificación, y no enfocarse en determinar si el programa cumple con su especificación (Escobar, y otros, 2015).

Tabla 6. Caso de prueba: Añadir incidencia (elaboración propia)

Caso de prueba Añadir incidencia	
Código de caso de prueba	**Nombre de requisito**: Añadir incidencia
Nombre de la persona que realiza la prueba: Reynaldo Columbié Mamalova	
Descripción de la prueba: Prueba a la funcionalidad añadir incidencia.	

Entrada/Pasos de ejecución: Se seleccionan los siguientes datos para crear la incidencia.
Servicio **Afectación** **Tipo de inmueble** **Inmueble** **Teléfono** **Local** **Asunto** **Descripción de la incidencia** **Adjunto** **Estado** **Brigada** **Fecha** El administrador presiona el botón guardar y si los datos están correctos se crea la incidencia que podrá ser visualizado desde la vista Incidencias para los usuarios autenticados en el sistema, si existe algún dato incorrecto el sistema mostrará un mensaje de error y señalará el campo erróneo en rojo para su posterior corrección.
Evaluación de la prueba: Satisfactoria.

Tabla 7. Caso de prueba: Editar incidencia

Caso de prueba Editar incidencia	
	Nombre de requisito: Editar incidencia
Nombre de la persona que realiza la prueba: Reynaldo Columbié Mamalova	
Descripción de la prueba: Prueba a la funcionalidad editar incidencia.	
Entrada/Pasos de ejecución: Se modifican los siguientes datos para editar un contenido de tipo incidencia	
Datos actuales **Servicio:** Carpintería **Afectación:** Bisagra rota **Tipo de inmueble:** Residencia **Inmueble:** Edificio 1	

Apartamento: 1101 **Teléfono:** 8851 **Local:** Cocina **Asunto:** nueva **Descripción de la incidencia:** bisagra rota **Adjunto:** **Estado:** Abierta **Brigada:** Carpintería **Fecha:** 22-3-2019 **<u>Datos después de editados</u>** **Servicio:** Plomería **Afectación:** Tupición **Tipo de inmueble:** Otro **Inmueble:** Edificio 2 Apartamento: 2101 **Teléfono:** 8841 **Local:** Comedor **Asunto:** nasd **Descripción de la incidencia:** tupición en fregadero **Adjunto:** **Estado:** Abierta **Brigada:** Plomería **Fecha:** 22-3-2019 El administrador presiona el botón guardar y si los datos están correctos actualiza la incidencia, que podrá ser visualizado desde la vista Incidencias, si existe algún dato incorrecto el sistema mostrará un mensaje de error y señalará el campo erróneo en rojo para su posterior corrección.
Evaluación de la prueba: Satisfactoria.

Resultado de las pruebas funcionales

Como resultado final de las pruebas funcionales, se obtuvo, en una primera iteración, un total de diez (10) no conformidades, divididas en tres (5) de ortografía, una (1) de redacción, dos (2) de funcionalidad y cuatro (2) de validación. De estas, se resolvieron seis (6), y cuatro (4) quedaron

pendientes. En una segunda iteración, no se identifican nuevas no conformidades y de las cuatro (4) pendientes, solo una (1) se mantuvo para la próxima iteración, donde fue resuelta y no se detectan nuevas no conformidades. En una cuarta iteración no se identifican nuevas inconformidades, obteniendo, de esta manera, resultados satisfactorios. La siguiente gráfica, muestra los resultados antes descritos:

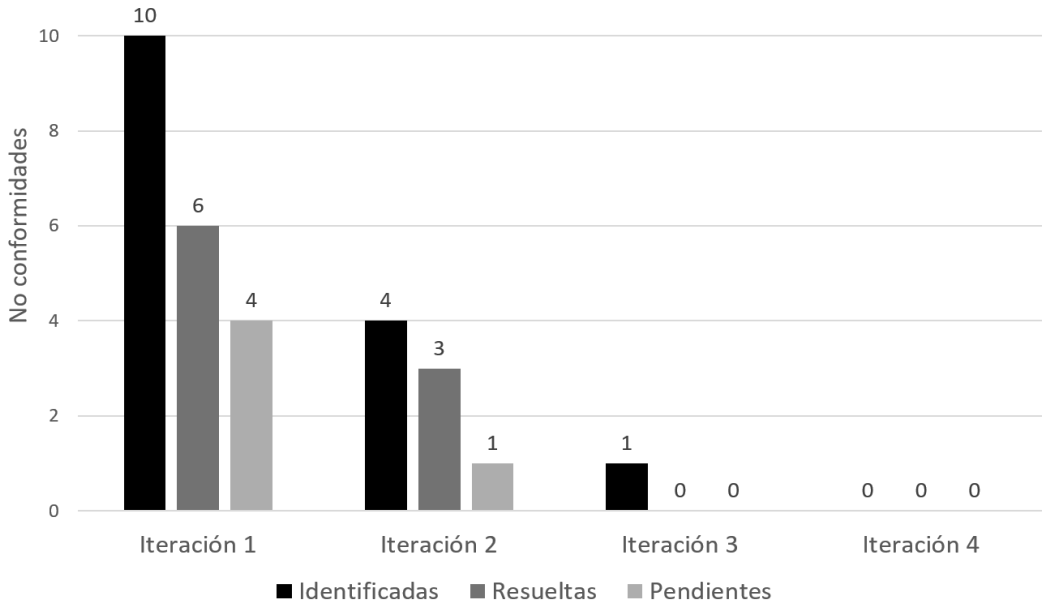

Figura 8. Gráfica de resultados de las pruebas funcionales (elaboración propia)

Entre las principales no conformidades detectadas en el proceso de pruebas funcionales se encuentran:

- Errores de estructuración de los contenidos mostrados en las vistas.
- Los datos introducidos por el usuario de forma incorrecta eran guardados en la base de datos sin validación previa.
- Opciones que no funcionaban.
- El sistema mostraba mensajes de error con datos sobre las variables.

3.5.3 Pruebas de seguridad

Las pruebas de seguridad se realizan para comprobar que los mecanismos de protección integrados en el sistema realmente lo protejan de irrupciones inapropiadas (Pressman, 2010). Además, se encargan de certificar que los datos y las funciones del sistema solo son accesibles por los actores debidamente autorizados (Toll, y otros, 2007).

Resultados de las pruebas de seguridad

Para la realización de este tipo de prueba, se empleó la herramienta *Acunetix Web Vulnerability Scanner,* caracterizada en el capítulo 2. En una primera iteración, se obtuvo un total de quince (15) no conformidades, divididas en seis (6) de nivel medio, seis (6) de nivel bajo y tres (3) de carácter informacional. De las de nivel medio, destacó el uso del protocolo no seguro para el envío de datos, así como campos de contraseña con auto completamiento activado. Las de nivel bajo estuvieron relacionadas con problemas para la protección contra ataques de fuerza bruta a la página de autenticación, así como directorios que pueden ser accesibles directamente sin pasar la autenticación y la protección de las cookies y las sesiones en el navegador. De carácter informativo fueron detectadas una dirección de correo y una posible cuenta de usuario en un fichero. Todas estas deficiencias fueron corregidas en la primera iteración, y para una segunda, no se identificó ninguna nueva, por lo cual se obtuvo finalmente una herramienta con seguridad y que cumple con el requisito de confiabilidad definido para la misma. Los resultados antes descritos, se muestran a continuación en la siguiente gráfica:

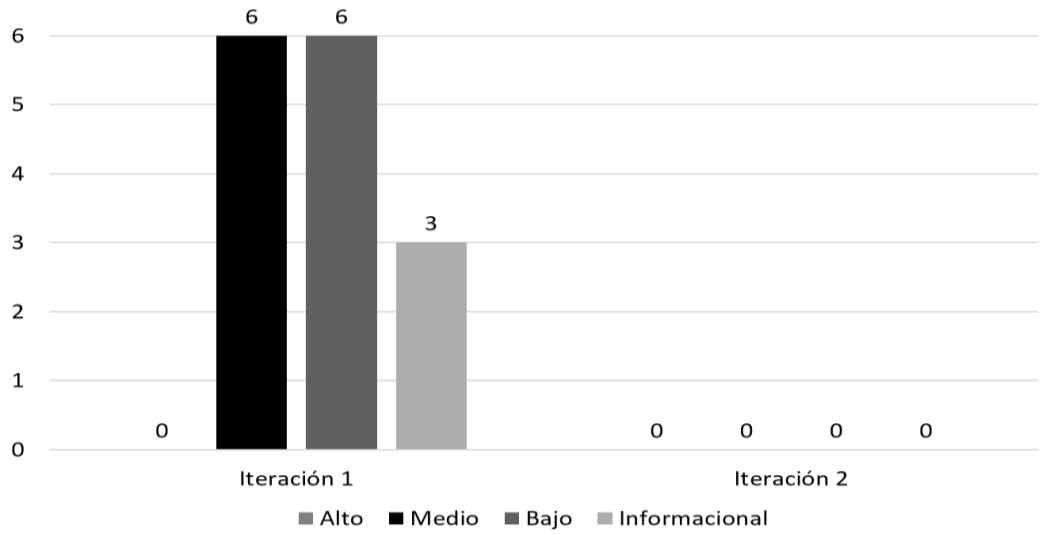

Figura 9. Gráfica de resultados de las pruebas de seguridad (elaboración propia)

3.5.4 Pruebas de aceptación

Según Huaraca (2013), el uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece. Sin embargo, antes de empezar a usarlo es muy difícil determinar si sus ventajas realmente justifican su uso. El mejor instrumento para esta determinación es la llamada prueba de aceptación. En esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique.

Por su parte, la Junta Internacional de Cualificaciones de Pruebas de Software (ISTQB por sus siglas en inglés) define la “Aceptación” como: Pruebas formales con respecto a las necesidades del usuario,

requerimientos y procesos de negocio, realizadas para determinar si un sistema satisface los criterios de aceptación que permitan que el usuario, cliente u otra entidad autorizada pueda determinar si acepta o no el sistema (PMOinformatica, 2015).

Para realizar la prueba de aceptación, se entregó la aplicación al cliente, el cual emitió su criterio a través de una carta de aceptación, a partir de sus consideraciones respecto a las ventajas que ofrece el sistema y las necesidades que resuelve.

3.5.5 Pruebas de usabilidad

Según diversos estándares de la Ingeniería de Software, se puede definir la usabilidad como el grado en el que un producto puede ser utilizado por usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción, en un determinado contexto de uso. Como se puede apreciar, la usabilidad de un sistema está ligada a usuarios, necesidades y condiciones específicas (Carcasés, 2016). De manera general, el término usabilidad es empleado para referirse a la capacidad que posee un producto de ser utilizado por los usuarios de forma fácil, eficiente y con satisfacción, en un determinado contexto de uso.

Se puede decir que el proceso de prueba de usabilidad se enfoca en satisfacer las necesidades de los usuarios finales basados en métricas definidas durante la planeación. Para la realización de las pruebas de usabilidad, se hace uso de la “Lista de Chequeo de Usabilidad para sitios web”, desarrollada por los especialistas del grupo de Seguridad del Departamento de Evaluación de Productos de Software (DEPSW), perteneciente al Centro Nacional de Calidad de Software (CALISOFT). A continuación, se muestran los resultados de dichas pruebas.

Tabla 8. Pruebas de usabilidad (elaboración propia)

Categoría de los indicadores	Indicadores	Proceden	Correctos	Incorrectos
Visibilidad del sistema	17	14	12	2
Lenguaje común entre sistema y usuario	12	10	9	1
Libertad y control por parte del usuario	29	24	13	11
Consistencia y estándares	33	23	19	4
Estética y diseño minimalista	17	15	15	0

Prevención de errores	8	7	6	1
Ayuda y documentación	11	11	6	5
Flexibilidad y eficiencia	6	4	3	1
Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de los errores	11	11	4	7
Total	144	119	87	32

Como se observa en la tabla, de los 144 parámetros originales de la lista de chequeo, solo proceden 119. En la primera iteración se evaluaron como correctos 87 parámetros, identificando 32 no conformidades, para un 73.10 % de usabilidad. Los principales problemas estuvieron relacionados con la libertad y control por parte del usuario en el sistema.

3.6 Interfaces principales del Sistema de gestión de reportes de incidencias de mantenimiento

Una vez desarrollado el Sistema de gestión de reportes de incidencias de mantenimiento, es posible visualizar las pantallas principales del mismo, donde se observa el resultado obtenido durante la implementación de los requisitos funcionales descritos en el capítulo 2.

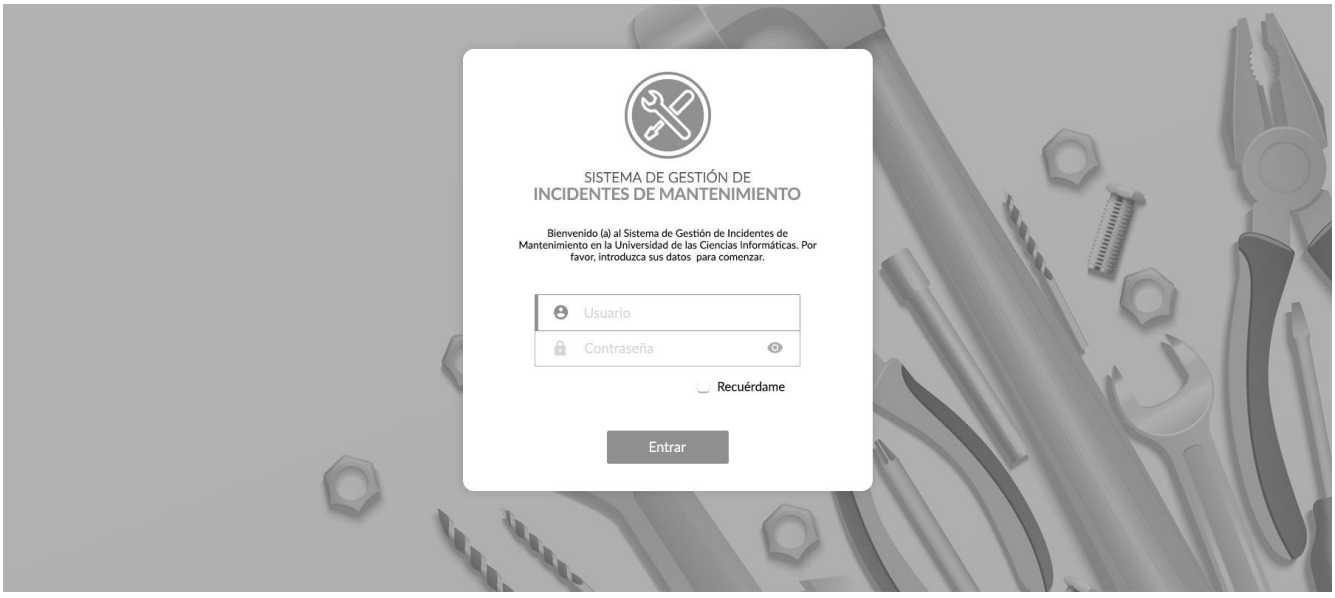

Figura 10. Captura de pantalla del sistema (Autenticar Usuario) (elaboración propia)

Figura 11. Captura de pantalla del sistema (Mostrar Incidencia) (elaboración propia)

3.7 Conclusiones del capítulo

En este capítulo se han abordado los elementos de la implementación de la propuesta de solución, así como las pruebas realizadas a la misma y los resultados obtenidos; lo cual permite arribar a las siguientes conclusiones:

1. La elaboración del diagrama de componentes permite una mejor comprensión de la estructura de los componentes del sistema implementado.
2. El correcto uso de los estándares de codificación, permite que el código del sistema desarrollado sea legible para lograr una fácil y mejor comprensión del mismo, lo cual es de utilidad para el mantenimiento del sistema.
3. La implementación del sistema permite la obtención de una aplicación funcional y completamente operativa.
4. El proceso de validación de la propuesta de solución, a través de la estrategia de pruebas especificada, arroja como resultado que el sistema implementado responde a los requerimientos definidos por el cliente.

Conclusiones generales

De manera general, la presente investigación concluyó con el desarrollo del Sistema de gestión de reportes de incidencias de mantenimiento de la Universidad de las Ciencias Informáticas, el cual sirve de apoyo al proceso de gestión de incidencias de mantenimiento de la Universidad de las Ciencias Informáticas.

1. El análisis y la fundamentación teórica de los principales conceptos asociados a la investigación, permitió lograr una mayor comprensión del alcance de la investigación y esclarecer su objeto de estudio.
2. La sistematización del marco teórico de la investigación científica y del estado actual de las herramientas informáticas para la gestión de reportes de incidencias, posibilitó la definición del ambiente de desarrollo para la implementación de la solución propuesta.
3. La integración de diversas áreas del conocimiento como son la ingeniería y gestión de software, base de datos, programación, entre otras, permitió el análisis, diseño e implementación del sistema.
4. La solución fue validada a partir de la definición correcta de una estrategia de pruebas, que permitió comprobar el correcto funcionamiento del Sistema para la gestión de reportes de incidencias de mantenimiento en la Universidad de las Ciencias Informáticas, a partir de los requerimientos definidos por el cliente.

Recomendaciones

Para el desarrollo de futuras investigaciones relacionadas con la presente, se propone:

1. Migrar el sistema a la versión 8 de Drupal.
2. Implementar una funcionalidad que permita importar los datos de los trabajadores de mantenimiento desde Sistema de Gestión Universitaria.
3. Implementar una funcionalidad que permita importar las áreas de la universidad desde el Sistema de Gestión Universitaria.

Referencias bibliográficas

Activalink. 2018. Activalink.com. [En línea] Noviembre de 2018. https://www.activalink.com/que-es-suite-crm.

Alvarez, S. 2018. Sistemas gestores de bases de datos. [En línea] 2018. http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html..

Araujo, Pedro Bernabé y Rodríguez, Sebastián Alberto. 2013. *Janeiro Studio.* 2013.

Carcasés, Arianne Ferrer. 2016. *Guía para los procesos de Usabilidad que se desarrollan en el Laboratorio de Pruebas de Software (LPS) de la Dirección de Calidad-UCI.* 2016.

CMSmatrix. 2018. [En línea] 2018. http://www.cmsmatrix.org/matrix/cms-matrix.

Drupal. 2018. [En línea] 2018. https://www.drupal.org/drupal-7.0/es.

—. 2018. Drupal.org. [En línea] 2018. https://www.drupal.org/docs/7/understanding-drupal/overview.

Escobar, Milton Eduardo y Fuertes, Walter Marcelo. 2015. *Modelo formal de pruebas funcionales de software para alcanzar el Nivel de Madurez Integrado 2.* 2015.

Exevi. 2019. [En línea] 2019. https://www.exevi.com/soluciones/otrs-software-libre-de-atencion-al-cliente/.

Fonseca, Yudelsi. 2015. *Sistema para la gestión de reportes de la Dirección de Mantenimiento de la Universidad de Ciencias Informáticas v2.0.* La Habana : Universidad de las Ciencias Informáticas, 2015.

Gamma, Erich, y otros. 1997. *Design Patterns: Elements of Reusable Object-Oriented Software.* 1997.

Gomez, Sandra. 2013. Reporte de incidencias con Mantis. [En línea] 2013. http://www.globetesting.com/2013/01/reporte-de-incidencias-mantis/.

HDKey. 2018. HDKey Help Desk Manager. [En línea] Noviembre de 2018. https://www.kmkey.com/software_help_desk/.

Huaraca, Abner Valdez. 2013. *Pruebas de Sistemas y Pruebas de Aceptación.* s.l. : FIS-UNICA, 2013.

Kemppainen, T. 2015. Data Archive Project. [En línea] Helsinki MetropoliaUniversity of Applied Sciences, 2015. http://www.theseus.fi/bitstream/handle/10024/93069/Kemppainen_TimoPekka.pdf?sequ.

Knowdo. 2018. Knowdo.org. [En línea] Octubre de 2018. http://www.knowdo.org/knowledge/39-sistemas-web.

Labrada, Evelyn y Aragón, Yaniel. 2013. *Desarrollo del Módulo de Gestión de Reportes Estadísticos para el sistema AiresProxyAudit.* La Habana : s.n., 2013.

Larman, Craig. 2004. *UML y patrones. Introducción al análisis y diseño orientado a objetos.* La Habana : Félix Varela, 2004.

LQMS. 2014. Occurrence management. *Sistema de gestión de la calidad en el laboratorio.* 2014.

Martínez, Alejandro y Martínez, Raúl. 2014. Guía a Rational Unified Process. Albacete : Escuela Politécnica Superior de Albacete. Universidad de Castilla la Mancha, 2014.

Martinez, Esteban. 2015. Prezi.com. [En línea] 2015. https://prezi.com/-zoqwmyhq2ls/que-es-un-reporte-y-cual-es-su-uso/.

MDN. 2019. MDN web docs moz://a. [En línea] 23 de Marzo de 2019. https://developer.mozilla.org/es/docs/Web/JavaScript.

Mening, Robert. 2017. Website Setup. *Popular CMS by Market Share.* [En línea] Diciembre de 2017. https://websitesetup.org/popular-cms/.

MES. 2019. [En línea] 2019. https://www.mes.gob.cu/es/ingreso/instituciones/universidad-de-las-ciencias-informaticas.

Mifsuf, E. 2013. Apache. España : s.n., 2013.

Muñoz, Isabela. 2016. *Estándares de codificación.* 2016.

Naranjo, Francisco José. 2015. Sistemas de Gestión: Valor Estratégico de las Organizaciones. 2015.

Nixon, R. 2014. *Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5.* s.l. : O'Reilly Media, Inc., 2014.

Peralta, Claudia y Durán, D. 2014. *Módulos de edición de plantillas y recepción de órdenes de impresión para el Sistema de Personalización de Documentos de Identidad basado en tecnologías libres. Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.* La Habana : Universidad de las Ciencias informáticas, 2014.

Pérez, Yoan y Hurtado, Gabriel. 2015. *SRMC: Sistema gestor para reportes de incidencias de mantenimiento constructivo en la Universidad de las Ciencias Informáticas.* La Habana : Universidad de las Ciencias Informáticas, 2015.

PHP. 2018. php.net. *PHP: Hypertext Preprocessor.* [En línea] 2018. php.net/manual/es/intro-whatis.php.

PMOinformatica. 2015. PMOinformatica.com. [En línea] 2015. PMOinformatica.com.

Pointeau, Arthur. 2018. [En línea] Marzo de 2018. https://openclassrooms.com/en/courses/4990961-planea-tu-proyecto-con-uml/4996511-diagramas-de-casos-de-uso.

Portilla, Johan y Bernal, Mireya. 2017. *Comparación del rendimiento de los comandos Insert, Select y Delet en los sistemas gestores de bases de datos Oracle y MYSQL.* Bogotá : Inventum, 2017. Vol. 12, 23.

PostgreSQL. 2014. PostgreSQL.org. [En línea] Diciembre de 2014. [Citado el: 20 de Noviembre de 2016.] https://www.postgresql.org/about/press/presskit94/es/.

Pressman, Roger S. 2010. *Ingeniería de Software, un enfoque práctico. Séptima Edición.* Madrid : s.n., 2010.

Protecnus. 2018. Protecnus. [En línea] Noviembre de 2018. https://www.protecnus.com/.

Puntoabierto. 2016. puntoabierto.net. [En línea] Agosto de 2016. https://puntoabierto.net/blog/que-es-bootstrap-y-cuales-son-sus-ventajas.

Revista digital. 2018. Revista digital INESEM. [En línea] 2018. https://revistadigital.inesem.es/informatica-y-tics/cms-vs-framework-ventajas-desventajas/.

Russo, Patricia. 2011. *Gestión documental en las Organizaciones.* Barcelona : UOC, 2011. 978-84-9788-863-9.

Samaniego, C. 2013. *Incidencias del control interno en la optimización de la gestión de las micro empresas en el distrito de Chaclacayo. .* Lima : Universidad San Martín de Porras, 2013.

Sánchez, Tamara Rodríguez. 2015. Metodología de desarrollo para la Actividad productiva de la UCI v1.2. . La Habana : s.n., 2015.

Solucionex. 2016. Solucionex. [En línea] octubre de 2016. https://www.solucionex.com/blog/un-apunte-sobre-buenas-practicas-de-programacion-para-desarrollar-modulos-en-drupal-8.

Sommerville, Ian. 2011. *Ingeniería de Software.* México : Pearson Educación de México, 2011. 978607-32-0603-7.

Techopedia. 2017. Modeling Lenguage. [En línea] 2017. https://www.techopedia.com/definition/20810/modeling-language.

Toll, Yuniet y Mendoza, Yilennis. 2007. *Propuesta de manual de procedimiento de Pruebas de Sistema y su aplicación en el Proyecto CICPC.* 2007.

Torres, Jose Raul. 2016. Slideshare.net. [En línea] 2016. https://es.slideshare.net/jrtorresb/que-es-el-mantenimiento.

Torres, Michelle. 2012. MIchelleTorres. [En línea] agosto de 2012. https://www.solucionex.com/blog/un-apunte-sobre-buenas-practicas-de-programacion-para-desarrollar-modulos-en-drupal-8.

UCI. 2018. Universidad de las Ciencias Informáticas. [En línea] 2018. https://www.uci.cu/universidad/la-uci-de-un-vistazo.

Anexos

Anexo 1. Entrevista al cliente para conocer la necesidad del desarrollo de la propuesta de solución y definir los requisitos funcionales y no funcionales

Estimado: Se necesita de su cooperación en una investigación para una tesis de

pregrado. Por ello, sería de gran ayuda que respondiera lo siguiente:

1. ¿Considera que a día de hoy la Dirección de Mantenimiento cumple con su objetivo de satisfacer al personal al que se atiende, teniendo en cuenta la gran cantidad de personal que reside en la UCI?

2. ¿Cree usted que se cuenta con los recursos y medios necesarios para dar una respuesta a la persona afectada?

3. ¿Existe alguna herramienta en la Universidad para la gestión de los reportes de incidencia?

4. En caso afirmativo, ¿cuáles son sus características?

5. A partir de esas características, ¿considera que el sistema posee lo necesario para atender las incidencias que se reportan diariamente?

6. ¿Considera que el sistema deba permitir que el sistema muestre al usuario autor de una incidencia siempre el estado actualizado de la misma desde que este la crea hasta que se cierra?

7. ¿Considera el sistema deba generar reportes, para una mejor gestión de las incidencias y contribución a la toma de decisiones?

8. ¿Qué otras características, considera que deba presentar el sistema, en cuanto usabilidad, seguridad, interfaz u otro aspecto que garantice su calidad?

Anexo 2. Diagrama de secuencia para el RF Actualizar Incidencia

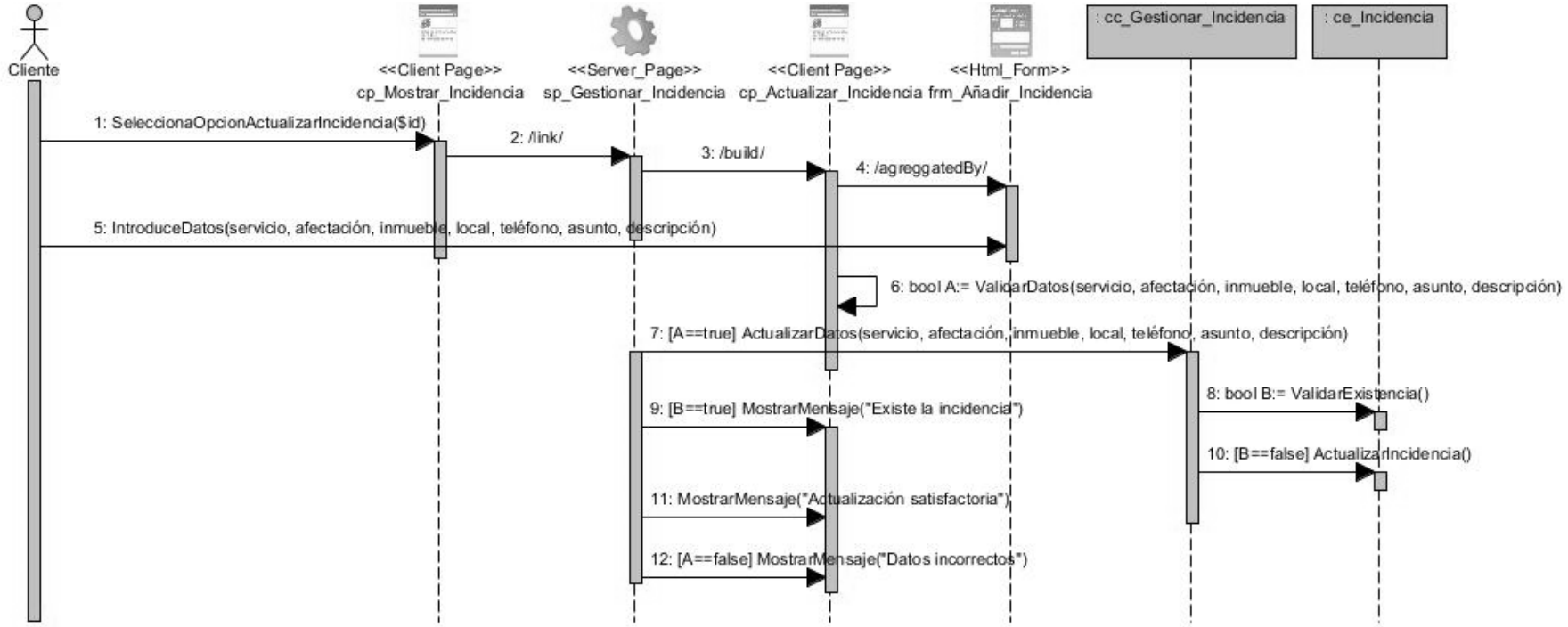

Figura 12. Diagrama de secuencia para el RF Actualizar Incidencia

Anexo 3. Diagrama de secuencia para el RF Eliminar Incidencia

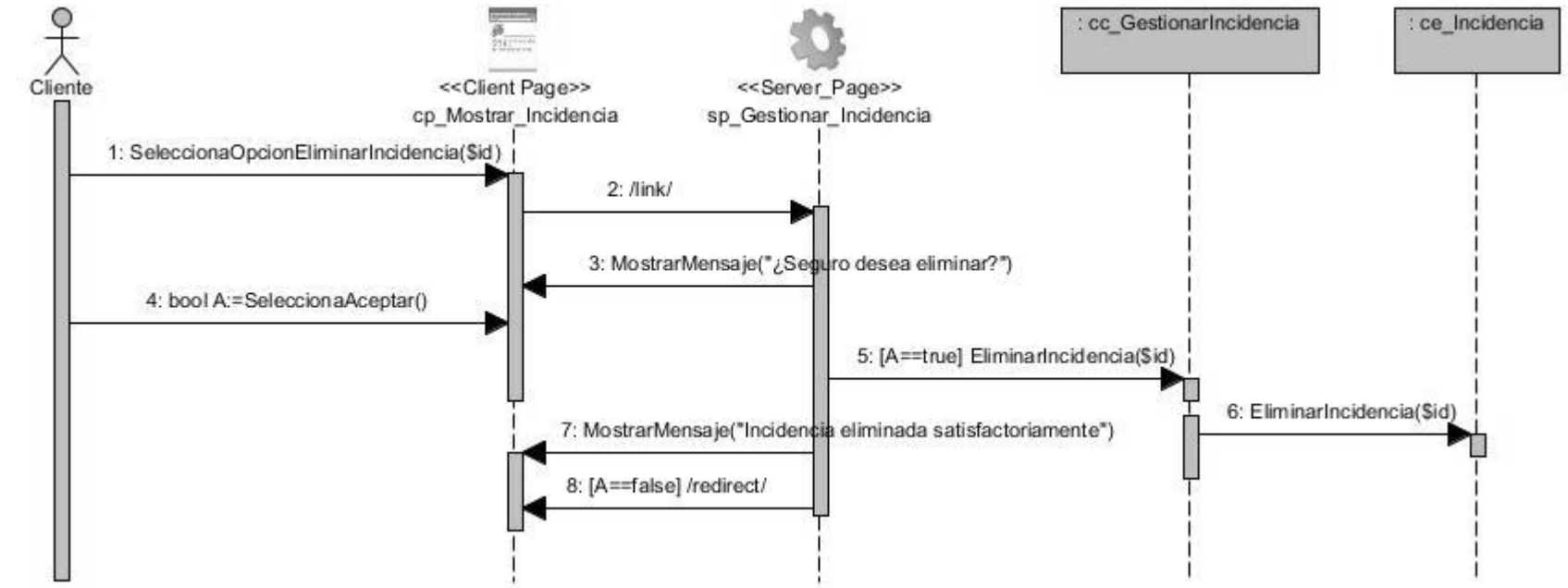

Figura 13. Diagrama de secuencia para el RF Eliminar Incidencia