Universidad de las Ciencias Informáticas

Facultad 4

Centro de Consultoría y Desarrollo de Arquitecturas Empresariales

Componente de gestión de incidencias para el portal Liferay Community

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

Autor: Addiel Tamayo Quintero

Tutores: Ing. Armando Masó Mosqueda

Ing. Susana Alba Silot

La Habana, Julio de 2017 "Año 59 de la Revolución"

Declaro ser el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Armando Masó Mosqueda — **Tutor**

Susana Alba Silot — **Tutora**

_____

Addiel Tamayo Quintero

Autor

Datos de contacto

Tutor: Amando Masó Mosqueda.

Ciudadanía: Cubana.

Institución: Unión de Informáticos en Cuba (UIC).

Título: Ingeniero en Ciencias Informáticas.

E-mail: amaso@uniondeinformaticos.cu

Tutora: Susana Alba Silot.

Ciudadanía: Cubana.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniera en Ciencias Informáticas.

E-mail: salba@uci.cu.

Dedicatoria

Al creador de todas las cosas. Definitivamente, Dios por estar siempre presente.

Mis padres que han sabido formarme con buenos sentimientos, hábitos y valores, lo cual me ha ayudado a salir adelante en los momentos más difíciles.

A mi hermano que siempre ha estado junto a mí y brindándome su apoyo.

A mi abuelo, que ya no está al lado mío, pero su cariño prevalece siempre en mi corazón, fue un hombre tolerante, honesto, bondadoso y generoso, todos estos valores me los inculcó él en mi niñez.

A mi familia en general, porque me han brindado su apoyo incondicional y por compartir conmigo buenos y malos momentos.

Agradecimientos

En primer lugar doy infinitamente gracias a Dios, por haberme dado fuerza y valor para culminar esta etapa de mi vida.

A mi novia Susana porque siempre ha estado ahí para mí cuando más la he necesitado, que durante estos años de carrera ha sabido apoyarme para continuar y nunca renunciar, gracias por su amor incondicional, TE AMO.

Agradecer hoy y siempre a mi familia porque a pesar de no estar presentes físicamente, sé que procuran mi bienestar, y está claro que si no fuese por el esfuerzo realizado por ellos, mis estudios no hubiesen sido posibles. A mis padres Addiel y Alina, mis abuelos, mi hermano Abdías, porque a pesar de la distancia, el ánimo, apoyo y alegría que me brindan me dan la fortaleza necesaria para seguir adelante.

A Maso y Yuni, por su amistad y apoyo incondicional, agradecerle por su paciencia, comprensibilidad y atención.

A todos mis amigos pasados y presentes; los buenos, los malos, por ayudarme a crecer y madurar como persona, a mis compañeros del aula, del Dota, del edificio, del apto, en especial al piquete del 87, Alexi, Ever, Pacheco, Roberto por brindarme su ayuda y por compartir conmigo todos los buenos y malos momentos durante 5 años a lo largo de esta carrera.

A todos los que de una forma u otra lograron que mi sueño de ser profesional se convirtiera en realidad, gracias.

Resumen

Las empresas a nivel mundial necesitan aumentar los canales de interacción con el cliente y mejorar su experiencia y satisfacción. Además de lograr diferencias competitivas y el uso efectivo de las nuevas tecnologías, de manera que no solo sea una mejora en el servicio[1] actual, sino una fuente para corregir y evitar que algunos incidentes ocurridos se repitan en el futuro. Permitiendo que todos estos componentes y gestiones se encuentren interconectados.

Una de las tecnologías que satisface muchos de estos objetivos es el llamado portal, al proporcionar a las personas el acceso a información, aplicaciones y procesos de negocio.

Portal *Liferay* es una de estas plataformas web, que tiene como objetivo ofrecer el máximo beneficio para sus clientes. Fue desarrollado principalmente para satisfacer las necesidades de las empresas.

En la presente investigación se argumenta el desarrollo de un *portlet*[2] de gestión de incidencias que permitirá a las entidades, administrar con mayor control los procesos y monitorizar los servicios a través del portal *Liferay*. El ciclo de desarrollo de la solución estuvo regido por la Metodología Unificada Ágil (AUP) adaptada a los procesos productivos de la Universidad de las Ciencias Informáticas (UCI). Las tecnologías y herramientas empleadas para la aplicación son libres (*Open Source*), con excepción de Visual Paradigm for UML.

Palabras clave: componente, gestión de incidencias, *liferay*, *portlet*.

[1] Servicio: Se refiere a la gestión de las incidencias.

[2] En la investigación se va a tratar el *portlet* como componente.

Índice de contenidos

Índice de figuras

Índice de tablas

Introducción

Las empresas de estos tiempos necesitan agilidad de negocio e interactividad con los clientes para así poder solucionar las necesidades o incidencias que surgen en el día a día. Además de desarrollar servicios a través del buen uso de las nuevas tecnologías que interconecten los procesos, la información y a las personas.

Actualmente existen herramientas que satisfacen muchas de estas necesidades, como es el caso de los llamados portales, que proveen mecanismos de integración, inicio único de sesión, seguridad y contenido personalizado. Además de ofrecer preferencias a medida de los usuarios, gestión de contenidos, opciones de colaboración entre personas, acceso desde múltiples dispositivos, opciones de búsqueda y navegación entre aplicaciones (1).

El portal *Liferay* es una de estas plataformas web basada en tecnología java y de código abierto. Tiene un gran número de características que hacen que sea una herramienta de trabajo muy funcional y útil para el desarrollo de las actividades de una compañía. Mientras que otras empresas tienen que pagar por características adicionales, esta plataforma cuenta con más de sesenta *portlets*[3] listos para usar, más de veinte temas, una serie de herramientas de desarrollo y un grupo de servicios profesionales entre los cuales brinda servicio de capacitación, consultoría y soporte empresarial a sus clientes. Cuenta con una gestión potente y flexible, permitiendo administrar la información y los usuarios, de tal forma que se pueden asignar roles a los mismos. Permite la autenticación por *SSO* (*Single sign-on*, por sus siglas en inglés), es decir que con solo loguearse una vez puede acceder a todos los servicios que se prestan por estar completamente integrado. Todo ello se realiza de forma eficaz a través de la Arquitectura Orientada a Servicios (*SOA*) y cuenta con al menos treinta idiomas disponibles debido a la globalización existente. Es una herramienta que permite que el manejo de los contenidos del portal sea fácil y funcional. Tiene una organizada y segura comunicación entre sus usuarios como: blogs, mensajería instantánea y correo electrónico (1). Presenta además una amplia compatibilidad con sistemas operativos, servidor de aplicaciones y base de datos. Eso quiere decir que se puede descargar, instalar y probar los portales que ofrece en los entornos informáticos con el personal de informática existente en cualquier institución.

[3] *Portlets*: pequeñas aplicaciones web que se ejecutan en una parte de una página web.

Corporaciones internacionales como Cisco Systems (empresa global con sede en San José, California, Estados Unidos), openTrends (tiene su sede en España) y Mexuz (tiene sede en Australia), entre otras, hacen uso de *Liferay* para desarrollar sitios web y portales que proporcionen soporte a sus negocios.

De igual forma, Cuba hace uso de esta herramienta con el objetivo de desarrollar aplicaciones de negocio, de forma rápida y sencilla. Instituciones como el Centro de Inmunología Molecular (CIM), la Unión de Informáticos de Cuba (UIC) y la Universidad de las Ciencias Informáticas (UCI) poseen proyectos reales desarrollados con la plataforma *Liferay*. Esta plataforma es potente y flexible al otorgar permisos de manera jerárquica y granular para gestionar los componentes. A pesar de estas características, no cuenta con un módulo que permita comunicar las incidencias que surgen en la organización, que pueden ser de tipo *hardware*, *software* o simplemente de gestión.

En los organismos mencionados anteriormente existen dificultades a la hora de gestionar las incidencias, debido a que se realizan con diferentes herramientas y tecnologías; teniendo que utilizar servidores más potentes que los convencionales y de mayor capacidad. También existen entidades que a la hora de realizar estas gestiones no siguen los procedimientos previstos, se resuelven sin registrarlas o se escalan innecesariamente omitiendo los protocolos preestablecidos, por lo que éstas no se registran adecuadamente. Todas estas dificultades conllevan a reducción de los niveles de servicio, demasiadas personas del nivel inadecuado trabajando concurrentemente en la resolución de la incidencia, se pierde valiosa información sobre las causas y efectos de las incidencias para futuras reestructuraciones y evoluciones, se crean clientes y usuarios insatisfechos por la mala y lenta gestión de sus incidencias.

Dada la situación problemática antes planteada, se propone el **problema de investigación**:

¿Cómo realizar la gestión de incidencias en las entidades que hacen uso del portal *Liferay Community*?

Objeto de estudio:

La gestión de incidencias.

Objetivo General:

Desarrollar un *portlet* que permita gestionar las incidencias que surgen en las entidades que hacen uso del portal *Liferay Community*.

Campo de acción:

La gestión de incidencias para el portal *Liferay Community*.

Como objetivos específicos se plantea:

- Elaborar el estado del arte sobre las tendencias actuales en gestión de incidencias.
- Seleccionar la metodología, herramientas y tecnologías idóneas para el desarrollo de la solución propuesta.
- Identificar los requisitos funcionales y no funcionales.
- Definir el análisis y diseño de la solución propuesta.
- Implementar las funcionalidades que den cumplimiento a los requisitos identificados.
- Realizar pruebas para validar la solución propuesta.

Idea a defender

Con el desarrollo del componente de gestión de incidencias para el portal *Liferay*, se obtendrán informes detallados de las incidencias. Este producto permitirá una monitorización y control del servicio.

Para el desarrollo de las tareas de investigación se emplearon varios métodos:

Métodos teóricos

El **Análisis-Síntesis** para una mayor comprensión del funcionamiento del portal *Liferay*, características, arquitectura que presenta y formular conclusiones a través de la síntesis de los conocimientos y resultados obtenidos.

La **Modelación** es considerada como uno de los métodos lógicos que consistirá en esquemas, diagramas o representaciones donde se refleje la estructura de relaciones y determinadas propiedades fundamentales del *portlet*.

Métodos empíricos

La **Observación Científica**, permite valorar los avances realizados en el estudio de las diferentes herramientas y tecnologías a usar.

El método de **Consulta de la Información** es utilizado para elaborar el marco teórico o el estado del arte de la investigación, permitiendo el conocimiento y el acceso a las referencias

bibliográficas de los múltiples criterios que han sido citados para comprender mejor el problema de investigación planteado.

Estructura de la tesis

Capítulo I. Fundamentación Teórica. En este capítulo se estudian las tendencias actuales de la gestión de incidencias, así como el desarrollo de los *portlets* en el portal *Liferay*. Se define el lenguaje de programación, metodología, tecnologías y herramientas para la implementación de la aplicación que dará solución al problema planteado.

Capítulo II. Características y diseño del sistema. *Portlet* de Gestión de Incidencias para *Liferay* como propuesta de solución que permitirá mejorar la productividad de los usuarios y un mayor control de los procesos. Se toma como base la metodología AUP para guiar el proceso de desarrollo. Se da a conocer la propuesta del sistema y los diagramas para apoyar la comprensión del funcionamiento del mismo.

Capítulo III. Implementación y prueba del sistema. Análisis y diseño del *Portlet* de Gestión de Incidencias para el portal *Liferay*. Se muestra todo lo referente al flujo de implementación. Se diseñan, describen y realizan los casos de prueba aplicados al sistema. También se valida el resultado mediante técnicas de partición de equivalencia y lista de chequeo.

Capítulo #1. Fundamentación teórica

1.1 Introducción

En el capítulo se presentan los elementos teóricos que sirven de base a la investigación del problema planteado. Se estudian las tendencias actuales; el uso de tecnologías, metodologías, arquitecturas y herramientas que brindan conocimiento y apoyo en el desarrollo del proceso de elaboración del producto final.

1.2 Gestión de Incidentes o incidencias

Se define que “la gestión de incidentes[4] tiene como objetivo resolver cualquier eventualidad que cause una interrupción en el servicio de la manera más rápida y eficaz posible. La gestión de incidentes no debe confundirse con la gestión de problemas, pues a diferencia de esta última, no se preocupa de encontrar y analizar las causas subyacentes a un determinado incidente sino exclusivamente a restaurar el servicio” (2).

Las actividades del proceso de gestión de incidencias constan de los siguientes pasos (3): Identificación, registro, clasificación, priorización, diagnóstico (inicial), escalado, investigación/ diagnóstico, resolución/recuperación y cierre.

Algunas precisiones sobre algunos de sus pasos:

- Cuando se registra una incidencia, es posible que los datos de los que se dispone estén incompletos o sean incorrectos. Por ello conviene comprobar la clasificación de la incidencia y actualizarla mientras se cierra la llamada. Un ejemplo de incidencia categorizada es el siguiente: *software*, aplicación.
- La prioridad de una incidencia se puede determinar a partir de su urgencia (la rapidez con que el negocio necesita una solución) e impacto (indicado por el número de usuarios a los que afecta).
- Se debe intentar registrar el mayor número posible de síntomas de la incidencia.
- También tiene que intentar determinar qué es lo que ha fallado y cómo se podría corregir.
- En este contexto pueden resultar muy útiles los guiones de diagnóstico y la información sobre errores conocidos. Si es posible, el agente del Centro de Servicio al Usuario

[4]En la investigación se define una incidencia como incidente, es decir, se refiere al mismo término.

resuelve la incidencia inmediatamente y la cierra. Si resulta imposible, el agente debe escalar la incidencia.

- El escalado funcional se da cuando la organización tiene un grupo de segunda línea de soporte y el Centro de Servicio al Usuario cree que ese grupo puede resolver la incidencia.
- Si se trata de una incidencia que requiere más conocimientos técnicos y la segunda línea de soporte no puede resolverla, tiene que ser escalada al grupo de tercera línea de soporte. El escalado jerárquico consiste en ir ascendiendo niveles en la cadena de mando de la organización para que los altos responsables conozcan la incidencia y puedan adoptar las medidas oportunas, como asignar más recursos o acudir a suministradores.
- Cuando se gestiona una incidencia, cada grupo de soporte investiga qué es lo que ha fallado y realiza un diagnóstico, todas estas actividades deben quedar documentadas en un registro de incidencias para disponer de una imagen completa de las actividades realizadas.
- Cuando se ha determinado una posible solución, lo siguiente que hay que hacer es implementar y probarla.

Se pueden llevar a cabo las siguientes acciones:

- Pedir al usuario que efectúe determinadas operaciones en su ordenador, el Centro de Servicio al Usuario puede ejecutar la solución de forma centralizada o utilizar *software* remoto para controlar el ordenador del usuario e implementar una solución o pedirá un proveedor que resuelva el error.
- El grupo de soporte devuelve la incidencia al Centro de Servicio al Usuario y éste procede a cerrar la incidencia, comprobando antes que ha sido resuelta y que los usuarios están satisfechos con la solución.
- También tiene que cerrar la clasificación, actualizar la documentación de la incidencia, determinar si se podría volver a producir la misma incidencia y decidir si hay que adoptar alguna medida para evitarlo.

“Una **incidencia** es toda interrupción o reducción de la calidad no planificada del servicio. Pueden ser fallos o consultas reportadas por los usuarios, el equipo del servicio o por alguna herramienta de monitorización de eventos” (4).

El autor del documento luego de haber consultado los diferentes conceptos antes mencionados, considera que la gestión de incidencia tiene como objetivo resolver cualquier eventualidad que cause una interrupción en el servicio prestado y todas las actividades deben quedar documentadas en un registro de incidencias con los datos principales como: tipo de incidencia, descripción, prioridad, responsable de dar solución, categorización, los estados por los que esta transcurre, fecha de detección y de resolución.

1.3 Sistemas de gestión de incidencias

Entre todas las aplicaciones dedicadas a la gestión de incidencias se tiene tanto soluciones gratuitas como de pago (2):

- BugTracker.NET: se presenta como una aplicación web donde datos y web están alojados en sus propios servidores. Algunas de sus características más interesantes son: la capacidad de crear informes propios, permitir definir el flujo de trabajo a seguir e importación de emails a “tickets”. No se da referencia a que la asignación de incidencias se pueda establecer de forma automática.

- Mantis Bug Tracker: aplicación en PHP (*Hypertext Preprocessor*). También puede ser usada como herramienta para gestión de proyectos. De esta solución se puede destacar que permite configurar la transición de estados de las incidencias y el soporte en varios idiomas. Pero como en el caso anterior parece que la asignación de incidencias es de forma manual o estática dependiendo del proyecto al que se refiera la incidencia.

- Request Tracker: de esta aplicación cabe señalar que permite a los usuarios configurar los campos de sus incidencias así como crear incidencias a partir de emails y generar respuestas automáticas.

Entre las soluciones de pago se encuentran:

- JIRA: al igual que Mantis Bug Tracker también posee características para ser usado como gestor de proyectos. Es altamente configurable, permite la integración con otros sistemas y ofrece estadísticas e informes bastante completos. El principal inconveniente que quizás a primera vista se le puede atribuir al producto, es que si lo que pretende tener es simplemente un gestor de incidencias, las tareas y opciones que la hacen versátil también como gestor de proyectos pueden dificultar su usabilidad.

- ServiceDesk Plus: la funcionalidad de esta solución es muy completa al poseer reglas de negocio para la asignación automática de solicitudes, base de conocimientos, gestión de

tareas programadas e integración con correo electrónico. No es sólo un gestor de incidencias sino también posee características de un gestor de cambios y problemas.

- I-Solver: al igual que el anterior programa permite la asignación automática, creación de incidencias por email y creación de incidencias programadas.

Se espera que el *portlet* que se propone sea una solución suficientemente completa, pero a la vez simple para ser atractiva y que contemple las necesidades mínimas que debe cumplir un gestor de incidencias.

1.4 Portal *Liferay*

Liferay es un portal web *Open Source* para la creación y gestión de escritorios personalizados que cuenta con una elevada escalabilidad y robustez para el manejo de las aplicaciones empresariales de todos los sectores (5).

Liferay está basado en los principios de la Arquitectura Orientada a Servicios (SOA), ofrece escalabilidad, fiabilidad y alto rendimiento, tanto en la nube como "*on premise*" (*software* instalado y operado desde un servidor local) (5). Soporta millones de usuarios online. Diseñado para que el contenido se muestre perfectamente en cualquier tamaño de pantalla. Se encuentra en más de veinte idiomas y proporciona una funcionalidad completa para la traducción de cada elemento del portal. Permite metodologías de escalado horizontal y vertical. Implementa las diez prácticas principales de seguridad recomendadas por *The Open Web Application Security Project* (OWASP), entre otras características destacables (6).

Hernández expresa que, “la adecuación de este portal a los estándares J2EE (*Java 2 Enterprise Edition*) hace posible implementar esta tecnología en multitud de contextos sin que existan restricciones a nivel de sistema operativo, servidor de aplicaciones o base de datos” (5).

1.5 Portlets

Los *portlets* son componentes modulares de interfaz de usuario gestionada y visualizada en un portal web. Producen fragmentos de código de marcado HTML (*HyperText Markup Language*), XHTML (*Extensible HyperText Markup Language*) que se agregan en una página de un portal (5). Además, una página de un portal se visualiza como una colección de ventanas de *portlet* que no se solapan, donde cada una de estas muestra un *portlet*. Por lo tanto un

portlet (o colección de *portlets*) se asemeja a una aplicación web que está hospedada en un portal (7).

Los *portlets* son módulos Web reutilizables que se ejecutan en un servidor del portal y proporcionan acceso al contenido, a las aplicaciones y a otros recursos basados en la web. Además, se ensamblan en una página más grande, con varias instancias del mismo *portlet* que muestran diferentes datos para cada usuario (8).

Características principales:

- Los *portlets* son manejados por un contenedor especializado, que controla su ciclo de vida.
- Los *portlets* generan contenido dinámico e interactúan con el cliente web mediante el uso del paradigma *request/response* (solicitud/respuesta).
- Los *portlets* son únicamente generados como parte de una página web y no como documentos completos, no están asociados directamente a una URL y no pueden generar contenido arbitrario, es por estas razones que son diferentes a los servlets.
- Los *portlets* proporcionan un objeto *Portlet Preferences* para almacenar las preferencias del usuario.

1.6 Tecnologías, herramientas y metodología a utilizar

En este epígrafe se expone la metodología que guiará el proceso de desarrollo del *software* y los conceptos relacionados a las herramientas y tecnologías empleadas en la aplicación Gestión de incidencia. Debido a que estas herramientas y tecnologías favorecen el incremento de la productividad, disminuyendo así el tiempo de desarrollo.

1.6.1 Metodología Unificada Ágil

Una metodología de desarrollo de *software* se refiere al entorno que se usa para estructurar, planificar y controlar el proceso de desarrollo de un sistema de información (9).

La metodología unificada ágil (AUP) es una forma simplificada del Proceso Racional Unificado (RUP, *Rational Unified Process*) desarrollada por Scott Ambler con el objetivo de mejorar la productividad, siendo un camino intermedio entre XP (*Extreme Programming*) y RUP. Describe un enfoque simple del desarrollo del *software* usando técnicas y conceptos ágiles. Algunas técnicas usadas por AUP incluyen el desarrollo orientado a pruebas, modelado, gestión de cambios ágiles y refactorización de base de datos para mejorar la productividad (10).

AUP establece cuatro fases: (Inicio, Elaboración, Construcción, Transición), la UCI decide para el ciclo de vida de los proyectos mantener la fase de Inicio, en la cual se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto. Se unifican las restantes tres fases de AUP en una sola, a la que se denomina Ejecución que sería aquella fase en la que se ejecutan las actividades requeridas para desarrollar el *software*, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del *software*, y se agrega una fase de cierre. En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto (11). AUP propone siete disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener ocho disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en tres disciplinas: Pruebas Internas, de Liberación y Aceptación y la disciplina Despliegue se considera opcional. Las restantes tres disciplinas de AUP asociadas a la parte de gestión, para la variación UCI, se cubren con las áreas de procesos que define CMMI-DEV (*Capability Maturity Model Integration for Development*) v1.3 para el nivel dos, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto) (11). En cada disciplina la metodología plantea las diferentes actividades y artefactos a producir, lo cual no implica que se realicen o se produzcan todos los planteados sino más bien lo que se necesita en el proyecto.

Ventajas de la Metodología AUP (10):

- El personal sabe lo que está haciendo y por tanto no obliga a conocer los detalles del proceso.

- Simplicidad: todo se describe concisamente utilizando un puñado de páginas, no miles de ellos.
- Permite centrarse en actividades de alto valor. La atención se centra en las actividades que son esenciales para el de desarrollo, no todas las actividades que suceden forman parte del proyecto.
- Permite hacer uso de un amplio conjunto de herramientas. Lo aconsejable es utilizar las que son las más adecuadas para el trabajo, que a menudo son las más simples o incluso herramientas de código abierto.
- Incluye explícitamente actividades y artefactos a los que la mayoría de desarrolladores ya están, de alguna manera, acostumbrados.
- Se preocupa especialmente de la gestión de riesgos. Propone que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo.

Desventajas de la Metodología AUP (10):

- Como es un proceso simplificado, muchos desarrolladores eligen trabajar con RUP, por tener a disposición más detalles en el proceso.

Por los recursos, la documentación necesaria, el escaso tiempo para la realización del proyecto, y personal, se decide escoger esta metodología, para guiar el proceso de desarrollo de la solución propuesta, adaptada al ciclo de vida definido para la actividad productiva de la UCI.

1.6.2 Lenguaje de programación Java

Como el código compilado de Java (conocido como *bytecode*) es interpretado, un programa compilado puede ser utilizado por cualquier computadora que tenga instalado el intérprete de Java. Este código es interpretado por diferentes computadoras de igual manera, solamente hay que instalar un intérprete para cada plataforma. De esa manera logra ser un lenguaje que no depende de una arquitectura computacional disponible. Además, no se requiere que se compilen todas las clases de un programa para que este funcione (12).

Debido a la Máquina Virtual de Java (JVM) las aplicaciones desarrolladas en Java funcionan en Linux, Windows, Mac OS, y en cualquier sistema operativo para el cual exista una JVM

garantizando que no se dependa de un único sistema operativo para el funcionamiento de la solución (13).

Java es un lenguaje de programación orientado a objetos, de arquitectura neutral, por lo que es altamente portable, segura, de multihilos y dinámica (14) (15).

Dado que *Liferay* es un portal de gestión de contenidos de código abierto escrito en Java, se escoge para el desarrollo del proyecto el lenguaje de programación Java para continuar la misma estructura.

1.6.3 *Framework* Bootstrap

Bootstrap es un *framework* desarrollado y liberado por Twitter que tiene como objetivo facilitar el diseño web. 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 *Open Source* o código abierto, por lo que se puede usar de forma gratuita y sin restricciones (16) (17) (18).

Ventajas de usar Bootstrap:

La más genérica es que permite simplificar el proceso de maquetación, sirviendo de guía para aplicar las buenas prácticas y los diferentes estándares. Aquí van unos cuantos beneficios más:

- Se puede tener una web bien organizada de forma visual rápidamente: la curva de aprendizaje hace que su manejo sea asequible y rápido si ya se sabe maquetar.
- Permite utilizar muchos elementos web: desde iconos a desplegables, combinando HTML5, CSS (*Cascading Style Sheets*) y Javascript.
- El diseño será adaptable, no importa el dispositivo, la escala o resolución.
- El Sistema de rejillas: maquetar por columnas nunca fue tan fácil. Además, son muy configurables.
- Se integra muy bien con las principales librerías Javascript.
- El haber sido creado por Twitter da ciertas garantías: está muy pensado y hay mucho trabajo ya hecho. Por lo tanto, hay una comunidad muy activa creando, dando mantenimiento, ofreciendo plugins y mucho más.
- Cuenta con implementaciones externas para WordPress, Drupal, etc.
- Permite usar Less, para enriquecer aún más los estilos de la web.

1.6.4 *Service Builder*

Liferay proporciona una buena manera de crear servicio y capa de persistencia. *Liferay Service Builder* es una herramienta para crear la capa de persistencia de servicio en *Liferay*. Genera un conjunto de clases e interfaces que se utiliza para interactuar con la base de datos (operación CRUD (Crear, Leer, Actualizar y Borrar)). De esta forma se ahorra tiempo de desarrollo de la capa de mantenimiento de servicio y de mantenimiento manualmente.

Liferay Service Builder utiliza Spring para proporcionar la implementación de la capa de servicio e Hibernate para la implementación de la capa de persistencia. Ambos marcos (Spring e Hibernate) son estándares de la industria y han demostrado construir aplicaciones web escalables (19) (20) (21).

1.6.5 Spring *Framework*

Spring es un *framework* liviano y no intrusivo: generalmente los objetos programados no tienen dependencias en clases específicas de Spring. Sus características principales son inyección de dependencias y programación orientada a aspectos (22).

Spring es un *framework* ligero formado por un conjunto de piezas de diversa índole: Modelo Vista Controlador (MVC), seguridad, transaccionalidad, Programación Orientada a Aspecto (AOP) , integración con Servicios Web, Hibernate; sin embargo, su componente principal, consiste en un contenedor de beans que permite, mediante Inversión de Control (IoC), acoplar las diversas partes de una aplicación (23).

Este *framework* está diseñado para no ser intrusivo, esto significa que no es necesario que el sistema extienda o implemente alguna clase o interface de Spring, por lo que el código de lógica quedará libre y completamente reutilizable para un proyecto que no lo utilice, o por si se debe quitar de una aplicación que ya lo esté usando. Gracias a esto es posible usar un POJO (*Plain Old Java Object*) o un objeto Java para realizar tareas que antes solo podían ejecutarse con *Enterprise Java Beans* (EJBs, una de las API (*Application Programming Interface*) que forman parte del estándar de construcción de aplicaciones empresariales *Java Enterprise Edition*). Sin embargo, la utilidad de este *framework*, no es solo para el desarrollo de aplicaciones web, o no solo en el servidor, sino que permite que cualquier aplicación Java pueda beneficiarse de su uso.

Además, al usarlo de la forma correcta la aplicación quedará dividida en capas bien delimitadas, y con buenas prácticas de programación.

El núcleo de Spring está basado en un principio o patrón de diseño llamado Inversión de Control (IoC). Las aplicaciones que usan este principio se basan en su configuración (que en este caso puede ser en archivos XML o con anotaciones como en Hibernate) para describir las dependencias entre sus componentes. En este caso “inversión” significa que la aplicación no controla su estructura, permitiendo que sea el *framework* de IoC (Spring) quien lo haga.

Se puede decir que Spring cambia las responsabilidades y en vez de que el propio desarrollador sea el encargado de generar los objetos de cada uno de los *frameworks*, es éste basándose en ficheros XML o anotaciones, el encargado de construir todos los objetos que la aplicación va a utilizar.

1.6.6 *Framework* Hibernate

Hibernate permite desarrollar clases persistentes a partir de clases comunes, incluyendo asociación, herencia, polimorfismo, composición y colecciones de objetos. El lenguaje de consultas de Hibernate HQL (*Hibernate Query Language*), diseñado como una mínima extensión orientada a objetos de SQL, proporciona un puente elegante entre los mundos objetual y relacional. Hibernate también permite expresar consultas utilizando SQL nativo o consultas basadas en criterios. Soporta todos los sistemas gestores de bases de datos SQL y se integra de manera elegante y sin restricciones con los más populares servidores de aplicaciones J2EE y contenedores web, y por supuesto también puede utilizarse en aplicaciones standalone (aplicación que puede ser usada en cualquier computadora sin necesidad de ser instalada o que no requiere necesariamente una conexión a la red para funcionar) (24).

Hibernate es un *Framework* que agiliza la relación entre la aplicación y la base de datos. Para poder aprender a utilizarlo es necesario contar con los conocimientos básicos de base de datos y SQL (*Structured Query Language*) así como manejar el lenguaje Java (25).

Se considera entonces, que las características principales de este *framework* son las siguientes:

- Hibernate puede operar proporcionando persistencia de una manera transparente para el desarrollador.
- Soporta el paradigma de orientación a objetos de una manera natural: herencia, polimorfismo, composición y el *framework* de colecciones de Java.
- Permite una gran variedad de mapeos para colecciones y objetos dependientes.

- No es necesaria la generación de código ni el procesamiento del *bytecode* en el proceso de compilación.
- Este lenguaje proporciona una independencia del SQL de cada base de datos, tanto para el almacenamiento de objetos como para su recuperación.
- Soporte para transacciones de aplicación.
- Soporta los diversos tipos de generación de identificadores que proporcionan los sistemas gestores de bases de datos, así como generación independiente de la base de datos, incluyendo identificadores asignados por la aplicación o claves compuestas.
- Escalabilidad extrema.

1.6.7 Sistema Gestor de Bases de Datos

Un sistema de gestión de bases de datos (SGBD) es un sistema de *software* que permite el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos. Brindan funciones con el objetivo de garantizar la confidencialidad, la calidad, la seguridad y la integridad de los datos. Permiten presentar la información de la base de datos en varios formatos. La mayoría incluyen un generador de informes. También pueden incluir un módulo gráfico que permita presentar la información con gráficos y tablas.

PostgreSQL Server

PostgreSQL es un Sistema Gestor de Bases de Datos Relacionales de código abierto y orientado a objetos. Brinda un control de concurrencia multi-versión, que permite trabajar con grandes volúmenes de datos. Funciona en todos los sistemas operativos Linux, UNIX y Windows.

El código fuente se encuentra disponible para todos sin costo alguno. Debido a la liberación de la licencia, PostgreSQL se puede usar, modificar y distribuir con fines privativo, comercial, así como académico. Por estas características es considerado como uno de los gestores de bases de datos más avanzado del mundo.

Se trabajará con PostgreSQL Server por ser muy usado en entornos de *software* libre y puede funcionar en múltiples plataformas.

1.6.8 Entorno de Desarrollo Integrado Eclipse Neon

Eclipse es una plataforma de desarrollo, diseñada para ser extendida de forma indefinida a través de plugins. Fue concebida desde sus orígenes para convertirse en una plataforma de integración de herramientas de desarrollo. Es un IDE (*Integrated Development Environment*) genérico, aunque goza de mucha popularidad entre la comunidad de desarrolladores del lenguaje Java usando el plugin JDT (*Java Development Toolkit*) que viene incluido en la distribución estándar del IDE (26).

Eclipse es una plataforma de desarrollo integrado, de código abierto y multiplataforma, potente y completa de programación, desarrollo y compilación de elementos variados como: sitios web, programas en C++ o aplicaciones Java. Es usado principalmente en la construcción de Aplicaciones de Cliente Enriquecido, opuesto a las aplicaciones Cliente-Liviano, basadas en navegadores. Posee una atractiva interfaz que lo hace fácil y agradable de usar.

Permite la instalación de plugins como es el caso del plugin *Liferay* IDE, brindando las opciones de desarrollar aplicaciones *portlets* y temas de interfaz visual, por lo que se escogió para el desarrollo de la solución propuesta.

1.6.9 Servidor de aplicaciones Apache Tomcat

Apache Tomcat es un servidor web multiplataforma que funciona como contenedor de servlets y se desarrolla bajo el proyecto Jackarta perteneciente a la Apache *Software Fundation*. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la licencia Apache 2.0 y que implementa las especificaciones de los servlets y de *Java Server Pages* (JSP).

Dado que Apache Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

Por lo que se escoge como servidor web, por poseer ayuda y abundante documentación, cuenta con una comunidad de usuarios, es libre y es uno de los más usados en el CDAE[5].

1.6.10 Herramientas CASE

Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de *Software* Asistida por Computadora) son aplicaciones informáticas destinadas a aumentar la productividad y la calidad en el desarrollo del *software* reduciendo los costos del mismo en

[5] CDAE: Centro de Consultoría y Desarrollo de Arquitecturas Empresariales.

términos de tiempo y dinero. Estas herramientas permiten tener una mejor organización y control del desarrollo de un sistema informático, en especial aquellos sistemas que sean grandes o robustos y que impliquen tener muchos componentes *software*, así como recursos humanos (27). Todo esto asistiendo en tareas como el diseño del proyecto, implementación de parte del código y la documentación asociada a la implementación (28).

Herramienta de modelado Visual Paradigm

Visual Paradigm for UML es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, implementación, pruebas y despliegue. Ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite construir diagramas de diversos tipos, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Teniendo en cuenta las características y los beneficios que brinda esta herramienta para la construcción de *software*, especialmente referente al modelado, se decidió utilizar Visual Paradigm for UML para el modelado. Además de ello, se tuvo en cuenta que esta constituye la herramienta que utiliza la Universidad, y dentro de ella el centro DATEC[6] para el desarrollo de *software* (29) (30).

Visual Paradigm for UML ofrece una variedad de funcionalidades muy ricas y potentes, como son:

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de usos.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidad de ingeniería directa e inversa.
- El modelado y el código permanecen sincronizado en todo el ciclo de desarrollo.
- Presenta una licencia gratuita y comercial.
- Se encuentra en varios idiomas.
- Soporta aplicaciones web.

[6] DATEC: Centro de Tecnologías de Gestión de Datos.

- Potente generador de informes en formato PDF/HTML.
- Facilidad de instalación y de actualización.
- Compatibilidad entre ediciones.
- Interoperabilidad con modelos UML2 (metamodelos UML2 para plataforma Eclipse) a través de XML.
- Soporte ORM- Generación de Objetos Java desde bases de datos.
- Generación de bases de datos – Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Editor de figuras.
- Ambiente visual superior de modelado.

Balsamiq Mockups

Balsamiq Mockups es una herramienta que permite crear borradores de la interfaz de usuario de la aplicación. Posee bastante documentación, así como un foro de ayuda y tiene una frecuencia de actualización de su producto muy buena. Además es una herramienta de pago, $ 79 para una licencia de usuario. Sin embargo, se puede evaluar una versión completamente funcional durante siete días. El grupo de Balsamiq incluso pone a disposición una licencia totalmente gratuita y es una herramienta que puede ser utilizada para generar borradores de manera rápida a la vez que bastante profesional para aplicaciones web (31).

Ambas herramientas CASE fueron usadas para los prototipos de interfaz de usuario, así como para el modelado de las clases y la arquitectura.

1.7 Conclusiones parciales

- Se logró realizar un análisis del concepto referente a gestión de incidentes, así como las herramientas, tecnologías y la metodología de guía para el desarrollo de la solución.
- Se definió para la interfaz de usuario el *framework* Bootstrap, el *Service Builder* con los *frameworks* asociados a este, Spring e Hibernate, Spring para la capa de negocios e Hibernate como acceso a datos. Lo principal de esta combinación es que hay configuraciones mínimas a través de XML y por anotaciones, dando más flexibilidad a los desarrolladores sólo para concentrarse en la interfaz de usuario y negocio.
- Se decide utilizar como gestor de bases de datos PostgreSQL Server, el lenguaje de programación Java, el entorno de desarrollo integrado Eclipse Neon, como servidor web

el Apache Tomcat, para guiar el proceso de desarrollo la metodología AUP, como herramientas CASE Visual Paradigm for UML y Balsamiq Mockups.

Capítulo #2. Características y diseño del sistema

2.1 Introducción

En el presente capítulo se dará a conocer la propuesta del sistema y los diagramas para apoyar la comprensión del funcionamiento del mismo, tomándose como base la metodología AUP para guiar el proceso de desarrollo. Esta metodología propone diferentes artefactos, la presente investigación utiliza los mencionados en la Tabla 8 (Artefactos de la metodología AUP generados en el documento) para la implementación del *software*. También se definirá la arquitectura que se usará para la implementación del módulo de gestión de incidencias.

2.2 Propuesta de solución

Portal *Liferay* actualmente no cuenta con un módulo que permita la gestión de incidencias en las organizaciones que hacen uso de esta herramienta, obligando de esta forma a que no se sigan los procedimientos previstos y se resuelven los incidentes sin registrarlos omitiendo los protocolos preestablecidos.

Para resolver este problema se propone desarrollar un *portlet* que permita a las entidades que usan el *Liferay*, realizar estas gestiones dinámicamente para aumentar la eficiencia a la hora de registrarlas, de manera que no se pierda información sobre las causas y efectos de las incidencias para futuras reestructuraciones y evoluciones. De igual forma, el *portlet* permitirá almacenar las incidencias, asignarle el responsable a resolverla, el departamento donde ocurrió o al que pertenece y la prioridad que tendría para la resolución, dígase baja, media y alta, además, el sistema permitirá gestionar los nomencladores antes mencionados (Insertar, Editar, Eliminar, Mostrar). También se podrán generar distintos reportes de incidencias, ya sea por rango de fechas, estado, categoría, prioridad, departamento, asignadas a un trabajador o registradas por un trabajador. Además, se podrá realizar una búsqueda de todas las incidencias registradas en el sistema al introducir un término o una frase, dígase título de la incidencia, descripción, lugar, estado, categoría prioridad o departamento. Mostrando la información de la incidencia que coincidieron según los parámetros mencionados anteriormente.

Partiendo de este conjunto de restricciones de negocio, con la implementación y puesta en marcha del módulo, los administradores de sistemas obtendrán información fiable referente a todas las incidencias que han ocurrido en el período dentro de la empresa.

2.3 Manejo de *Service Builder* en la solución

Service Builder es una herramienta de generación de código basada en modelos construida por *Liferay* que permite a los desarrolladores definir modelos de objetos personalizados llamados entidades. *Service Builder* genera una capa de servicio a través de la tecnología de correlación objeto-relacional (ORM) que proporciona una separación clara entre el modelo de objeto y el código para la base de datos subyacente. Esto le libera para agregar la lógica de negocio necesaria para su aplicación. *Service Builder* toma un archivo XML como entrada y genera las capas de modelo, persistencia y servicio necesarias para su aplicación. Además, genera la mayor parte del código común necesario para implementar operaciones de creación, lectura, actualización, eliminación y búsqueda en la base de datos, lo que le permite centrarse en los aspectos de nivel superior del diseño del servicio. Principales beneficios del uso de *Service Builder* (32):

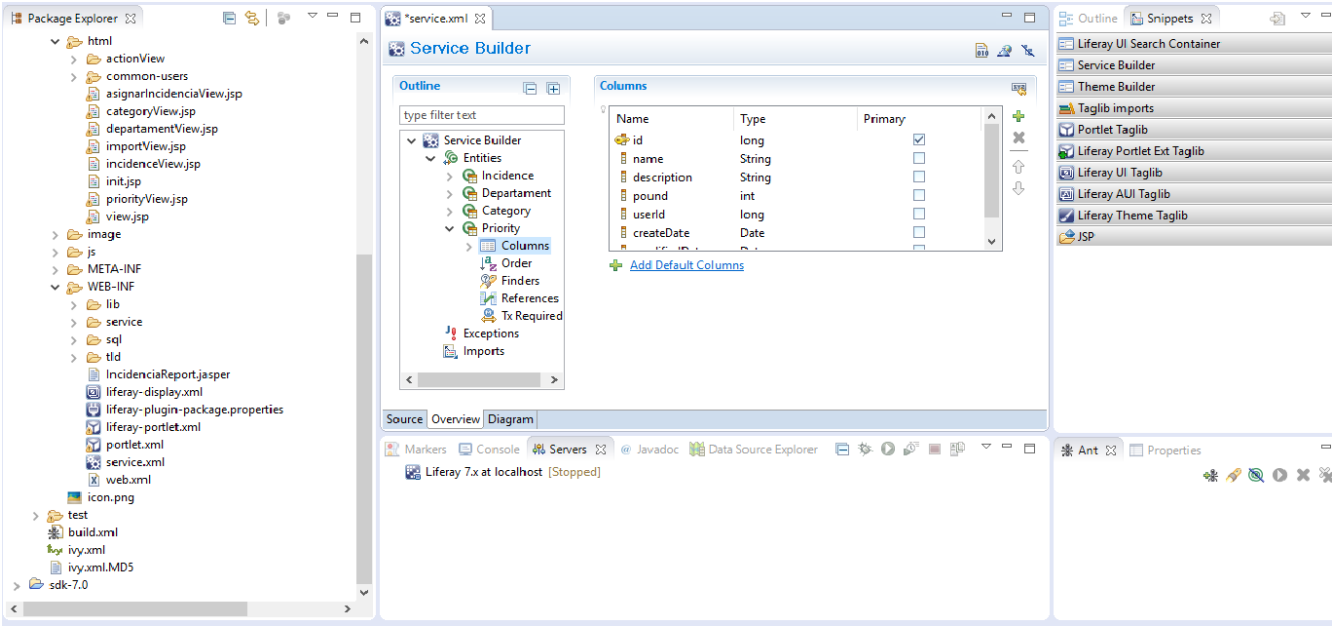
- Integración con *Liferay*.
- Modelo generado automáticamente, persistencia y capas de servicio.
- Servicios locales y remotos generados automáticamente.
- Configuraciones Hibernate y Spring generadas automáticamente.
- Soporte para generar métodos finder para entidades y métodos finder que tengan en cuenta permisos.
- Soporte de caché de entidad incorporado.
- Compatibilidad con consultas SQL personalizadas y consultas dinámicas.

Liferay utiliza *Service Builder* para generar todo el código de persistencia de la base de datos interna. Además, todos los módulos de servicios de *Liferay*, tanto locales como remotos, son generados por *Service Builder*. *Service Builder* es fácil de usar y puede ahorrar mucho tiempo de desarrollo, aunque el número de archivos generados por éste puede parecer abundante al principio, los desarrolladores sólo necesitan trabajar con unos pocos para poder personalizar sus aplicaciones y agregar lógica empresarial (32).

Una de las principales formas en que *Service Builder* ahorra tiempo de desarrollo es eliminar por completo la necesidad de escribir y mantener el código de acceso a la base de datos. Para generar una capa de servicio básica, sólo necesita crear un archivo service.xml y ejecutar *Service Builder*. Esto genera un nuevo archivo .jar de servicio para su proyecto. El archivo .jar de servicio generado incluye una capa de modelo, una capa de persistencia, una capa de

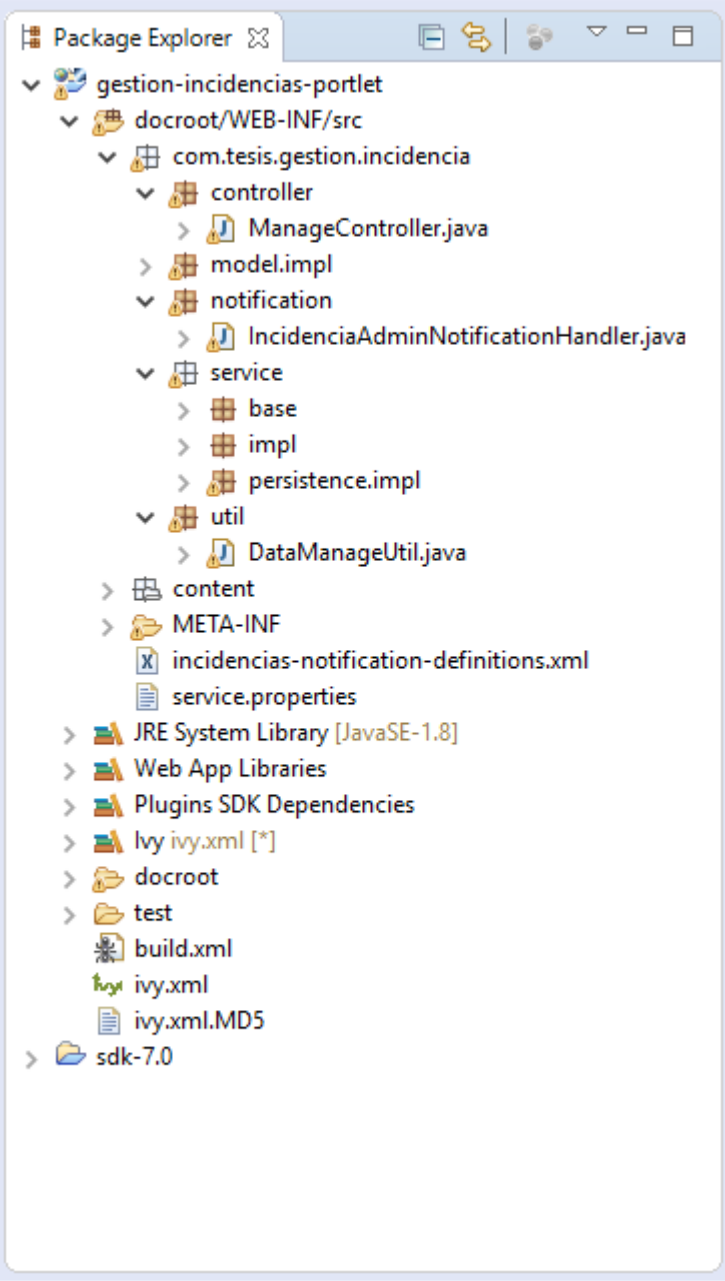
servicio e infraestructura relacionada. La capa de modelo es responsable de definir objetos para representar las entidades de su proyecto, la capa de persistencia es responsable de guardar entidades y recuperar entidades de la base de datos y la capa de servicio es responsable de exponer CRUD y métodos relacionados para sus entidades como API.

Figura 1 Estructura del componente haciendo uso de Service Builder. Elaboración propia.



El código Generador de servicios es independiente de la base de datos, como es *Liferay* en sí. Cada entidad generada por *Service Builder* contiene una clase de implementación de modelo. Cada entidad también contiene una clase de implementación de servicio local, una clase de implementación de servicio remoto o ambas, dependiendo de cómo configure *Service Builder* en su archivo service.xml. Las personalizaciones y la lógica empresarial pueden implementarse en estas tres clases; de hecho, éstas son las únicas clases generadas por *Service Builder* que están destinadas a personalizarse. Asegurar que todas las personalizaciones tengan lugar en sólo unas pocas clases hace que los proyectos de *Service Builder* sean fáciles de mantener. La clase de implementación de servicio local es responsable de llamar a la capa de persistencia para recuperar y almacenar entidades de datos. Los servicios locales contienen la lógica de negocio y acceden a la capa de persistencia. Pueden ser invocados por el código de cliente que se ejecuta en la misma máquina virtual Java. Los servicios remotos suelen tener código adicional para la comprobación de permisos y están destinados a ser accesibles desde cualquier lugar a través de Internet o de su red local. *Service Builder* genera automáticamente el código necesario para permitir el acceso a los servicios remotos (32).

Figura 2 Estructura por paquetes. Elaboración propia.



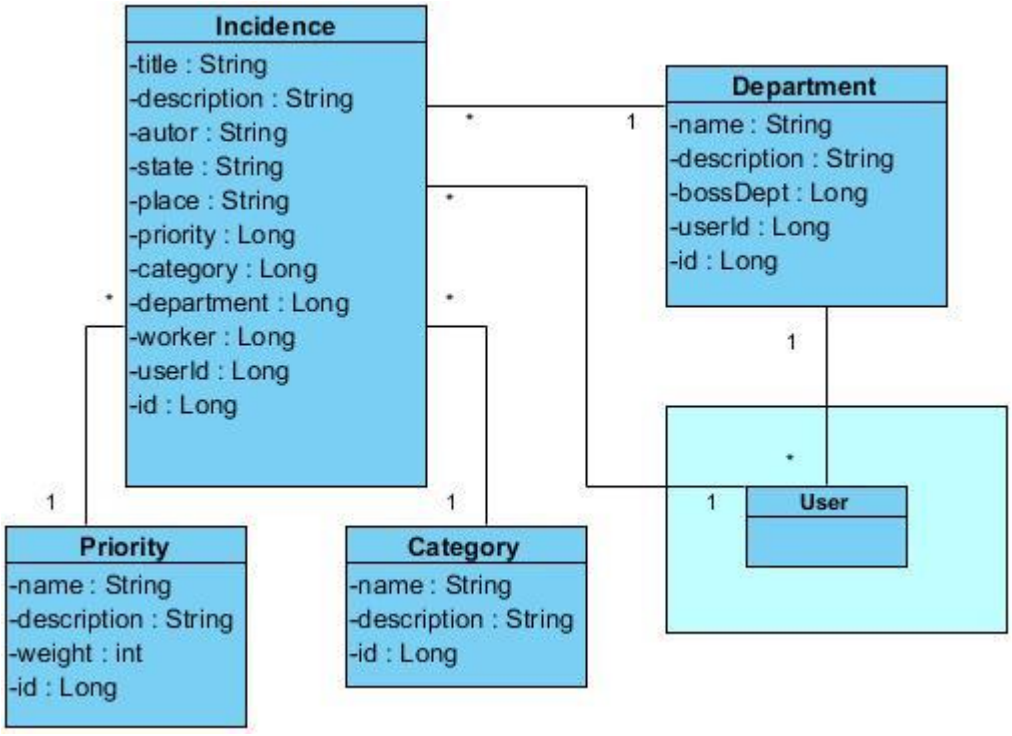
Otra forma en que *Service Builder* ahorra tiempo de desarrollo es proporcionar configuraciones Spring e Hibernate para su proyecto. *Service Builder* utiliza la inyección de dependencia de Spring para hacer que las clases de implementación de servicio estén disponibles en tiempo de ejecución y use Spring AOP (Programación Orientada a Aspectos) para la administración de transacciones de base de datos. *Service Builder* también utiliza el marco de persistencia de Hibernate para el mapeo objeto-relacional. Como conveniencia para los desarrolladores,

también oculta las complejidades de usar estas tecnologías. Los desarrolladores pueden aprovechar la inyección de dependencias (DI), AOP y la asignación de objetos y relaciones (ORM) en sus proyectos sin tener que configurar manualmente un entorno Spring o Hibernate ni realizar ninguna configuración (32).

2.4 Modelado conceptual

El modelado conceptual permite describir, de un modo totalmente independiente de la implementación, los datos que el usuario quiere recoger en el sistema. Dependiendo de la cantidad de información que se desee representar, se tendrán aplicaciones más o menos orientadas a los datos (33).

Figura 3 Modelo conceptual. Elaboración propia.



A continuación se describen las clases del modelo conceptual.

Tabla 1 Descripción del diagrama de clases del modelo conceptual.

Nombre	Descripción
Incidence	Es la clase que almacena los datos de todas las incidencias ocurridas.
Priority	Es la clase que almacena el tipo de impacto que tendría para la empresa la incidencia ocurrida, dígase, si tiene prioridad alta, que sería que tiene prioridad el darle solución sobre las demás, medio cuando la solución es parcialmente inmediata y baja cuando no es significativo el impacto en la entidad. Estos valores son adaptables a la empresa o institución que lo implemente.
Category	Es la clase que almacena las tipos de incidencias por categorías, dígase de *software*, *hardware* o simplemente de gestión.
Department	Es la clase que almacena los departamentos que tiene la entidad a la que pertenecen los trabajadores y que poseen un rol determinado en la misma, además de contener el jefe del departamento.
User	Son las personas físicas que trabajan en la empresa y que pueden o no hacer uso del sistema.

2.5 Modelado de requisitos del sistema

La gestión de requisitos es una parte vital en el desarrollo de proyectos de *software* puesto que define el propósito, la dirección y el tamaño del proyecto y por tanto, en mayor o menor medida, condicionará el éxito global del mismo. Es importante la comunicación iterativa con el cliente, con el fin de definir y registrar adecuadamente qué se espera del proyecto. Se fomenta la calidad a través de la trazabilidad, es decir, el proyecto finalizará correctamente si se trabaja con el menor número de cambios o errores en los objetivos. En todo momento hay que tratar y controlar las actualizaciones y cambios en los requisitos y actuar en consecuencia para poder seguir teniendo como objetivo principal la satisfacción del cliente (34).

A continuación se exponen los requisitos del sistema, obtenidos a través de entrevistas a jefes de proyectos y desarrolladores de portales pertenecientes al CDAE con experiencia en el uso

del *Liferay*, así como a trabajadores de las instituciones mencionadas en la introducción del documento. Consultándoles las deficiencias que les ocasiona la ausencia de un componente capaz de gestionar las incidencias en el portal *Liferay* a la hora de la toma de decisiones.

2.5.1 Requisitos funcionales (RF)

Tabla 2 Requisitos funcionales.

No	Nombre	Descripción	Complejidad
RF1	Gestionar incidencia. • Registrar incidencia. • Actualizar incidencia. • Mostrar incidencia. • Asignar incidencia. • Reasignar incidencia.	El administrador podrá gestionar las incidencias, dígase: registrar una nueva incidencia, editar los datos de la misma, asignar y reasignar la persona encargada de darle solución a la misma. Permitiendo además mostrar todos los datos o información de ésta.	Alta
RF2	Generar reportes. • Generar reporte por incidencia. • Generar reporte de incidencias por intervalo de fecha. • Generar reporte de incidencias por estado. • Generar reporte de incidencias por categoría. • Generar reporte de incidencias por prioridad. • Generar reporte de incidencias por departamento.	Permitirá generar varios tipos de reportes según sean las necesidades, dígase reportes por incidencia, por fecha, por estado, por categoría, por prioridad, por departamento, reportes asignados a un trabajador o creados por un trabajador y todos estos en formato PDF.	Media

	• Generar reporte de incidencias asignadas a un trabajador. • Generar reporte de incidencias realizadas por un trabajador.		
RF3	Asignar trabajador.	Permitirá al administrador asignar el personal de la empresa a un determinado departamento.	Alta
RF4	Gestionar departamento. • Registrar departamento. • Actualizar departamento. • Mostrar departamento. • Eliminar departamento.	Permitirá al administrador gestionar los departamentos de la empresa, dígase: registrar un nuevo departamento, visualizar todos los datos del mismo, así como eliminarlo del sistema y actualizar la información referente a él.	Alta
RF5	Gestionar prioridad. • Registrar prioridad. • Actualizar prioridad. • Mostrar prioridad. • Eliminar prioridad.	Permitirá al administrador gestionar las prioridades que puede tener una incidencia, dígase: registrar una nueva prioridad, visualizar todos los datos de la misma, así como eliminarla del sistema y actualizar la información referente a ella.	Alta
RF6	Gestionar categoría. • Registrar categoría. • Actualizar categoría.	Permitirá al administrador gestionar las categorías que puede tener una incidencia, dígase: registrar una nueva categoría, visualizar todos los datos de	Alta

	• Mostrar categoría. • Eliminar categoría.	la misma, así como eliminarla del sistema y actualizar la información referente a ella.	
RF7	Realizar búsqueda de incidencia.	Permitirá realizar una búsqueda de todas las incidencias registradas en el sistema, actuando por orden del usuario que especifica un término o una frase, dígase título de la incidencia, descripción, lugar, estado, categoría prioridad o departamento. Una vez finalizado el análisis, el sistema mostrará la incidencia o las incidencias que coincidieron según los parámetros mencionados anteriormente, permitiendo consultar todos sus detalles.	Media

2.5.2 Requisitos no funcionales (RNF)

Tabla 3 Requisitos no funcionales.

No	Nombre	Descripción
RNF1	Soporte	El servidor de base de datos será PostgreSQL Server 9.4.x o puede ser MySQL u otro superior a éstos.
RNF2	Soporte	El sistema se ejecutará en los navegadores Mozilla Firefox 51.0, Internet Explorer 11.0, Chrome 51, Safari for Windows 5.1.7 u Opera 41 o en otro superior a los mencionados.
RNF3	Soporte	Se usará el servidor web Apache Tomcat 8.0.x.

2.6 Historias de Usuario

Se hace uso de las historias de usuario debido a que son una forma rápida de administrar los requisitos, sin tener que elaborar gran cantidad de documentos formales.

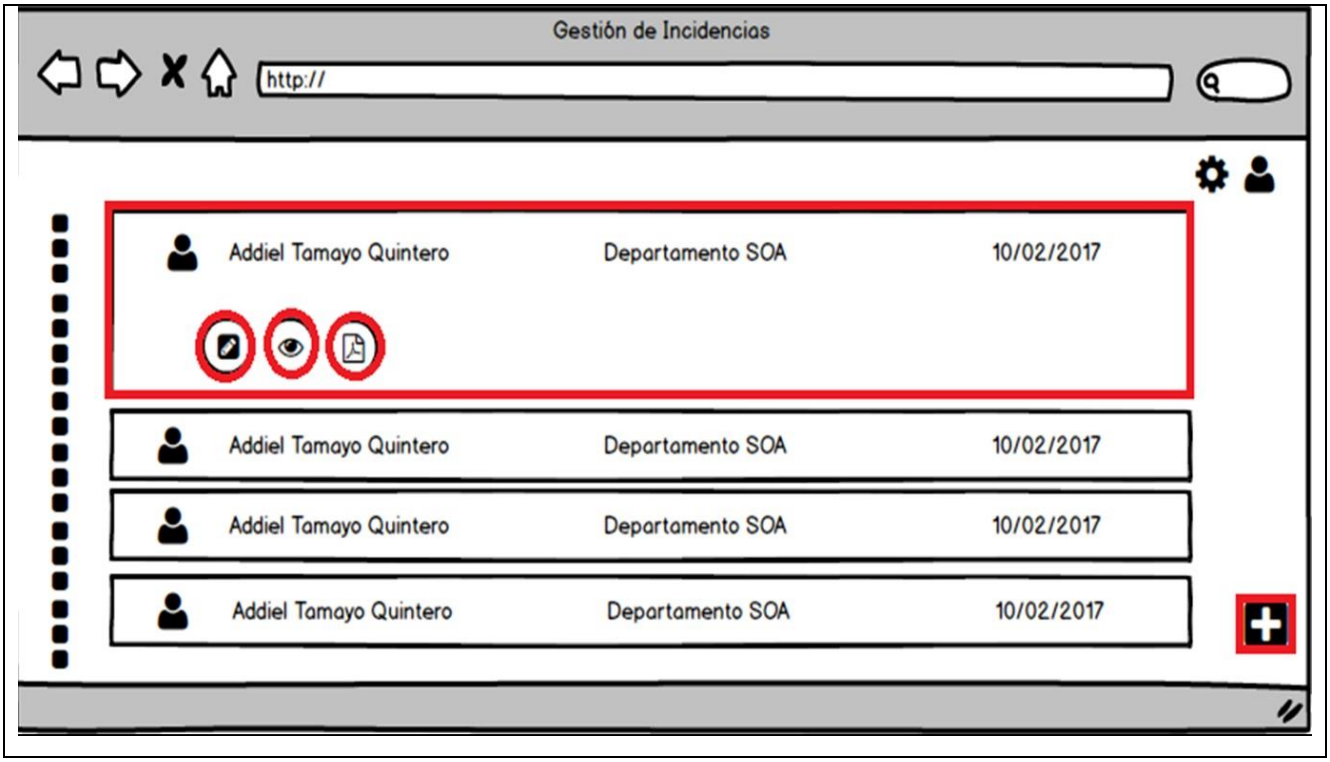
Otras de las ventajas que brindan las historias de usuario son (35):

- Potencian la participación del equipo en la toma de decisiones.
- Se crean y evolucionan a medida que el proyecto avanza.
- Son peticiones concretas y pequeñas.
- Contiene la información imprescindible.
- Apoyan la cooperación, colaboración y conversación entre los miembros del equipo, lo que es fundamental.

A continuación se describen las historias de usuario.

Tabla 4 Historia de usuario obtener listado de roles.

Historia de Usuario	
Número: HU1	**Nombre del requisito:** Gestionar incidencia.
Programador: Addiel Tamayo Quintero	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 10 día
Riesgo en desarrollo: o Problemas eléctricos. o Problemas técnicos de los servicios de redes.	**Tiempo real:** 80 horas
Descripción: El administrador luego de haber seleccionado el *portlet* gestión de incidencias que se encuentra en el panel de control, será capaz de obtener la lista de todas las incidencias que fueron registradas en el sistema. Una vez seleccionada la incidencia, se mostrarán las diferentes funcionalidades: en la parte inferior se mostrará la opción de actualizar incidencia a través de un icono representado por un lápiz, mostrar incidencia a través de un icono representado por un ojo , y por último generar reporte de la incidencia seleccionada, con el logo del documento PDF. También en la parte inferior derecha de la interfaz principal estarán las diferentes funcionalidades donde se podrá registrar una nueva incidencia al accionar el botón de color rojo con el signo de “+”.	
Observaciones:	
Prototipo de interfaz:	



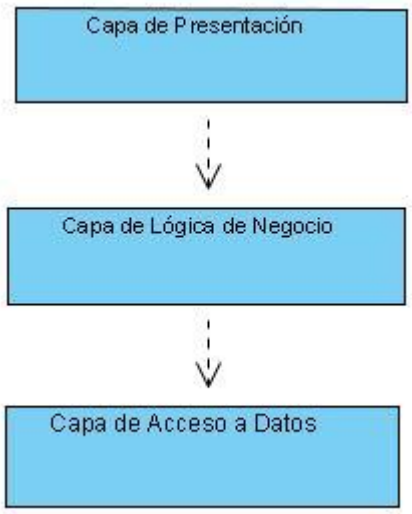
Las demás Historias de usuario se pueden ver en el Anexo 2.

2.7 Arquitectura de la aplicación

"La AS (Arquitectura de Sistema) empieza a ser formada desde que inicia el ciclo de la especificación de los requisitos de *software*, cuando se establecen los puntos de referencias del dominio del problema. Para su entendimiento el arquitecto debe interpretar, conciliar y proponer alternativas de solución en función del conocimiento de los expertos del dominio y de los planificadores del proyecto" (36).

Para el desarrollo de la solución se decidió utilizar una arquitectura en tres capas, lo que permitirá que las distintas partes de la aplicación se puedan modificar sin que esto afecte las demás capas, es decir, separar los diferentes aspectos del desarrollo, tales como las cuestiones de presentación, lógica de negocio y mecanismos de almacenamiento.

Figura 4 Arquitectura de la aplicación. Elaboración propia.



Capa de presentación: Se refiere a la presentación del programa frente al usuario, esta presentación debe cumplir su propósito con el usuario final, una presentación fácil de usar y amigable.

Para el desarrollo de la solución la capa de presentación fue basada en el *framework* Bootstrap para facilitar el manejo de las peticiones realizadas por los usuarios (37).

Capa de lógica de negocios: En esta capa es donde se encuentran los programas que son ejecutados, recibe las peticiones del usuario y posteriormente envía las respuestas tras el proceso. Esta capa es muy importante pues es donde se establecen todas aquellas reglas que se tendrán que cumplir (37).

La lógica de negocio es la intermediaria entre la capa de presentación y la capa de acceso a datos, haciéndose uso de Spring contenido dentro del *Service Builder* como mecanismo de comunicación.

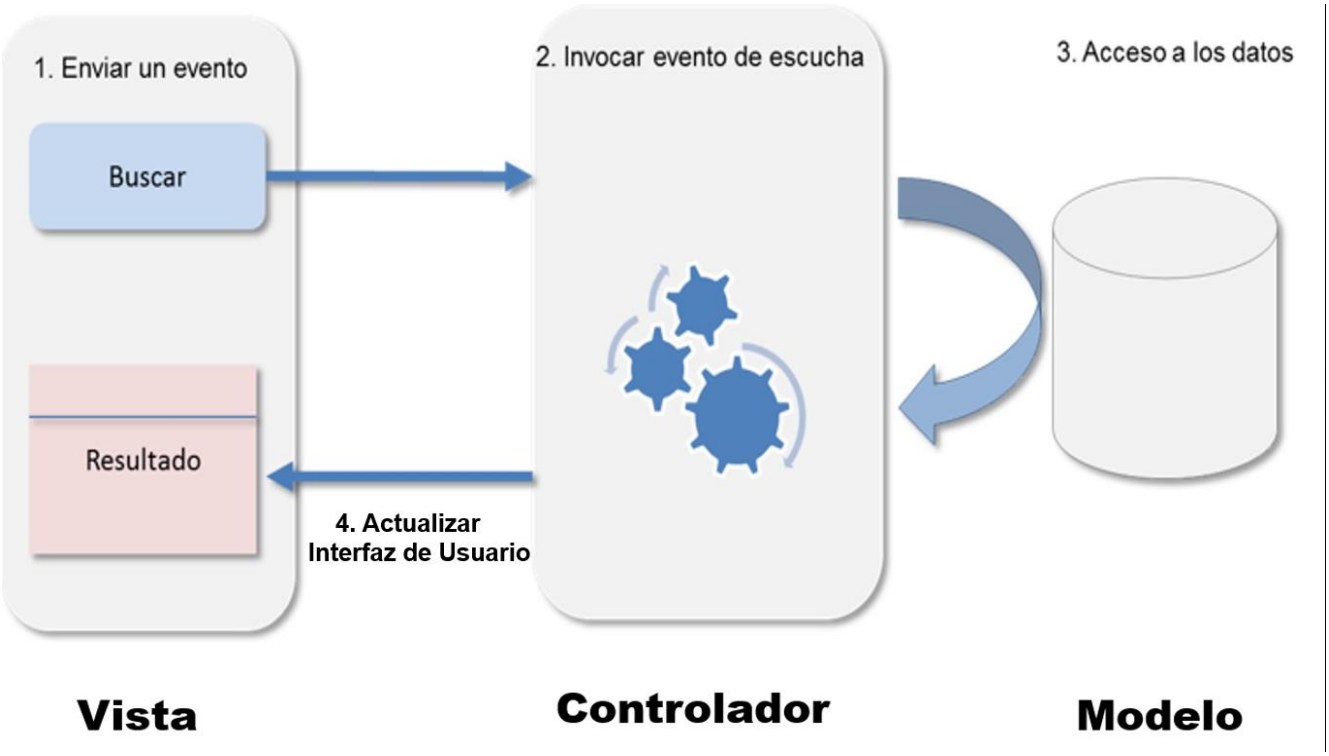
Capa de acceso a datos: Esta capa es la que se encarga de hacer las transacciones con la base de datos y con otros sistemas para descargar o insertar información al sistema. La consistencia en los datos es sumamente importante, es decir, los datos que se ingresan o insertan deben ser precisos y consistentes (37). Esta capa mantendrá la comunicación con la lógica de negocio al enviar la información que será procesada e ingresada en objetos según sea necesario.

La capa de acceso a datos es la encargada de satisfacer las peticiones del negocio a través de los Objetos de Acceso a Datos (DAO), para ello, se utilizan interfaces DAO y sus implementaciones, haciendo uso del *framework* Hibernate para persistir los datos, donde este *framework* se encuentra embebido en el *Service Builder*.

2.8 Patrón arquitectónico Modelo Vista Controlador

Modelo Vista Controlador (MVC): Es una propuesta de diseño de *software* utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. Surge de la necesidad de crear *software* más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos (38).

Figura 5 Patrón Modelo Vista Controlador.



A continuación se describe el funcionamiento del patrón arquitectónico MVC (39):

El **modelo** representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El modelo no tiene conocimiento específico de los controladores o de las vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el modelo y sus vistas, y notificar a las vistas cuando cambia el modelo.

La **vista** maneja la presentación visual de los datos representados por el modelo. Genera una representación visual del modelo y muestra los datos al usuario. Interactúa preferentemente con el controlador, pero es posible que trate directamente con el modelo a través de una referencia al propio modelo.

El **controlador** proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el modelo, centra toda la interacción entre la vista y el modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del modelo o por alteraciones de la vista. Interactúa con el modelo a través de una referencia al propio modelo.

Queda evidenciado el uso del patrón en la capa de presentación, donde la vista va a estar representada a través de las JSP, el controlador lo implementa la clase *ManageController*, y el modelo, es simbolizado por las entidades: *Incidence*, *Category*, *Priority* y *Department*, donde la clase *DataManageUtil* es la encargada de proveer al controlador el modelo de dato con el cual se construirá la vista.

2.9 Patrones de diseño

Un patrón es un conjunto de información que proporciona respuesta a un conjunto de problemas similares. Para ello se aíslan los aspectos comunes y su solución y se añaden cuantos comentarios y ejemplos sean oportunos. Los patrones ayudan a capturar conocimiento y a crear un vocabulario técnico, hacen el diseño orientado a objetos más flexible, elegante y en algunos casos reusable (39).

Patrón DAO (*Data Access Object*): el problema que viene a resolver este patrón es el de contar con diversas fuentes de datos (base de datos, archivos, servicios externos). De tal forma que se encapsula la forma de acceder a la fuente de datos. Este patrón surge históricamente de la necesidad de gestionar una diversidad de fuentes de información, aunque su uso se extiende al problema de encapsular no sólo la fuente de datos, sino además ocultar la forma de acceder a estos.

Queda evidenciado el uso del patrón a través de las interfaces, y sus implementaciones, haciendo uso de los *frameworks* Spring e Hibernate para persistir los datos.

2.9.1 Patrones GRASP

Se emplearon los patrones de asignación de responsabilidades para el desarrollo del proceso, conocidos como patrones **GRASP**[7] los cuales describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (40).

Controlador: el patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es el controlador quien recibe los datos del usuario y quien los envía a las distintas clases según el método llamado (41).

Asigna la responsabilidad de gestionar un mensaje de un evento del sistema a una clase que represente una de estas opciones:

1. Representa el sistema global, dispositivo o un subsistema (controlador de fachada).
2. Representa un escenario de caso de uso en el que tiene lugar el evento del sistema (controlador de caso de uso o sesión) (42).

Queda evidenciado el uso del patrón a través de la clase *ManageController*, que va a facilitar las peticiones realizadas por los usuarios y que tendrán algún impacto o requieran de una respuesta de la aplicación.

Experto en información: el patrón experto sugiere asignar la responsabilidad al objeto que posea la información necesaria para desempeñarla (42) (43).

Queda evidenciado el uso del patrón a través de las clases del modelo: *Incidence*, *Category*, *Priority*, *Department*, que son las entidades del negocio.

Alta cohesión: el patrón alta cohesión es una medida que determina cuán relacionadas y adecuadas están las responsabilidades de una clase, de manera que no realice un trabajo colosal; una clase con baja cohesión realiza un trabajo excesivo, haciéndola difícil de comprender, reutilizar y conservar (42) (44).

Queda evidenciado el uso del patrón a través de la clase: *ManageController*, que sería aquel que controla el estado de las páginas web.

Bajo Acoplamiento: es una medida de la fuerza con que una clase se relaciona con otras; es decir, no posee numerosas relaciones (42) (44).

[7] Patrones Generales de *Software* para Asignar Responsabilidades, del inglés *General Responsibility Assignment Software Patterns.*

Queda evidenciado a través de la arquitectura en capas, donde estaría estructurada por paquetes: interfaces, *service*, *model*, evitando el acceso directo a los datos, es decir, que si existe algún cambio en las capas inferiores no afectaría las capas superiores (42) (44).

2.9.2 Patrones GOF

También se emplearon los patrones **GOF**[8] los cuales se clasifican en dependencia del propósito para los que hayan sido definidos: creación, estructurales y de comportamiento (45).

Facade: Se basa en poner una clase principal de control como fachada de otros objetos tras estos. El objetivo es simplificar al máximo la gestión y las operaciones de diversos tipos de objetos a través de un controlador único que nos permita operaciones más simples y unificadas (46).

El uso del patrón mencionado anteriormente, queda evidenciado a través de las clases *LocalServiceUtil* y *DataManageUtil*.

Singleton: garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella (47).

El patrón de diseño *singleton* (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto (47).

El uso del patrón mencionado anteriormente, queda evidenciado a través de la clase *Service*, que proporciona un punto de acceso global, manejado por Spring, actuando como mecanismo de comunicación con la capa de acceso a datos la cual es responsable del manejo de los datos.

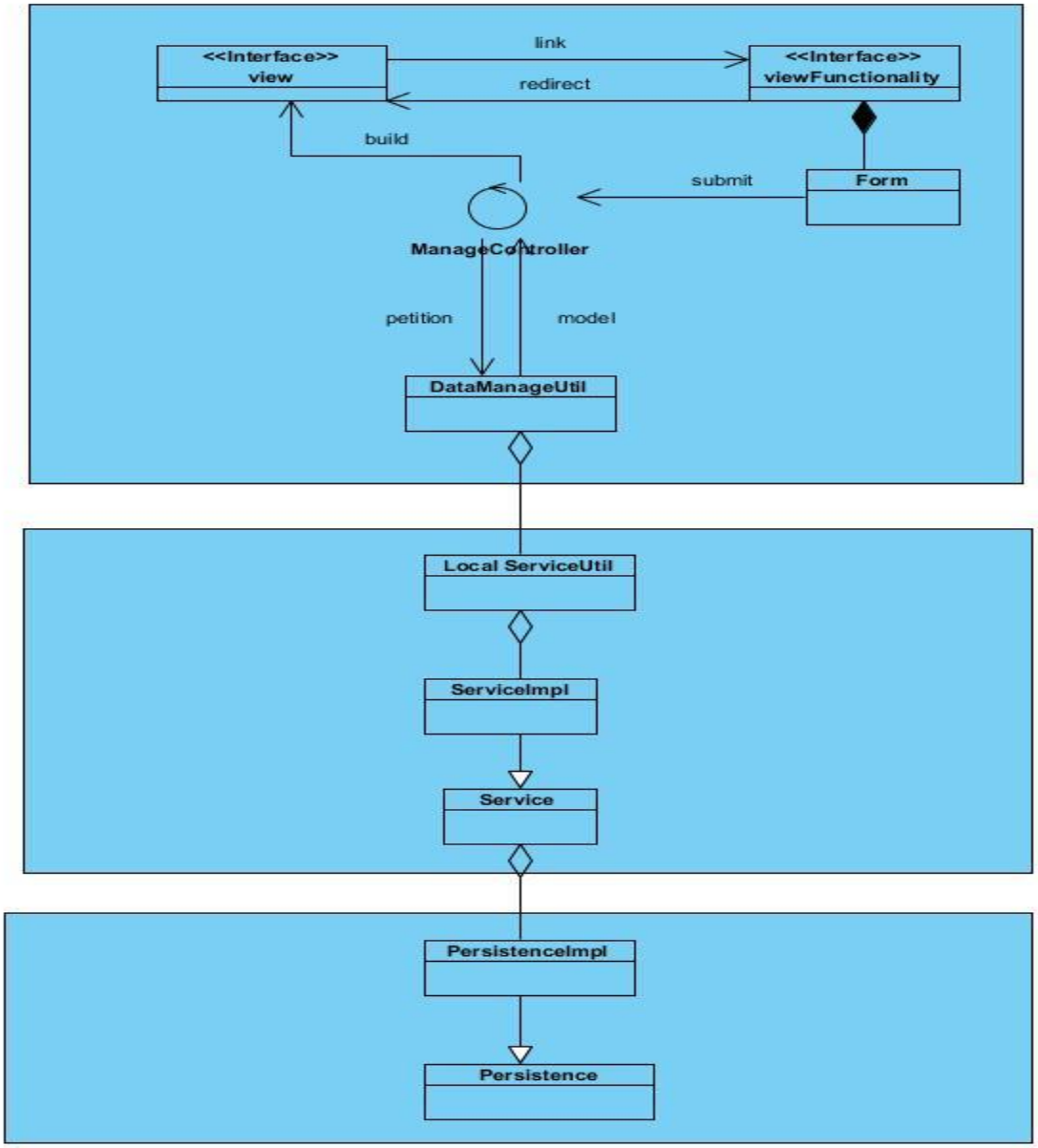
[8] Banda de los Cuatro, del inglés *Gang-Of-Four*

2.10 Modelo de diseño

El modelo de diseño es una representación conceptual, gráfica o visual de sistemas o procesos, donde se describen o muestran las relaciones entre las clases que involucra el sistema. Se obtiene la información acerca del dominio de aplicación para el *software* que se va a construir.

Diagramas de clases:

Figura 6 Diagrama de clases. Elaboración propia.



Descripción de las clases del diseño

Tabla 5 Descripción de las clases del diseño.

Nombre	Descripción
View	Es la interfaz principal mostrada al usuario al solicitar el *portlet* que gestiona las incidencias, permitiéndole navegar por las demás interfaces.
***[9]ViewFuntionality**	Conjunto de vistas que referencian las distintas funcionalidades que muestra el sistema. Estas interfaces de usuario tienen como objetivo comunicar información detallada a los mismos, como por ejemplo, gestionar departamentos, categorías y prioridades. Así como también asignar incidencias y trabajadores.
ManageController	Es el encargado de recibir las peticiones realizadas por el usuario, para una vez procesadas, dar la respuesta obtenida de la capa de lógica de negocio.
DataManageUtil	Procesa las peticiones realizadas al controlador, sirviendo de enlace entre las capas de presentación y lógica de negocio. Se utiliza como un punto de acceso común a todas las funcionalidades de las clases del negocio.
[9]LocalServiceUtil	Estas clases son expertas en información, utilizando las funcionalidades que implementa el *ServiceImpl* especializándose en cada modelo de datos.
[9]ServiceImpl	Estas clases son las que implementan las interfaces *Service*.
[9]Service	Estas clases son interfaces para hacer más fácil el acceso a las funcionalidades declaradas unificando las acciones más comunes,

[9] Referencia a un conjunto de clases.

	permitiendo comunicar la capa lógica de negocio con la capa de acceso a datos.
[9]***PersistenceImpl***	Estas clases implementan todas las funcionalidades declaradas en las clases *Persistence* para acceder a los datos almacenados o por almacenar.
[9]***Persistence***	Estas clases declaran todas las funcionalidades necesarias para acceder a los datos almacenados o por almacenar.

2.11 Conclusiones parciales

- Para un correcto trabajo en el cumplimiento de los requisitos definidos, se estableció la arquitectura que debe tener la aplicación a desarrollar, así como los patrones de diseño.
- Se expusieron y explicaron los artefactos necesarios generados por la metodología AUP, para entender el negocio y dar cumplimiento a las tareas que impuso la solución propuesta.

Capítulo #3. Implementación y prueba del sistema

3.1 Introducción

En el capítulo se describe el proceso de implementación y prueba del Componente de gestión de incidencias para el portal *Liferay Community*, plasmando los casos de prueba realizados y demostrando el cumplimiento de los requisitos trazados.

3.2 Estándar de código

Un estándar de programación es (48):

- Una forma de "normalizar" la programación de forma tal que al trabajar en un proyecto, cualquier persona involucrada en el mismo tenga acceso y comprenda el código.

Y permite:

- Definir la escritura y organización del código fuente de un programa.
- Facilita a un programador la modificación de un código fuente aunque no esté trabajando en el equipo.
- Definir la forma en que deben ser declaradas las variables, las clases y los comentarios.
- Especificar qué datos deben incluirse acerca del programador y de los cambios realizados al código fuente.

Variables:

Para los identificadores se hace uso de la variante *lowerCamelCase*, que es una convención de nomenclatura en la que un nombre está formado por varias palabras que se unen como una sola palabra y empiezan en minúscula y las siguientes palabras en mayúscula (49).

Métodos:

Los nombres de los métodos deben iniciar con un verbo, como se muestra en la sentencia de código:

```
public void registerCategoryActionURL(ActionRequest actionRequest, ActionResponse actionResponse)
```

Los métodos para obtener campos privados en las clases tienen el prefijo "*get*", como se muestra en la sentencia de código:

```
public ExpandoBridge getExpandoBridge();
```

Los modificadores de campos privados en las clases tienen el prefijo "*set*", como se muestra en la sentencia de código:

```
public void setCachedModel(boolean cachedModel);
```

Los métodos para obtener el resultado de booleano tienen el prefijo "*is*", como se muestra en la sentencia de código:

```
public boolean isEscapedModel();
```

Constantes:

Las constantes o campos finales son escritos en letras minúsculas, como se muestra en la sentencia de código:

```
IncidenceSoap soapModel = new IncidenceSoap();
```

Clases:

Para estos identificadores se hace uso de la variante *UpperCamelCase* (49). Todas las palabras que componen a dichos identificadores empezarán con mayúscula, como se muestra en la sentencia de código:

```
public class IncidenceSoap implements Serializable
```

Comentarios de las clases:

Al inicio de cada clase se debe describir con un comentario de bloque el propósito de la clase e instrucciones de uso, como se muestra en el siguiente ejemplo:

```
/**
 * The extended model interface for the Incidence service. Represents a row in the &quot;gestion_Incidence&quot; database table, with each column mapped to a
 * property of this class.
 *
 * @author Addiel
 * @see IncidenceModel
 * @see com.tesis.gestion.incidencia.model.impl.IncidenceImpl
 * @see com.tesis.gestion.incidencia.model.impl.IncidenceModelImpl
 * @generated
 */
```

3.3 Pruebas

Descripción de algunos de los diferentes tipos de pruebas (50):

Un conjunto de actividades de pruebas suele orientarse a comprobar determinados aspectos de un sistema de *software* (o de una parte del mismo).

En primer lugar, se tienen las pruebas funcionales. Típicamente se encuentra el comportamiento del sistema, subsistema o componente *software* descrito en especificaciones de requisitos o casos de uso, aunque también puede no estar documentado ("que funcione como el sistema al que sustituye"). Es decir, con las funciones se establece "lo que el sistema hace".

Estas pruebas se definen a partir de funciones o características (bien descritas en documentos o bien interpretadas por los probadores) y su interoperabilidad con sistemas específicos, pudiendo ejecutarse en todos los niveles de pruebas (unitarias, integración, sistema y aceptación).

Se consideran pruebas de caja negra ("*black-box testing*") puesto que se valora el comportamiento externo del sistema. Las pruebas de seguridad o las pruebas de interoperabilidad entre sistemas o componentes son casos especializados de las pruebas funcionales.

En segundo lugar figuran las pruebas no funcionales que incluyen las pruebas de: rendimiento, carga, estrés, usabilidad, mantenibilidad, fiabilidad o portabilidad, entre otras. Por tanto se centran en características del *software* que establecen "cómo trabaja el sistema".

Estas pruebas también pueden ejecutarse en todos los niveles de pruebas. Las características no funcionales del *software* se pueden medir de diversas maneras, por ejemplo, por medio de tiempos de respuesta en el caso de pruebas de rendimiento o por número máximo de sesiones en pruebas de estrés.

Puesto que las pruebas no funcionales normalmente consideran el comportamiento externo del sistema, en la mayoría de los casos se utilizan técnicas de pruebas de caja negra.

A continuación, en tercer lugar, se tienen las pruebas estructurales. Nuevamente pueden ejecutarse en todos los niveles de pruebas (unitarias, integración, sistema, aceptación) y encajan muy bien si son utilizadas técnicas de especificación de la estructura o arquitectura del *software*. Es posible aplicar técnicas estáticas de análisis de código.

Para expresar el alcance con un conjunto de pruebas ("*test suite*") que ha cubierto la estructura o arquitectura en cuestión, se utiliza el concepto de cobertura ("*Coverage*"), normalmente en forma de porcentaje.

Es especialmente habitual utilizar herramientas de apoyo para calcular la cobertura del código en el caso de pruebas de componentes o en pruebas de integración de componentes (por ejemplo, trazando la jerarquía de llamadas entre elementos). Puesto que se indaga en el comportamiento interno, estas pruebas se denominan también pruebas de caja blanca ("*white-box testing*").

Finalmente, el cuarto tipo de pruebas que presenta el *ISTQB (International Software Testing Qualifications Board)* son las pruebas derivadas de la realización de cambios: las pruebas de regresión y las re-pruebas.

Una vez que un defecto ha sido corregido, toca volver a probar el *software* para confirmar que el defecto ha sido eliminado. Son pruebas repetidas o re-pruebas.

Las pruebas de regresión consisten en volver a probar un componente, tras haber sido modificado, para descubrir cualquier defecto introducido, o no cubierto previamente, como consecuencia de los cambios. Los defectos pueden encontrarse tanto en el *software* que se ha cambiado como en algún otro componente. Se ejecutan cuando se cambia el *software* o su entorno. El criterio para decidir la extensión de estas pruebas de regresión está basado en el riesgo de no encontrar defectos en el *software* que anteriormente estaba funcionando correctamente.

Las pruebas de regresión se realizan sobre un componente ya probado, para verificar que no presenta nuevos defectos cuando se realiza una modificación después de dichas pruebas.

Este tipo de pruebas deben ser repetibles si han de usarse para pruebas de confirmación (o aseguramiento) y regresión (como sondas de disponibilidad, por ejemplo). Los conjuntos de pruebas de regresión ("*Regression test suites*") suelen ser bastante estables por lo que son muy buenos candidatos para actividades de automatización de pruebas.

Luego de realizar un estudio de las diferentes pruebas definidas dentro de la ingeniería de *software*, se escogen mediante técnicas experimentales las pruebas de caja negra para determinar el nivel de factibilidad y calidad del producto.

Pruebas de caja negra

- Pruebas que se llevan a cabo sobre la interfaz del *software*.
- El objetivo es demostrar que las funciones del *software* son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene (no se ve el código).

Técnica de la partición de equivalencia

La partición de equivalencia es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

De tal modo que se pueda asumir razonablemente que una prueba realizada con un valor representativo de cada clase es equivalente a una prueba realizada con cualquier otro valor de dicha clase. Esto quiere decir que si el caso de prueba correspondiente a una clase de equivalencia detecta un error, el resto de los casos de prueba de dicha clase de equivalencia deben detectar el mismo error. Y viceversa, si un caso de prueba no ha detectado ningún error, es de esperar que ninguno de los casos de prueba correspondientes a la misma clase de equivalencia encuentre ningún error.

Casos de prueba

Las celdas de la tabla contienen **V, I,** o **N/A**. **V** indica válido, **I** indica inválido, y **N/A** que no es necesario proporcionar un valor del dato en este caso, dado que es irrelevante.

Tabla 6 Caso de prueba gestionar incidencia.

Id del escenario	**Escenario**	**Variable 1** Autor	**Variable 2** Título	**Variable 3** Descripción	**Variable 4** Departamento	**Variable 5** Categoría	**Variable 6** Prioridad	**Variable 7** Estado	**Variable 8** Lugar	**Respuesta del sistema**	**Resultado de la prueba**
EC 1	Registrar incidencia.	V	V	V	V	V	V	V	V	Mensaje de confirmación.	Se agrega una incidencia al sistema.
		I	V	V	V	V	V	V	V	Notificación de error.	
		V	I	V	V	V	V	V	V	Notificación de error.	
		V	V	I	V	V	V	V	V	Notificación de error.	
		V	V	V	I	V	V	V	V	Notificación de error.	

		V	V	V	V	I	V	V	V	Notificación de error.	
		V	V	V	V	V	I	V	V	Notificación de error.	
		V	V	V	V	V	V	I	V	Notificación de error.	
		V	V	V	V	V	V	V	I	Notificación de error.	
EC 2	Actualizar incidencia.	V	V	V	V	V	V	V	V	Mensaje de confirmación.	Se actualizó satisfactoriamente la incidencia.
		I	V	V	V	V	V	V	V	Notificación de error.	
		V	I	V	V	V	V	V	V	Notificación de error.	
		V	V	I	V	V	V	V	V	Notificación de error.	

		V	V	V	I	V	V	V	V	Notificación de error.	
		V	V	V	V	I	V	V	V	Notificación de error.	
		V	V	V	V	V	I	V	V	Notificación de error.	
		V	V	V	V	V	V	I	V	Notificación de error.	
EC 3	Mostrar incidencia.	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Se muestran los detalles de la incidencia seleccionada.	Se muestra la incidencia con todos los datos referentes a la misma.

Los demás casos de prueba se pueden ver en los Anexo 3.

Lista de chequeo

Lista de chequeo es una herramienta de gran importancia, que en una empresa que está desarrollando o tiene implementado un sistema de gestión, le va a servir para detectar peligros antes de iniciar una labor, actualizar el panorama de peligros, materia prima para elaborar o ajustar procedimientos seguros (51).

Tabla 7 Lista de chequeo.

Característica de soporte por comprobar	**Ponderación de importancia 1(poco), 5 (fundamental)**	**Nivel de cumplimiento (0-100)**	**Justificación**
¿Se utilizó como servidor de base de datos PostgreSQL 9.4 para el desarrollo del sistema?	4	100	Se utilizó el gestor de base de datos PostgreSQL 9.4, por ser un requisito establecido por el cliente, además, es uno de los más usados en la Universidad de Ciencias Informáticas. Debido a la liberación de la licencia. El código fuente se encuentra disponible sin costo alguno.
¿El *portlet* administrativo es adaptable a los navegadores Mozilla Firefox 51.0, Internet Explorer 11.0, Chrome	5	100	El sistema se ejecutó en diferentes entornos de navegación: Mozilla Firefox, Internet Explorer, Chrome, Safari y Opera.

51, Safari for Windows 5.1.7 u Opera 41.0?			
¿Para desarrollar la aplicación se utilizó como servidor web el Apache Tomcat 8.0.32?	5	100	Se utilizó el servidor web el Apache Tomcat por ser requerido por el cliente. Posee ayuda y abundante documentación. Cuenta con una comunidad de usuarios, es libre y uno de los más usados en el CDAE. Para trabajar con esta herramienta se instaló la Máquina Virtual de Java.

3.4 Resultados

Se realizaron tres iteraciones de las pruebas de caja negra específicamente con la utilización de casos de prueba y técnica de partición de equivalencia, para comprobar que la solución funcionara correctamente y si satisfacía los requisitos del cliente. Durante la primera iteración se encontraron ocho no conformidades, una de prioridad alta y siete de prioridad baja. La no conformidad de prioridad alta fue con respecto a mal funcionamiento de validación de la aplicación y las bajas estuvieron relacionadas con mal funcionamiento de mensajes de confirmación. Se realizó una segunda iteración, en la cual se encontraron dos no conformidades de prioridad baja relacionadas con los mensajes de confirmación, quedando resueltas por el desarrollador de la aplicación y por último se realizó una tercera iteración donde no se detectaron fallos en el sistema, evaluándose de esta manera todos los casos de prueba como satisfactorios. Permitiendo validar el buen funcionamiento del *portlet y* el correcto registro de las incidencias en el mismo. Además de monitorizar y controlar todo el proceso a través de los reportes generados con los detalles de las incidencias y por los registros históricos que son almacenados en el componente.

3.5 Conclusiones parciales

- En el capítulo para sintetizar la implementación y pruebas del sistema, se presentaron los casos de prueba, a través del uso de técnicas de partición de equivalencia y lista de chequeo.
- Las pruebas que se le realizaron al sistema trajeron consigo mejoras en la implementación de las diferentes capas de la aplicación, logrando una apariencia más amigable con el cliente.
- Se logró validar el correcto registro de las incidencias en el sistema, de manera que puedan ser monitorizadas constantemente.

Conclusiones generales

- Se desarrolló un sistema robusto construido sobre una base teórica sólida a través de la puesta en práctica de las herramientas y tecnologías abordadas en el marco teórico.
- La metodología de desarrollo de *software* utilizada permitió la definición de las Historias de Usuario, logrando así ver con claridad los requisitos funcionales.
- El componente “Gestión de Incidencias” permitió un control y monitorización del servicio, a través del registro histórico de la información detallada de cada incidencia.
- Se validó la solución a través del tipo de prueba funcional caja negra, arrojando como resultado el buen funcionamiento del *portlet* desarrollado, quedando de esta forma resuelto el problema investigativo.

Recomendaciones

- Mantener capacitado al personal de soporte tecnológico e involucrado en el proceso de gestión de incidencia y cambios, para un óptimo funcionamiento del sistema.
- Implementar el proceso de Gestión de Problemas, para garantizar el registro de soluciones de incidencias recurrentes.

Bibliografía

1. Rosas, Juan Eladio Sánchez. El Mundo es Open Source. Blog sobre Productos y Soluciones de Software Libre y código abierto de nivel empresarial. [En línea] 04 de Diciembre de 2012. [Citado el: 05 de Noviembre de 2016.]

2. Quetglas, Patricia Riera. Memoria Final. Proyecto Fin de Carrera. Catalunya : Universitat Oberta de Catalunya, 2013.

3. José Álex Evangelista Casas, Luis Daniel Uquiche Chircca. Mejora de los procesos de gestión de incidencias y cambios aplicando ITIL en la Facultad de Ingeniería y Arquitectura. Tesis para optar por el título profesional de Ingeniero de computación y sistemas. [En línea] 2014. [Citado el: 18 de Noviembre de 2016.] http://www.repositorioacademico.usmp.edu.pe/bitstream/usmp/1158/1/evangelista_c.pdf .

4. ITIL v3. Gestión de Incidencias. Qué es la Gestión de incidencias y sus principales actividades según ITIL v3. [En línea] ServiceTonic, 2017. [Citado el: 16 de Enero de 2017.] https://www.servicetonic.es/itil/itil-v3-gestion-de-incidencias/.

5. Hernández, Isabel Cuellar. Universidad Complutense Madrid. Proyecto MERLIN: plataforma de aprendizaje a distancia basada en tecnología de Portlets y Web 2.0 para una enseñanza participativa. [En línea] El repositorio de producción académica en abierto de la UCM, 06 de Febrero de 2014. [Citado el: 05 de Febrero de 2017.] http://eprints.ucm.es/9083/.

6. Liferay Portal Security Overview. Liferay Portal Security Overview. [En línea] Liferay Developer Network, 2017. [Citado el: 16 de Enero de 2017.] https://dev.liferay.com/discover/deployment/-/knowledge_base/7-0/liferay-portal-security-overview.

7. Corporation, Copyright IBM. Portlet concepts. [En línea] IBM Knowledge Center, 11 de Julio de 2011. [Citado el: 26 de Enero de 2017.] http://www.ibm.com/support/knowledgecenter/SSYJ99_6.1.0/com.ibm.wp.exp.doc_v6101/dev/wpsbpc.html?lang=es..

8. Scamarcio, Francesco. Introducción al Liferay Portal. [En línea] 30 de Septiembre de 2011. [Citado el: 11 de Enero de 2017.] http://francescoscamarcio.com/2010/12/10/introducion-a-liferay-portal/.

9. Romero, H. Metodologias de desarrollo. [En línea] es.slideshare.net, 2012. [Citado el: 12 de Febrero de 2017.]

10. Nieves, I.I.S. Metodologías Ágiles. Proceso Unificado Ágil (AUP). Ingeniería del Software II – Análisis de Sistemas. 2014. p.9.

11. Sánchez, T.R. PROGRAMA DE MEJORA Metodología de desarrollo para la Actividad productiva de la UCI. 2014.

12. Perry, J. Steven. Introducción a la programación Java, parte 1: Conceptos básicos del lenguaje Java. [En línea] 03 de Diciembre de 2012. [Citado el: 16 de Enero de 2017.] http://www.ibm.com/developerworks/ssa/java/tutorials/j-introtojava1/.

13. Salle, Universidad La. Laboratorio de Cómputo de Ingenierí. Curso de java básico. s.l. : Aprender Gratis, 2007.

14. Interactive: The Top Programming Languages 2016. Find the programming languages that are most important to you. [En línea] IEEE Spectrum, 2016. [Citado el: 18 de Enero de 2017.] http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2016.

15. ¿Qué lenguaje de programación aprender? [En línea] Universia España, 18 de Marzo de 2016. [Citado el: 18 de Enero de 2017.] http://noticias.universia.es/ciencia-tecnologia/noticia/2016/03/18/1137432/lenguaje-programacion-aprender.html.

16. Twitter. Bootstrap is the most popular HTML, CSS, and JS framework in the world for building responsive, mobile-first projects on the web. Easy to get started. [En línea] 19 de Mayo de 2017. [Citado el: 23 de Enero de 2017.] https://v4-alpha.getbootstrap.com/.

17. W3Schools.com. Bootstrap 3 Tutorial. Try it Yourself Examples. [En línea] W3.CSS., 2017. [Citado el: 23 de Enero de 2017.] https://www.w3schools.com/bootstrap/.

18. aulaformativa. Framework responsive alternativas a Bootstrap. [En línea] 18 de Febrero de 2016. [Citado el: 23 de Enero de 2017.] http://blog.aulaformativa.com/framework-responsive-alternativas-a-bootstrap/.

19. Liferay Developer Network. What is Service Builder? [En línea] 2017. [Citado el: 23 de Enero de 2017.] https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/what-is-service-builder.

20. LC, Jesús. Service Builder para desarrollar un portlet con gestión de base de datos Parte 1/3. [En línea] 10 de Mayo de 2013. [Citado el: 23 de Enero de 2017.] https://jesuslc.com/2013/05/10/utilizando-spring-para-desarrollar-un-portlet-para-liferay-con-base-de-datos-service-builder/.

21. Islam, Hamidul. Pro Liferay. An introduction to Liferay Service Builder. [En línea] 27 de Abril de 2015. [Citado el: 23 de Enero de 2017.] http://proliferay.com/an-introduction-to-liferay-service-builder/.

22. Briano, F. Introducción a Spring Framework Java. [En línea] 2010. [Citado el: 24 de Enero de 2017.] http://picandocodigo.net/2010/introduccion-a-spring-framework-java/.

23. Fulguera, J.O. Spring Framework. 2014.

24. Hernández, I.C. Plataforma de Aprendizaje a Distancia Basada en tecnología de Portlets y Web 2.0 para una Enseñanza Participativa. Sistemas Informáticos. 2008. p. 1-97.

25. EducacionIT. ¿Qué es Java Hibernate? [En línea] 2013. [Citado el: 24 de Enero de 2017.] http://blog.educacionit.com/2013/02/07/que-es-java-hibernate/.

26. Dev, G. Eclipse IDE. [En línea] 2014. [Citado el: 15 de Febrero de 2017.] http://www.genbetadev.com/herramientas/eclipse-ide.

27. Lorena CHAVARRÍA-BÁEZ, Departamento de Posgrado, Nancy OCOTITLA ROJAS, Departamento de Ingeniería en Sistemas Computacionales, Instituto Politécnico Nacional – Escuela Superior de Cómputo. Sobre el uso de herramientas CASE para la enseñanza de bases de datos. México : s.n., 2016.

28. Miguel Ángel Montiel Peña. UNIVERSIDAD CENTRAL “MARTA ABREU” DE LAS VILLAS. FACULTAD MATEMÁTICA FÍSICA Y COMPUTACIÓN. INGENIERÍA INFORMÁTICA. Desarrollo de Herramienta para Arquitectura Dirigida por Modelos JMDA versión 2.0 Modulo PSM-Código Fuente. 2012.

29. Sadradín, L.C. Universidad de las Ciencias Informáticas. Sistema automatizado para el diseño de prototipos de interfaces de usuario. 2009. p.102.

30. González, L.C. y E.R.P. Torres. Extensión de Visual Paradigm for UML para el desarrollo dirigido por modelos de aplicaciones de gestión de información. Serie Científica de la Universidad de las Ciencias Informáticas. 2012. p.11.

31. Balsamiq. Balsamiq Mockups, diseño interfaz de usuario. [En línea] 13 de Agosto de 2011. [Citado el: 19 de Marzo de 2017.] https://hop2croft.wordpress.com/2011/08/13/balsamiq-mockups/.

32. What is Service Builder? [En línea] Liferay Developer Network, 2017. [Citado el: 15 de Marzo de 2017.] https://dev.liferay.com/develop/tutorials/-/knowledge_base/7-0/what-is-service-builder.

33. Esperanza Marcos,Escuela Técnica Superior de Ingeniería Informática, Universidad Rey Juan Carlos. Modelado Conceptual. 2012.

34. Elena Sánchez Pescador, Universidad Carlos III de Madrid. Análisis e implantación de una herramienta de gestión de requisitos para la gestión de servicios basado en la filosofía de ITIL V3, CMMI de servicios y MOF. [En línea] 2009. [Citado el: 20 de Abril de 2017.] http://e-archivo.uc3m.es/bitstream/handle/10016/6686/PFC_Elena_Sanchez_Pescador.pdf;jsessionid=703C3E30597AF9EFB1D938597EBB22FD?sequence=5.

35. Historias de usuario¿Por qué? ¿Qué son? ¿Cómo son? [En línea] Miquel Mora, 10 de Diciembre de 2010. [Citado el: 14 de Marzo de 2017.] https://es.slideshare.net/MiquelMora/historias-de-usuario.

36. Rogelio Noé Limón Cordero, Valencia, España, Universidad Politécnica de Valencia, Departamento de Sistemas Informáticos y Computación. Tesis Doctoral, Las vistas arquitectónicas de software y sus correspondencias mediante la gestión de modelos. 2009.

37. Construcción de Aplicaciones por Capas. [En línea] Junta de Andalucía, 2017. [Citado el: 15 de Marzo de 2017.] http://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/desarrollo/construccion-aplicaciones-por-capas.

38. Juan Pavón Mestras. Estructura de las Aplicaciones Orientadas a Objetos. Programación Orientada a Objetos. [En línea] 2009. [Citado el: 18 de Marzo de 2017.] https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf.

39. Tema 6 Patrones de diseño. Ingeniería del Software II. 2011.

40. LARMAN, C. UML y Patrones. Introducción al análisis y diseño orientado a objetos y al proceso unificado. [En línea] 2003. [Citado el: 13 de Marzo de 2017.] http://www.fmonje.com/UTN/ADES%20-%20208/UML%20y%20Patrones%20%202da%20Edicion.pdf.

41. GRASP: Controlador. [En línea] 14 de Marzo de 2012. [Citado el: 14 de Marzo de 2017.] http://juan-garcia-carmona.blogspot.com/2012/09/grasp-controlador.html.

42. UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. [En línea] [Citado el: 14 de Marzo de 2017.] http://s3.amazonaws.com/academia.edu.documents/32421917/PREVIEW-LIBRO-9788483229279.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1489548679&Signature=iOM6bMDFGDAITnLmICrGtFdoV4A%3D&response-content-disposition=inline%3B%20filename%3DUML_y_patrones.pdf.

43. Diseño Dirigido por Responsabilidades con los patrones GRASP. [En línea] [Citado el: 14 de Marzo de 2017.]

44. Rey, E.M., Diseño Dirigido por Responsabilidadesconlospatrones GRASP. 2015 [cited 2015], Pearson Educación, S.A. Todos los derechos reservados. [En línea] 2017. [Citado el: 15 de Marzo de 2017.]

45. ARLOS A. GUERRERO, Johanna M. Suaréz y LUZ E. GUTIÉRREZ. Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. [En línea] 2013. [Citado el: 15 de Marzo de 2017.] http://www.scielo.cl/pdf/infotec/v24n3/art12.pdf.

46. Fernández, Julio César. Apple Coding. applecoding academy. [En línea] 11 de Enero de 2016. [Citado el: 15 de Marzo de 2017.] https://applecoding.com/guias/patrones-diseno-software-facade.

47. Patrones de diseño. [En línea] 2008. [Citado el: 15 de Marzo de 2017.]

48. 2017, LinkedIn Corporation ©. Estándares y Buenas Practicas . s.l. : https://es.slideshare.net/PiXeL16/estandares-de-codigo-emanuel, 2013.

49. Creative Commons BY NC SA. Glosario Terminología Informática. lowercamelcase. [En línea] GTI, 07 de Agosto de 2008. [Citado el: 18 de Mayo de 2017.] http://www.tugurium.com/gti/termino.php?Tr=lowercamelcase.

50. Panel Testing - Centro de Excelencia. Panel Sistemas. Software QA – ¿Cuáles son los tipos de pruebas software? [En línea] 11 de Febrero de 2015. [Citado el: 15 de Mayo de 2017.] http://www.panel.es/blog/software-qa-cuales-son-los-tipos-de-pruebas-software/.

51. Ruíz, Asesor en Higiene y Seguridad Industrial Mario Ramón Mancera. Listas de Chequeo. Listas de Chequeo. [En línea] 19 de Abril de 2009. [Citado el: 19 de Mayo de 2017.] https://es.slideshare.net/manceramr/listas-de-chequeo.

52. Modelo vista controlador (MVC). [En línea] Universidad de Alicante, 2017. [Citado el: 16 de Marzo de 2017.] https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html.

Anexo 1

Tabla 8 Artefactos de la metodología AUP generados en el documento.

Artefactos generados de la metodología AUP
Modelado conceptual Modelado de requisitos del sistema • Requisitos funcionales • Requisitos no funcionales Historia de usuario Arquitectura de la aplicación Modelo de diseño • Diagrama de clases Pruebas del sistema

Anexo 2

Tabla 9 Historia de usuario generar reporte de incidencia.

Historia de Usuario	
Número: HU2	**Nombre del requisito:** Generar reporte de incidencia.
Programador: Addiel Tamayo Quintero	**Iteración asignada:** 1
Prioridad: Media	**Tiempo estimado:** 5 día
Riesgo en desarrollo: o Problemas eléctricos. o Problemas técnicos de los servicios de redes.	**Tiempo real:** 40 horas
Descripción: El usuario podrá generar varios tipos de reportes según sean las necesidades, dígase reportes por incidencia, fecha, estado, categoría, prioridad, departamento, reportes asignados a un trabajador, creados por un trabajador o al seleccionar una incidencia especifica podrá generar un reporte de la misma, para ello debe accionar en el logo del documento PDF.	
Observaciones:	
Prototipo de interfaz:	

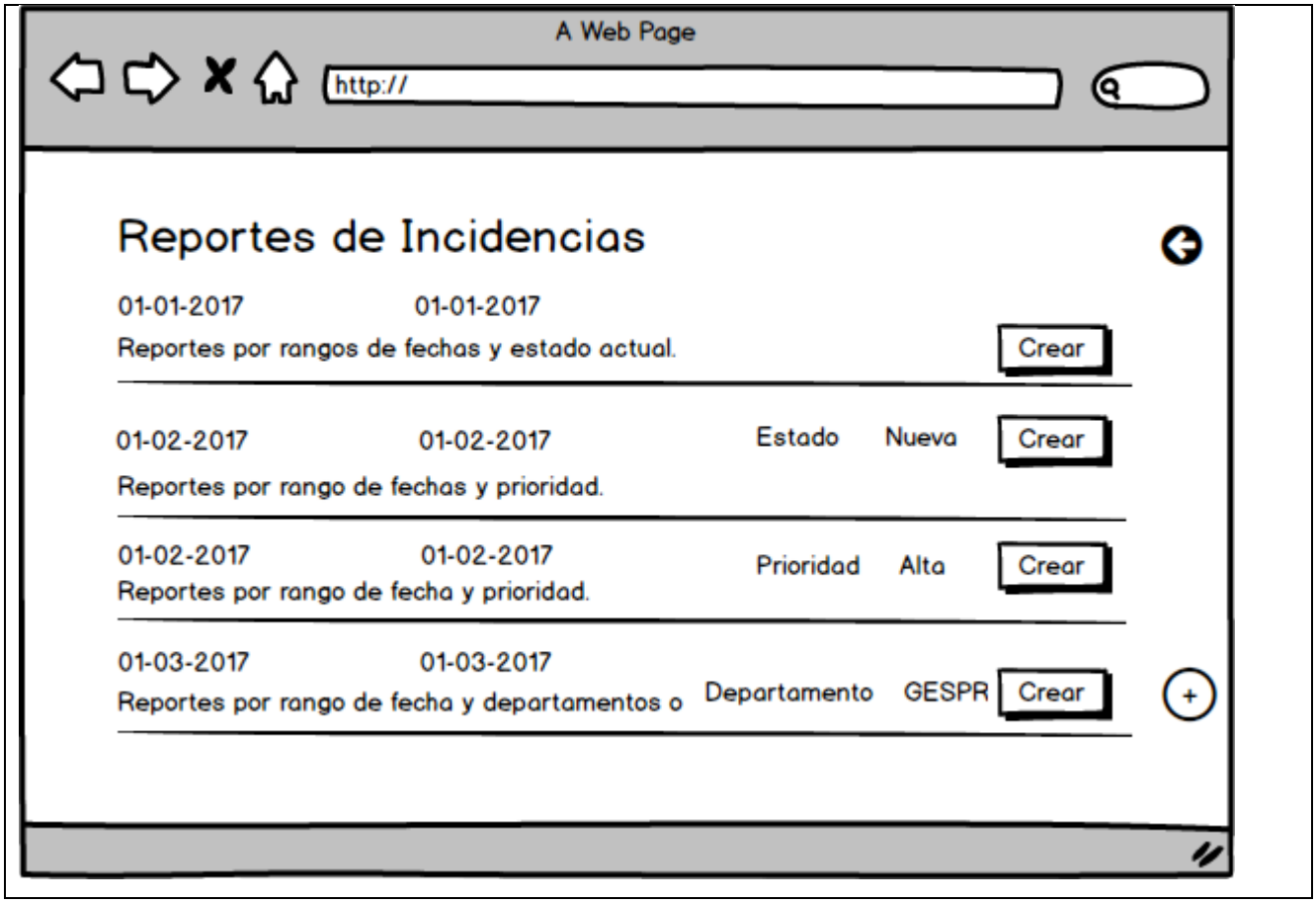


Tabla 10 Historia de usuario gestionar departamento.

Historia de Usuario	
Número: HU3	**Nombre del requisito:** Gestionar departamento.
Programador: Addiel Tamayo Quintero	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 10 día
Riesgo en desarrollo: o Problemas eléctricos. o Problemas técnicos de los servicios de redes.	**Tiempo real:** 80 horas
Descripción: Permitirá al administrador gestionar los departamentos de la empresa, dígase: registrar un nuevo departamento con el nombre, la descripción, el jefe del departamento y la fecha en que fue creado, para ello debe accionar el botón que se encuentra en la parte	

inferior derecha con el signo “+”, así como editarlo al accionar en el icono en forma de lápiz y eliminarlo del sistema al accionar en el icono en forma de latón.
Observaciones:
Prototipo de interfaz: 

Tabla 11 Historia de usuario gestionar prioridad.

Historia de Usuario	
Número: HU4	**Nombre del requisito:** Gestionar prioridad.
Programador: Addiel Tamayo Quintero	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 10 día
Riesgo en desarrollo: ○ Problemas eléctricos. ○ Problemas técnicos de los servicios de redes.	**Tiempo real:** 80 horas
Descripción: Permitirá al administrador gestionar las prioridades de la empresa, dígase: registrar una nueva prioridad con el nombre, la descripción y la fecha en que fue creada, para ello debe accionar el botón que se encuentra en la parte inferior derecha con el signo	

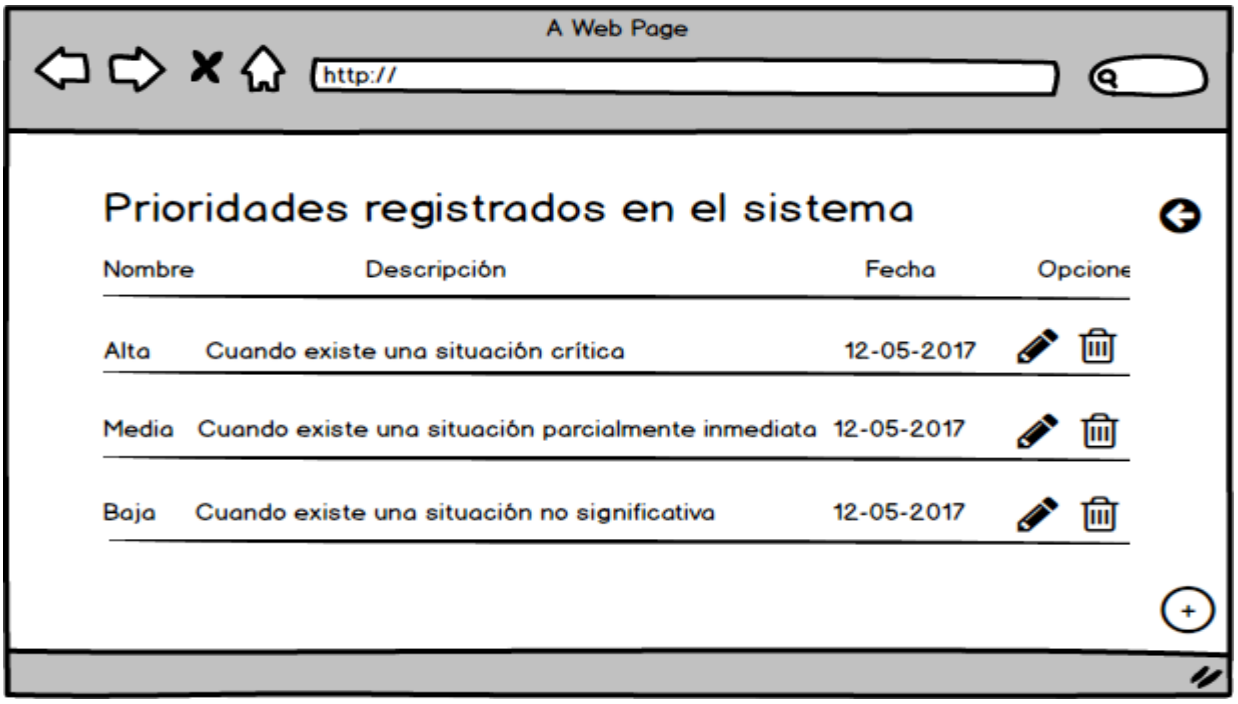
"+", así como editarla al accionar en el icono en forma de lápiz y eliminarla del sistema al accionar en el icono en forma de latón.
Observaciones:
Prototipo de interfaz: 

Tabla 12 Historia de usuario gestionar categoría.

Historia de Usuario	
Número: HU5	**Nombre del requisito:** Gestionar categoría.
Programador: Addiel Tamayo Quintero	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 10 día
Riesgo en desarrollo: o Problemas eléctricos. o Problemas técnicos de los servicios de redes.	**Tiempo real:** 80 horas
Descripción: Permitirá al administrador gestionar las categorías de la empresa, dígase: registrar una nueva categoría con el nombre, la descripción y la fecha en que fue creada, para ello debe accionar el botón que se encuentra en la parte inferior derecha con el signo	

"+", así como editarla al accionar en el icono en forma de lápiz y eliminarla del sistema al accionar en el icono en forma de latón.

Observaciones:

Prototipo de interfaz:

Anexo 3

Tabla 13 Caso de prueba gestionar departamento.

Id del escenario	**Escenario**	**Variable 1** Nombre	**Variable 2** Descripción	**Variable 3** Jefe de departamento	**Variable 4** Fecha	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1	Registrar departamento.	V	V	V	N/A	Mensaje de confirmación.	Se agrega un departamento al sistema.
		I	V	V	N/A	Notificación de error.	
		V	I	V	N/A	Notificación de error.	
		V	V	I	N/A	Notificación de error.	
EC 2	Actualizar departamento.	V	V	V	N/A	Mensaje de confirmación.	Se actualiza satisfactoriame

							nte el departamento.
		I	V	V	N/A	Notificación de error.	
		V	I	V	N/A	Notificación de error.	
		V	V	I	N/A	Notificación de error.	
EC 3	Eliminar departamento.	N/A	N/A	N/A	N/A	Mensaje de confirmación.	El sistema elimina el departamento seleccionado.
EC 4	Mostrar departamento.	N/A	N/A	N/A	N/A	Se muestran los detalles del departamento seleccionado.	El sistema muestra los departamentos con todos sus datos.

Tabla 14 Caso de prueba gestionar categoría.

Id del escenario	**Escenario**	**Variable 1** Nombre	**Variable 2** Descripción	**Variable 3** Fecha	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1	Registrar categoría.	V	V	N/A	Mensaje de confirmación.	Se agrega una categoría al sistema.
		I	V	N/A	Notificación de error.	
		V	I	N/A	Notificación de error.	
EC 2	Actualizar categoría.	V	V	N/A	Mensaje de confirmación.	Se actualiza satisfactoriamente la categoría.
		I	V	N/A	Notificación de error	

		V	I	N/A	Notificación de error.	
EC 3	Eliminar categoría.	N/A	N/A	N/A	Mensaje de confirmación.	El sistema elimina la categoría seleccionada.
EC 4	Mostrar categoría.	N/A	N/A	N/A	Se muestran los detalles de la categoría seleccionada.	El sistema muestra las categorías con todos sus datos.

Tabla 15 Caso de prueba gestionar prioridad.

Id del escenario	**Escenario**	**Variable 1** Nombre	**Variable 2** Descripción	**Variable 3** Peso	**Variable 4** Fecha	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1	Registrar prioridad.	V	V	V	N/A	Mensaje de confirmación.	Se agrega una prioridad al sistema.
		I	V	V	N/A	Notificación de error.	
		V	I	V	N/A	Notificación de error.	
		V	V	I	N/A	Notificación de error.	
EC 2	Actualizar prioridad.	V	V	V	N/A	Mensaje de confirmación.	Se actualiza satisfactoriamente la prioridad.

		I	V	V	N/A	Notificación de error.	
		V	I	V	N/A	Notificación de error.	
EC 3	Eliminar prioridad.	N/A	N/A	N/A	N/A	Mensaje de confirmación.	El sistema elimina la prioridad seleccionada.
EC 4	Mostrar prioridad.	N/A	N/A	N/A	N/A	Se muestran los detalles de la prioridad seleccionada.	El sistema muestra las prioridades con todos sus datos.