Universidad de las Ciencias Informáticas

Facultad 3

Título: Análisis y Diseño de la Plataforma de Gestión de Servicios V.1.0 del Centro de Soporte de la Universidad de Ciencias Informáticas.

Autor(es): Yobalis Hernández Santos

Tutor(es): Ing. Frank David Avalos Palomo

Ing. Yulio Seriocha García Gallardo

La Habana, Junio de 2011

"Año 53 de la Revolución"

Declaración de Autoría

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

Para que así conste se firma la presente a los ____ días del mes de _____ del año 2011.

_____

Yobalis Hernández Santos

_____ _____

Ing. Frank David Avalos Palomo Ing. Yulio Seriocha García Gallardo

Resumen.

Resumen

Las entidades empresariales y presupuestadas se han insertado en el mundo de la informatización debido a la necesidad de gestionar un gran volumen de recursos que necesitan ser controlados. La gestión de incidencias tecnológicas a nivel mundial se lleva a cabo mediante la utilización de sistemas informáticos que durante años ha venido desarrollando la industria de software.

Con el propósito de crear un sistema para la gestión de incidencias que satisfaga las exigencias de la universidad (sólo gestionar incidencias tecnológicas), se estudia el proceso de la gestión de incidencias tecnológicas, se realiza un análisis de los sistemas existentes y se elaboran los artefactos para cada uno de los flujos de trabajo correspondientes a la metodología presentada.

El presente trabajo de diploma tiene como finalidad brindar el análisis y diseño de la solución de un sistema capaz de gestionar las incidencias tecnológicas, desarrollado sobre tecnologías libres, que brinde en tiempo real la información necesaria a los usuarios sobre las incidencias.

Para el desarrollo del presente trabajo se exponen herramientas y tecnologías a utilizar, tales como: UML como lenguaje de modelado y Visual Paradigm como herramienta de modelado, las que fueron utilizadas en el modelado del negocio y el diseño del sistema en cuestión.

Todo lo modelado en este trabajo constituye un punto de partida para la futura implementación de los requerimientos definidos.

PALABRAS CLAVES

Análisis, Diseño, Sistema, Gestión, Incidencia.

Índice.

Índice

Índice.

Índice.

Índice.

Índice de tablas.

Índice de Tablas

Índice de figuras.

Índice de Figuras

Índice de figuras.

Introducción

En el mundo de hoy la mayoría de las operaciones y actividades giran entorno a una computadora. La informática soporta importantes y continuos cambios que repercuten en la sociedad y en el desarrollo del mundo empresarial.

Las Tecnologías de la Información y las Comunicaciones (TIC), producto de sus avances y como integrantes de esta ciencia, han posibilitado el incremento rápido del desarrollo de la economía y la sociedad. Estos avances vienen dados gracias al interés de conocimiento del ser humano y a su naturaleza creadora, los mismos son aplicados en gran medida en el desarrollo de la economía y la sociedad mediante la producción de software. Esta actividad demanda cada vez más un mayor estudio y una mejor planificación porque a medida que el ser humano y el mundo evolucionan se solicitan programas más complejos y con mayor calidad.

Cuba, en aras de estar al tanto de los adelantos tecnológicos que marcan un determinado nivel en la industria de software y para lograr cierto desarrollo informático en la sociedad, ha destinado grandes cantidades de recursos de acuerdo a las posibilidades económicas del país, en ese sentido se ha ampliado el conocimiento y se ha propiciado la creación de soluciones informáticas a la altura de las nuevas tecnologías.

Las entidades empresariales, presupuestadas y de servicios no están ajenas a este proceso, debido a la existencia de recursos que deben ser controlados de forma eficaz, tales como: los recursos financieros, los energéticos, los recursos humanos y los recursos de transporte, los cuales son algunos de los ejemplos a citar. En lo que refiere a los recursos tecnológicos, se producen incidencias: Eventos que no forman parte de la operación estándar de un servicio y puede o no causar una interrupción, o una reducción de la calidad del mismo. Estas incidencias son controladas mediante la gestión de incidencias la cual contribuye de una forma u otra a que el funcionamiento, desarrollo y la eficiencia de los recursos tecnológicos sea mayor.

Fundamentación teórica.

De forma general la gestión de incidencias tiene como objetivo fundamental restablecer el funcionamiento normal del servicio lo más rápido posible y con el menor impacto sobre la actividad del negocio. (1)

En Cuba, hoy en día se lleva a cabo la gestión de incidencias de forma manual debido a que no se cuenta con un sistema capaz de realizar estas tareas con el máximo de interoperabilidad y control de los recursos, por lo que se ha incentivado la búsqueda minuciosa de soluciones informáticas extranjeras relacionadas con esta área de conocimientos, demostrando que hasta el momento no existe una herramienta capaz de ejecutar estas operaciones que pueda ser utilizado en las entidades del país, teniendo en cuenta que muchos de estos han sido desarrollados sobre tecnologías propietarias que conlleva a gastos significativos por concepto de compra de licencias de software y mantenimiento y además no se ajustan a lo que en realidad se quiere, debido a que estos sistemas gestionan incidencias relacionadas con todos los recursos que son controlados por las entidades y lo que se necesita es un sistema que sólo gestione incidencias tecnológicas.

La Universidad de las Ciencias Informáticas (UCI), cuenta con 7 facultades, en cuyos centros de desarrollo se gestionan proyectos y servicios informáticos, la universidad tiene además un Centro de Soporte encargado de brindar mantenimiento y resolver las incidencias tecnológicas tanto de hardware como de software que se produzcan en la infraestructura productiva. Al igual que la mayoría de las entidades presupuestadas, la UCI lleva la gestión de incidencias de forma manual, existiendo para ello pequeños centros de gestión de llamadas en donde se notifican dichas incidencias, muchas de las cuales no quedan registradas o las que se registran están archivadas en papel, información que se puede extraviar o sencillamente se puede deteriorar y además puede ser duplicada debido a que no existe una base de datos que registre toda la información de incidencias pasadas.

Teniendo en cuenta la situación descrita con anterioridad, el presente trabajo de diploma plantea el siguiente **problema científico:**

Fundamentación teórica.

¿Cómo lograr que la gestión de incidencias en la Universidad de las Ciencias Informáticas sea óptima y menos engorrosa?

Objeto de estudio:
Proceso para la gestión de incidencias.

Campo de acción se enmarca en:
La gestión de incidencias en la Universidad de las Ciencias Informáticas.

Para la solución de este problema se plantea como:

Objetivo general: Desarrollar el análisis y diseño de una aplicación web que permita una correcta gestión de los servicios que brinda el Centro de Soporte de la Dirección General de Producción. De aquí se derivan los siguientes **objetivos específicos**:

1. Realizar la fundamentación teórica de la investigación.
2. Realizar el modelado del negocio y describir los requisitos funcionales del sistema.
3. Diseñar la propuesta de un sistema que gestione las incidencias.
4. Validar los requisitos y el diseño de clases.

Para desarrollar el presente trabajo se llevaron a cabo las siguientes tareas:

1. Realizar estudio del estado del arte.
2. Elaborar el diagrama de casos de uso del negocio.
3. Elaborar el diagrama de procesos del negocio.
4. Realizar levantamiento y especificación de requisitos.
5. Realizar los prototipos de interfaz de usuario.

6. Realizar los diagramas de clases del análisis.
7. Realizar el diagrama de clases del diseño.
8. Realizar el diagrama de modelo de datos.
9. Realizar el diagrama de paquetes.
10. Realizar la validación de los requisitos y del diseño.
11.

El presente trabajo consta de tres capítulos:

Capítulo 1:

Fundamentación teórica de la investigación: En este capítulo se exponen conceptos fundamentales relacionados con la gestión de incidencias, se realiza un análisis de los diferentes sistemas de gestión de incidencias existentes en el mundo y se finaliza con una caracterización de la metodología de desarrollo de software, las herramientas y tecnologías a utilizar para la solución.

Capítulo 2:

Modelado del negocio y captura de requisitos: En este capítulo se realiza el diagrama de casos de uso del negocio correspondiente a la Modelación del Negocio, se determinan los requisitos funcionales y no funcionales con los que debe cumplir el sistema, se lleva a cabo la especificación de los requerimientos de software, se presentan los prototipos de interfaz de usuario y por último se realiza la validación de los requisitos.

Capítulo 3:

Análisis y diseño de la solución: En el capítulo se realiza el análisis y el diseño de la solución, se muestran los diagramas de clases del análisis y del diseño, se representa el diagrama de paquetes y el modelo de datos, se especifica la utilización de patrones y se valida el diseño propuesto través de métricas.

Capítulo 1 - Fundamentación teórica

1.1 Introducción.

Para lograr una mayor comprensión del funcionamiento de la Gestión de Incidencias se exponen en el presente capítulo conceptos fundamentales del tema, se analiza el estado del arte de los sistemas que informatizan el proceso a nivel mundial y se caracteriza la realización de los procesos asociados a la gestión de incidencias en la universidad.

Se describen además las tecnologías, las herramientas y la metodología que se utilizarán a la hora del desarrollo del producto, así como los patrones a utilizar en la captura de requisitos y en el diseño de la propuesta de solución.

1.2. Gestión de Incidencias.

Los vertiginosos cambios y el desarrollo producido en el ámbito de las TIC, de una u otra forma han ocasionado fuertes reestructuraciones en los espacios de trabajo y tanto las empresas internacionales como las entidades empresariales y presupuestadas han tratado de nutrirse con su uso, en ese sentido la gestión de incidencias ha pasado a ser una tarea difícil teniendo en cuenta que en muchas ocasiones se realiza de forma manual.

Una **incidencia** puede definirse como una interrupción no planificada o una reducción de la calidad de un servicio, es cualquier evento que no forma parte de la operación estándar de un servicio y que puede causar una interrupción o una reducción de la calidad del mismo.
La **gestión de incidencias** cubre todo tipo de incidencias (las incidencias que se tratarán con el desarrollo del software serán solo tecnológicas), ya sean fallos, faltas o dificultades planteadas por los usuarios, o el personal técnico. Tiene como objetivo resolver de la manera más rápida y eficaz posible cualquier incidente que cause una interrupción en el servicio.

En la gestión de incidencias hay que tener en cuenta los siguientes elementos:

Límites de tiempo: Se deben definir límites de tiempo para todas las fases.

Modelos de incidencias: Los modelos de incidencias son una manera de determinar los pasos necesarios para ejecutar correctamente los procesos, lo que significa que las incidencias estándares se gestionarán de forma correcta y en el tiempo establecido.

Impacto: Efecto de una incidencia sobre los procesos de negocio.

Urgencia: Medida del tiempo disponible hasta que la incidencia tenga un impacto significativo en los procesos de negocio.

Prioridad: Categorización de la importancia relativa de la incidencia, en función del impacto y la urgencia.

Incidencias graves: Incidencias que tienen un grado de impacto extremo para la comunidad de usuarios.

La gestión de incidencias tiene como principales objetivos:

- Restablecimiento del servicio acordado lo antes posible y con el mínimo impacto al negocio.
- Minimizar el impacto negativo de un incidente en el negocio, garantizando el más alto nivel de calidad y disponibilidad del servicio.
- Registro permanente del incidente.
- Identificar mejoras del servicio proactivamente.
- Revisar la exactitud de los detalles de la base de datos de conocimiento.
- Minimizar el riesgo de incidentes perdidos.
- Recolección de información sobre la gestión.

Además los principales beneficios de una correcta gestión de incidencias incluyen:

- Mayor productividad por parte de los usuarios.
- Mayor control de los procesos y monitorización del servicio.
- Optimización de los recursos disponibles.
- Aumento en la satisfacción general de clientes y usuarios.

Por otra parte una incorrecta gestión de incidencia puede acarrear efectos adversos tales como:

- Reducción de los niveles de servicio.
- Se malgastan valiosos recursos, demasiada gente o gente del nivel inadecuado trabajando concurrentemente en la resolución del incidente.
- Se pierde valiosa información sobre las causas y efectos de los incidentes para futuras reestructuraciones y evoluciones.
- Se crean clientes y usuarios insatisfechos por la mala y/o lenta gestión de sus incidentes.

A la hora de implementar la gestión de incidencia las principales dificultades se resumen en:

- No se siguen los procedimientos previstos y se resuelven las incidencias sin ser registradas o se escalan innecesariamente y/o se omiten los protocolos preestablecidos.
- No existe un margen operativo que permita gestionar los “picos” de incidencias por lo que éstas no son registradas adecuadamente e impiden la correcta operación de los protocolos de clasificación y escalado.
- No están bien definidos los niveles de calidad de servicio ni los productos soportados. Lo que puede provocar que se procesen peticiones que no se incluían en los servicios previamente acordados con el cliente. (2) (16)

Todos los departamentos de Tecnologías Informáticas (TI) atienden fallos en hardware o software, por lo que es de vital importancia la realización de una eficiente y lucrativa gestión de incidencias.

1.3 Sistemas informáticos relacionados con la gestión de incidencias.

Las TIC se han desarrollado con el fin de mejorar la calidad de vida de las personas en los diferentes entornos y están presentes en la educación, la salud, la cultura, la economía y en el resto de las esferas que constituyen la sociedad.

Como parte de las tecnologías de la información y las comunicaciones, la informatización de los procesos ha sido un motor impulsor para potenciar el cumplimiento de las funciones a todos los niveles, con un máximo de racionalidad y control en las diferentes áreas de la misma. En las empresas es esencialmente importante llevar a cabo el control de las incidencias tecnológicas, pues se guía a las entidades hacia el aumento de la productividad con un elevado nivel de eficiencia.

1.3.1 Sistemas gestores de incidencias.

Service Desk:

La función del Service Desk es registrar todas las llamadas de los usuarios reportando incidencias, categorizarlas, priorizarlas, canalizar su resolución, documentar el progreso y mantener continuamente informado al usuario del estado de la misma, es decir, iniciar para cada llamada de servicio el proceso adecuado para su tratamiento y en el caso concreto las llamadas de tipo “Incidencia” iniciar el proceso de Gestión de Incidencias. (3)

Xperta:

Xperta es una aplicación Web, los requisitos para lograr su funcionamiento son: un navegador Web y una conexión a Internet. Con la utilización de Xperta los usuarios tienen la facilidad de consultar el estado y los trabajos realizados por el servicio técnico en sus incidencias, así como un histórico de las mismas. A

través de la interfaz los usuarios pueden informar de sus incidencias, sólo con introducir un título descriptivo de la incidencia y una breve descripción. De este modo los técnicos tendrán constancia desde este mismo momento de la incidencia (dentro del panel de administración de Xperta) y a través de un correo electrónico (si así está configurado en la aplicación).

Xperta contiene una base de datos de todos los usuarios y personal del servicio técnico, de todas las incidencias actuales y solucionadas. Esto permite que los técnicos tengan acceso en todo momento a los datos del usuario o cliente: dirección, teléfono, e-Mail… (4)

CADdy SAT:

El gestor de incidencias CADdy SAT permite contar con un histórico de las incidencias que ha sufrido cada máquina a lo largo del tiempo, así como del modo en que fueron resueltas y del tiempo que se tardó en encontrar la solución. Los usuarios tienen mucho más control sobre el estado de la reparación de sus máquinas y disponen de un canal privilegiado para contactar con los técnicos y exponer los problemas. Da la posibilidad a los usuarios de ser notificados vía email, haciéndoles saber de que hay una nueva respuesta para la incidencia, esto ocurre cada vez que el servicio técnico la actualiza. (5)

KMKey Help Desk:

Es un software de gestión de incidencias conveniente para servicios de mantenimiento, ayuda al usuario y resolución de problemas en cualquier sector. Permite definir flujos de trabajo para abordar problemáticas derivadas de anomalías en servicios y maquinaria. La incidencia puede recibirse de forma automática (e-mail, entrada a través de una web, desde un dispositivo móvil) o bien ser abierta por el servicio de atención. Una vez en marcha seguirá el flujo diseñado por el cliente para su resolución. Permite: resolución inmediata, escalado, consulta de información anterior, reparto de recursos. Si la incidencia da origen a una intervención de mayor orden se puede enlazar con la Gestión de Proyectos o la Gestión de Calidad.

Fundamentación teórica.

Mediante **KMKey Help Desk** se podrá disponer, desde cualquier acceso a Internet, de toda la información relevante.

Este sistema maneja otras áreas de la gestión, como son:

Gestión de trabajos:

- SAT: Servicio de Asistencia Técnica.
- Mantenimiento preventivo: Planificación, gestión y control de acciones.
- Mantenimiento correctivo: Recepción y resolución de incidencias.
- Soporte (Hot line): Atención a usuarios, resolución remota de dudas.

Gestión de recursos:

- Humanos: Perfiles de trabajo. Permisos. Horas/hombre valoradas. Accesos restringidos.
- Materiales: Asignación de herramientas, espacios. Control de disponibilidad.
- Económicos: Previsiones, ofertas, facturas. Análisis económico de la incidencia. Real contra previsto. Gastos personales.

Gestión de la información:

- Gestión del Conocimiento: Toda la información introducida queda disponible y se puede localizar con una búsqueda indexada tipo Google.
- Documentos y archivos: Generación automática y salida de informes en varios formatos: Open Office, MS Office, PDF. Gestión documental asociada: versiones, autores, reservas.
- Agenda: Base de datos de empresas y contactos. Calendario de actividades.
- Integración e-mail y SMS: Notificaciones a terceros vía mail de acciones y tareas. Recepción automática de mails. Envío SMS. (6)

1.3.2 Valoración de los sistemas.

Teniendo en cuenta que la universidad no cuenta con un sistema que realice el proceso de gestión de incidencias y después de haber realizado un análisis de los Sistemas Gestores de Incidencias de carácter extranjeros, se concluye que no resultan soluciones factibles para las entidades cubanas debido a que fueron desarrollados sobre plataformas de software propietario, lo que implica incrementos de gastos en licencias de uso y mantenimiento del software y además no se ajustan a las necesidades internas de la universidad debido a que estos sistemas no sólo van a gestionar las incidencias tecnológicas, sino, que además gestionan las incidencias producidas en los recursos que son controlados por las entidades empresariales, presupuestadas y de servicio.

1.4 Tendencias al desarrollo del software libre.

En Cuba se ha desencadenado un auge nacional del software libre, debido al principio de Independencia Tecnológica (El acceso al código fuente permite el desarrollo de nuevos productos sin la necesidad de desarrollar todo el proceso partiendo de cero). El país se encuentra enfrascado en realizar un crecimiento en la industria del software, que da sus primeros pasos y cuya única alternativa sostenible para su avance es la migración al software libre dándole un gran impulso a esta corriente de desarrollo. (7)

La UCI, que acertadamente ha dedicado un por ciento de su matrícula al desarrollo de las tecnologías libres, hoy es pilar en este campo con un trabajo ascendente. Ya cuenta con una distribución de GNU/Linux orientada a Escritorio llamada Nova que responde a las necesidades de las instituciones cubanas. (8)

Hoy el desarrollo de software con la utilización de las tecnologías libres hará que Cuba obtenga como valores fundamentales los logros de un estado de soberanía e independencia tecnológica. En consecuencia las tecnologías y herramientas que serán presentadas posteriormente para el desarrollo de esta aplicación estarán enfocadas a este principio.

1.5 Metodología de desarrollo, lenguaje y herramienta de modelado del sistema.

En la industria de software las metodologías de desarrollo desempeñan un papel fundamental a partir de que las exigencias de los clientes son cada vez mayores y por lo tanto es necesario organizar el trabajo basándose en métodos comunes para el desarrollo. Existen las metodologías agiles y robustas, cada una con sus particularidades, utilizadas teniendo en cuenta las características de lo que se quiere lograr.

De acuerdo con lo planteado anteriormente el grupo de arquitectura del proyecto determinó emplear la metodología que aparece a continuación:

1.5.1 Metodología.

- **Rational Unified Process (RUP)**

Es una metodología que ayuda a desarrollar y desplegar software fragmentando su desarrollo en fases y ciclos, de forma iterativa e incremental, guiado por casos de uso y basado en la arquitectura. Es una colección de buenas prácticas experimentadas en la ingeniería de software que provee a sus usuarios de una fundamentación arquitectónica fuerte que permite construir el software con las necesidades requeridas.

Puede especializarse para gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto. Utiliza el Lenguaje Unificado de Modelado (UML) para preparar todos los esquemas de un sistema de software. De hecho, UML es una parte esencial del Proceso Unificado, sus desarrollos fueron paralelos.

RUP presenta algunas características importantes:

1. Dirigido por casos de uso.

2. Iterativo e incremental.
3. Centrado en la arquitectura.
4. Adaptable a proyectos de largo plazo.
5. Genera muchos artefactos, convirtiéndose en una metodología muy usada para lograr una certificación de desarrollo de software.
6. Destaca la importancia de lograr una buena captura de requisitos.
7. Maneja como actividades paralelas la gestión de la calidad y la gestión de riesgos.
8. Trabaja con la herramienta Rational Rose y se basa en UML como lenguaje orientado a objetos para visualizar, especificar, construir y documentar los artefactos.

Los flujos de trabajo que propone RUP se presentan a continuación. En este trabajo se desarrollan los tres primeros:

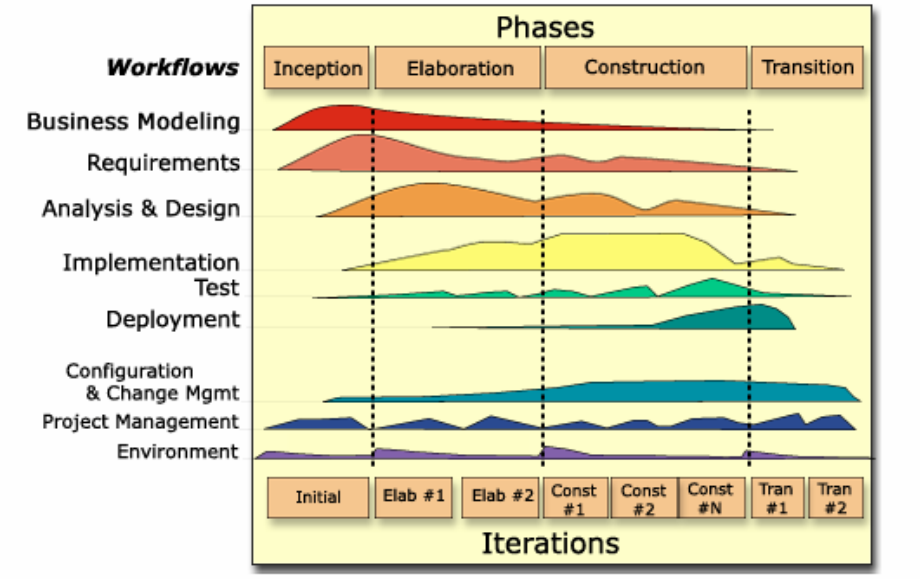


Figura 1 Flujos de trabajo que propone RUP.

1.5.2 Lenguajes de Modelado.

- **UML (Unified Modeling Language).**

Es un lenguaje gráfico para visualizar, especificar, construir, documentar y comunicar los artefactos de un sistema de software. Especifica varios diagramas que permiten crear artefactos durante el proceso de desarrollo del software.

UML es un lenguaje flexible de modelado que permite definir modelos de análisis y modelos del diseño. Beneficios de UML:

1. Soporta tecnología orientada a objetos.
2. Soporta RUP.
3. Los errores son tratables en todas las etapas del desarrollo del software.
4. Especifica, mediante los diagramas, cómo se estructura y se comporta el sistema.
5. Permite obtener un “plano del sistema”.
6. Reduce los costos y el tiempo de desarrollo.
7. Representa una colección de las mejores prácticas de ingeniería que tienen una probación exitosa en la modelación de sistemas largos y complejos. (19)

1.5.3 Herramienta CASE.

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son el mejor soporte para el proceso de desarrollo de software, es un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software.

- **Visual Paradigm.**

Para UML es una herramienta CASE multiplataforma de modelado visual UML, muy potente y fácil de utilizar. Soporta el ciclo de vida completo del desarrollo de software, ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

1. Soporte de UML versión 2.1.
2. Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
3. Ingeniería inversa - Código a modelo, código a diagrama.
4. Generación de código - Modelo a código, diagrama a código.
5. Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
6. Poderosa herramienta de generación de PDF/HTML a partir de diagramas UML.
7. Sincronización entre el código fuente y el modelo en tiempo real.

Se estará haciendo uso del Visual Paradigm en su versión 6.4. (9) (18)

1.6 Patrones.

¿Qué es un patrón?

Un patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas.

Formato de un patrón.

- Nombre
- Solución
- Problema
- Explicación
- Ejemplo de utilización

Los patrones son los siguientes:

1.6.1 Patrones de Casos de uso.

- **Patrón CRUD (Creating, Reading, Updating, Deleting).**

Este patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual.

Completo: Este patrón consta de un caso de uso, llamado Información CRUD o gestionar información, modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples.

Parcial: Este patrón alternativo modela una de las vías de los casos de uso como un caso de uso separado. Es preferiblemente utilizado cuando una de las alternativas de los casos de uso es más significativa, larga o más compleja que las otras.

- **Nombres que revelan la intención (Intention Revealing Name).**

El nombre debe reflejar la intención del caso de uso y reflejar un único objetivo e intención que el actor está intentando lograr. Se debe asignar un nombre apropiado que facilite el manejo del caso de uso, permitiendo tener una vista general del trabajo en su conjunto.

- **Preciso y Legible (Precise and Readable).**

Cada caso de uso que se escriba debe exactamente describir una Meta única y completa sin ser tan verboso que la audiencia no lo pueda leer o de tan alto nivel que no comunique la suficiente información para entenderlo adecuadamente. Los niveles más altos de formalidad en las especificaciones dan a los desarrolladores un sentido falso de seguridad. Nada puede remplazar el diálogo con el cliente.

- **Adorno, Decoración (Adornments).**

Este patrón plantea que el usuario a la hora de leer un caso de uso debe entender cómo el sistema entrega los valores sin preocuparse de detalles de la interfaz de usuario. La idea es crear campos dentro de la plantilla del caso de uso que fuera del texto del escenario apoyen la información auxiliar que es útil asociar con el caso de uso.

- **Pasos Nivelados (Leveled Steps).**

Pasos excesivamente pequeños hacen el caso de uso largo, difícil de leer y bloquean la visión. Pasos excesivamente largos pueden enterrar comportamientos importantes. Lo contrario ocasionalmente pasa, que el escritor escribe en un nivel muy alto de abstracción y hace largo el salto en la narrativa, omitiendo acciones claves que los desarrolladores deben saber. Mezclar niveles de detalle en un escenario es entretener. Ocasionalmente se deben escribir pasos continuos a diferentes niveles de abstracción. Demasiado de esto distrae al lector de lo que se supone que está pasando y le hace difícil la interpretación correcta de las instrucciones. (14)

1.6.2 Patrones de Diseño.

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptadas para resolver un problema de diseño general en un contexto particular. Identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades.

- **Patrones de asignación de responsabilidades.**

Patrones Generales de Software para Asignación de Responsabilidades), que describen los principios fundamentales de la asignación de responsabilidades a objetos. El empleo de los patrones **GRASP** tiene como objetivo la descripción de los principios fundamentales del diseño de objetos para la asignación general de responsabilidades.

Alta cohesión: Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. Su alto grado de funcionalidad, combinada con una reducida cantidad de operaciones, también simplifica el mantenimiento y los mejoramientos. La ventaja que significa una gran funcionalidad también soporta un aumento de la capacidad de reutilización.

El patrón alta cohesión presenta semejanzas con el mundo real. Se sabe que si alguien asume demasiadas responsabilidades, sobre todo las que debería delegar, no sería eficiente.

Controlador: En la asignación de responsabilidades, es contradictorio definir ¿Quién debería encargarse de atender un evento del sistema?, sin embargo, el uso de este patrón elimina la incógnita anterior sugiriendo asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una de las clases.

Bajo acoplamiento: Patrón evaluativo que asigna responsabilidades de modo que se mantenga un engranaje pobre entre las clases y objetos, reduce el impacto de los cambios y aumenta la reutilización.

1.6.3 Patrones Arquitectónicos.

- **Patrón arquitectónico Modelo Vista Controlador.**

El patrón de arquitectura conocido como Modelo-Vista-Controlador (MVC), separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario; es decir separa en tres capas diferentes los datos de una aplicación, la interfaz de usuario, y la lógica de control:

Modelo: Esta capa administra el comportamiento y los datos del dominio de la aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: Esta capa maneja la visualización de la información, es decir que presenta el modelo en un formato adecuado para interactuar, que usualmente es la interfaz de usuario.

Controlador: Esta capa controla el flujo de datos entre la vista y el modelo; es decir que responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. (10) (15)

Esta separación permite construir y probar el modelo, independientemente de la representación visual.

1.7 Ambiente de Desarrollo.

Es importante hacer una determinación precisa del contexto donde se debe desarrollar el software. De la apropiada selección de las tecnologías a utilizar depende en gran medida el diseño de la solución, constituyendo esto un aspecto a tener en cuenta, debido a que influye considerablemente en el tiempo de desarrollo y la calidad final del producto.

En el Documento de Arquitectura definido por el grupo de arquitectura del Centro de Soporte están reflejadas tanto las tecnologías como las herramientas a utilizar en la elaboración del software en cuestión. En esta sección del capítulo se caracteriza el ambiente de desarrollo, el cual comprende las tecnologías, y las herramientas necesarias en estas primeras etapas así como las que se deben emplear en la fase posterior al análisis y diseño de la solución a modo de propuesta.

1.7.1 Lenguaje de programación.

1.7.1.1 Lenguaje de programación del lado del cliente.

- **JavaScript.**

Es un lenguaje de programación utilizado para crear en su mayoría pequeños programas encargados de realizar acciones dentro del ámbito de una página web. Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado.

Con javascript se pueden crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones javascript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador. Entre las acciones típicas que se pueden realizar en javascript están: por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo, por el otro, javascript permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que se pueden crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo. Javascript es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas.

1.7.1.2 Lenguaje de programación del lado del servidor.

- **PHP.**

Para el desarrollo del producto se escogió el lenguaje PHP como lenguaje script del lado del servidor utilizado para la generación de páginas web dinámicas, debido a que es el utilizado y estandarizado por la UCI. Específicamente la versión 5.2.3 que se encuentra dentro de las versiones 5.x (que revolucionan el lenguaje y la manera de enfocar la programación Orientada a Objetos) por su estabilidad y rendimiento.

Características.

1. Es un lenguaje multiplataforma.
2. Soporte para varios servidores Web.
3. Sintaxis clara y bien definida.
4. Bastante sencillo de aprender y utilizar.
5. Permite las Técnicas de Programación Orientada a Objetos.
6. Tiene manejo de excepciones (desde PHP5).
7. Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad: MySQL, PostgreSQL, Oracle, MSSQL Server.
8. Ofrece una solución simple y universal para las paginaciones dinámicas de la web de fácil programación.
9. Soportado por una gran comunidad de desarrolladores, como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente.
10. El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP. (17)

1.7.2 Tecnologías.

- **AJAX (Asynchronous JavaScript and XML).**

Esta tecnología, incluye un conjunto de tecnologías aplicadas de forma concreta, permite crear "aplicaciones web" más eficientes en la interacción con el usuario. AJAX aúna todas estas tecnologías:

- Estructura y presentación de la información basada en estándares mediante XHTML y CSS.
- Document Object Model (DOM) para interactuar dinámicamente con los datos.

- Intercambio y manipulación de datos usando XML and XSLT.
- Recuperación de datos asincrónica usando XMLHttpRequest y JavaScript.

Ajax puede utilizarse hoy en día como una opción más rápida y eficiente de navegación web, no como el único método de navegación y es bueno utilizarse en webs que intentan ser "aplicaciones".

1.7.3 Framework.

- **Symfony.**

Symfony es un framework para construir aplicaciones web con PHP. Incluye un grupo de frameworks probados que reducen considerablemente el tiempo del desarrollo, encargándose de las tareas engorrosas y dejando espacio principalmente para las cuestiones relacionadas con el negocio. Entre los frameworks que incluye están: Propel como mapeador Objeto-Relacional, Creole como capa de abstracción de la Base de Datos, Lime como mecanismo para realizar pruebas unitarias y funcionales, Prototype para el desarrollo de aplicaciones web ricas y trabajo con AJAX. Además puede ser fácilmente configurable con otros frameworks o plugins probados y tiene elementos que permiten garantizar la seguridad, el cacheo de información, la validación de datos, además obliga a desarrollar según patrón arquitectónico Modelo-Vista-Controlador. Se estará haciendo uso del Framework Symfony en su versión .

1.7.4 Sistema Gestor de Base de Datos.

- **PostgreSQL.**

Está ampliamente considerado como el sistema de bases de datos de código abierto. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre. Es usado para manejar grandes cantidades de información y está basado en el modelo relacional, aunque incorpora conceptos del modelado orientado a objeto. Se pueden definir consultas anidadas, vistas, crear funciones por el usuario, no sólo en el lenguaje natural SQL, sino en varios más, entre ellos C, lenguaje nativo

PostgreSQL, Perl, PHP y Java. Es multiplataforma, soporta múltiples transacciones, integridad de datos, presenta una estabilidad muy alta, gran seguridad de los datos, soporta la réplica y procedimientos almacenados. Propone un tamaño ilimitado para las base de datos, lo que da la medida de un gestor de base de datos robusto, y con grandes funcionalidades. Se estará haciendo uso del Gestor de Base de Datos PostgreSQL en su versión 8.4.

1.7.5 Servidor de Aplicaciones.

- **Apache.**

Servidor web, solución libre, altamente utilizado en el mundo y que tiene una estabilidad probada. Además, permite agregar disimiles módulos y cuenta con una comunidad de desarrollo extensa en donde se pueden encontrar variedad de soluciones. Se estará haciendo uso del Servidor Web Apache en su versión 2.2. (20)

1.7.6 Entorno Integrado de Desarrollo (IDE).

- **NetBeans.**

Es una plataforma universal para integrar herramientas de desarrollo, con una arquitectura abierta y basada en plugins. El entorno integrado de desarrollo (IDE) de NetBeans emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad al frente de la plataforma que se muestra del lado del cliente, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software, adicionalmente al permitirle a NetBeans extenderse usando otros lenguajes de programación como son C/C++, Phyton, Ruby. Se estará haciendo uso del NetBeans en su versión 6.7. (11)

1.8 Conclusión del capítulo.

Luego de un estudio detallado de sistemas relacionados con la gestión de incidencias en el ámbito internacional, se concluye que no existe una solución de este tipo capaz de realizar dichas tareas que pueda ser utilizado en la UCI.

Es necesaria la utilización de herramientas y tecnologías libres que hoy día están alcanzando un determinado nivel en la industria de software, que pueden contribuir al desarrollo de aplicaciones robustas enfocadas a un modelo más racional para los usuarios, con menos costes de licencia, donde se intensifique la prestación de servicios, que reduzca el tiempo de desarrollo e incremente la calidad y en este sentido se presentó la propuesta elaborada por la dirección del proyecto para el desarrollo de la solución.

Capítulo 2 – Modelado del negocio y captura de requisitos.

2.1 Introducción.

El presente capítulo muestra las características que tendrá el sistema, partiendo de la representación del negocio mediante el diagrama de casos de uso del negocio, el modelo de objeto y el diagrama de procesos del negocio, la definición de los requerimientos funcionales y no funcionales del sistema y la especificación de los requisitos funcionales, así como los prototipos de interfaz de usuario del sistema y la validación de los requisitos.

2.2 Reglas del Negocio.

Las políticas que deben cumplirse o condiciones que deben satisfacerse para que se ejecuten los casos de uso, de manera que regulen los procesos de negocio de acuerdo a las especificidades de cada uno, son:

- El Notificador debe confeccionar la incidencia con todos los datos necesarios, para que posteriormente sea enviada al Especialista o al Técnico del Centro de Soporte.
- El Notificador es la persona autorizada a solicitar la gestión de incidencias.
- El Especialista y el Técnico solo pueden tener acceso a la información que se maneja para la gestión de estas incidencias.
- El Especialista y el Técnico son los encargados de revisar las notificaciones que hacen los notificadores, le dan solución a dicho problema y posteriormente le notifican al usuario (notificador) la solución que se le dio a la incidencia.

Modelado del negocio y captura de requisitos.

2.3 Modelado del negocio.

El modelo de negocio representa y describe detalladamente los procesos del negocio, de forma que constituye el artefacto esencial para comprenderlos. “Está soportado por dos tipos de modelos UML: modelo de casos de uso y modelo de objetos”, por tanto, en este trabajo se desarrollan estos artefactos, además del diagrama de procesos del negocio.

2.3.1 Procesos del negocio.

Llevar a cabo un control de la gestión de incidencias tecnológicas es una tarea fundamental para cualquier entidad, contribuyendo a que el desarrollo, el funcionamiento y la eficiencia de los recursos tecnológicos sea mayor.

Se identificaron procesos básicos dentro de la gestión de incidencias, los cuales se muestran a continuación:

Registro de incidencias: Un usuario de servicios le informa al centro de soporte una incidencia, el técnico que recibe la incidencia la analiza, en caso de poder resolverla la registra en su lista de incidencias, en caso de no poder resolverla se la asigna a un especialista, el cual la registra en su lista de incidencias.

Resolver incidencia: El proceso se inicia todos los días a un ahora determinada, y en consecuencia un técnico o un especialista resuelven la incidencia informada por un usuario, firman la incidencia resuelta, la eliminan de su lista de incidencias y registran en otro documento la incidencia resulta con su respectiva solución.

2.3.2 Diagrama de casos de uso del negocio.

Modelado del negocio y captura de requisitos.

El diagrama de casos de uso del negocio, es un modelo que refleja gráficamente las metas y funciones que persigue el negocio.

Muestra los casos de uso del negocio, actores del negocio, trabajadores del negocio y las interacciones entre ellos. Modela lo que hace una compañía, quién está dentro y quién está fuera de la compañía. Da el alcance de la organización, visualizando lo que abarca y cuáles son sus fronteras.

- Diagrama de casos de uso del negocio.

Modelado del negocio y captura de requisitos.

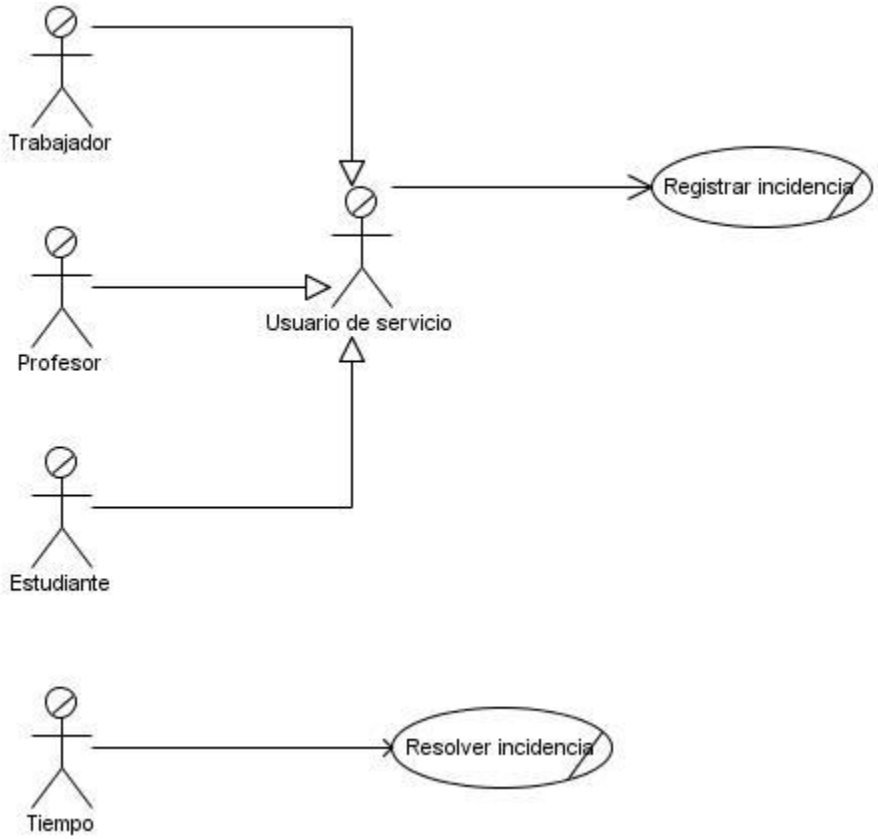


Figura 2 Diagrama de casos de uso del negocio.

- Especificación de casos de uso del negocio.

Tabla 1 Especificación del caso de uso del negocio: Registrar incidencia.

Casos de uso:	Registrar incidencia.
Actores:	Usuario de servicio.
Trabajadores:	Técnico, Especialista.

Modelado del negocio y captura de requisitos.

Resumen:	El caso de uso se inicia cuando el usuario de servicio notifica la incidencia al centro de soporte, ahí es registrada.
Precondiciones:	N/A
Flujo normal de eventos	
Acción del actor	**Respuesta del negocio**
1. El usuario de servicio informa la incidencia.	2. El técnico recibe la notificación de la incidencia. 3. Analiza la incidencia si le puede dar solución. 4. Registra la incidencia en su lista de incidencias.
Flujos alternos	
	3.1 Si no le puede dar solución se la asigna a un especialista y este la registra en su lista de incidencias.

Tabla 2 Especificación del caso de uso: Resolver incidencia.

Casos de uso:	Resolver incidencia.
Actores:	Tiempo
Trabajadores:	Técnico, Especialista.
Resumen:	El CU se inicia cuando el usuario de servicio notifica la incidencia al centro de soporte, ahí es resuelta.
Precondiciones:	La incidencia debe estar notificada.
Flujo normal de eventos	
Acción del actor	**Respuesta del negocio**
	1. El técnico busca la incidencia en su lista de incidencias. 2. Si existe resuelve la incidencia. 3. Firma la incidencia resuelta. 4. Registra la incidencia en el informe de incidencias.

Modelado del negocio y captura de requisitos.

	5. Notifica al usuario de servicio.
Flujos alternos 2	
	2.1 Si no le puede dar solución informa al especialista. 2.2 El especialista busca la incidencia en su lista de incidencias. 2.3 Si existe resuelve la incidencia. 2.4 Firma la incidencia resuelta. 2.5 Registra la incidencia en el informe de incidencias. 2.6 Notifica al usuario de servicio.
Flujos alternos 2.3.1	
	2.3.1 Si no existe, se le notifica al usuario de servicio.

- Actores del negocio.

Actor	Descripción
Usuario de servicio	Es el encargado de notificar una incidencia. La incidencia puede ser notificada por un estudiante, profesor o un trabajador, independientemente del rol que desempeña en su centro.

2.3.3 Modelo de objeto del negocio.

A continuación se muestra el modelo de objeto del negocio el cual representa las relaciones entre los trabajadores y las entidades del negocio.

- Modelo de objetos del negocio.

Modelado del negocio y captura de requisitos.

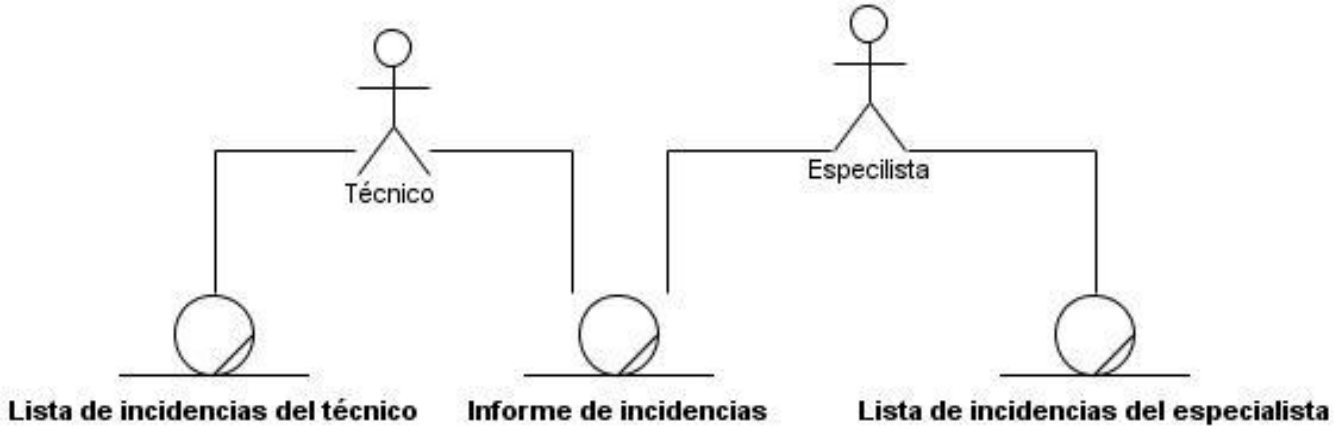


Figura 3 Modelo de objetos del negocio.

- Trabajadores del negocio.

Trabajadores	Descripción
Técnico	Es el encargado de recibir la notificación de la incidencia y darle solución.
Especialista	Se encarga de darle solución a las incidencias que el técnico no pudo resolver.

- Descripción de las entidades del negocio.

Lista de incidencias del técnico: Es la lista en formato duro que posee el técnico donde escribe las incidencias que resolverá posteriormente y donde son firmadas por los usuarios luego de ser resueltas.

Lista de incidencias del especialista: Es la lista en formato duro que posee el especialista donde escribe las incidencias que resolverá posteriormente y donde son firmadas por los usuarios luego de ser resueltas.

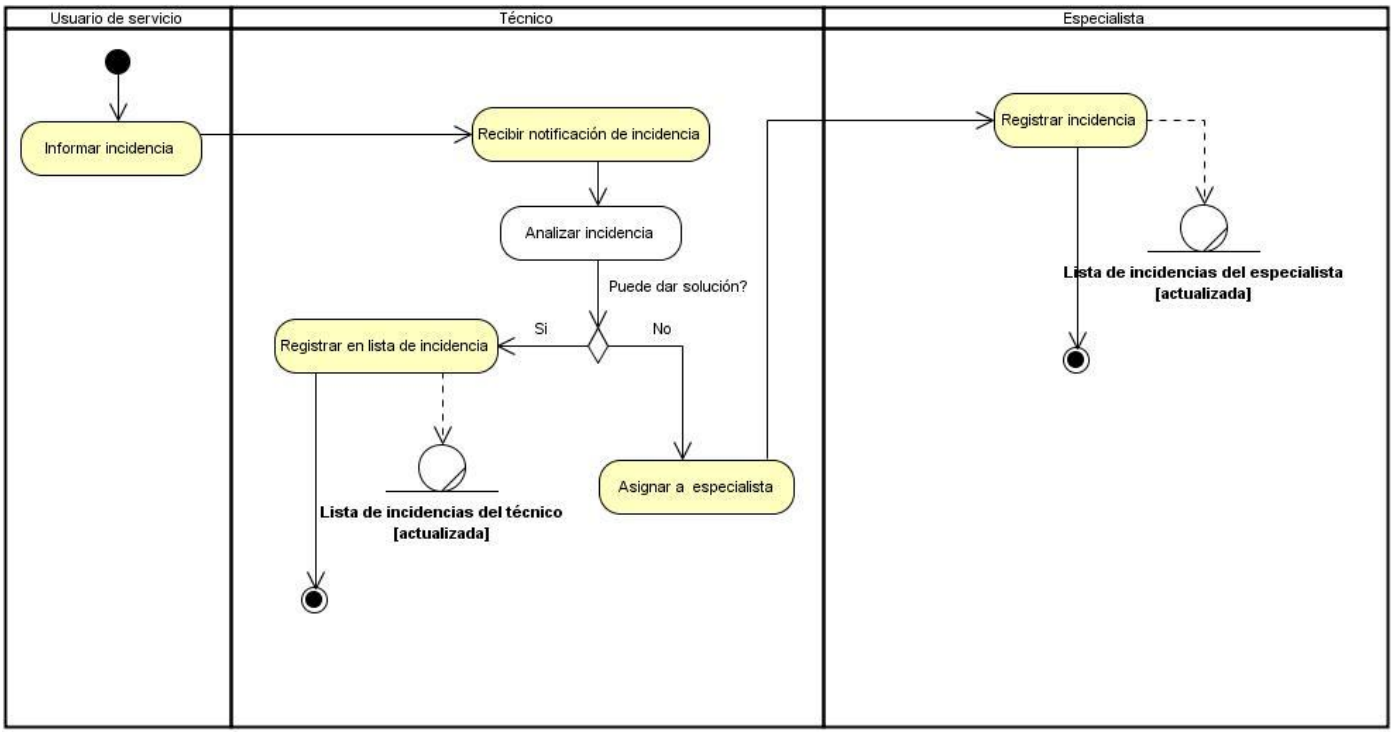
Modelado del negocio y captura de requisitos.

Informe de incidencias: Documento en formato duro donde se registran las incidencias resueltas con la información de quién la notificó, quién la solucionó y qué solución se le dio.

2.3.4 Diagrama de procesos del negocio.

Representa las actividades y el valor creado por el negocio e ilustra la interacción entre los procesos y los recursos para alcanzar la meta de cada proceso. Es una manera de modelar el flujo de trabajo de un caso de uso de manera gráfica. Se enfoca en el flujo de eventos internos de un proceso.

- Diagrama de procesos de negocio del caso de uso Registrar incidencia.



Modelado del negocio y captura de requisitos.

Figura 4 Diagrama de procesos del negocio: Registrar incidencia.

- Diagrama de procesos del negocio del caso de uso Resolver incidencia.

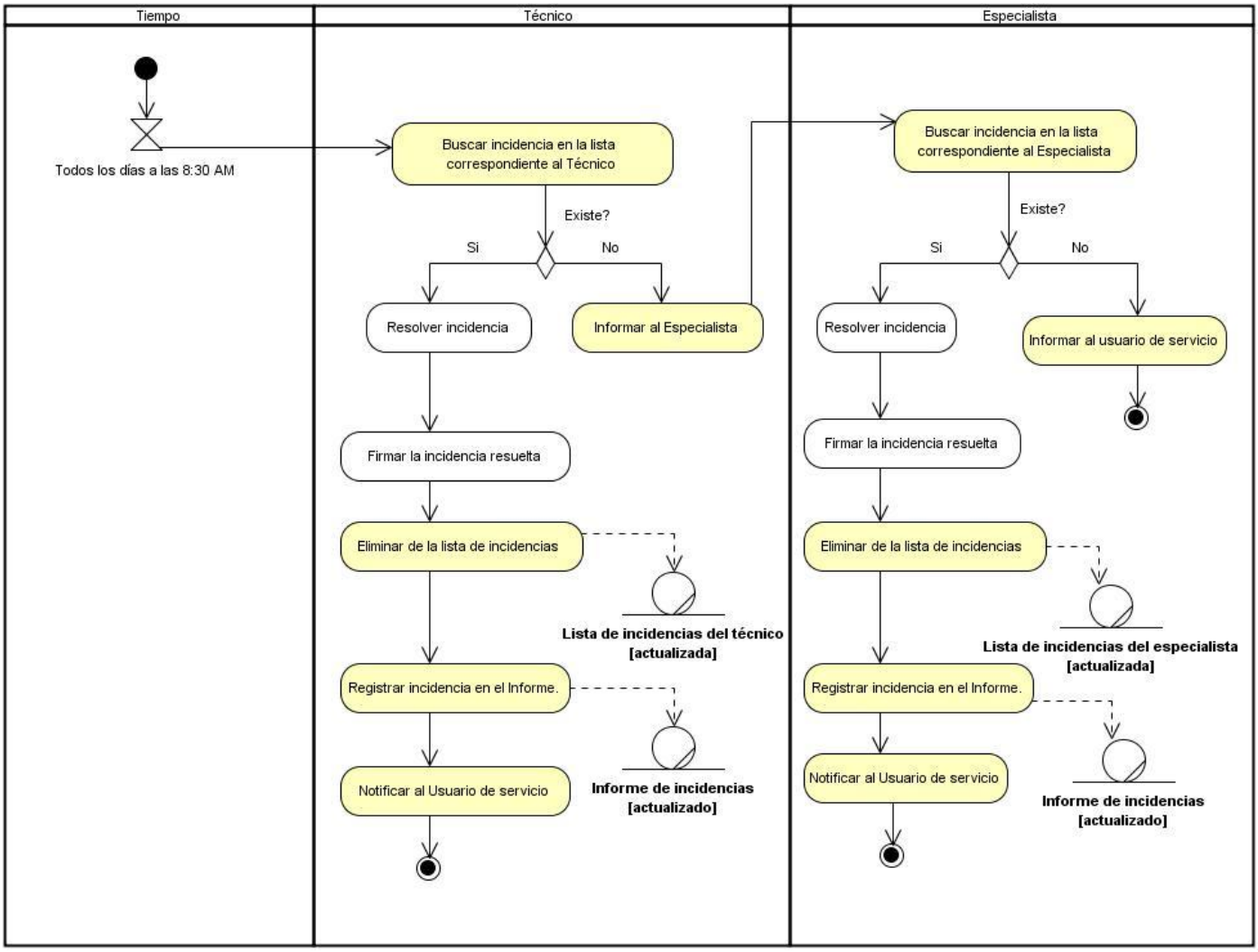


Figura 5 Diagrama de procesos del negocio: Resolver incidencia.

Modelado del negocio y captura de requisitos.

2.4 Técnicas utilizadas en las actividades de la ingeniería de requisitos.

Se hace referencia a las técnicas empleadas en la actividad de captura de requisitos necesarios para definir las funcionalidades que debe cumplir el sistema de acuerdo con las necesidades del cliente.

- **Entrevistas y Cuestionarios:**

Las entrevistas y cuestionarios se emplean para reunir información proveniente de personas o de grupos. Específicamente esta técnica se aplicó a los diferentes especialistas del Centro de Soporte para conocer de manera clara y concreta cómo se lleva a cabo la gestión de incidencia en la universidad y a partir de ahí definir las principales actividades a automatizar.

- **Sistemas existentes:**

Esta técnica consiste en analizar distintos sistemas ya desarrollados que están relacionados con el sistema a ser construido. El análisis de dichos sistemas se llevó a cabo a través de interfaces de usuario, observando el tipo de información, cómo ésta es manejada en la aplicación y las distintas salidas que producen, para determinar las principales ventajas y deficiencias en aras de desarrollar una aplicación que reúna las mejores prácticas de cada uno de ellos.

2.5 Patrones utilizados en la especificación de requisitos.

- **Patrón CRUD (Creating, Reading, Updating, Deleting)**

En los requerimientos identificados se pone de manifiesto el patrón CRUD, que aunque está dirigido a casos de uso está presente de manera completa y parcial en los requisitos de software definidos.

Modelado del negocio y captura de requisitos.

- **Nombres que revelan la intención (Intention Revealing Name)**

La utilización de este patrón se evidencia en la definición de todos los requisitos de manera que los nombres de los mismos fueran entendibles por los usuarios y por el equipo de desarrollo. Los nombres deben comenzar con un verbo y seguido de una frase que refleje su objetivo. Ejemplo: Mostrar Reportes, Adicionar Usuario y Eliminar Incidencia.

- **Preciso y legible (Precise and Readable).**

La utilización de este patrón se evidencia en la definición de todos los requisitos de manera que fueran entendibles por los usuarios y por el equipo de desarrollo. Ejemplo: Gestionar Ticket, Gestionar Componente de Software y Gestionar Servicio.

- **Adorno, Decoración (Adornments)**

La utilización de este patrón se evidencia en la definición de todos los requisitos, con la creación de campos dentro de la plantilla del caso de uso (requerimientos) que fuera del texto del escenario apoye la información auxiliar que es útil asociar con el caso de uso. En este caso las plantillas de especificación establecidas incluyen campos adicionales con el objetivo de una mayor compresión de la solución.

- **Pasos Nivelados (Leveled Steps)**

Este patrón se evidencia en la definición de todos los requisitos, donde cada escenario debe tener de tres a nueve pasos. Todos a niveles similares. Esto no quiere decir que un caso de uso (requerimiento) no pueda tener más de 9 pasos si lo requiere.

2.6 Requisitos del sistema.

Para dar cumplimiento al objetivo planteado y conocidos ya los conceptos relacionados con el objeto de estudio, se deben definir una serie de requisitos funcionales que sean capaces de resolver el problema existente. A continuación se muestra un listado de los requisitos funcionales del sistema.

Modelado del negocio y captura de requisitos.

2.6.1 Requisitos funcionales del sistema.

RF1. El sistema debe ser capaz de: Autenticar usuario.

RF2. El sistema debe ser capaz de: Gestionar usuario.

2.1 Insertar usuario.

2.2 Eliminar usuario.

2.3 Buscar usuario.

2.4 Modificar usuario.

RF3. El sistema debe ser capaz de: Gestionar tickets.

3.1 Adicionar un nuevo tickets.

3.2 Eliminar tickets.

3.3 Cerrar un ticket.

3.4 Liberar un ticket.

3.5 Escalar un ticket.

3.6 Responder un ticket.

3.7 Auto asignar un ticket.

3.8 Mostrar detalles de un ticket.

3.9 Mostrar tickets cerrados.

3.10 Adjuntar archivos.

3.11 Imprimir ticket.

Modelado del negocio y captura de requisitos.

3.12 Filtrar tickets.

3.13 Buscar tickets.

3.14 Mostrar Tickets No Asignados.

3.15 Mostrar Tickets Asignados.

3.16 Mostrar Tickets Resueltos.

3.17 Mostrar Trazas.

3.18 Mostrar Reportes.

RF4. El sistema debe ser capaz de: Gestionar departamento.

4.1 Insertar un nuevo departamento.

4.2 Eliminar departamento.

4.3 Modificar datos de un departamento.

4.4 Mostrar datos de un departamento existente.

4.5 Permitir adjuntar.

4.6 Permitir evaluar.

16.7 Permitir habilitar escalado.

RF5. El sistema debe ser capaz de: Gestionar componente de software.

5.1 Insertar un nuevo componente de software.

5.2 Eliminar componente de software.

5.3 Modificar datos de un componente de software.

5.4 Mostrar datos de un componente de software existente.

5.5 Permitir Seleccionar el lenguaje de programación.

RF6. El sistema debe ser capaz de: Gestionar componente de hardware.

6.1 Insertar un nuevo componente de hardware.

6.2 Eliminar componente de hardware.

6.3 Modificar datos de un componente de hardware.

6.4 Mostrar datos de un componente de hardware.

RF7. El sistema debe ser capaz de: Gestionar solución integral.

7.1 Insertar una nueva solución integral.

7.2 Eliminar solución integral.

7.3 Modificar datos de solución integral.

7.4 Mostrar datos de solución integral.

RF8. El sistema debe ser capaz de: Gestionar contratos.

8.1 Insertar un nuevo contrato.

8.2 Eliminar contrato.

8.3 Modificar datos de un contrato.

8.4 Mostrar datos de un contrato.

8.5 Permitir seleccionar catálogos.

RF9. El sistema debe ser capaz de: Gestionar catálogos de servicios.

9.1 Insertar un nuevo catálogo de servicios.

9.2 Eliminar catálogos de servicios.

9.3 Modificar datos de un catálogo de servicios.

9.4 Mostrar datos de un catálogo de servicios.

9.5 Permitir seleccionar portafolio de servicios.

RF10. El sistema debe ser capaz de: Gestionar cuentas.

10.1 Insertar una nueva cuenta.

10.2 Eliminar cuentas.

10.3 Modificar datos de una cuenta.

10.4 Mostrar datos de una cuenta.

10.5 Permitir especificar si es proveedor.

10.6 Permitir especificar si es cliente.

RF11. El sistema debe ser capaz de: Gestionar servicios.

11.1 Insertar un nuevo servicio.

11.2 Eliminar servicios.

11.3 Modificar datos de un servicio.

11.4 Mostrar datos de un servicio.

11.5 Opinar sobre un servicio.

RF12. El sistema debe ser capaz de: Gestionar cursos.

12.1 Insertar cursos.

Modelado del negocio y captura de requisitos.

12.2 Eliminar cursos.

12.3 Modificar datos de un curso.

12.4 Mostrar datos de un curso existente.

12.5 Buscar curso.

2.6.1.1 Especificación de los requisitos funcionales del sistema.

En las especificaciones de requisitos se registran las características y condiciones definidas que debe cumplir cada requisito funcional.

A continuación se lleva a cabo la especificación de los requisitos funcionales que tienen mayor impacto en el sistema:

Tabla 3 Especificación del requisito Adicionar ticket.

Precondiciones	El usuario debe estar autenticado con anterioridad. La incidencia debe tener el formato definido.
Flujo de eventos	
Flujo básico	
1	El usuario selecciona la opción Incidencias en la sección Adicionar del módulo Centro de Gestión.
2	El sistema muestra una pantalla para que los datos de la incidencia sean insertados, estos son: Centro/Dirección al que pertenece, Rol que desempeña, Servicio asociado a la incidencia, Asunto, Prioridad, Adjunto (opcional) y Contenido (que es donde va el texto que describe la incidencia).
3	El usuario inserta los datos correspondientes a la incidencia y presiona el botón Enviar.

Modelado del negocio y captura de requisitos.

4	El sistema procede al registro de la incidencia en la base de datos.
5	El sistema muestra una pantalla con el total de tickets enviados por el usuario.
6	Concluye el requisito.
Pos-condiciones	
	La incidencia queda registrada en el sistema.
Flujos alternativos	
Flujo alternativo 3.a Información incompleta	
	El sistema muestra un mensaje de error indicando que debe llenar todos los campos.
	El usuario corrige los datos.
	Volver al paso 3 del flujo básico.
Pos-condiciones	
	N/A
Flujo alternativo *.a El usuario cancela la acción	
	Concluye el requisito.
Pos-condiciones	
	N/A
Validaciones	

Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A
Conceptos	**Incidencia**	Visibles en la interfaz: Incidencia Rol Servicio Prioridad

Modelado del negocio y captura de requisitos.

Requisitos especiales	N/A
Asuntos pendientes	N/A

Tabla 4 Especificación del requisito Eliminar ticket.

Precondiciones	El usuario con el rol de administrador se debe haber autenticado con anterioridad. El administrador debe Listar o Buscar incidencia.
Flujo de eventos	
Flujo básico	
1	El administrador elige la incidencia y selecciona la opción Eliminar.
2	El sistema muestra una confirmación para estar seguro que lo desea eliminar.
3	El administrador confirma que desea eliminar la incidencia presionando el botón Aceptar.
4	El sistema procede a la eliminación de los datos de la incidencia seleccionada.
5	Concluye el requisito.
Pos-condiciones	
	El sistema permite que los datos de la incidencia seleccionada queden eliminados.
Flujos alternativos	
Flujo alternativo 2.a Información errónea	
	Si presiona el botón Cancelar no se ejecuta la acción.
	Concluye el requisito.
Pos-condiciones	
	N/A
Flujo alternativo *.a El usuario cancela la acción	

Modelado del negocio y captura de requisitos.

	Concluye el requisito.	
Pos-condiciones		
	N/A	
Validaciones		
Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A
Conceptos	N/A	
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Tabla 5 Especificación del requisito Buscar ticket.

Precondiciones	El usuario con el rol de administrador se debe haber autenticado con anterioridad.
Flujo de eventos	
Flujo básico	
1	El administrador selecciona la opción Buscar en la sesión de Incidencia del módulo Centro de Gestión.

Modelado del negocio y captura de requisitos.

2	El sistema muestra los parámetros disponibles para hacer la búsqueda.	
3	El administrador ingresa los datos por los cuales desea realizar la búsqueda.	
4	El sistema realiza la búsqueda.	
5	Concluye el requisito.	
Pos-condiciones		
	El sistema muestra los datos referentes al parámetro seleccionado.	
Flujos alternativos		
Flujo alternativo 4.a No es satisfactoria la búsqueda		
	El sistema muestra un mensaje indicando que no se encuentran los parámetros de búsqueda.	
	Volver al paso 2 del flujo básico.	
Pos-condiciones		
	N/A	
Flujo alternativo *.a El usuario cancela la acción		
	Concluye el requisito.	
Pos-condiciones		
Validaciones		
Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A
Conceptos	**Incidencia**	Visibles en la interfaz: Incidencia Ubicación Solución integral

Modelado del negocio y captura de requisitos.

Requisitos especiales	N/A
Asuntos pendientes	N/A

Tabla 6 Especificación del requisito Autenticar usuario.

Precondiciones	NA
Flujo de eventos	
Flujo básico	
1. El usuario ingresa los datos en el formulario (usuario y contraseña del ldap).	
2. El sistema verifica que los campos del formulario no estén vacios.	
3. El sistema verifica la existencia del usuario.	
4. El sistema le da acceso al usuario de interactuar con la aplicación y termina el caso de uso.	
5. Concluye el requisito.	
Pos-condiciones	
1. El usuario queda autenticado con un nuevo nivel de permiso.	
Flujos alternativos	
Flujo alternativo 2.a Información incompleta	
1. El sistema muestra un mensaje de error indicando que debe llenar todos los campos.	
2. El usuario corrige los datos.	
3. Volver al paso 2 del flujo básico.	
Pos-condiciones	

Modelado del negocio y captura de requisitos.

N/A		
Flujo alternativo 3.a Información errónea		
1. El sistema muestra un mensaje de error indicando que la cuenta no existe.		
2. El usuario corrige los datos.		
3. Volver al paso 2 del flujo básico.		
Pos-condiciones		
N/A		
Validaciones		
Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A
Conceptos	N/A	
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Consultar Anexo 1: Especificación del requisito Gestionar curso.

2.6.2 Requisitos no funcionales del sistema.

RNF1. Usabilidad.

1.1 El sistema debe estar disponible las veinticuatro horas del día, sin ninguna interrupción.

Modelado del negocio y captura de requisitos.

1.2 El sistema debe ser accesible desde todos los puntos donde exista una máquina conectada a la red.

1.3 El sistema debe tener una interfaz que le sea familiar al usuario para aprovechar sus conocimientos en el manejo de herramientas de software.

1.4 El sistema debe tener una interfaz de fácil aprendizaje para que usuarios inexpertos puedan familiarizarse lo más pronto posible y le sea cómodo el manejo del software.

1.5 El sistema debe diferenciar las interfaces gráficas y opciones para los usuarios que accedan al sistema con diferentes roles.

RNF2. Confiabilidad y Seguridad.

2.1 El sistema debe tener la capacidad de identificar con certeza a los diversos usuarios o entidades que interactúan con él.

2.2 El sistema debe tener la capacidad de darle seguridad al usuario, de que las informaciones solo serán vistas por quien esté capacitado para esto.

2.3 El servicio del sistema debe tener una disponibilidad aceptable (99%).

2.4 El tiempo entre fallos debe ser breve o cero, haciendo lo posible para que esto no ocurra.

2.5 La base de datos debe estar fraccionada sobre varios esquemas, dividiendo así de una forma lógica las funcionalidades, evitando así la pérdida total de la información en caso de algún accidente o ataque.

2.6 Permitir autenticarse a través de un servidor de dominio.

RNF3. Rendimiento.

3.1 El sistema debe ser capaz de mantener el mismo rendimiento y estabilidad a medida que aumenta la cantidad de datos a gestionar.

3.2 El Sistema debe ser capaz de correr al ser montado en una misma PC todos sus componentes tales como Gestor de bases de datos, Aplicación Web.

Modelado del negocio y captura de requisitos.

RNF4. Soportabilidad y Operatibilidad.

4.1 El software podrá operar en cualquier sistema operativo debido a que se desarrollará en Java y herramientas de software libre como es el caso de PostgreSQL. En cuanto al soporte debe tenerse en cuenta en el sistema operativo en que se esté trabajando y buscar la tecnología necesaria para las herramientas que requiere el software en el mismo.

RNF5. Mantenimiento y Actualización.

5.1 El ambiente de desarrollo donde se implementará dicho software será: NetBeans 6.7.

5.2 Gestor de base de datos: PostgreSQL.

5.3 Lenguaje de programación: PHP.

5.4 El sistema debe ser construido en un código estándar, cada procedimiento debe estar comentado.

RNF6. Funcionalidad.

6.1 Capacidades de búsqueda con velocidad apropiada.

6.2 El sistema debe ser multiplataforma.

RNF7. Desempeño y Escalabilidad.

1.1 El tiempo de respuestas debe ser corto, con respuestas rápidas y eficientes.

1.2 El sistema podrá soportar un gran número de clientes online.

1.3 El sistema debe tener un rendimiento óptimo debido a que presta servicios a un gran número de usuarios.

2.7 Prototipos de interfaz de usuarios propuestos.

Los prototipos de interfaz de usuario ayudan a identificar, comunicar y probar un producto antes de crearlo. La realización de los mismos, logra un entendimiento común entre el cliente y los desarrolladores,

solucionando así la mayoría de los cambios posteriores en los proyectos de desarrollo de software. A continuación se muestran los prototipos de interfaz de usuario de los escenarios más importantes correspondientes a los requisitos funcionales Autenticar usuario y el Gestionar tickets.

Figura 6 Autenticar usuario.

Modelado del negocio y captura de requisitos.

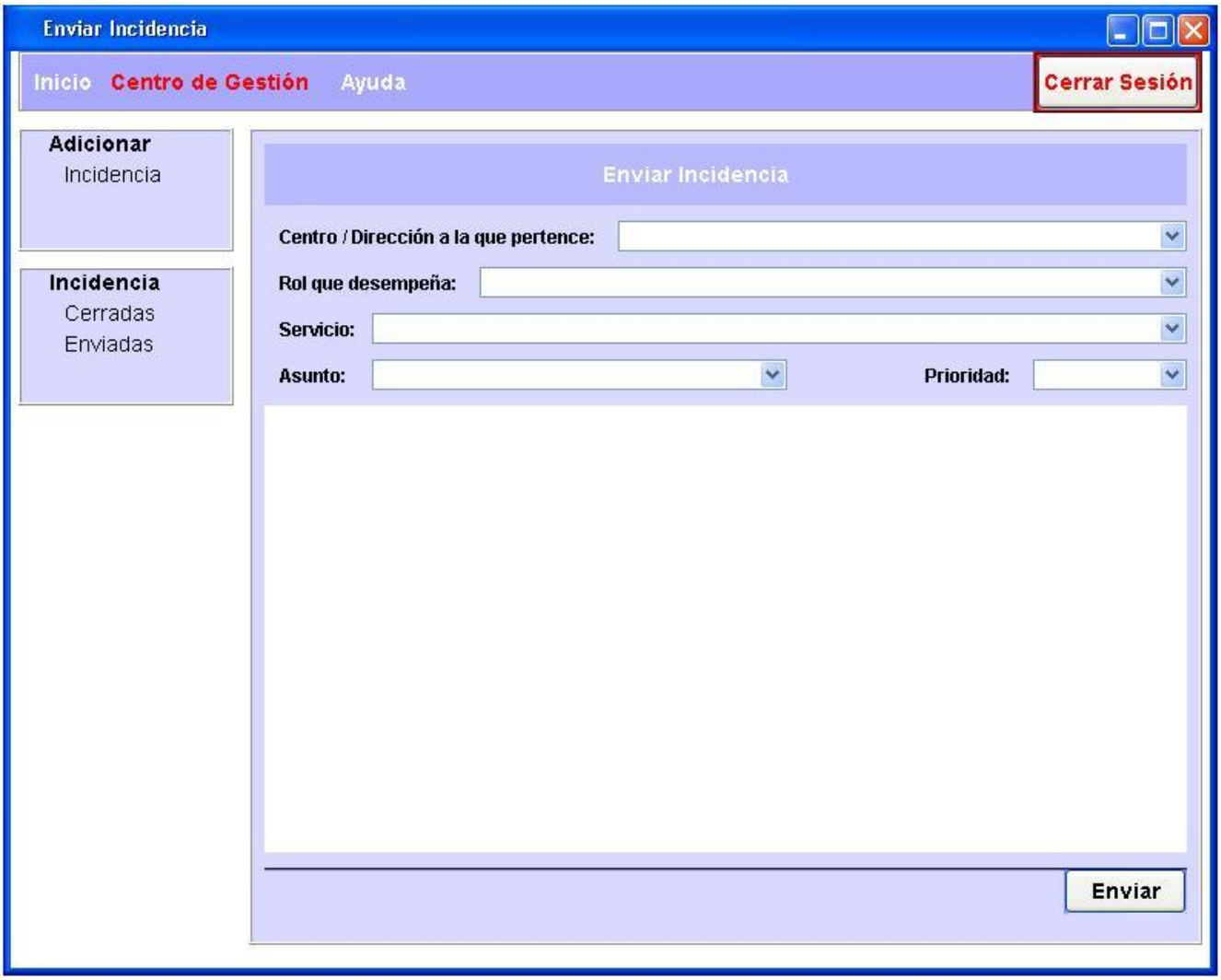


Figura 7 Adicionar ticket.

Modelado del negocio y captura de requisitos.

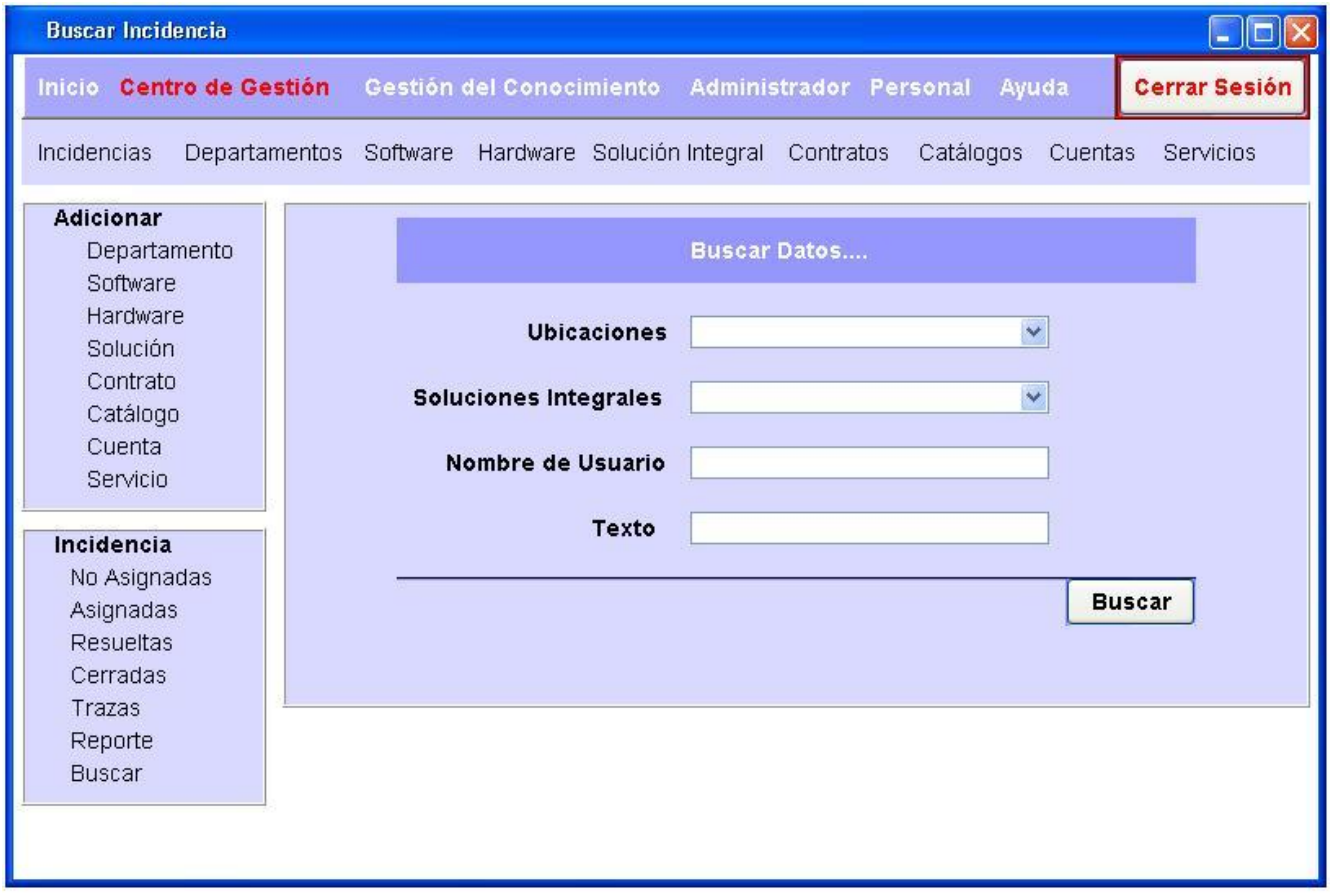


Figura 8 Buscar ticket.

Modelado del negocio y captura de requisitos.

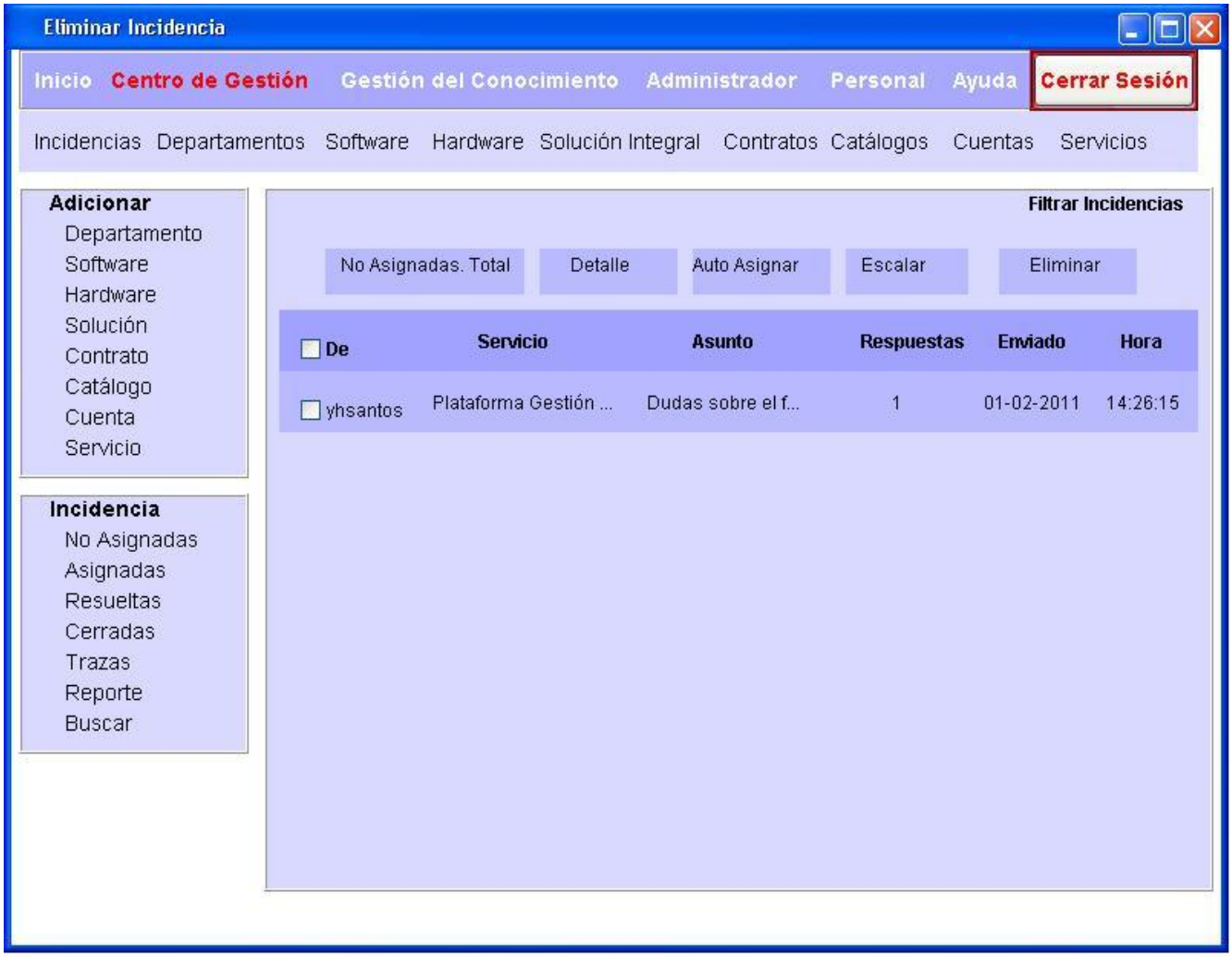


Figura 9 Eliminar ticket.

2.8 Validación de los requisitos.

Modelado del negocio y captura de requisitos.

La validación de los requisitos mediante técnicas y métricas para demostrar que los requisitos definidos para el sistema están acordes con las expectativas del cliente.

2.8.1 Pasos para la validación de requisitos.

Esta actividad tiene como propósito validar que se ha descrito de manera correcta lo que se debe automatizar.

La validación se realiza en tres pasos:

- Revisión técnica por el equipo de analistas principales: El objetivo de este paso es verificar que se hayan construido correctamente los artefactos correspondientes a la Ingeniería de Requisitos.
- Revisión funcional: Se realizará con el propósito de validar que las funcionalidades descritas satisfacen las expectativas de los interesados y que desde el punto de vista funcional se han descrito correctamente.
- Taller de aprobación: El objetivo de este taller es presentar las soluciones a las observaciones detectadas en los pasos anteriores y aprobar la especificación de requisitos según los criterios para la evaluación y aceptación de los requisitos.

Si como resultado de este taller aún persisten observaciones y se decide aprobar los requisitos, se llegará a un acuerdo de cómo resolver cada observación. Los acuerdos se registran en la sección observaciones de la especificación de requisitos y se procede a su aprobación. Una vez aprobados los requisitos no pueden ser modificados, pero el equipo de desarrollo está obligado a resolver en el sistema las observaciones registradas y a actualizar la especificación de requisitos. Estos pasos se llevaron a cabo y los requisitos fueron firmados por el cliente (El director del centro de soporte) y un conjunto de representantes de equipo de desarrollo del proyecto.

Modelado del negocio y captura de requisitos.

2.8.2 Técnicas de validación de los requisitos.

Prototipos: Algunas propuestas de prototipos de interfaz de usuario se basan en obtener de la definición de requisitos, prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema.

- Un prototipo es una versión inicial de un sistema de software que se utiliza para demostrar los conceptos, probar las opciones de diseño y entender mejor el problema y su solución.
- Un prototipo puede revelar errores u omisiones en los requerimientos propuestos, favorece la comunicación entre clientes y desarrolladores, da una primera visión del producto.

Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final.

2.8.3 Métricas para la validación de los requisitos.

La especificación de requisitos es un factor principal para el buen funcionamiento del desarrollo de software. De acuerdo con el estándar IEEE 830, se considera que una especificación es de calidad cuando se puede decir que es correcta, no-ambigua, completa, consistente, ordenada por importancia y estabilidad, verificable, modificable y trazable.

Especificación correcta.

La corrección no se puede establecer a priori, sino que depende fundamentalmente del usuario final del sistema representado. Quien debe decidir si una especificación es correcta o no es el cliente que solicita el sistema. Por eso la corrección de una especificación debe ser verificada a través de la revisión y aceptación del usuario.

Modelado del negocio y captura de requisitos.

Especificación no ambigua.

El equipo de desarrollo que interviene en el proceso de especificación de requisitos suele tener varios puntos de vista. Por este motivo es difícil asegurar la ausencia de ambigüedad en la especificación. Aunque a priori parece un problema insalvable, existen formas de limitar los efectos negativos de la ambigüedad

Especificación completa.

Una especificación es completa si, y sólo si, describe todos los requisitos relevantes para el usuario, incluyendo requisitos asociados con funcionalidad, actuación, restricciones de diseño, atributos o interfaces externas.

Especificación consistente.

Los mayores problemas relacionados con la consistencia son los que tienen que ver con las incoherencias lógicas entre distintos requisitos (requisitos incompatibles, incoherentes o mutuamente excluyentes), la repetición de la misma información a lo largo de distintos requisitos (requisitos repetitivos o redundantes) o la referencia en distintos requisitos a ítems que usan la misma palabra para designar conceptos del problema diferentes (incoherencia respecto al dominio del problema).

Especificación organizada.

La categorización de los requisitos por orden de importancia es una recomendable práctica que permite establecer prioridades a la hora de abordar el desarrollo. Esta categorización por el atributo importancia o prioridad es necesaria desde un punto de vista práctico.

Modelado del negocio y captura de requisitos.

Otra posible categorización que resulta interesante es la de la estabilidad de la especificación. El cambio de los requisitos de usuario es algo intrínseco al propio cambio en el problema.

Especificación verificable.

Se considera que una especificación es verificable si lo son cada uno de los requisitos constituyentes. A su vez, se considera que un requisito individual es verificable si existe un proceso acotado que permita determinar que el sistema construido satisface lo descrito en el propio requisito.

Una forma de conseguir que los requisitos sean verificables es describirlos con suficiente detalle, o teniendo en cuenta que una de las premisas que se debe cumplir es que sean probados una vez implementados.

Especificación modificable.

Se considera que una especificación es modificable si su estructura permite realizar cambios sobre los requisitos que contiene de forma sencilla, completa y consistente, manteniendo la estructura inicial del conjunto. Esto implica que debe existir una buena organización de la información y que el acoplamiento entre requisitos sea el menor posible.

Especificación trazable.

Una especificación se considera trazable si el origen de cada requisito individual está claro y existe algún mecanismo que permita seguir el impacto de dicho requisito a lo largo del resto de actividades del ciclo productivo. (12)

Tabla 7 Métricas auxiliares aplicadas a la especificación de requisitos.

Modelado del negocio y captura de requisitos.

	Descripción	**Valor**
TR	Total de requerimientos evaluados	131
NUI	Número de requisitos para los que todos los revisores tuvieron una misma interpretación	131
RC	Cantidad de requisitos cambiados (insertados, modificados y eliminados)	5
NNV	Número de requisitos no válidos	0
NC	Número de requisitos considerado válidos	131

Tabla 8 Métricas principales aplicadas a la especificación de requisitos.

No	**Métrica**	**Fórmula**	**Propiedad**	**Valor**
1	Especificidad	Q1 = NUI / RT * 100	No ambigüedad	100%
2	Estabilidad	Q2 = RC / TR *100	Estabilidad	3.8%
3	Grado de validación	Q3 = NC / (NC + NNV)* 100	Grado de validación	100%

Interpretación.

Modelado del negocio y captura de requisitos.

- Una especificidad de requerimientos de un 100% es una especificación realizada con una alta calidad, con ausencia de ambigüedades, demostrando que todas las personas involucradas en el proceso de revisar los requisitos coincidieron con la interpretación de los mismos.
- La inestabilidad es de un 3.8%, por lo que las especificaciones se consideran lo bastante estable (96.2%) con respecto a los cambios que se realizaron.
- A partir del resultado que arrojó la métrica Grado de validación se demuestra que todos los requisitos están en estado validado.

2.9 Conclusiones del capítulo.

Una vez realizadas las tareas definidas para dar cumplimiento a los objetivos específicos que tributan a este capítulo, se concluye que se le puede dar comienzo a la construcción de la propuesta de solución, teniendo en cuenta que la realización del modelo de dominio como parte de la modelación del negocio, las especificaciones de los requisitos del sistema y la presentación de los prototipos de interfaz de usuario, constituyen una entrada válida para las fases posteriores del desarrollo a partir de los resultados arrojados luego de la validación de los requerimientos funcionales son satisfactorios.

Capítulo 3 – Análisis y diseño de la solución.

3.1 Introducción.

Teniendo en cuenta las tareas que se deben llevar a cabo para dar por cumplido el presente capítulo, se realizará el análisis y el diseño de la solución a partir de la especificación de requisitos previamente efectuada. Inicialmente dichos requisitos serán traducidos a un lenguaje más entendible por el equipo de desarrollo y posteriormente serán diseñados para formar parte del futuro sistema, adecuándose a las características de la arquitectura base definida y sustentado en la utilización de patrones como elemento fundamental para el desarrollo.

3.2 Diagramas de clases del análisis.

- Diagrama de clase del análisis para el requisito funcional Autenticar usuario.

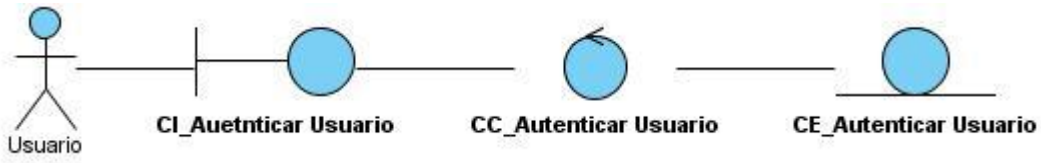


Figura 10 Diagrama de clases del análisis: Autenticar usuario.

- Diagrama de clase del análisis para el requisito funcional Gestionar usuario.

Análisis y diseño de la solución.

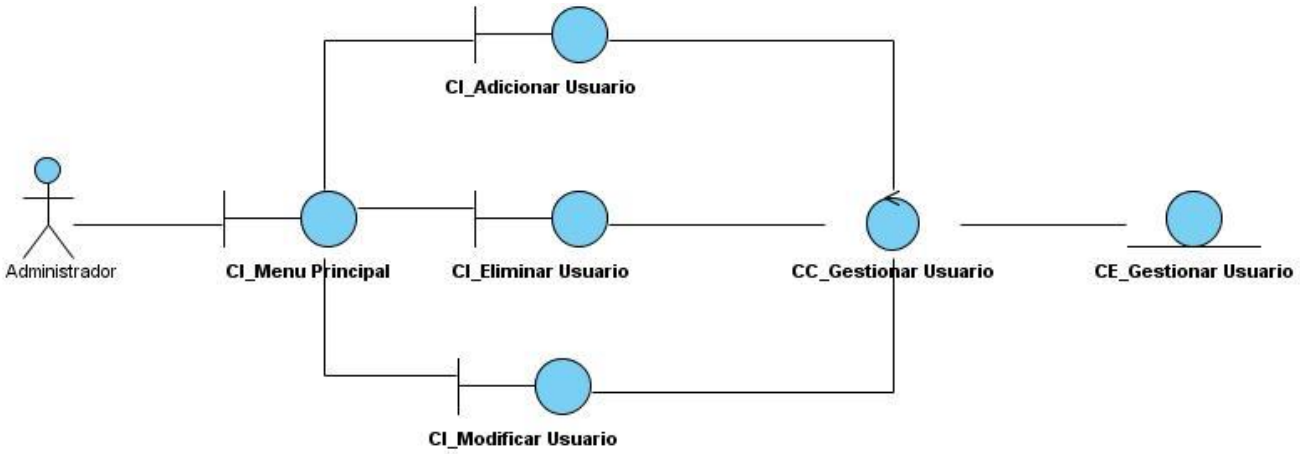


Figura 11 Diagrama de clases del análisis: Gestionar usuario.

- Diagrama de clase del análisis para el requisito funcional Gestionar ticket.

Análisis y diseño de la solución.

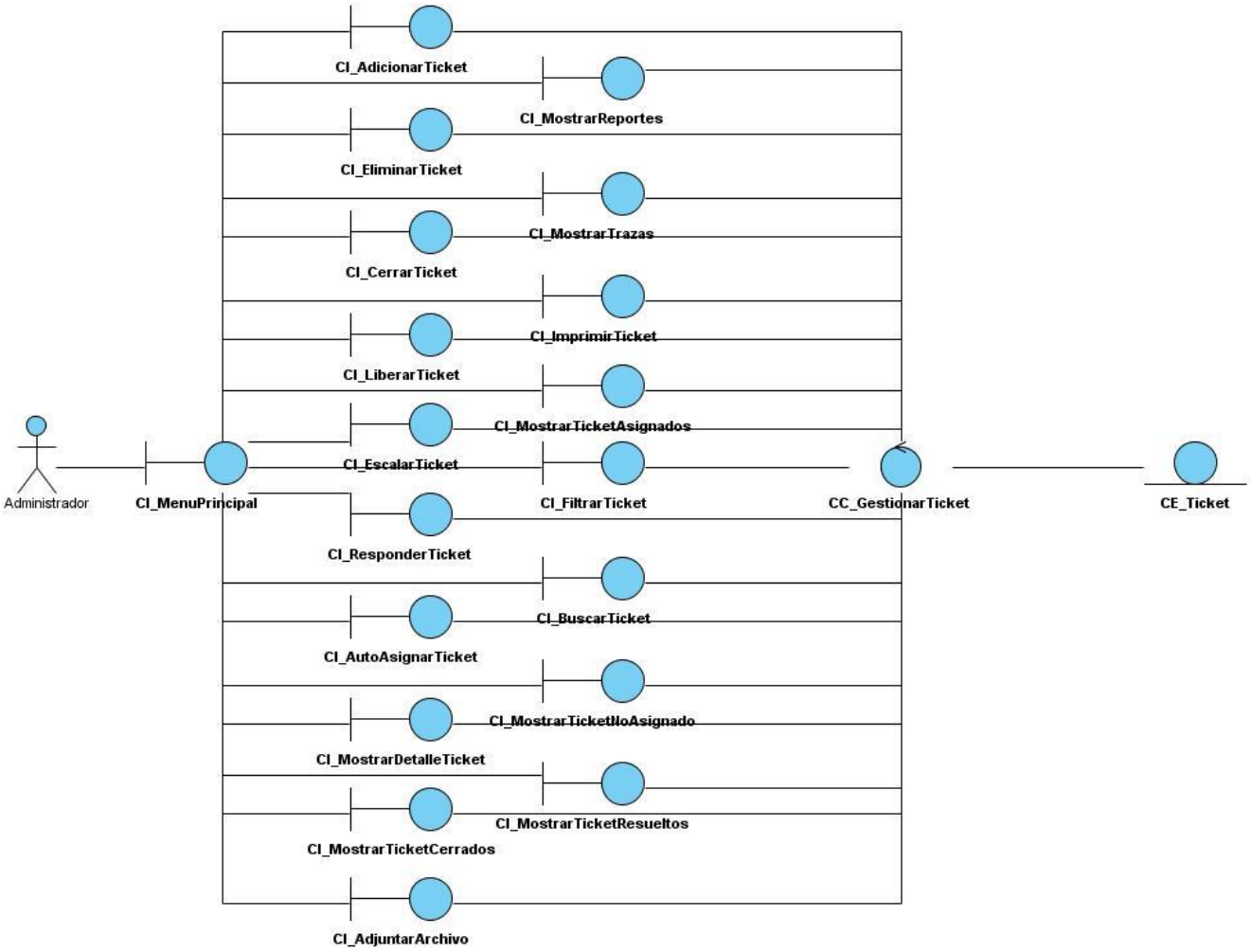


Figura 12 Diagrama de clases del análisis: Gestionar ticket.

Consultar Anexo 2: Diagrama de clases del análisis Gestionar usuario.

3.2.1 Diagramas de interacción del análisis.

Análisis y diseño de la solución.

- Diagrama de interacción del análisis para el requisito funcional Gestionar ticket escenario Adicionar incidencia.

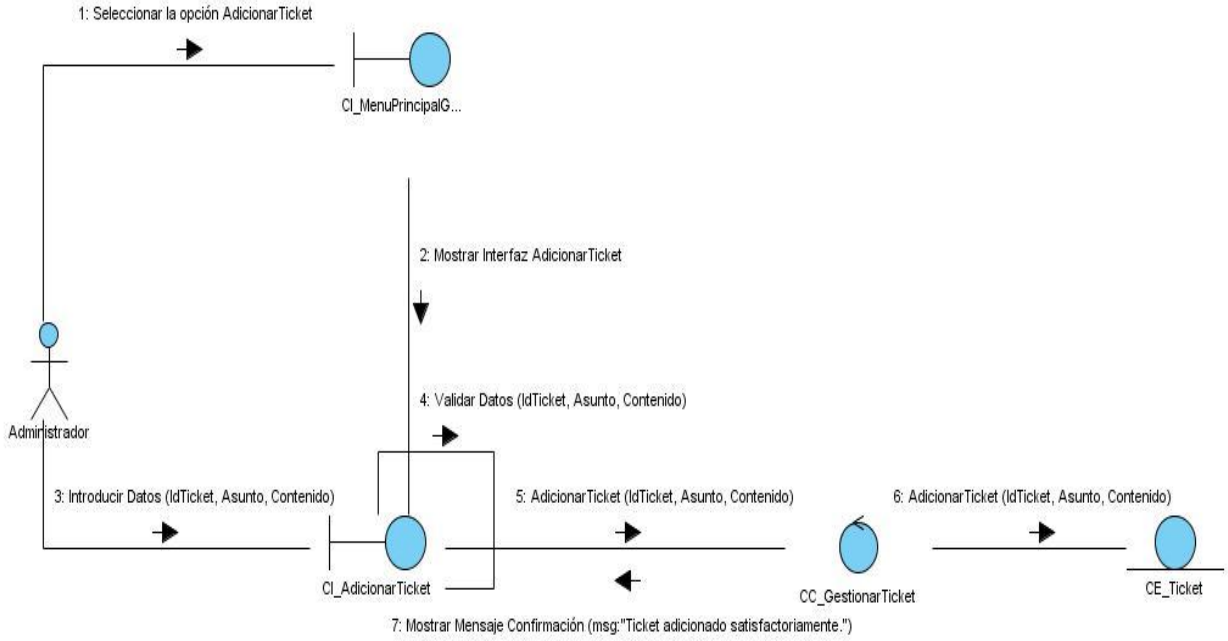


Figura 13 Diagrama de interacción del análisis: Gestionar ticket escenario Adicionar ticket.

- Diagrama de interacción del análisis para el requisito funcional Gestionar ticket escenario Eliminar incidencia.

Análisis y diseño de la solución.

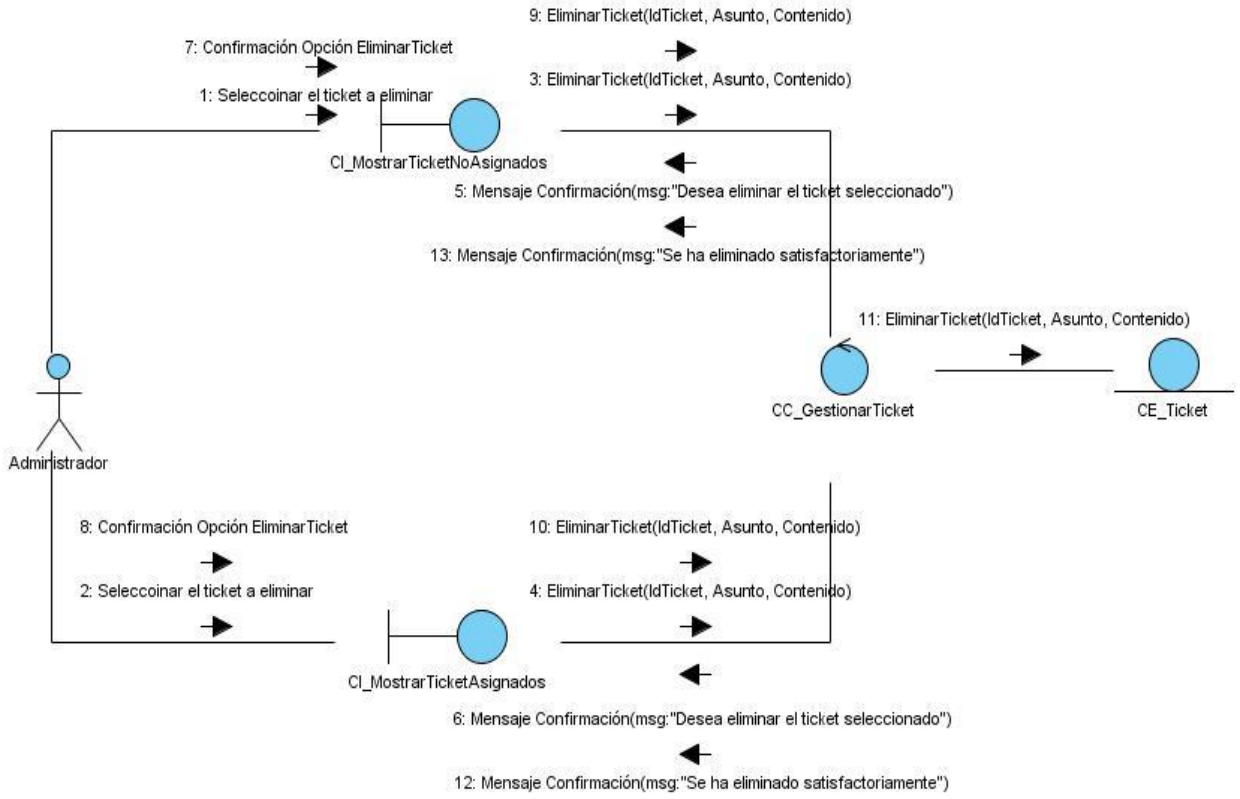


Figura 14 Diagrama de interacción del análisis: Gestionar ticket escenario Eliminar ticket.

- Diagrama de interacción del análisis para el requisito funcional Gestionar ticket escenario Mostrar detalles de incidencia.

Análisis y diseño de la solución.

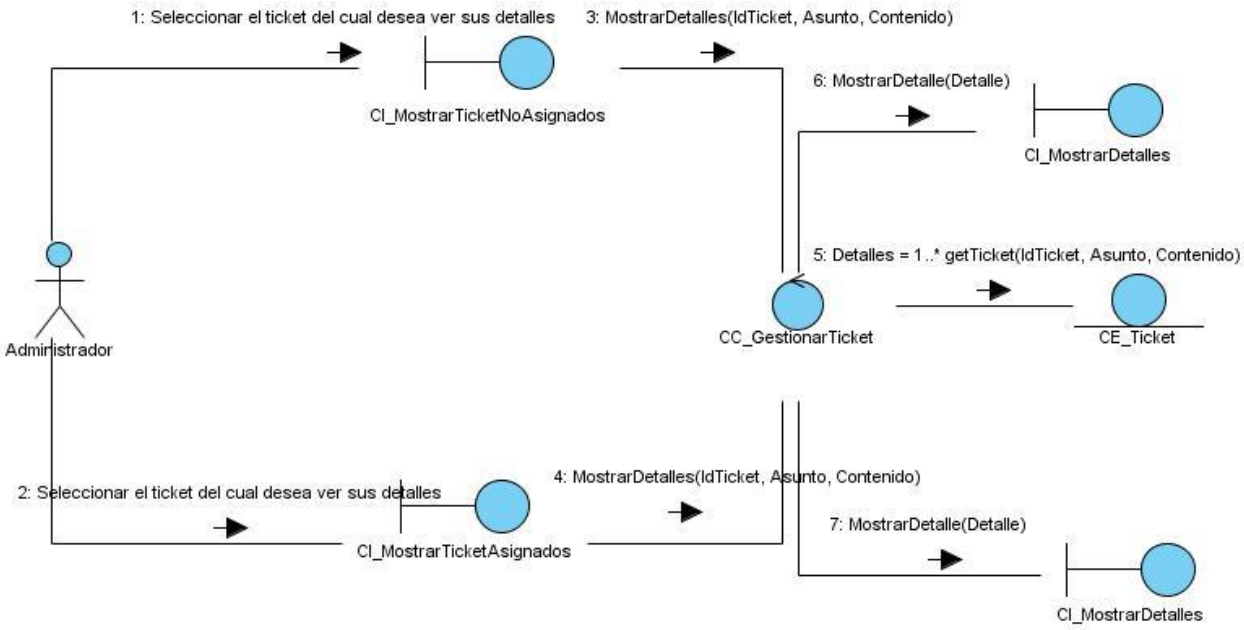


Figura 15 Diagrama de interacción del análisis: Gestionar ticket escenario Mostrar detalles del ticket.

- Diagrama de interacción del análisis para el requisito funcional Gestionar ticket escenario Buscar incidencia.

Análisis y diseño de la solución.

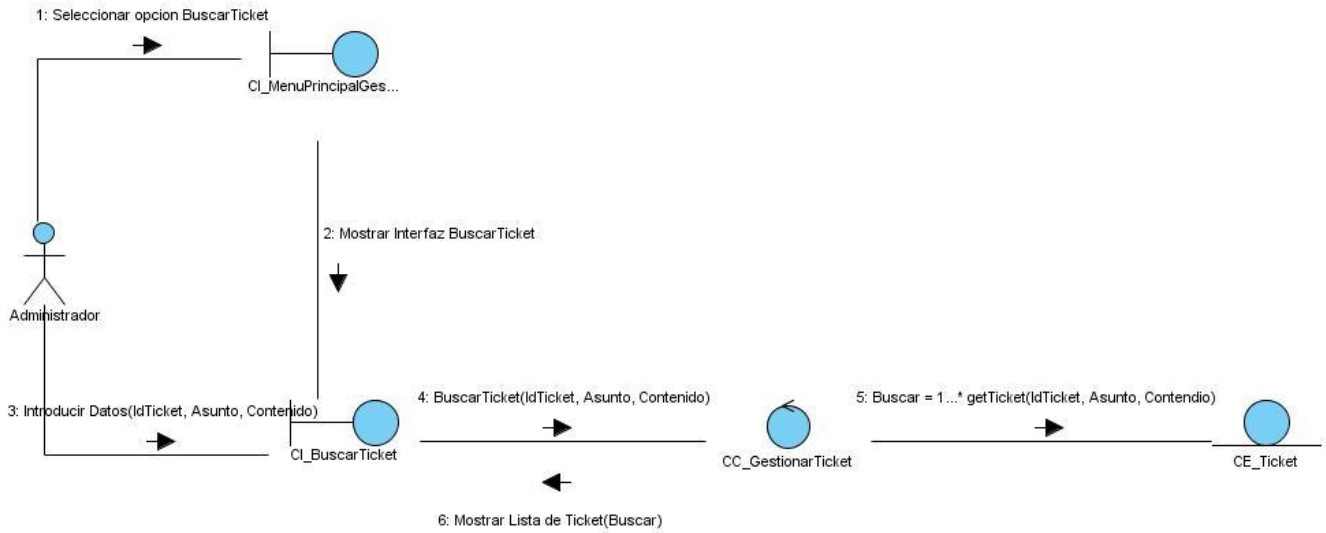


Figura 16 Diagrama de interacción del análisis: Gestionar ticket escenario Buscar ticket.

Consultar Anexo 3: Diagrama de interacción del análisis Gestionar curso para el escenario Adicionar curso.

3.3 Diagramas de clases del diseño.

En el diseño se modela el sistema y se encuentra su forma, incluyendo la arquitectura, para que soporte todos los requisitos y otras restricciones relacionadas con el entorno de la implementación. El modelo de diseño sirve de abstracción a la implementación del sistema y es utilizado como entrada fundamental de las actividades de implementación.

- Diagrama de clase del diseño para el requisito funcional Autenticar usuario.

Análisis y diseño de la solución.

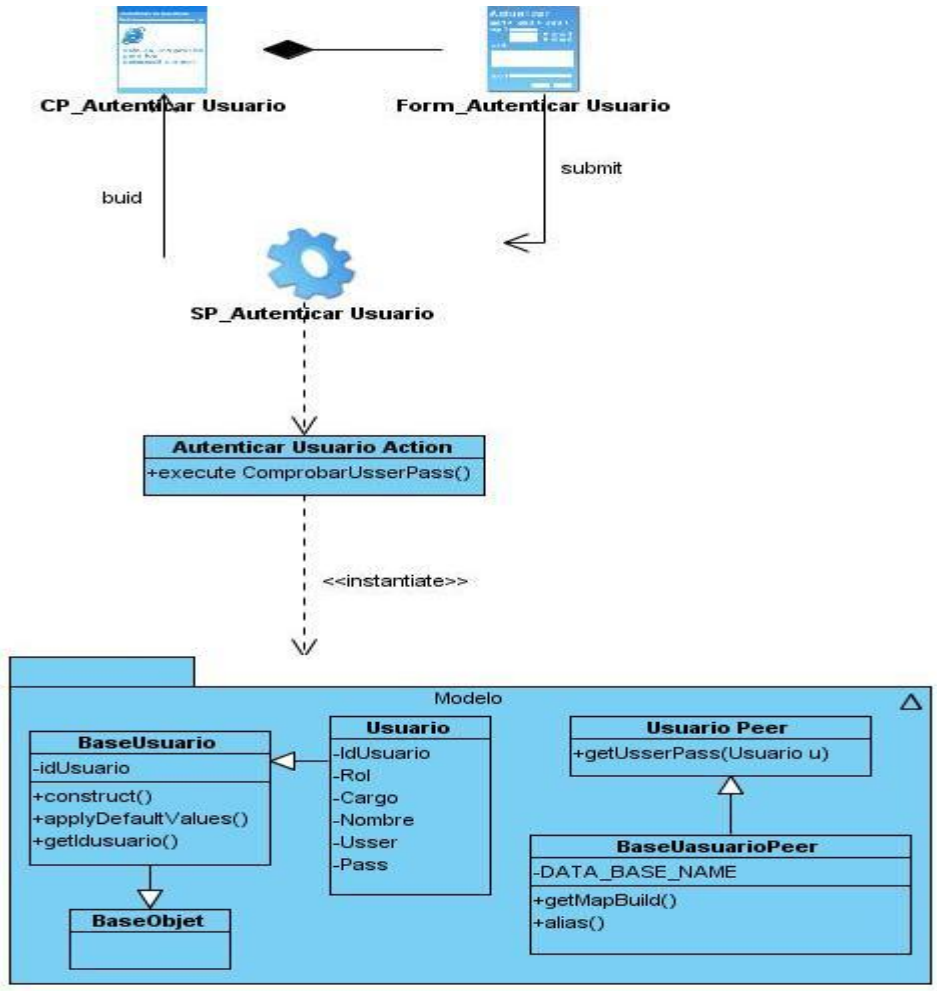


Figura 17 Diagrama de clases del diseño: Autenticar usuario.

- Diagrama de clase del diseño para el requisito funcional Gestionar usuario.

Análisis y diseño de la solución.

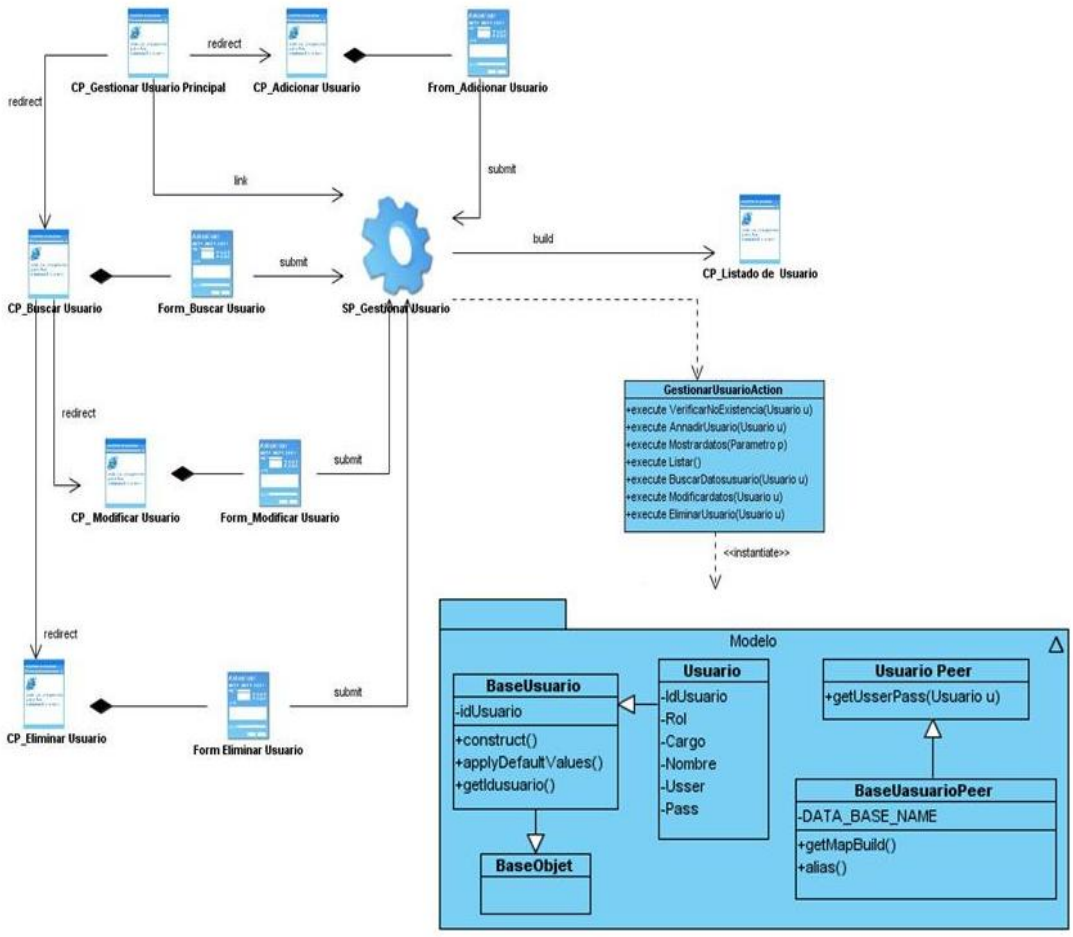


Figura 18 Diagrama de clases del diseño: Gestionar usuario.

- Diagrama de clase del diseño para el requisito funcional Gestionar ticket.

Análisis y diseño de la solución.

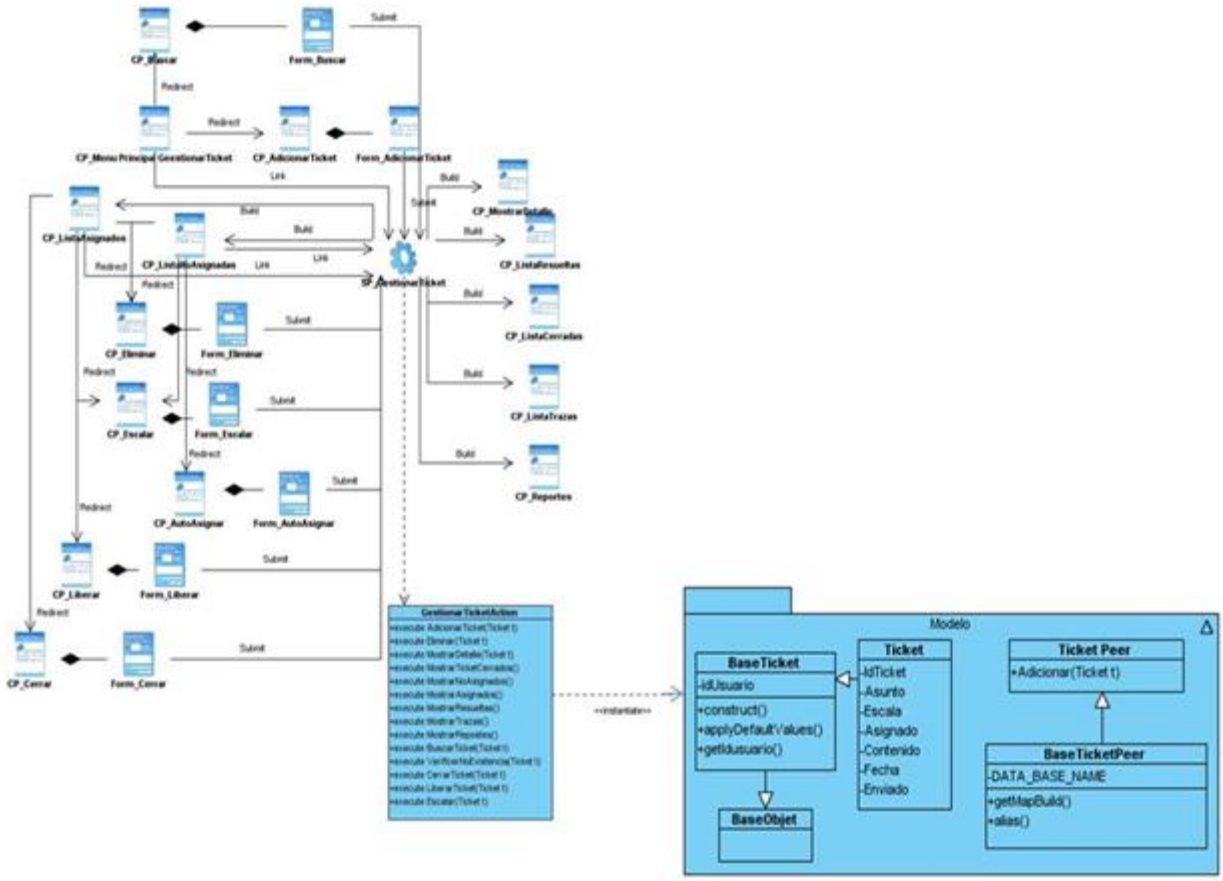


Figura 19 Diagrama de clases del diseño: Gestionar ticket.

Anexo 4: Diagrama de clases del diseño del requisito funcional Gestionar curso.

Anexo 5: Prototipo de interfaz de usuario del requisito funcional Gestionar curso.

3.3.1 Diagramas de secuencia del diseño.

Análisis y diseño de la solución.

- Diagrama de secuencia para el requisito funcional Gestionar usuario escenario Adicionar usuario.

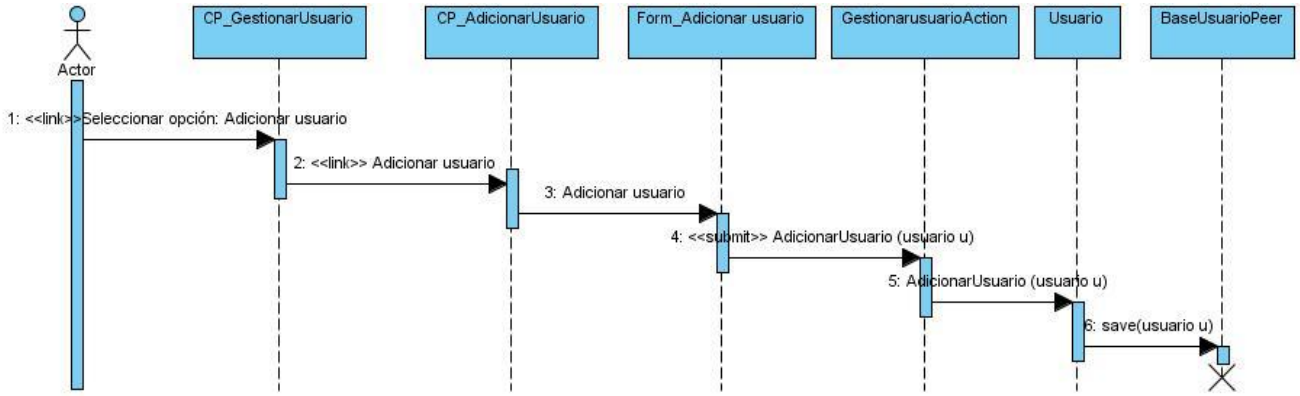


Figura 20 Diagrama de secuencia del diseño: Gestionar usuario escenario Adicionar usuario.

- Diagrama de secuencia para el requisito funcional Gestionar usuario escenario Eliminar usuario.

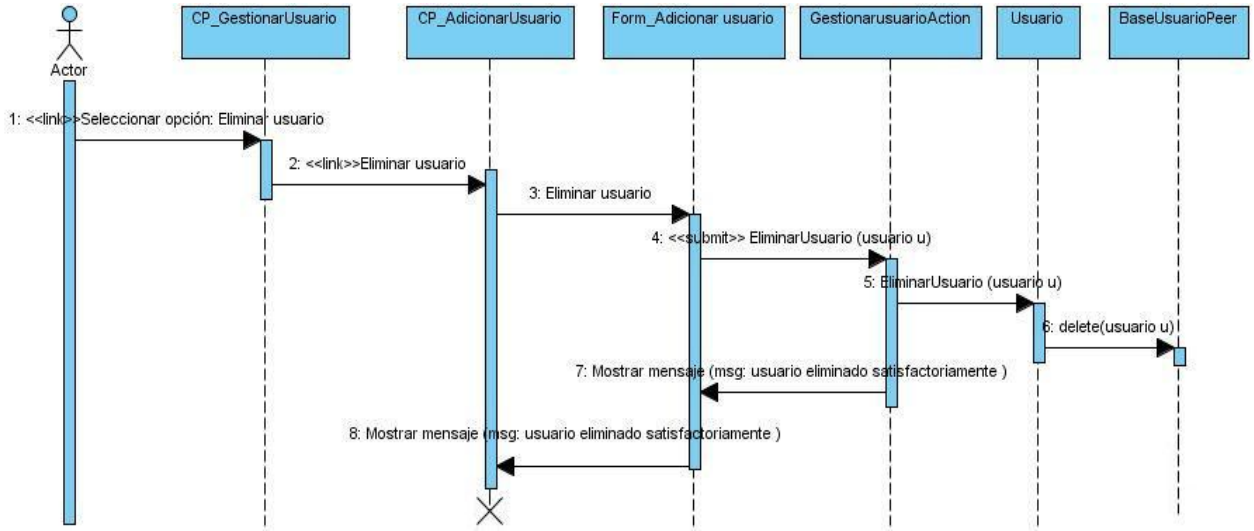


Figura 21 Diagrama de secuencia del diseño: Gestionar usuario escenario Eliminar usuario.

Análisis y diseño de la solución.

3.4 Diagrama de paquetes.

A continuación se muestra la estructura en paquetes de la aplicación teniendo en cuenta el patrón MVC. En el paquete View (Vistas) se representa la vista; la cual contiene las clases phtml que son las que interactúan directamente con los usuarios, el paquete Controller (Controlador) por su parte; agrupa las clases Action encargadas de manejar toda la lógica de negocio de la aplicación y por último el Model,(Modelo) que representa las clases necesarias para el trabajo con la Base de datos según lo que propone le framework Symfony.

- Diagrama de paquete.

Análisis y diseño de la solución.

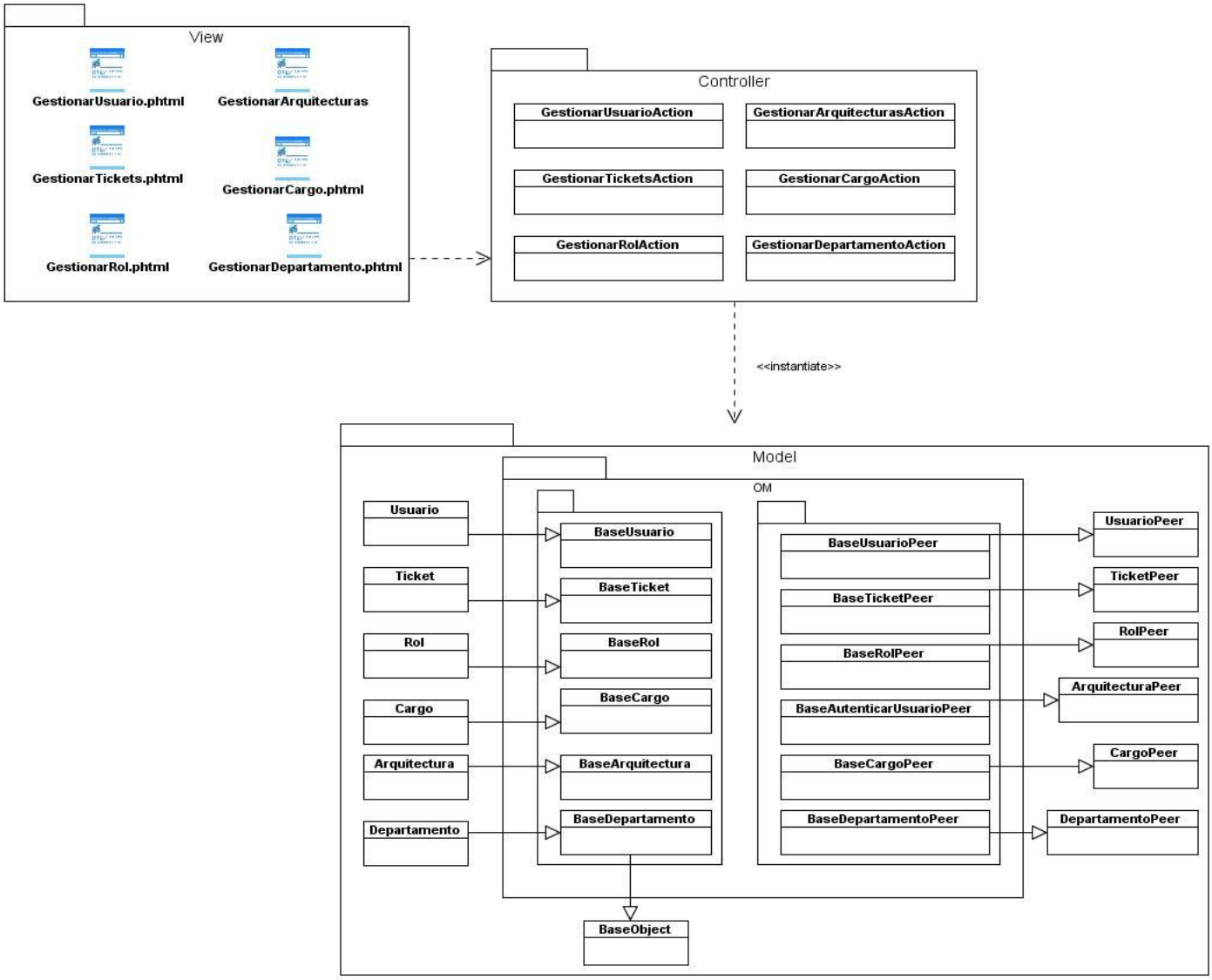


Figura 22 Diagrama de paquete.

A continuación se representa la estructura interna de las clases que se encuentran en el paquete Model para una mejor comprensión del mismo.

Análisis y diseño de la solución.

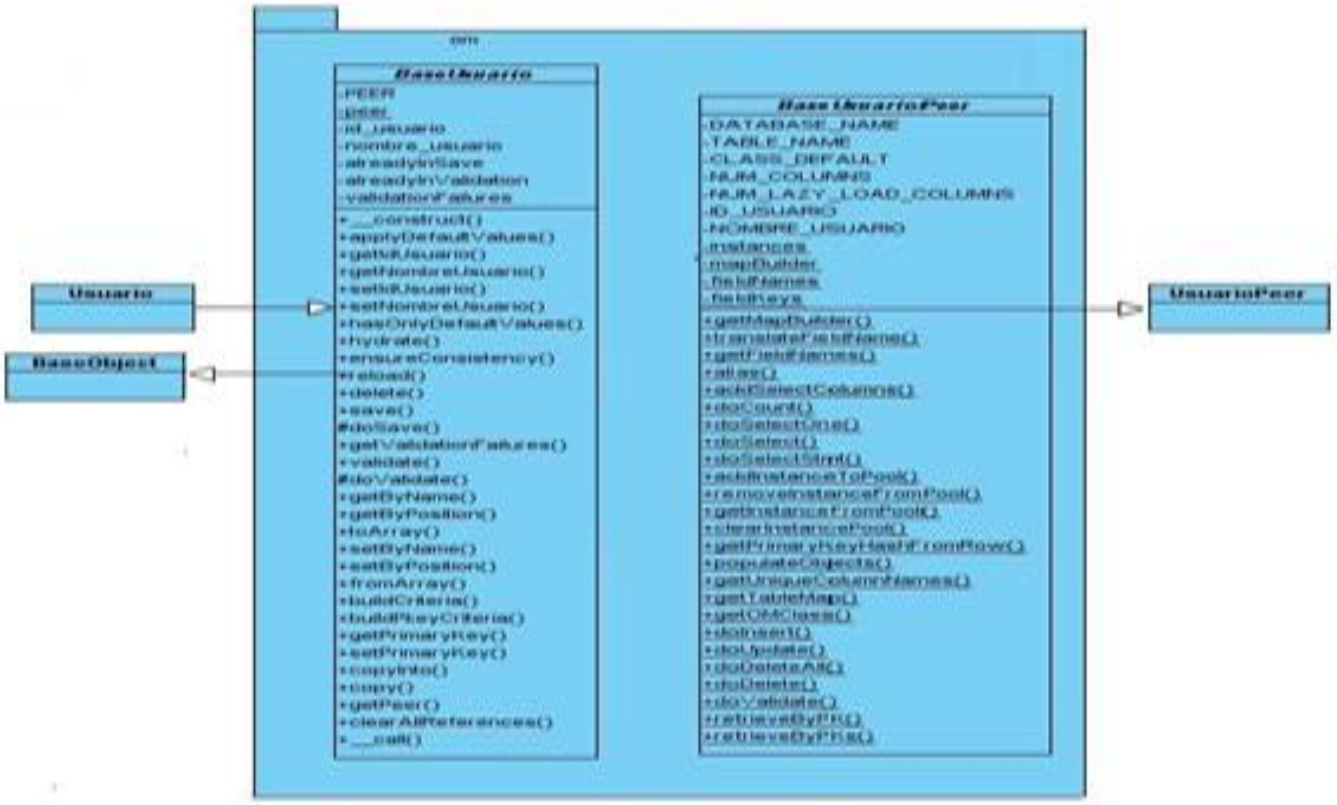


Figura 23 Estructura interna de clases de Symfony.

3.5 Modelo de datos.

Un modelo de datos es una colección de conceptos bien definidos matemáticamente que ayudan a expresar las propiedades estáticas y dinámicas de una aplicación con un uso de datos intensivo. El modelo de datos propuesto en la solución cuenta con un total de 57 tablas. Para su construcción se tuvo en cuenta la reducción a la mínima expresión de los campos nulos y la persistencia de campos resúmenes para agilizar recuperaciones frecuentes de algunos datos que son complejos de calcular.

Análisis y diseño de la solución.

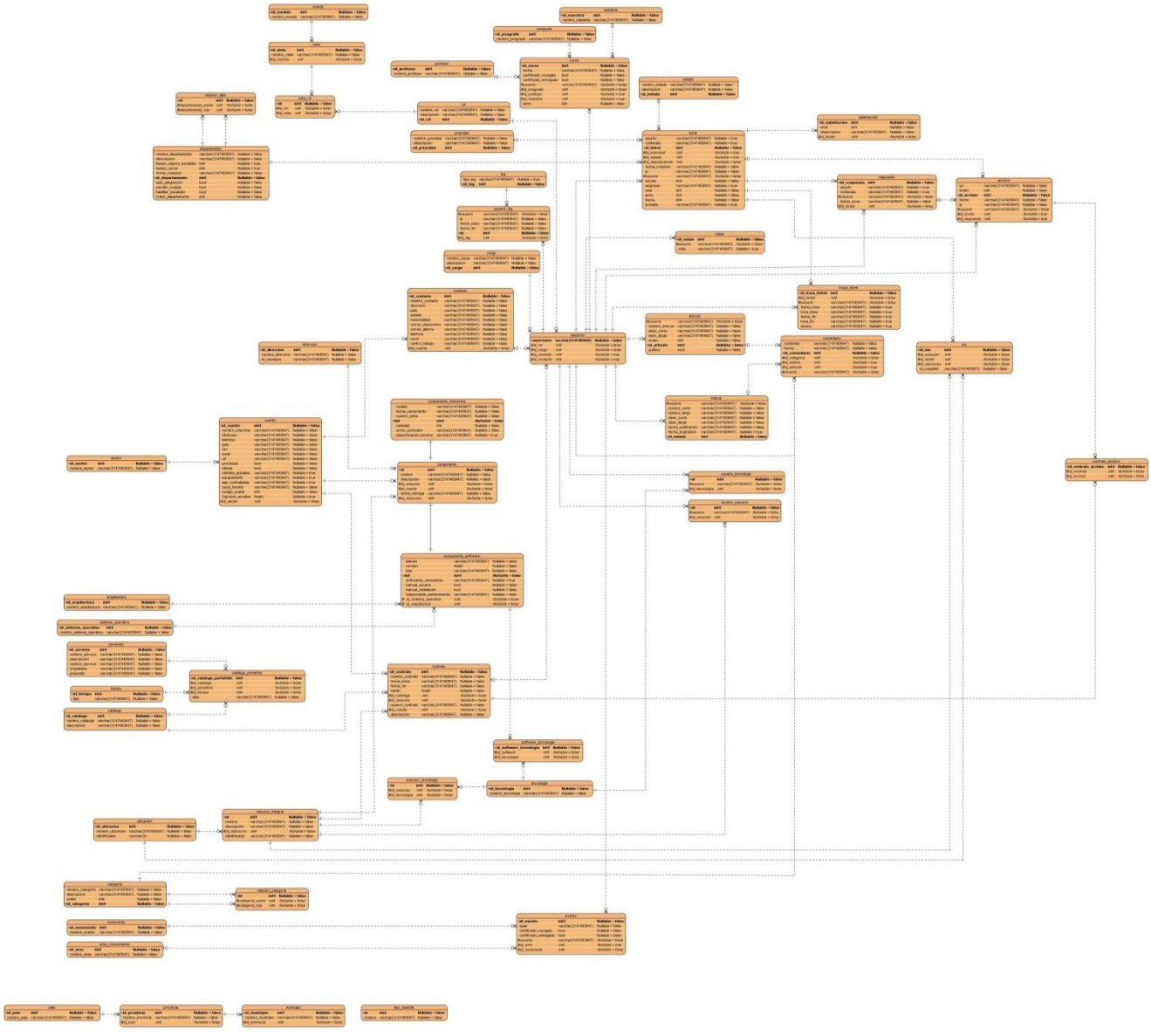


Figura 24 Modelo de datos.

Análisis y diseño de la solución.

3.6 Patrones utilizados en el diseño del sistema.

Patrones de asignación de responsabilidades.

El diseño fue elaborado siguiendo patrones basados en la experiencia, que de manera general constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. En este caso se emplearon los patrones GRASP (en ingles General Responsability Assignment Software Patterns), los que describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Los patrones GRASP que se utilizaron son los siguientes:

Alta cohesión: Este patrón fue utilizado en el diseño de la aplicación de manera general; donde se agruparon las clases en dependencia de los requerimientos a los que se les debía dar respuesta, según la premisa de que cada clase debe implementar las operaciones que estén sobre la misma área funcional.

Bajo acoplamiento: En el modelo de datos se definieron un conjunto de clases persistentes, entre las cuales se establecieron las relaciones necesarias de manera que fueran más independientes y reutilizables para reducir el impacto de los cambios y acrecentar la oportunidad de una mayor productividad.

Controlador: Las clases controladoras definidas: AutenticarUsuarioAction, GestionarUsuarioAction, GestionarTicketAction, son un ejemplo de la aplicación de este patrón, las mismas tendrán a cargo la responsabilidad de manejar los eventos asociados a ellas dentro de la aplicación.

Patrón arquitectónico Modelo Vista Controlador.

Este patrón separa en tres componentes distintos la interfaz de usuario, la lógica de negocio y los datos persistentes. En los diagramas de clases presentados con anterioridad se representa el contenido

dinámico sobre el cual el usuario puede realizar operaciones, la vista, mediante la página phtml, el controlador a través de las clases action, y el modelo como las clases de dominio, evidenciándose de esta manera la utilización de este patrón, potenciándose así la flexibilidad y la adaptabilidad a futuros cambios.

3.7 Validación del diseño.

A continuación se validará el diseño propuesto para analizar en qué medida constituye una adecuada entrada a las posteriores fases de desarrollo de la solución.

3.7.1 Métricas para la validación del diseño.

La aplicación de métricas al diseño de un producto de software constituye un elemento fundamental a la hora de evaluar la calidad del mismo. Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software e incluyen entre otras medidas la cohesión, acoplamiento y complejidad del módulo, medidas que pueden ayudar al desarrollador de software a juzgar la calidad de un diseño a nivel de los componentes. (13)

Seguidamente se hará una evaluación del diseño propuesto para la solución donde se tienen en cuenta los atributos y la métrica de calidad que a continuación se presentan.

Atributos de calidad que se abarcan:

Responsabilidad: Responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta.

Complejidad de implementación: Grado de dificultad que tiene implementar un diseño de clases determinado.

Análisis y diseño de la solución.

Reutilización: Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.

Métrica de calidad:

Tamaño Operacional de Clase (siglas: TOC): Se refiere al número de métodos pertenecientes a una clase. Está determinada por los atributos: Responsabilidad, Complejidad de implementación y la Reutilización, existiendo una relación directa con los dos primeros e inversa con el último antes mencionado.

Instrumento de medición de la métrica Tamaño operacional de clase (TOC).

	Categoría	Criterio
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Complejidad implementación	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	<= Prom.

Figura **25 Instrumento de medición de la métrica Tamaño operacional de clase (TOC).**

Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad.

Análisis y diseño de la solución.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
GestionarTicket	21	Alta	Alta	Baja
Gestionar Usuario	14	Baja	Baja	Alta
Autenticar Usuario	6	Baja	Baja	Alta
Gestionar Roles	12	Baja	Baja	Alta
Gestionar Cargos	12	Baja	Baja	Alta
Gestionar Estados	11	Baja	Baja	Alta
Gestionar Ubicaciones.	12	Baja	Baja	Alta
Gestionar Tecnología	10	Baja	Baja	Alta
Gestionar Tiempos	10	Baja	Baja	Alta
Gestionar Sector	10	Baja	Baja	Alta
Gestionar Arquitecturas	10	Baja	Baja	Alta
Gestionar Sistemas Operativos	12	Baja	Baja	Alta
Gestionar Prioridad	12	Baja	Baja	Alta
Gestionar Cursos	17	Media	Media	Media
Gestionar Contactos	14	Baja	Baja	Alta
Gestionar Departamento	14	Baja	Baja	Alta
Gestionar Componente de Software	21	Media	Media	Media
Gestionar Componente de hardware	20	Media	Media	Media
Gestionar Solución Integral	14	Baja	Baja	Alta
Gestionar Contratos	15	Baja	Baja	Alta
Gestionar Catálogos de Servicios	13	Baja	Baja	Alta
Gestionar Cuentas.	20	Alta	Alta	Baja
Gestionar Servicios	7	Baja	Baja	Alta
Subir documentos	3	Baja	Baja	Alta
Gestionar Noticias	14	Baja	Baja	Alta
Gestionar Artículos	18	Media	Media	Media
Gestionar Categorías	16	Baja	Baja	Alta

Figura 26 Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad.

Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Análisis y diseño de la solución.

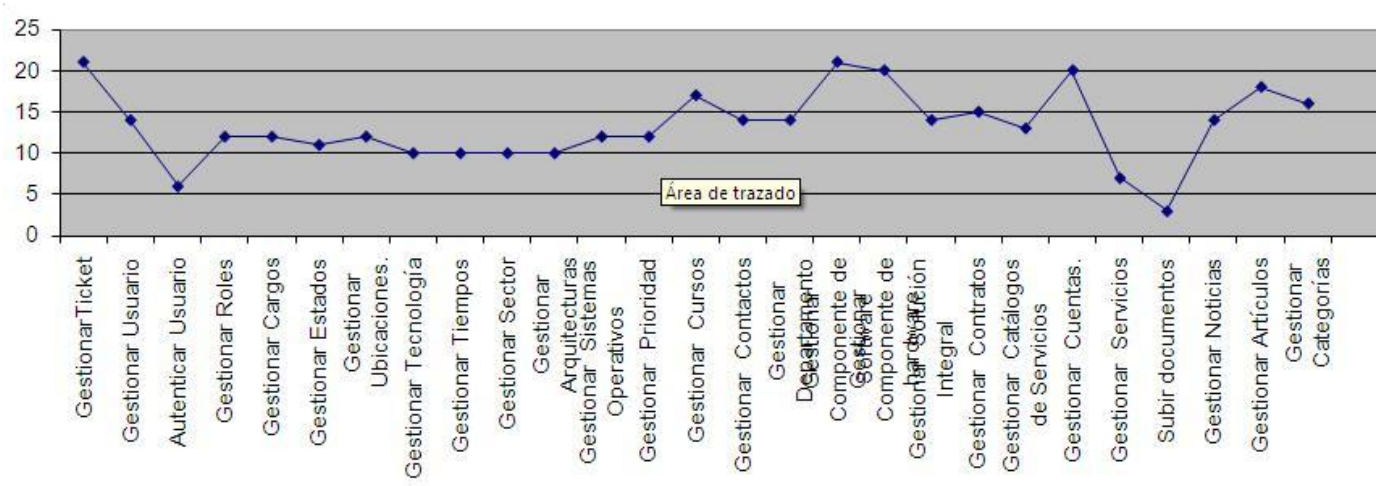
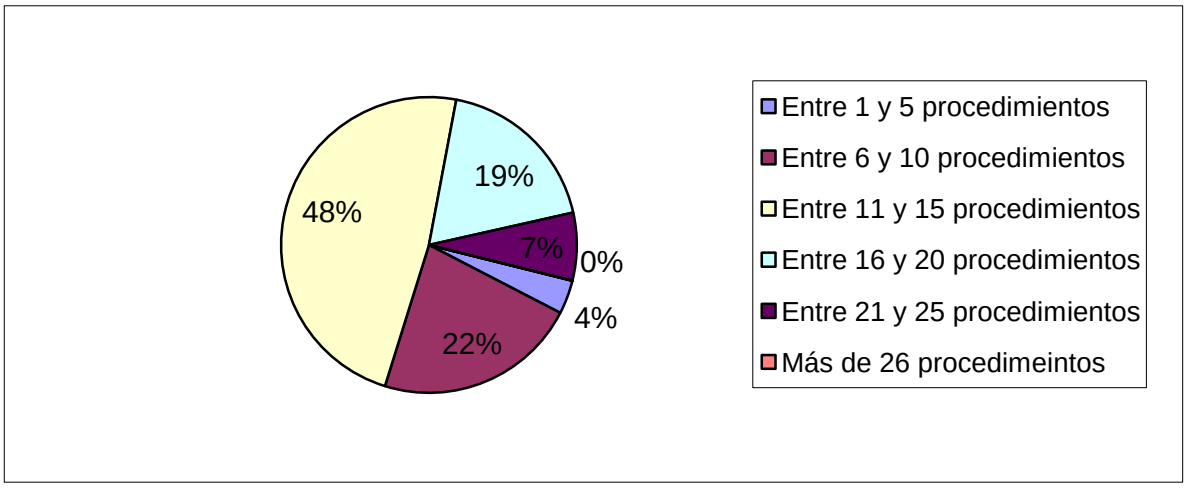


Figura 27 Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos:



Análisis y diseño de la solución.

Figura **28 Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.**

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad:

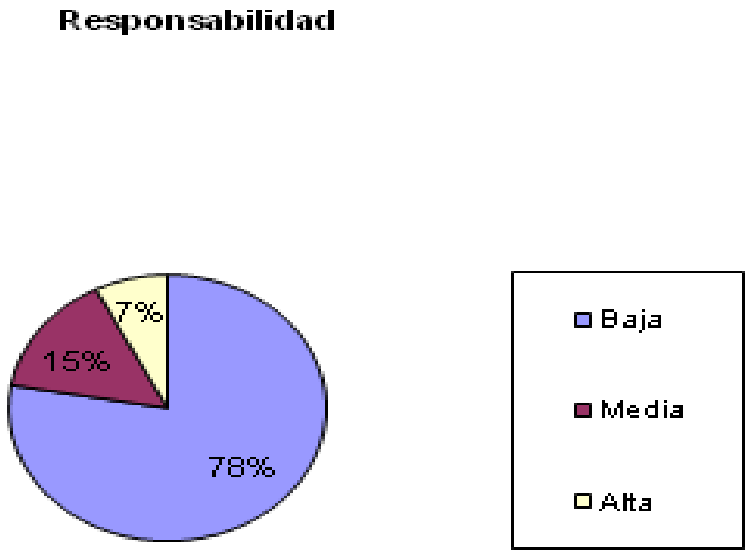


Figura 29 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación:

Análisis y diseño de la solución.

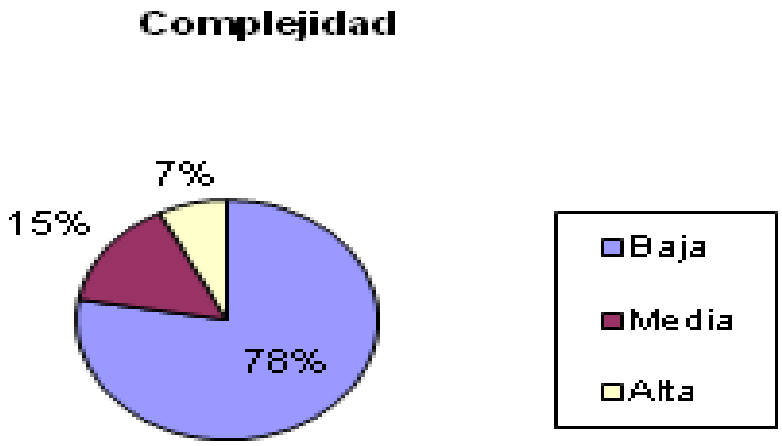


Figura 30 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización:

Análisis y diseño de la solución.

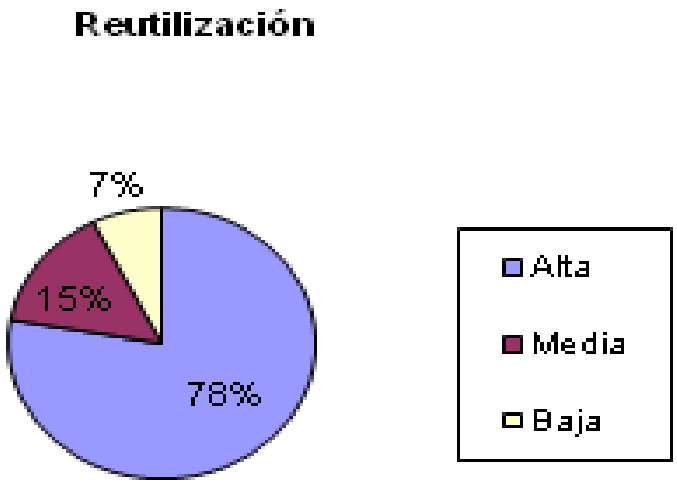


Figura 31 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica TOC, se puede concluir que el diseño de la solución propuesto está entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases (74%) posee menos cantidad de operaciones que la media registrada en las mediciones.

Los atributos de calidad se encuentran en un nivel satisfactorio, en el 78% de las clases; de manera que se puede observar cómo se fomenta la Reutilización (elemento clave en el proceso de desarrollo de software) y cómo están reducidas en menor grado la Responsabilidad y la Complejidad de implementación.

3.8 Conclusiones del capítulo.

Análisis y diseño de la solución.

Mediante el diseño de la solución realizado en el capítulo que recién concluye, donde su principal objetivo fue elaborar el Modelo de diseño, validar el mismo y definir el Modelo de datos correspondiente, queda presentada la propuesta del sistema permitiendo dar paso a la fase posterior dentro del proceso de desarrollo, en este caso; implementación.

Conclusiones generales

Una vez terminado el presente trabajo de diploma se puede concluir que se desarrollaron todas las tareas a fin de cumplir los objetivos propuestos, para esto:

- Se analizaron ventajas y deficiencias de sistemas informáticos en el mundo vinculados a la gestión de incidencias; evidenciándose de esta manera la no existencia de una solución informática capaz de ejecutar las funcionalidades referentes a la gestión de incidencias tecnológicas, que se adapte a las necesidades de la universidad.
- Se realizó el modelado de los procesos de negocio, con el objetivo de lograr un mayor entendimiento y comprensión del proceso de gestión de incidencias.
- Con cada especificación de requisito se obtuvo un prototipo de interfaz de usuario, sirviendo de guía para toda la implementación que se llevará a cabo.
- La validación de los requisitos identificados a través de las diferentes técnicas demuestra que cuentan con la claridad y calidad que necesitan para su informatización.
- La aplicación de las métricas de calidad para la especificación de requisitos arrojaron resultados satisfactorios para cada una de las características evaluadas.
- Se realizó el diseño del sistema utilizando patrones obteniendo de esta manera una definición más adecuada para que el futuro sistema pueda tener un buen funcionamiento.
- Se evaluó el diseño del sistema aplicando métricas que demostraron que el diseño de clases no presenta dificultades.

La solución propuesta es novedosa, su importancia radica en la presentación del análisis y diseño de los procesos de gestión de incidencia como base para el desarrollo de un único sistema capaz de realizar dichas operaciones, para un mayor control de los recursos tecnológicos.

Conclusiones generales.

Recomendaciones.

Recomendaciones

Al concluir el presente trabajo de diploma, considerando cumplidos los objetivos trazados en el mismo, se recomienda:

- Seguir mejorando el modelado de dominio y la captura de requisitos atendiendo a nuevos procesos que puedan surgir en el futuro.
- Continuar el estudio del tema de las incidencias con el objetivo de incluir nuevas funcionalidades en el proceso de gestión de incidencias.
- Realizar la implementación del sistema propuesto.
- Realizar el despliegue del sistema propuesto.

Referencias bibliográficas

1. addlink. [Online] http://www.addlink.es/productos.asp?pid=542..

2. itil. [Online] http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_incidentes/introduccion_objetivos_gestion_de_incidentes/introduccion_objetivos_gestion_de_incidentes.php.

3. Gestión de Incidencias. [Online] http://www.gobiernotic.es/2006/07/servicedesk-o-gestin-de-incidencias.html.

4. Xperta. [Online] http://www.xperta.es/upload/web/parrafos/00572/docs/xperta_w6588.pdf.

5. CADdySpain. [Online] http://www.caddyspain.com/noticias/man_sat.htm.

6. KmKey. [Online] http://www.kmkey.com/productos/software_help_desk.

7. **Merconchini, David Grau.** Juventud Rebelde. [Online] http://www.juventudrebelde.cu/cuba/software-libre-ii-una-estrategia-decisiva-de-desarrollo/.

8. Juventud Rebelde. [Online] http://www.juventudrebelde.cu/cuba/software-libre-ii-una-estrategia-decisiva-de-desarrollo/..

9. **Soporte, Grupo de Arquitectura del Centro de.** *Documento de Arquitectura del Crento de Soporte.*

10. Arquitectura y Patrones de diseño. Colectivo de autores. 2008-2009.

11. **Soporte, Grupo de Arquitectura del Centro de.** *Documento de Arquitectura del Centro de Soporte.*

12. **software, Departamento de ingeniería de.** *Conferencias de ingeniería de requisitos.* 2008.

13. **Fernández González, Mairelys.** *44- Pérez Hernández, Yorji. 2008. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.* 2010.

14. **Larman, Craig.** *UML y Patrones.* 1999.

15. *Arquitectura y Patrones de diseño.* **Colectivo de autores.** 2008-2009.

Referencias bibliográficas.

16. Consultoría y Soluciones para la Gestión de Servicios TI. http://www.overti.es/procesos-itsm/gestion-incidencias-itil-v3.aspx

17. **Angel Álvarez, Miguel.** DesarrolloWeb. [En línea] http://www.desarrolloweb.com/articulos/25.php.

18. **Paradigm, Visual**. visual-paradigm.com/product/vpuml/. visual-paradigm.com. [Online] Visual Paradigm,2008.http://www.visual-paradigm.com/product/vpuml/.

19. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El lenguaje unificado de modelado.*

20. **apache.** apache.com. *apache.com.* [Online] http://www.apache.com.

Anexos.

Glosario de términos

Gestión de Incidencias: Cubre todo tipo de incidencias, ya sean fallos, faltas o dificultades planteadas por los usuarios, o el personal técnico. (Solucionar las incidencias).

Incidencia: Es un evento que no forma parte de las operaciones de un sistema, ocasionando una interrupción en el servicio o perdida de la calidad del mismo.

Sistema gestor de base de datos: Es el software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez.

SQL (Structured Query Language): Conjunto estándar de comandos para gestionar bases de datos relacionales por sus mismas características relacionales.

UML (Unified Modeling Language): Lenguaje gráfico que brinda un vocabulario y reglas para especificar, construir, visualizar y documentar los artefactos de un sistema utilizando el enfoque orientado a objetos.