Universidad de las Ciencias Informáticas
Facultad 3

Título: Análisis y Diseño de una plataforma para brindar soporte al proceso de producción de software.

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

Autores

Anisbert Suárez Batista

Ginisleisy Morales Gómez

Tutor

Ing. Dairo Reyes Rodríguez

Asesores

MsC. José Raúl Rodríguez Galera

Ing. Dayana Caridad Tejera Hernández

Ciudad de la Habana, Junio del 2008

"Podrán morir las personas, pero jamás sus ideas."

Ernesto Che Guevara.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Anisbert Suárez Batista
Ginisleisy Morales Gómez

Ing. Dairo Reyes Rodríguez

_____ _____

Firma de los Autores

_____

Firma del Tutor

DATOS DE CONTACTO

Tutor: Ing. Dairo Reyes Rodríguez

Correo Electrónico: dreyes@uci.cu

Asesor: MsC. Jose Raúl Rodríguez Galera

Correo Electrónico: rgalera@uci.cu

Asesor: Dayana de la Caridad Tejera Hernández

Correo Electrónico: dtejera@uci.cu

Autora: Anisbert Suárez Batista

Correo Electrónico: abatista@estudiantes.uci.cu

Autora: Ginisleisy Morales Gómez

Correo Electrónico: gmorales@estudiantes.uci.cu

AGRADECIMIENTOS

A mi mamita por todo el amor que me da, por sus tantos sacrificios para que yo llegara hasta aquí, este logro no es mío, es nuestro. A partir de este momento, nuestras vidas cambian, mamita. Yo seré más independiente y tu ya te darás cuenta que he crecido, soy la mujer que siempre quisiste que fuera y hoy quiero ser tu orgullo. A mi papito, por amarme tanto, por protegerme siempre y ser sus ojitos. Se que hoy estás orgulloso de mi, y eso me complace más que nada. Siempre he luchado porque nunca se te quite el brillo de los ojos cuando hablas de mí a tus amigos y les cuentas de mis logros y de la niña que tienes. A mi hermano, al que le debo muchos te quiero, por ser la alegría de mi casa y protegerme siempre. A mi tía Xiomara por quererme tanto. Los años de mi vida no alcanzarán para retribuirte todo lo que has hecho por mí y los míos. Eres la persona que siempre admiré desde niña y quise ser como ella. A mis abuelos, por ser tan lindos y quererme tanto. A mi tía Elena por ser mi ejemplo a seguir, por incitarme a mi superación profesional y darme tanto cariño. A todos los tíos y primos que siempre me han apoyado desde la distancia.

A mis viejas amistades de la Las Tunas, Yanara, Katia, Fabián, Karen y Yadira por nunca olvidarme y acompañarme siempre en las risas y los llantos. Los quiero mucho. A mis queridas mixitas: Ginita, Llilian, Olguita y Mariam, por toda la ayuda que me han dado y ser tan unidas. Espero logremos seguir juntas por mucho más tiempo. ¡Qué quinteto de ingenieras! A Raúl, por ser único para nosotras. A mis hermanitos Yordan e Ibrael, por estar conmigo siempre que los he necesitado, por ser tan especiales en mi vida y demostrarle al mundo que la amistad es lo principal. A mi grupo 3301. Nunca los olvidaré. Mis mejores recuerdos universitarios quedan entre ustedes.

A esa persona que me enseñó que las lágrimas son nuestros tesoros y que hay que ser fuertes y seguros en la vida.

A mis amistades Olga, Héctor, Flor, Fernando y Beba. Qué tanto me han ayudado en la Universidad y me han tratado como una familia más. A Luis Enrique por apoyarnos tanto y confiar en nosotros mientras estuvo presente, esta era tu idea. A Dayana por ayudarnos y siempre estar dispuesta, casi te vuelvo loca. A nuestro actual tutor Dairo, por ayudarnos en todo lo que le fue posible.

A la Revolución por crear este sueño que es la Universidad de las Ciencias Informáticas y darme la oportunidad de ser parte de ella. A mi Fidel por ser mi guía y ejemplo de por vida.

Anisbert Suárez Batista

Agradecimientos

A mis padres, los dos, que a pesar de estar separados siempre han estado juntos para cuidar de mí. No saben cuán orgullosa me siento de tenerlos. Ustedes son la base de mi existencia y espero recompensarlos por todo lo que han hecho, que no ha sido poco. Y gracias por confiar en mí, trataré con todas mis fuerzas de no decepcionarlos. A mi hermana, que siempre ha sido guía y capitán de mi vida. Tú eres todo lo que yo tengo, Manita. Y siempre voy a estar ahí para ti. A mis abuelos, que aunque ya ninguno esté siempre me dieron el honor de decir: Mis abuelos son lo mejor del mundo. Y todo el cariño que me dieron nunca lo voy a olvidar. A todos mis tíos y mis tías, y primos y primas, por ser esa gran familia que siempre he admirado. Me considero dichosa por formar parte de ella.

A mis antiguas amistades, esas que han quedado después de mis primeros pasos en la vida, a ustedes les debo la comprensión de que la amistad sincera existe. ¡Gracias! A mis amistades de ahora: Ani, Olgui, Mariam y Llilian, que luego de 5 años llegamos a unirnos como Voltus V. A ustedes les debo muchos momentos de alegrías que jamás olvidaré. A Raúl, por ser el recipiente de nuestros problemas y ese Banco Nacional que él mismo dice, me sacaste de muchos apuros. Yo siempre te llamé: Amigo, y me siento halagada de que me llames igual. A mi hermanito Ibrael, que en muy poco tiempo me hizo quererle. ¡Gracias por esas vacaciones maravillosas!

A mi grupo de siempre, ese con el que comencé mis primeros años de Universidad. Desde el primer día me sentí satisfecha por compartir con ustedes las primeras experiencias, los primeros tropiezos aquí en la UCI. Sé que las cosas han cambiado, pero yo aún recuerdo a mi grupo.

A nuestro tutor, que en la medida de lo posible nos brindó su ayuda. A Luis Enrique, por hacerle brillar los ojitos a Ani con su idea y darle ánimos para seguirla.

A los padres de Olga, que han sido nuestro apoyo y sostén en multitud de ocasiones. ¡Ustedes son maravillosos!

A la Revolución por darme la oportunidad de escoger lo que quiero ser, siempre he estado muy orgullosa de ser cubana. ¡Muchas Gracias!

Ginisleisy Morales Gómez

DEDICATORIA

A mis padres, a mi tiita, a mi Virgen de la Caridad, a mis amigos, a mi trova…

Anisbert

A mis padres, por confiar en mí.

A mi hermana, por estar siempre a mis espaldas.

Ginisleisy

RESUMEN

El proceso de desarrollo de software en la Universidad de las Ciencias Informáticas (UCI) se ha desarrollado vertiginosamente a partir de su creación. El mismo se desempeña dentro de diversos grupos de proyectos en los cuales sus integrantes generan un conocimiento diario en las actividades que realizan. Pero, a pesar de que este conocimiento generado durante el proceso de desarrollo de software existe y tiene un valor primordial, no hay una forma definida de almacenarlo y gestionarlo para un soporte posterior al proceso productivo, ya sea para beneficios de los demás integrantes del equipo u otro proyecto dentro de la Universidad.

Con la ayuda de un sistema que permita gestionar el conocimiento generado en los proyectos productivos y almacenarlos en una base de conocimiento, los integrantes de estos proyectos podrían hacer uso de una información ya registrada, clasificada y fácil de acceder. De esta forma se estaría contribuyendo a la reutilización, socialización de la información y trabajo colaborativo.

Con el presente trabajo se garantizan las bases para dicho sistema a través del análisis y diseño del mismo, el cual permite un fácil entendimiento entre futuros desarrolladores y usuarios.

Mediante el flujo de trabajo de análisis y diseño se obtienen artefactos entendibles que viabilizan a los implementadores la automatización de una plataforma que sea capaz de brindar soporte al proceso productivo de software.

PALABRAS CLAVES

Soporte, Aplicación web, Base de conocimiento, Gestión de la información, Análisis, Diseño

TABLA DE CONTENIDOS

INTRODUCCIÓN

En la actualidad, la sociedad se ha desarrollado en el mundo de las tecnologías vertiginosamente. Cuba, en su empeño por alcanzar un nivel mundial en el marco de la informática y a pesar de su situación económica, ha tratado de insertarse en el nuevo mundo tecnológico. Gracias al aprovechamiento del considerable capital humano disponible, la industria cubana del software ha sido llamada a transformarse en una significativa fuente de ingresos nacionales. El sistema de empresas cubanas vinculadas a la rama, junto a la Universidad de las Ciencias Informáticas, juegan un papel cimero en el desarrollo de esta industria y en la materialización de los proyectos asociados al programa cubano de informatización.

La gran credibilidad con que cuenta el país en sectores como el deporte, la salud y la educación ha influido en la creciente promoción de la industria cubana del software en el ámbito internacional. Partiendo de que estos sectores satisfacen sus necesidades dado la producción sostenida de software de alta calidad en prestaciones, imagen y soporte, cada día aumenta más la cantidad de interesados en el software cubano.

La UCI se propone ser la vanguardia del desarrollo de las empresas de software en Cuba, posibilitar la informatización de todos los sectores de la sociedad, regir y propiciar un avance tecnológico de la industria del software nacional, hasta que se convierta en un renglón fundamental de la economía y posibilite insertarse en el mercado internacional. Es un programa de la Revolución que garantiza la formación de personal como Ingeniero en Ciencias Informáticas que acabará con la crisis de especialistas en esta rama. (Casañola Trujillo, 2006)

A lo largo del proceso productivo de software el equipo de desarrollo se enfrenta constantemente a obstáculos o problemáticas que comprometen la productividad, eficiencia, calidad y tiempo de desarrollo de un producto. Los programadores son los que más afectados se ven con respecto a esto debido al nivel de complejidad y dificultad de su trabajo. Durante el proceso de implementación surgen dudas, errores y situaciones complejas, que los programadores tratan de solucionar instantáneamente, aunque estas traigan una serie de problemas consigo:

Preguntar a algún compañero de labor, que tiene más experiencia y que probablemente sabrá solucionarlo es algo muy común entre profesionales, pero ciertamente desconcentra al camarada y retrasa su trabajo, además de crear una dependencia de conocimiento de terceras personas cuando se puede llegar a la solución por medios propios.

Buscar en Internet o estudiar tutoriales es muy efectivo en algunas ocasiones, pero en otras no, debido a que se emplea mucho tiempo de búsqueda sin estar seguros de un resultado satisfactorio y con riesgos de generarse más dudas sobre el tema en cuestión. Otro punto importante es que es un recurso que cuesta y no ciertamente barato, por lo que algunas empresas no cuentan con la posibilidad de obtenerlo.

Consultar la ayuda del entorno de desarrollo integrado (IDE) parece ser la mejor opción que tiene el implementador cuando existe algún problema que solucionar, mas estas muestran información de manera general y en ocasiones para casos específicos no son útiles.

Independientemente de la forma en que se hayan encontrado las repuestas, el desarrollador genera, asimila, aplica y enriquece un conocimiento o experiencia que puede ser aprovechada por el mismo individuo o por otras personas en situaciones posteriores.

Sin embargo, estos conocimientos que se generan a diario tienden a perderse, desechando así la posible socialización de la nueva información con el resto del equipo de desarrollo. Los programadores creen una pérdida de tiempo el documentar las ideas o experiencias generadas en tiempo real de codificación. Les resulta tedioso y un retraso en su planificación laboral, aunque en un futuro, frecuentemente cercano, las pueden volver a necesitar en su equipo de desarrollo.

En La Declaración de Principios de la Cumbre Mundial sobre la sociedad de la información Ginebra-2003 se plantea: “La capacidad universal de acceder y contribuir a la información, las ideas y el conocimiento es un elemento indispensable en una sociedad de la información integradora.”

Por ello, las organizaciones deben de ser capaces de preservar el conocimiento, de identificarlo, capturarlo, recuperarlo y compartirlo, es decir de gestionarlo. Si los profesionales lo llevaran acabo de manera individual transformarían los conocimientos de simples datos e ideas a todo un activo intelectual de esencial beneficio para la organización y la sociedad. Gestionar el conocimiento generado en el proceso de desarrollo de software es una de las mejores y más necesarias prácticas del profesional.

La UCI ha ganado mucho en cuanto al concepto de soporte se trata, se da soporte a la tecnología, a las comunicaciones, a la televisión, a las construcciones y soporte a soluciones de software. Esta última consiste en proporcionarle mantenimiento a las soluciones de software que están implantadas y es la más aplicada mundialmente y en la universidad. Por ejemplo, la UCI cuenta con un conjunto de aplicaciones de forma virgen implantadas en el nodo central que son atendidas por el grupo de

soporte, mientras que también la mayoría de las empresas brindan servicio de soporte a la aplicación una vez desplegados los sistemas como parte del contrato. (Saborit Ramírez, 2007)

Los equipos de desarrollo no disponen de soporte en el proceso productivo que viabilice su labor. Existen estándares, metodologías y modos de trabajo establecidos, pero no un proceso definido que brinde soporte a los proyectos productivos en el proceso de desarrollo de software, dígase un apoyo integrado que brinde facilidades para realizar el trabajo con calidad. La existencia de una plataforma que posibilite documentar los conocimientos generados por los desarrolladores garantiza que se conserve información valiosa y reutilizable en otro instante, se logre una socialización de experiencias, se perpetúe y materialice el potencial de la organización, se ahorre tiempo al tener un acceso rápido y cómodo a los apuntes, y se gane en productividad, motivación a la innovación y desarrollo de habilidades personales.

Debido a la versatilidad y magnitud de tendencias de gestión de conocimientos en el proceso de producción de software, no existe una visión detallada de cómo se desarrolla el mismo, ni un convenio establecido de cómo gestionar los conocimientos.

Problema Científico de Investigación

De acuerdo a la situación problémica expuesta anteriormente se identifica el siguiente problema científico:

¿Cómo elaborar el análisis y el diseño de un sistema, que automatice la gestión del conocimiento generado en el proceso de producción de software, de manera que se logren artefactos lo suficientemente entendibles que garanticen una correcta implementación del sistema?

El **objeto de estudio** es la gestión del conocimiento en el proceso de producción de software.

El **objetivo** propuesto consiste en realizar el análisis y diseño de una plataforma que garantice la retroalimentación a partir de la gestión del conocimiento.

Se determinó como **campo de acción** el análisis y diseño de una plataforma para la automatización de la gestión del conocimiento.

Hipótesis: Con el análisis y diseño de la plataforma los programadores podrán desarrollar de manera eficiente una aplicación que garantice la retroalimentación a partir de la gestión del conocimiento y brinde soporte en línea al proceso de producción de software lo cual ahorraría tiempo de

implementación aumentando la productividad, la socialización, reutilización y retroalimentación de la información.

Para cumplir exitosamente la meta antes planteada se propusieron una serie de **tareas de investigación** a desarrollar:

- Realización de un estudio del estado del arte, a partir de literatura especializada, de conceptos como la gestión del conocimiento, las bases de conocimientos, el soporte en línea y tecnologías de desarrollo actuales.
- Identificación de las principales áreas que demandan la gestión del conocimiento en el proceso productivo.
- Definición de una política de soporte en línea para apoyar el proceso de desarrollo de software en la UCI.
- Análisis y diseño de una plataforma que permita la gestión del conocimiento generado por los desarrolladores en los proyectos productivos para brindar servicios de soporte en línea a los mismos.

Para llevar a cabo las tareas se emplearon los siguientes **métodos de investigación:**

Métodos teóricos:

- **Analítico – sintético:** Para arribar a conclusiones a partir de la documentación consultada para entender fenómenos relacionados con la gestión y socialización de la información.
- **Histórico – lógico:** Para conocer las tendencias actuales en cuanto a bases de conocimientos, servicios de soporte y uso de tecnologías de desarrollo.
- **Inductivo – deductivo:** A través de la tormenta de ideas desarrolladas por el equipo de trabajo se llegaron a deducciones valiosas que brindaron aportes a la definición de objetivos y tomas de decisiones.

Métodos empíricos:

- **Entrevista:** Para obtener información actualizada sobre las tendencias de servicios de soporte al proceso productivo y criterio especializado sobre la gestión del conocimiento.

Una plataforma como la propuesta anteriormente constituiría una herramienta muy útil para los profesionales de los equipos de desarrollo. Este medio para la gestión de la información y las

experiencias facilita en alta medida la consolidación del potencial profesional. Le hace más productivo su trabajo, más rápidas las consultas, más cercanas las soluciones. Sin contar que promueve el trabajo colaborativo y la extensión del conocimiento.

Estructuración del contenido con una breve explicación de sus partes.

El presente trabajo consta de cuatro capítulos de desarrollo, los cuales presentan las siguientes características:

Capítulo 1: En el desarrollo de este capítulo se hace un estudio del estado del arte de los diferentes conceptos vinculados a la solución propuesta, como por ejemplo soporte en línea, gestión de conocimiento y base de conocimiento. Se realiza un análisis de las metodologías, tecnologías candidatas, y roles fundamentales del flujo de trabajo análisis y diseño, así como de algunos patrones posibles a necesitar a lo largo del trabajo.

Capítulo 2: Este capítulo aborda las características específicas del sistema que se desea desarrollar. Particularmente contiene el modelo de dominio con sus conceptos fundamentales y relaciones que modelan el dominio de la solución, además de los requisitos tanto funcionales como no funcionales que condicionan la organización por casos de uso y sus respectivas especificaciones. El objetivo fundamental de este capítulo es garantizar la mayor comprensión, por parte de los desarrolladores y clientes, de las características del sistema a proponer.

Capítulo 3: Este capítulo comprende todo los artefactos resultantes del flujo de trabajo de análisis y diseño. Se presentan las realizaciones de casos de uso, con sus respectivos diagramas de clases, diagramas de interacción y las descripciones de las principales clases de diseño. Además del modelo Entidad-Relación para la modelación de la Base de Datos.

Capítulo 4: En este capítulo se realiza un análisis de los resultados obtenidos en los capítulos anteriores, a través del uso de métricas de calidad, aplicadas a los distintos artefactos resultantes del análisis y diseño. Las métricas a usar se refieren a las especificaciones de casos de uso, diagrama de casos de uso y diagramas de diseño. De esta forma se garantiza la calidad de los artefactos generados, que servirán de entrada para una futura implementación del sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

Debido al creciente desarrollo de la industria del software, muchas empresas han comenzado a proveer servicios de soporte como parte de los productos ofrecidos, con el propósito de garantizar una mejora continua de la calidad de sus servicios. Es a partir de aquí y gracias al desarrollo vertiginoso de Internet y los servicios web, que surge una nueva alternativa: el soporte en línea. En este capítulo se estarán abordando los temas asociados al soporte en línea, así como las ventajas que lo hacen atractivo para la comunidad de usuarios, se abarcará las tendencias actuales de la gestión de conocimiento y las bases de conocimiento por la repercusión que estas han tenido en el trabajo colaborativo y la socialización del conocimiento. Se presentará un análisis detallado de las herramientas de desarrollo, metodologías y roles asociados al análisis y el diseño de la plataforma propuesta.

1.1. Soporte

Según la Real Academia de la Lengua Española, el término soporte es definido como apoyo o sostén. La práctica lo define como la asistencia o ayuda que viabiliza un proceso o resuelve problemas. Informáticamente existe el llamado soporte técnico que consiste en la asistencia especializada ofrecida por una empresa de hardware o software para resolver los problemas que enfrentan sus clientes al utilizar un ordenador, un sistema operativo o un programa determinado.

La mayoría de las compañías informáticas que venden hardware o software brindan soporte técnico a los clientes que contratan el uso de sus licencias para los productos que comercializan. Entre los medios de solicitud que brindan están las vías telefónicas y el Internet mediante el correo electrónico y la asistencia en línea.

1.1.1. Soporte en línea

El soporte en línea es una modalidad muy difundida en el ciberespacio actual, dado por el continuo desarrollo tecnológico y la proliferación del comercio electrónico. Es muy productivo encontrar las soluciones de los problemas a la distancia de un clic o descargando un simple archivo. Las ventajas son muchas:

- **Disponibilidad**: los clientes podrán establecer comunicación con la empresa sin depender de horarios, pudiendo solicitarlo en el momento que estimen necesario.
- **Velocidad**: los accesos a los servicios no requieren colas de espera, ni tramitaciones especiales, por lo que los clientes obtienen lo que necesitan en el instante.
- **Comodidad**: a los clientes no les supone desempeñar esfuerzos adicionales para conseguir sus objetivos, los pueden llevar a cabo cómodamente sentados frente al ordenador.
- **Economía**: los clientes ahorrarán tiempo y dinero en las comunicaciones establecidas con la empresa.

Estas razones son las que hacen válido y atractivo el establecimiento de un servicio de soporte en línea al proceso productivo de software. A través de este y de una política de soporte, el desarrollo de aplicaciones puede ser aún mucho más eficiente y fructífero.

Esta investigación propone un soporte al trabajo de los desarrolladores, que a partir de sus propias gestiones de información, le podrán brindar ayuda al resto del equipo en cuanto a soluciones de problemáticas enfrentadas, resueltas y documentadas individualmente. Una aplicación que permita la documentación de experiencias a lo largo del tiempo constituirá toda una base de conocimiento para consultar. Además, si también se permite el intercambio de opiniones y consejos a través de foros, la descarga de software, la actualización en el acontecer tecnológico y otros tipos de prestaciones, el trabajo colaborativo y la socialización del conocimiento se verán estimulados.

Por otra parte una política de soporte, que posibilite establecer contratos con clientes necesitados de servicios de asesoría, entrenamientos y capacitación, provee mecanismos de formación y adiestramiento para los equipos de desarrollo. De esta forma el proyecto aseguraría la preparación de su personal a través de profesionales en el tema y dispondría de sus recursos humanos solo para superarse y producir.

Algunos ejemplos de empresas o instituciones que brindan soporte en línea son:

- **El Centro Nacional de Información de Ciencias Médicas de Cuba mediante el portal Infomed.**
  Este sitio está dirigido fundamentalmente a profesionales de la salud. Brinda servicios de documentación digital con libros, tutoriales y manuales sobre la especialidad. Posibilita búsquedas en bases de datos de interés para los médicos, por ejemplo la que contiene la información de todos los medicamentos aprobados para su uso en Cuba. Proporciona utilidades para conversión de extensiones. También permite la descarga de antivirus, drivers de hardware, herramientas de multimedia y software médicos.

- **Microsoft** *(Microsoft Corporation, 2008)*

  Esta empresa brinda servicios de soporte técnico como:

  Soporte Personal: Para personas que utilizan ordenadores en casa o trabajan desde su domicilio y precisan de soporte para los productos de escritorio y de consumo de Microsoft en un entorno independiente o en una red sencilla.

  ***Soporte Profesional*:** Para los que participan en el desarrollo, la implementación y la administración de las soluciones de software de Microsoft en entornos comerciales.

  ***Soporte Esencial*:** Para empresas que necesitan de acceso directo las 24 horas del día al equipo técnico de cuentas de Microsoft, que le proporciona soporte técnico para solución de problemas y servicio de soporte técnico proactivo.

- **Oracle Company** *(Oracle Company, 2008)*

  Oracle ofrece una lista de servicios de soportes en línea como son actualizaciones de software, soporte de producto, repositorio técnico, respuesta inmediata a consultas de soluciones e información necesaria para aprovechar la tecnología Oracle por parte de sus especialistas y clientes.

- **IBM Company** *(IBM Company, 2008)*

  La IBM también por su parte ofrece la posibilidad de descargas de drivers, patches, actualizaciones, soluciones a problemas, documentación etc.

1.1.2. Algunas herramientas utilizadas para brindar soporte en línea

- **SupportNow 3.1**

SupportNow 3.1 proporciona soporte en línea, salvando las barreras entre el simple acceso a los datos y las aplicaciones de e-business[1] basadas en la red. Es la herramienta interactiva de soporte visual de Netmanage. Reduce significativamente el tiempo de resolución del problema, con una interfaz intuitiva, acceso remoto y capacidades de procesamiento de color que permiten el soporte del monitor personal, acceder a la información sobre múltiples detalles de llamada, acceso seguro y el mantenimiento de máquinas remotas a partir de un entorno de soporte sencillo.

- **Helpdesk** (Soporte Bankoi S.L., 2007)

[1] e-business: Cualquier tipo de actividad empresarial realizada a través de las tecnologías de las comunicaciones.

Bankoi desarrolló una herramienta Helpdesk para gestionar las consultas de sus clientes y resolver tanto problemas técnicos como propiedades de sus productos. Brinda la opción de envío de correos al usuario cuando su duda ha sido respondida. Es personalizable tanto en el aspecto como en la funcionalidad. Cuenta con un sistema de preguntas frecuentes integrado, que incluye un sistema de seguimiento de las últimas consultas y las más vistas, para darse cuenta de cuáles son los problemas que más preocupan o hacen dudar a los usuarios. También incorpora un sistema de funcionamiento público o privado donde se puede escoger entre tener el centro de soporte abierto a todo el mundo o restringido de forma que los usuarios tengan que ser previamente validados por el administrador. Su administración es sencilla y el sistema seguro con encriptación de contraseñas.

- **Oracle MetaLink**

Oracle Metalink ofrece a los clientes de Soporte Oracle, sin cargo adicional, un acceso completo a soporte vía Internet que facilita el trabajo de sus divisiones de tecnología de la información. Es una aplicación web que brinda varias facilidades y servicios como información y soluciones técnicas, patches, bugs, repositorio de artículos técnicos, herramientas y pruebas de diagnóstico, foros interactivos, personalización de la página principal, acceso 24 x 7, poderoso motor de búsqueda y una base de conocimientos con más de 400 000 soluciones, todas desarrolladas por expertos Oracle, alrededor del mundo.

1.2. Gestión del Conocimiento

La gestión del conocimiento es un proceso mediante el cual se desarrolla, estructura y mantiene la información, con el objetivo de transformarla en un activo crítico y ponerla a disposición de una comunidad de usuarios, definida con la seguridad necesaria. Incluye el aprendizaje, la información, las aptitudes y la experiencia desarrollada durante la historia de la organización. (García Robles, 2001)

De esta forma, dicho proceso asegura constantemente el desarrollo y aplicación de todo tipo de conocimientos pertinentes en una organización, con la meta de mejorar su capacidad de resolución de problemas, optimizar el flujo de información e incentivar la interacción y cooperación entre las personas.

Su fin está muy estrechamente relacionado con conceptos como socialización de la información, reutilización de la información, retroalimentación de experiencias, transformación de la información en activo crítico y aprovechamiento del capital intelectual.

Gestionar el conocimiento generado en los procesos de desarrollo constituye una inapreciable práctica que transforma el razonamiento y experiencia del hombre en un activo intelectual. Como activo en fin, posibilita la persistencia del contenido y la reutilización continua del mismo por todo el equipo de trabajo, contribuyendo a la productividad, eficiencia y calidad de la labor.

Un sistema, que viabilice la gestión del conocimiento, sistematice y racionalice la información disponible en una organización en busca de un mayor rendimiento en el trabajo, impulsará a una cultura de cooperación entre trabajadores, a un buen clima de entendimiento, a la satisfacción del personal por el reconocimiento de sus ideas y a un mayor potencial cognoscitivo en la empresa.

En la actualidad el avance hacia una sociedad del conocimiento ha pasado de ser una meta a un proceso de optimización de logros. Se vive ya en una sociedad con capacidad para generar, apropiar y utilizar el conocimiento en pos de atender las necesidades actuales, convirtiendo la creación y trasferencia del conocimiento en herramienta propia para su mismo beneficio. Los productos hoy valen en el mercado no solo por la materia prima que los conforma sino por el valor agregado que constituye el conocimiento invertido.

Aunque el capital intelectual de la Universidad de las Ciencias Informáticas se nutre constantemente de los conocimientos y las experiencias adquiridos a lo largo del proceso de desarrollo de software, le falta mucho por consolidar y ganar en este sentido. Si las grandes magnitudes de información circuladas dentro de los proyectos productivos fueran debidamente gestionadas e interpretadas, todas ellas quedarían transformadas en conocimientos, ya sean explícitos o tácitos, que se incorporarían al proceso productivo y aumentarían el valor de la institución y sus productos.

1.3. Base de conocimiento.

1.3.1. ¿Qué son las bases de conocimientos?

Una base de conocimiento es una representación abstracta de un tema en un área, o un ambiente en particular, incluyendo los conceptos principales de interés en dicha área y las relaciones entre las entidades. (Vieyra, 2001)

Una Base de Conocimiento se puede ver como una Base de Datos (en lo adelante BD) organizada en procesadores de cualquier tipo. La información que guardan es una fuente de conocimiento cierto, válido y aprovechable. (Knowledge Master Corporation, 2007)

Ahora, una Base de Conocimiento no es lo mismo que una Base de Datos. Las bases de conocimientos representan una evolución lógica de los sistemas de base de datos tradicionales. El primer objetivo de una BD es, como su nombre lo indica, almacenar grandes cantidades de datos o información, de forma organizada, siguiendo un determinado esquema o modelo de datos que faciliten su almacenamiento, recuperación y modificación. Mientras que las bases de conocimientos son herramientas para buscar y hacer un mejor uso del conocimiento, información y experiencias que han sido capturadas, organizadas y almacenadas por individuos de una organización o comunidad y que se ponen a disposición de todos.

Las bases de conocimientos toman valor en la medida en que su información almacenada pasa de ser datos para convertirse en conocimientos explícitos. Generalmente no tienen un tamaño acotado, se documentan, enriquecen y refinan con el paso de tiempo y el surgimiento de nuevos contenidos. Un ejemplo son los temas referentes al proceso de desarrollo de software, para el cual sería óptimo disponer de una plataforma, como sistema gestor, que posibilite el proceso de documentación de una manera fácil e intuitiva para los profesionales. Un medio a través del cual los integrantes del equipo puedan documentar cada una de sus experiencias, las problemáticas afrontadas, las soluciones y publicaciones personales.

Hoy en día las tecnologías están siendo usadas a mayores escalas como soporte para el aprendizaje educacional y la socialización de la información.

Ejemplos:

- Base de conocimiento jurídico: www.Iustel.com.
- Base de conocimiento sobre arquitectura de software: www.epidataconsulting.com
- Base de conocimiento sobre antivirus: www.sophos.com

1.4. Metodologías de desarrollo

Una metodología de desarrollo es un medio de estandarización que cubre completamente el proceso de desarrollo de un software. Posibilita el movimiento en doble sentido entre las fases del ciclo de desarrollo para verificar trabajos realizados y corrección de los errores detectados. Provee un entendimiento efectivo entre los analistas, programadores, clientes, usuarios y gestores para una correcta toma de decisiones. Especifica claramente cuáles son las tareas a realizar y quiénes son los responsables de las mismas. Sin embargo, es necesario tener en cuenta que no es lo mismo hacer

uso de un modelo de estandarización para un sitio Web que para el desarrollo de una aplicación Desktop. Por lo que existe una gran diversificación de metodologías de desarrollo. Por eso es necesario hacer una buena elección de la metodología a utilizar en cada proceso que se desee desarrollar.

Las metodologías se dividen en dos categorías actualmente, metodologías tradicionales (pesadas) y metodologías ágiles (livianas). Las metodologías tradicionales se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Las metodologías ágiles son más factibles usarlas en proyectos pequeños que no requieran de tanta documentación. Estas últimas se centran más en que el software funcione que en la documentación exhaustiva, en los individuos y sus interacciones que en los procesos y herramientas, en la colaboración con el cliente que en la negociación de contratos, por solo mencionar algunos aspectos. Un ejemplo de metodología ágil es la metodología Programación Extrema (Extreme Programming, XP), la cual se explicará a continuación.

- **Extreme Programming, XP**

XP es una metodología ágil nacida en el verano de 1996 de la mano de Kent Beck cuando éste se incorpora a Chrysler Corporation para desarrollar una aplicación de nóminas, en donde las condiciones existentes conducían a este tipo de metodologías no convencionales que él propuso aplicar. La misma se encuentra centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. Le da prioridad a las tareas que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación.

XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y responsabilidad a la hora de afrontar los cambios. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Prácticas básicas de la programación extrema. (Marquetti, y otros, 1997)

1. **Equipo completo:** forman parte del equipo todas aquellas personas que tienen algo que ver con el proyecto.

2. **Planificación:** se planifica en qué orden se van a hacer las cosas y las mini-versiones.

3. **Test del cliente**: el cliente propone sus propias pruebas con el objetivo de evaluar las mini-versiones.

4. **Versiones pequeñas:** las versiones deben ser pequeñas y deben ofrecer algo útil al usuario final.

5. **Diseño simple:** mantener siempre sencillo el código.

6. **Pareja de programadores:** los programadores trabajan por parejas y éstas se intercambian con frecuencia.

7. **Desarrollo guiado por las pruebas automáticas:** se deben realizar programas de prueba automática.

8. **Mejora del diseño:** extraer funcionalidades comunes y eliminar líneas de código innecesarias.

9. **Integración continua:** las funcionalidades deben de recompilarse y probarse a medida que sean desarrolladas.

10. **El código es de todos:** cualquiera puede y debe tocar y conocer cualquier parte del código.

11. **Normas de codificación:** debe haber un estilo común de codificación.

12. **Metáforas:** buscar frases o nombres que definan cómo funcionan las distintas partes del programa.

13. **Ritmo sostenible:** se debe trabajar a un ritmo que se pueda mantener durante todo el proceso de desarrollo del software.

Características fundamentales de la programación extrema

- Desarrollo iterativo e incremental.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas.
- Programación en parejas.
- Frecuente interacción del equipo de programación con el cliente o usuario.
- Corrección de todos los errores antes de añadir una nueva funcionalidad.

- Propiedad del código compartido.
- Simplicidad en el código.

➢ Proceso Unificado de Desarrollo

El Proceso Unificado de Desarrollo (en lo adelante RUP) constituyó el resultado de tres décadas de desarrollo y trabajo práctico. Transitó desde el Proceso Objectory con su primera publicación en 1987 hasta el Proceso Objectory del Rational en 1997, para luego en 1998, finalmente, tomar el nombre de Proceso Unificado de Rational. Es un proceso de desarrollo de software que ofrece un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el Proceso Unificado es más que un simple proceso: es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. (Jacobson, y otros, 2000)

Este proceso está basado, en componentes y como aspectos definitorios se señala que es dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental.

- ***Dirigido por casos de usos:***

Un caso de uso es un fragmento de funcionalidad del sistema que devuelve un resultado importante para el usuario. Representan los requisitos funcionales. La unión de todos los casos de usos describe la funcionalidad del sistema en su totalidad y se representa a través del modelo de casos de uso.

- ***Centrado en la arquitectura:***

La arquitectura abarca los aspectos estáticos y dinámicos más relevantes del sistema como son: la estructura y el comportamiento, la funcionalidad, la facilidad de comprensión, la reutilización, la flexibilidad, el rendimiento, las restricciones y compromisos económicos y tecnológicos y la estética. La arquitectura debe de trabajar equilibradamente y mantener una interacción con los casos de uso, ya que ambos determinan la forma y la función respectivamente.

- ***Iterativo e incremental:***

El trabajo se divide en partes más pequeñas denominadas iteraciones y que resultan incrementos. Las iteraciones hacen referencias a pasos en los flujos de trabajo (Modelamiento del Negocio,

Requerimientos, Análisis y Diseño, Implementación, Pruebas, Instalación o Despliegue, Administración del proyecto, Administración de configuración y Cambios y Ambiente), y los incrementos al crecimiento del producto. La selección de lo que se implementará está basada en los casos de uso más significativos y los riesgos de mayor relevancia. Una iteración es una secuencia de actividades con un plan establecido y un criterio de evaluación.

Las mejores prácticas que propone RUP (Rational Software Corporation, 2003) son:

- **El desarrollo iterativo del software:** El software es desarrollado bajo un ciclo de iteraciones. Una iteración incorpora una serie de actividades secuenciales en el modelado de negocios, requerimientos, análisis y diseño, implementación, prueba, despliegue y, en varias proporciones dependiendo en qué parte del ciclo de desarrollo se encuentre la iteración.
- **La gestión de los requerimientos:** Es un enfoque sistemático para encontrar, documentar, organizar requisitos, además de un sistema de seguimiento de la evolución de las necesidades.
- **Una arquitectura basada en componentes:** se basan en la independencia, componentes modulares, ayuda a administrar la complejidad y fomentar la reutilización. La arquitectura permite obtener y conservar el control intelectual sobre el proyecto, es una base efectiva para la reutilización a gran escala y proporciona una base para la gestión de proyectos. El sistema estará constituido de discretos componentes ejecutables, que se desarrollan independientemente unos de otros, posiblemente creados por diferentes equipos.
- **El uso de software de modelado visual:** el modelado visual es el uso de notaciones gráficas y textuales de diseño semánticamente ricas para capturar el diseño de software. Una notación, como por ejemplo la UML, permite plantear un nivel de abstracción, manteniendo una rigurosa sintaxis.
- **La verificación de la calidad del software:** es importante que la calidad de todos los artefactos sea evaluada en varios momentos del ciclo de vida del proyecto a medida que madura el mismo. Deben ser comprobados tanto los artefactos como las actividades que los producen y se completan al término de cada iteración.
- **El control de cambios del software:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a utilización y actualización concurrente de elementos y control de versiones.

1.5. Lenguaje Unificado de Modelado

Capítulo 1

Fundamentación teórica.

El lenguaje unificado de modelado (en lo adelante UML) es un lenguaje visual que permite a los creadores de sistemas generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas. El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas que no son más que la representación gráfica de un conjunto de elementos. Visualizan un sistema desde diferentes perspectivas a las cuales se les conoce como modelo UML y describe lo que supuestamente hará un sistema, pero no dice como implementarlo *(Schumller, 2000)*

UML es una especificación de notación orientada a objetos. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto:

- Vista casos de uso: Se forma con los diagramas de casos de uso, colaboración, estados y actividades.
- Vista lógica: Se forma con los diagramas de clases, objetos, colaboración, estados y actividades.
- Vista de procesos: Se forma con los diagramas de la vista de diseño, recalcando las clases y objetos referentes a procesos.
- Vista de implementación: Se forma con los diagramas de componentes, colaboración, estados y actividades.
- Vista de despliegue: Se forma con los diagramas de despliegue, interacción, estados y actividades.

La esencia del UML son sus diagramas, cada uno de ellos usa la anotación pertinente y la suma de estos diagramas crean las diferentes vistas antes mencionadas. Se dispone de dos tipos diferentes de diagramas los que dan una vista estática del sistema y los que dan una visión dinámica:

Diagramas de estructura estática: Describen las propiedades estructurales del sistema.

Diagrama de clases: Conjunto de clases, interfaces y colaboraciones.

Diagrama de objetos: Conjunto de objetos y sus relaciones.

Diagrama de casos de uso: Conjunto de casos de uso y actores y sus relaciones.

Diagramas de comportamiento:

Diagramas de interacción (secuencia y colaboración): Objetos y sus relaciones, incluyendo los mensajes que pueden ser enviados entre ellos.

Diagrama de estados: Muestra una máquina de estado que consta de estados, transiciones, eventos y actividades.

Diagrama de actividad: Es un tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema.

Diagramas de implementación:

Diagrama de componentes: Organización y las dependencias entre un conjunto de componentes.

Diagrama de despliegue: Configuración de nodos de procesamiento en tiempo de ejecución y los componentes que residen en ellos.

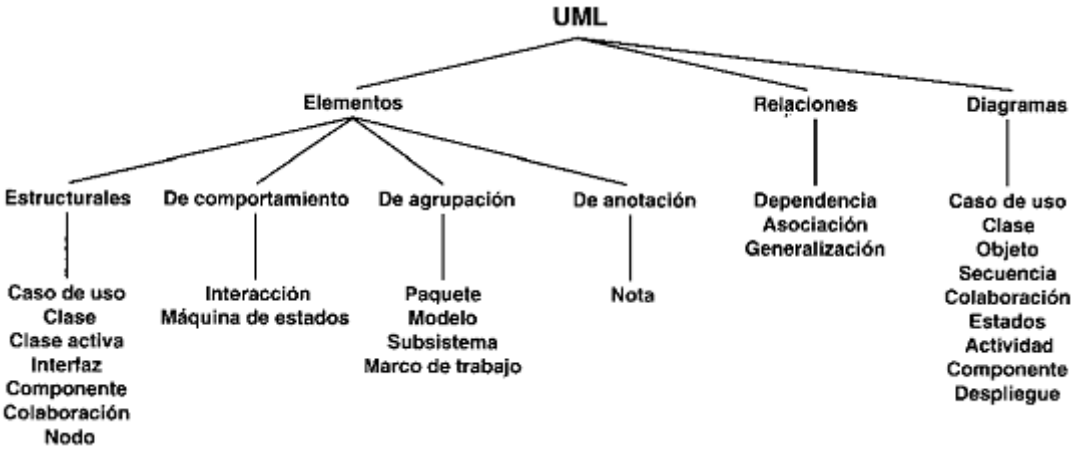


Fig.1 El vocabulario de UML (Jacobson, y otros).

1.6. Herramientas Case.

Las herramientas CASE (Computer Aided/Assisted Software/System Engineering) son un conjunto de herramientas que brindan soporte a un enfoque de ingeniería en el desarrollo de software en alguna o en todas las fases de este proceso. (Kendall & Kendall, 2005)

Tienen como objetivo:

- Permitir la aplicación práctica de metodologías estructuradas.
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.

- Simplificar el mantenimiento de los programas.
- Mejorar y estandarizar la documentación.
- Aumentar la portabilidad de las aplicaciones.
- Facilitar la reutilización de componentes del software.
- Permitir un desarrollo y un refinamiento visual de las aplicaciones.

Existen varias herramientas de modelado UML que posibilitan el modelado visual de proyectos de software, viabilizando la comunicación entre los integrantes del equipo de desarrollo. Algunos ejemplos son Rational Rose y Visual Paradigm.

- **Rational Rose**

El Rational Rose es una herramienta de modelado visual con plataforma independiente, que ayuda a la comunicación entre los miembros del equipo. Tiene como ventaja que utiliza la notación estándar en la arquitectura de software (UML), la cual permite a los arquitectos y desarrolladores de software visualizar todo el sistema mediante el uso de un lenguaje común. Mantiene la consistencia de los modelos del sistema de software. Permite el chequeo de la sintaxis UML y realizar ingeniería inversa. Posibilita la generación de documentación automáticamente y la generación de código a partir de los modelos. A través de él los diseñadores pueden modelar sus componentes e interfaces en forma individual y luego unirlos con los demás componentes del proyecto.

- **Visual Paradigm para UML**

Visual Paradigm para UML es una herramienta CASE, que utiliza UML 2.1 como lenguaje de modelaje y soporta el ciclo de vida completo del desarrollo de software. Es multiplataforma y posibilita la generación de código y documentación. Permite dibujar 13 tipos de diagramas diferentes a través de un intuitivo modelado visual y la aplicación de ingeniería inversa. Admite la importación y exportación de XML e imágenes, la administración de requerimientos, la creación de esquemas de clases a partir de una base de datos y viceversa, sin contar también que cuenta con un soporte que facilita el trabajo simultáneo sobre un mismo diagrama entre dos desarrolladores en un tiempo real.

1.7. Fundamentación de la metodología, lenguaje y herramientas de modelado seleccionadas para el desarrollo.

Luego de haber realizado un minucioso análisis de las metodologías candidatas se decidió seleccionar la metodología RUP debido a ser una metodología adaptable al contexto y necesidades de la solución.

Posibilita una definición inicial y acertada del sistema que evitan las reconstrucciones parciales posteriores debido a que se desarrolla un proceso de reducción de riesgos desde etapas tempranas. Implementa muy buenas prácticas de desarrollo de software debido a que es iterativa e incremental, dirigida por casos de usos y basada en la arquitectura. El hecho de utilizar la Programación Orientada a Objetos permite obtener un sistema escalable en el tiempo que no necesitará grandes inversiones de recursos en sus modificaciones posteriores. Por otra parte, un punto muy importante es lo referente a la organización del trabajo, esta metodología define quién, cómo, qué y cuándo se realizan cada una de las actividades o los artefactos necesarios.

Se ha decidido utilizar el lenguaje de modelado UML debido a que está consolidado como el lenguaje estándar en el análisis y diseño de sistemas informáticos. Sus características visuales facilitan a los integrantes de un equipo multidisciplinario, ya sean analistas, diseñadores, programadores u cualquier otro rol, una fácil comunicación e intercomunicación que tributa a la común comprensión de lo que se quiere hacer. A través de él se logran robustas aplicaciones, bien diseñadas y de fácil mantenimiento como resultados de ser desarrolladas también bajo la tecnología orientada objetos y la metodología de desarrollo RUP.

Como herramienta de modelado se seleccionó el Visual Paradigm debido a su intuitivo modelado, es multiplataforma y cuenta con amplias funcionalidades que satisfacen las necesidades de los desarrolladores. Con ella se agiliza el modelado del software, aumenta la productividad y viabiliza el trabajo en equipo.

1.8. Análisis y Diseño de Sistemas.

El objetivo principal de esta disciplina es transformar los requerimientos a una especificación que describa cómo implementar el sistema. El análisis fundamentalmente consiste en obtener una visión que se preocupa de ver qué hace el sistema de software a desarrollar, por tal motivo éste se interesa en los requerimientos funcionales. Por otro lado, el diseño es un refinamiento que toma en cuenta los requerimientos no funcionales, por lo cual se centra en cómo el sistema cumple sus objetivos. (Marrero, y otros, 2008)

Al principio de la fase de elaboración hay que definir una arquitectura candidata: crear un esquema inicial de la arquitectura del sistema, identificar clases de análisis y actualizar las realizaciones de los casos de uso con las interacciones de las clases del análisis. Durante la fase de elaboración se va refinando esta arquitectura hasta llegar a su forma definitiva. En cada iteración hay que analizar el comportamiento para diseñar componentes.

El análisis se encarga de representar una vista interna del sistema en la cual usando el lenguaje de los desarrolladores se refina los requisitos y se estructuran en base a clases y paquetes. En esta etapa se procura identificar y describir los objetos dentro del dominio del problema. Luego este proceso continúa en el diseño hasta obtener los objetos lógicos que interactúan y darán cumplimiento a los requerimientos funcionales y no funcionales planteados.

Propósitos generales del análisis y el diseño:

- Transformar los requerimientos en un diseño de lo que será el sistema.
- Evolucionar una arquitectura robusta para el sistema.
- Adaptar el diseño al ambiente de implementación, diseñándolo para el desarrollo.

Tabla 1.1 Propósitos particulares del análisis y diseño. (Jacobson, y otros).

Propósitos del análisis	Propósitos de diseño
Comprensión perfecta de los requisitos, refinarlos y estructurarlos.	Comprensión exacta de aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario.
Análisis profundo de los requisitos funcionales utilizando el lenguaje de los desarrolladores.	Brindar una entrada apropiada para actividades de implementación ya sean requisitos o subsistemas individuales interfaces y clases.
Proporcionar una visión general del sistema.	Descomponer los trabajos de implementación en partes más manejables para ser desarrolladas por diferentes equipos.
	Capturar las interfaces entre los subsistemas antes en el ciclo de vida del software.

Tabla 1.2. Diferencias entre el Modelo de Análisis y el Modelo de Diseño. (Jacobson, y otros).

Modelo de Análisis	Modelo de Diseño
Modelo conceptual (abstracción del sistema y permite aspectos de la implementación)	Modelo físico (plano de la implementación)
Genérico respecto al diseño (aplicable a varios diseños)	No genérico, específico para una implementación.
Tres estereotipos conceptuales sobre las clases: Control, Entidad e Interfaz.	Cualquier número de estereotipos (físicos) sobre las clases, dependiendo del lenguaje de implementación.
Dinámico (no muy centrado en la secuencia)	Dinámico (centrado en las secuencias)
Bosquejo del diseño del sistema, incluyendo su arquitectura.	Manifiesto del diseño del sistema, incluyendo su arquitectura (una de sus vistas)
Puede no estar mantenido durante todo el ciclo de vida del software.	Debe ser mantenido durante todo el ciclo de vida del software.
Define una estructura que es una entrada esencial para modelar el sistema, incluyendo la creación del modelo de diseño.	Da forma al sistema mientras que intenta preservarla estructura definida por el modelo de análisis lo más posible.

La etapa de análisis y diseño es la que más aporta a la arquitectura del sistema debido a que en ella se definen tres vistas arquitectónicas, la vista lógica, la de procesos y la de despliegue.

1.9. El rol de analista

El analista dirige y coordina la etapa de elicitación de requerimientos y modelación de casos de uso de manera que va perfilando las funcionalidades del sistema y delimitando el mismo. Por ejemplo él delimita quiénes son los actores y cuáles son los casos de uso que propician la interacción con el sistema. (Rational Software Corporation, 2003)

La persona que trabaje como el rol de analista debe ser experta identificando y entendiendo problemas y oportunidades. Ha de contar con habilidades para ofrecer soluciones a las necesidades del problema. Debe ser muy comunicativa. El conocimiento del negocio y el dominio de tecnologías han de ser otras habilidades importantes en este rol, que vienen a perder un poco de importancia si es un

profesional que entiende y absorbe las nuevas informaciones rápidamente. Algo que no debe de faltar es que debe ser capaz de colaborar eficazmente con los otros integrantes del equipo.

1.10. El rol del diseñador

El diseñador es el responsable de diseñar partes del sistema, teniendo en cuenta las restricciones de los requerimientos, la arquitectura y el proceso de desarrollo para el proyecto. Él identifica y define las responsabilidades, funcionamientos, atributos, y relaciones de los elementos del diseño. Asegura que el diseño es consistente con la arquitectura del software, y la detalla a un punto donde la aplicación puede proceder. (Rational Software Corporation, 2003)

La persona que desempeñe el rol de diseñador debe contar con sólidos conocimientos en cuanto a los requisitos del sistema, la arquitectura del mismo, las técnicas de diseño de software, incluyendo el análisis orientado objeto y el lenguaje unificado de modelado, las tecnologías en las que el sistema será implementado y las líneas bases del proyecto.

1.11. Estrategia de captura de requisitos.

En todo el proceso de desarrollo de software sentar las bases de lo que se va a realizar es lo más complicado. Si no se tienen claras las necesidades o expectativas del cliente todo trabajo posterior es en vano. Para evitar este problema lo principal es desarrollar un sólido proceso de captura de requisitos que posibilite comprender el dominio de la aplicación, el problema en cuestión, el contexto del negocio y las restricciones y necesidades de los usuarios finales. Lo esencial para alcanzar estos objetivos es elegir entre las diferentes estrategias existentes para la captura de requisitos aquella que satisfaga las necesidades y se ajuste en mayor medida a las condiciones existentes.

Entre las más destacadas podemos mencionar algunas:

Entrevistas: Intento sistemático de recoger información de otra persona a través de una comunicación interpersonal que se lleva a cabo por medio de una conversación estructurada. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada.

Cuenta con tres fases que son la identificación del entrevistado, la preparación de la entrevista, la realización de la entrevista y la documentación de los resultados. (Escalona, y otros, 2002)

Workshop colaborativos (Trabajo en grupo y aprendizaje colaborativo): Técnica de extracción de requisitos en la que se promueve la cooperación entre usuarios y analistas en las que se aprovechan

las ventajas de la dinámica de grupos. Se desarrolla mediante un trabajo sistemático y organizado (reglas de la reunión, tipo de participación, mantener el alcance, tiempos de discusión, tamaño del equipo, etc.) Para ello se realizan varias reuniones en intervalos de 2 a 4 días y se hace uso de ayudas visuales de comunicación. A través de él se puede discrepar sobre opiniones encontradas, todo el grupo puede actuar como revisor y detectar defectos y se profundiza la participación más profunda de los usuarios en el proyecto.

Prototipado: Consiste en la elaboración de un modelo o maqueta del sistema. Son versiones reducidas, demos o conjuntos de pantallas que no son totalmente operativas. Se construye para evaluar mejor los requisitos que se desean cumplir. Existen tres tipos: el prototipo de interfaz de usuario, los modelos de rendimientos y un prototipado funcional.

Cuestionarios: Esta técnica requiere que el analista conozca el ámbito del problema en el que está trabajando. Consiste en redactar un documento con preguntas cuyas respuestas sean cortas y concretas, o incluso cerradas por unas cuantas opciones en el propio cuestionario.

Tormentas de ideas: Es también una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas. El grupo de personas que participa en estas reuniones no debe ser muy numeroso (máximo 10 personas), una de ellas debe asumir el rol de moderador de la sesión, pero sin carácter de controlador. (Escalona, y otros, 2002)

Como técnica de captura de requisitos es sencilla de usar y de aplicar. Suele ofrecer una visión general de las necesidades del sistema, pero normalmente no sirve para obtener detalles concretos del mismo, por lo que suele aplicarse en los primeros encuentros.

1.12. Patrones

¿Qué es un Patrón?

Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos sobre cómo aplicarlo en nuevas situaciones, o sea, un patrón es una descripción de un problema bien conocido que suele incluir: descripción, escenario de uso, solución concreta, consecuencias de utilizar el patrón, ejemplos de implementación y lista de patrones relacionados. (Larman, 2008)

1.12.1. Patrones de Casos de uso.

El nombre revela la intención: Hace un llamado al uso de nombres descriptivos para los casos de usos, que ubiquen expectativas en el lector. Es decir que revelen la intención del caso de uso y refleje el único objetivo que el actor está intentando lograr. Para ello este patrón propone como buena práctica nombrar los casos de uso comenzando con un verbo activo que describa el uso del caso de uso y seguido del verbo con una frase que describa su propósito. Ser conciso pero lo suficientemente descriptivo para capturar la esencia del caso de uso. (Övergaard & Palmkvist, November 12, 2004)

Completar una única meta: Consiste en escribir cada caso de uso dirigido hacia una completa y bien definida meta.

Alternativas exhaustivas, íntegras: Trata de capturar todos los fallos y alternativas que deben ser manejados en el caso de uso. Una vez se tienen identificados todos los casos de uso y su flujo principal, se identifican y capturan todas las variaciones del flujo principal que se quiera que el sistema maneje.

Patrón Preciso y Legible: El presente patrón plantea escribir cada caso de uso lo suficientemente legible que los clientes los lean, evalúen y precisen con exactitud y que los implementadores entiendan qué están construyendo. 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. (Övergaard & Palmkvist, November 12, 2004)

El patrón "CRUD[2]: Completo": Este patrón plantea la definición de un solo caso de uso que fusione las diferentes operaciones que pueden ser realizadas como simples casos de uso. Se habla de casos como adicionar, eliminar, actualizar, consultar segmentos de información dentro de un caso de uso, formando una sola unidad conceptual. Este patrón es aplicado cuando todo el flujo tributa a un mismo valor dentro del negocio y este es corto y simple. (Övergaard & Palmkvist, November 12, 2004)

Patrón Claro Conjunto de Roles: Plantea identificar los actores y el rol que cada uno juega con respecto al sistema. Describir claramente cada uno de los actores con que el sistema debe actuar recíprocamente. Esto es porque si se analizan solo los usuarios del sistema y preferiblemente después los roles que los usuarios juegan con respecto al sistema o las partes interesadas en esos roles,

[2] CRUD: Crear, Obtener, Actualizar y Borrar

entonces se pierden comportamientos importantes del sistema o se introducen comportamientos redundantes. (Övergaard & Palmkvist, November 12, 2004)

1.12.2. Patrones de Diseño.

Un patrón de diseño es una descripción de clases y objetos, comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Identifica clases, instancias, roles, colaboraciones y la distribución de responsabilidades. (Gracia, 2005)

Un patrón de diseño es:

- Una solución estándar para un problema común de programación.
- Una técnica para flexibilizar el código.
- Un lenguaje de programación de alto nivel.
- Una manera más práctica de describir ciertos aspectos de la organización de un programa.
- Conexiones entre componentes de programas.

Características de los patrones de diseño:

- Son soluciones concretas y técnicas.
- Se utilizan en situaciones frecuentes.
- Favorecen la reutilización de código.
- Es difícil reutilizar la implementación de un patrón.

Ventajas de los patrones de diseño:

La principal ventaja es que proponen una forma de reutilizar la experiencia de expertos en el proceso de desarrollo de software, en busca de no cometer los mismos errores. Por ello clasifican formas de solucionar problemas que ocurren de forma frecuente en el desarrollo y se describen como experiencias reales, probadas y funcionales.

Patrones GRASP[3]

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. (Larman, 2008)

Se pueden destacar 5 patrones principales que son:

1. **Experto:** la responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados. Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Con el uso de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece el hecho de tener sistemas más robustos y de fácil mantenimiento. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases "sencillas" y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

2. **Creador:** se asigna la responsabilidad de que una clase B cree un objeto de la clase A solamente cuando:

- B contiene a A.
- B es una agregación (o composición) de A.
- B almacena a A.
- B tiene los datos de inicialización de A.
- B usa a A.

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte a bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.

[3] GRASP: Patrones Generales de Software para Asignación de Responsabilidades.

3. **Alta Cohesión:** cada elemento del diseño debe realizar una labor única dentro del sistema. Ejemplos de una baja cohesión son las que hacen demasiadas cosas. Este patrón mejora la claridad y la facilidad con que se entiende el diseño, simplifica el mantenimiento y las mejoras en funcionalidad y a menudo se genera un bajo acoplamiento.

4. **Bajo Acoplamiento:** debe haber pocas dependencias entre las clases. Estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento y produzca resultados negativos propios de un alto acoplamiento. Esto contribuye que las clases no se afecten por cambios de otros componentes, sean posibles de entender por separado y fáciles de reutilizar.

5. **Controlador:** asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades, mayor potencial de los componentes reutilizables y la garantía de que los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la de la interfaz. Desde el punto de vista técnico, las responsabilidades del controlador podrían cumplirse en un objeto de interfaz, pero esto supone que el código del programa y la lógica relacionada con la realización de los procesos del dominio puro quedarían incrustados en los objetos interfaz o ventana.

También este patrón posibilita reflexionar sobre el estado del caso de uso. A veces es necesario asegurarse de que las operaciones del sistema sigan una secuencia legal o poder razonar sobre el estado actual de la actividad y las operaciones en el caso de uso subyacente.

Patrones GOF[4]

Existen tres tipos de patrones GOF:

1. **Patrones de Comportamiento.**

Los patrones de comportamiento están relacionados con algoritmos y asignación de responsabilidades a los objetos. Estos patrones describen no solo patrones de objetos sino también patrones de comunicación entre ellos. Dentro de ellos se pueden clasificar en función de que trabajen con clases (Template Method, Interpreter) u objetos (Chain of Responsibility, Command, Iterator, Mediator, Memento, Observer, State, Strategy, Visitor). (Albacete, 2002)

2. **Patrones de Estructura.**

[4] GOF: Guest Observer Facility.

Los patrones estructurales están relacionados con cómo las clases y los objetos se combinan para dar lugar a estructuras más complejas. También se puede hablar de patrones estructurales asociados a clases (Adapter) y asociados a objetos (Bridge, Composite, Decorator, Facade, Flyweight, Proxy), los primeros utilizan la herencia y los últimos la composición. (Albacete, 2002)

3. **Patrones de Creación.**

Los patrones de creación proporcionan ayuda a la hora de crear objetos, principalmente cuando esta creación requiere de la toma de decisiones, la cual puede ser dinámica. Estos patrones ayudan a estructurar y encapsular dichas decisiones. En algunas ocasiones existe más de un patrón que se puede aplicar a la misma situación y en otras se pueden combinar múltiples patrones convenientemente. Un patrón de creación asociado a clases usa la herencia para variar la clase que se instancia, mientras que un patrón de creación asociado a objetos delegará la instanciación a otro objeto. Ejemplos de estos patrones son Singleton, Builder, Prototype. (Albacete, 2002)

Conclusiones parciales

En este capítulo se realizó un estado del arte sobre los servicios de soporte existentes, las bases de conocimientos, la gestión del conocimiento y su importancia, además de un análisis de las diferentes tecnologías y metodologías de desarrollo de software, particularizando el flujo de trabajo de análisis y diseño y particularmente el rol del analista y el diseñador. También se fundamentaron algunas de las decisiones tomadas en el desarrollo del sistema, como son la selección de la metodología RUP, el lenguaje UML y la herramienta de modelado Visual Paradigm.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.

Introducción

En el presente capítulo se describe la solución propuesta. Principalmente se basará en el modelo de dominio a través del cual se muestran los principales conceptos que se manejan en el dominio del sistema, ya sean objetos o eventos pertenecientes al entorno. Esto posibilita el establecimiento de un vocabulario común para los involucrados, que va en pos de un mejor entendimiento del proceso en cuestión. Se describirán los requisitos funcionales y no funcionales que ha de satisfacer el sistema. Se representará la modelación en términos de casos de usos del mismo. Además, se detallará cada caso de uso mediante sus especificaciones.

2.1. Modelo de Dominio.

2.1.1. ¿Qué es Modelo de Dominio?

El modelo del dominio se considera un subconjunto del llamado modelo de objetos del negocio. Es una representación visual mediante diagramas UML de los conceptos, productos y eventos que suceden en el contexto del sistema. La mayoría de los objetos del dominio o clases son obtenidos a través de la especificación de requisitos o mediante una entrevista con los expertos del dominio. En este modelo solo se describe información significativa en el negocio donde no se encuentran incluidas las responsabilidades de los asociados. (Rational Software Corporation, 2003)

El propósito de esta alternativa de trabajo es identificar todos los productos y entregables importantes para el dominio del sistema, detallar entidades del negocio y proporcionar una comprensión común de los conceptos encontrados en el entorno del dominio. En resumen, comprender y describir las partes más importantes del contexto del sistema sin tener en cuenta cronología alguna.

Además, el modelo de dominio es global, es decir se realiza para todos los procesos en general. Se considera que un escenario apropiado para esta alternativa es aquel donde el objetivo primario es la gestión y presentación de información, tales como sistemas de gestión de órdenes y sistemas bancarios. El glosario y el modelo de dominio posibilitan a usuarios, clientes, desarrolladores y otros interesados el uso de un vocabulario común.

2.1.2. ¿Por qué Modelo de Dominio?

Durante el desarrollo del proceso de producción de software se generan muchas experiencias y conocimientos nuevos. Documentar y gestionar todos estos no es cuestión de minutos, ni lleva un procedimiento particular. Por ejemplo, entre los integrantes de los proyectos existentes en la Universidad de las Ciencias Informáticas no existe un convenio en cuanto a las maneras de gestionar los conocimientos asimilados por los desarrolladores en el marco de trabajo, ni siquiera existe la tendencia de hacerlo en la mayoría de los casos. Solo algunos valoran las ventajas de materializar esta información y lo hacen de forma empírica y espontánea.

Por la situación antes planteada es que se ha decidido realizar un modelo de dominio. Los eventos del negocio en cuestión no tienen fronteras completamente definidas y se hace difícil delimitar los procesos del mismo. Aunque sí es posible identificar las personas involucradas o que se benefician en él. Esta técnica de modelado puede hacer trazas de clases hasta la experiencia de los expertos, pero imposible entre el modelo del dominio y los casos de uso de sistema.

2.2. Glosario de términos.

Los glosarios de términos propician el establecimiento de un vocabulario común y una mejor comprensión del trabajo debido a que se definen las entidades y conceptos principales asociados al entorno del dominio. Para construir una aplicación, cualquiera que sea su tamaño, los ingenieros han de hablar un mismo lenguaje consistente. Una terminología común es necesaria para compartir conocimientos con otros, de no realizarse abundaría la confusión en el proceso de ingeniería y el trabajo sería muy difícil, si no imposible. (Rational Software Corporation, 2003)

Conceptos	Descripción
Asesoría	Servicio de ayuda sobre materias en las que el cliente presente problemas.
Aplicación	Sistema informático que permite a un usuario utilizar una computadora con un fin específico.
Artículo	Texto que presenta la postura personal de un individuo frente a un acontecimiento, un problema actual o de interés general.
Base de Conocimiento	Donde se almacenan los conocimientos generados durante el proceso de desarrollo de software.
Capacitación	Servicio de enseñanza a través del cual se desarrollan conocimientos y habilidades en el individuo.

Conocimiento	Conjunto de datos sobre hechos, verdades o de información almacenada a través de la experiencia.
Contrato	Documento que oficializa un acuerdo legal sobre prestación de servicios entre la entidad y el cliente.
Documento	Escrito que contiene información.
Entrenamiento	Servicio de preparación para el uso de herramientas o metodologías.
FAQ (Frecuent answer question)	(En español) Preguntas y respuestas más frecuentes propuestas por el usuario.
Foro	Espacio de intercambio de opiniones y consejos sobre temas determinados.
Grupo de Soporte	Grupo de personas que brindan servicios.
Listas de Chequeo	Documento que sirve para verificar el grado de cumplimiento de determinadas reglas establecidas.
Nota Técnica	Documento dónde se documentan problemas afrontados y la solución encontrada.
Noticias	Publicación con noticias, comunicados y artículos sobre temas específicos.
Servicios	Asistencia prestada a un cliente.
Solicitud de Contrato	Documento de petición de contratos por parte del cliente.
Solicitud de Servicio	Documento de petición de asistencia a la entidad por parte del cliente.
Soporte en línea	Asistencia brindada para resolver necesidades a través de la web.
Usuario	Persona autorizada que interactúa con el sistema y accede a información.

2.3. Reglas del Negocio.

Las reglas del negocio son la declaración de políticas y condiciones que han de satisfacerse. Se utilizan cuando hay muchas o complejas condiciones guiando las operaciones del negocio.

1. El usuario debe autenticarse por su usuario del dominio para acceder al sistema.

2. El usuario para contar con un servicio debe haber establecido un contrato.

3. Un contrato puede quedar cancelado por violaciones de las políticas de establecimiento del contrato.

4. El contrato se establece bajo varias reglas:

- El contrato queda cancelado por negligencias del cliente en cuanto al cumplimiento de un servicio solicitado.

- El cliente se responsabiliza por el contenido insertado.

- El contenido creado no puede contener información censurable, ni un vocabulario obsceno y agresivo, ni violar los principios de propiedad intelectual.

- El cliente una vez establecido el contrato tiene derecho a reclamar por la calidad del servicio.

5. El cliente puede solicitar servicios de: capacitación, entrenamiento y asesoría. Además de poder crear contenido y realizar búsquedas avanzadas de software.

6. Solo el jefe de soporte es el autorizado de invalidar un contrato.

2.4. Modelo de Dominio.

El modelo de dominio tiene como objetivo fundamental representar y comprender, a través de diagramas UML las principales clases dentro del contexto del sistema, lo que también contribuye a la comprensión de los requisitos del sistema que se desprenden de dicho contexto. (Jacobson, y otros, 2000)

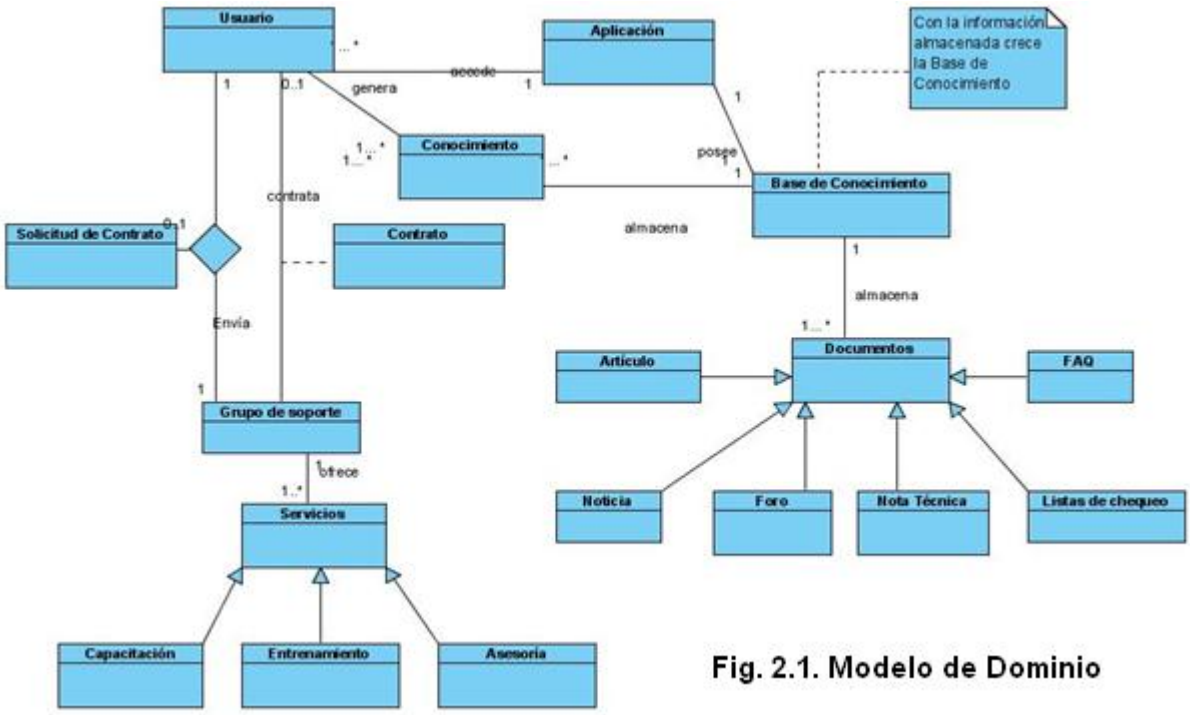


Fig. 2.1. Modelo de Dominio

2.5. Requerimientos del sistema.

Los requisitos del sistema son las capacidades o condiciones que un sistema debe cumplir para lograr la satisfacción del cliente en cuanto a las funcionalidades requeridas o las restricciones que se imponen.

2.5.1. Técnicas de captura de requisitos utilizadas.

Como estrategia de captura de requisitos se aplicaron tres estrategias fundamentalmente: la entrevista, la tormenta de ideas y el prototipado. Primeramente el equipo de desarrollo se reunió y debatieron en cuanto a ideas generales sobre el sistema y todas las funcionalidades a fines con el objetivo de la solución propuesta. De este modo surgieron una serie de requisitos funcionales para el desarrollo, pero solo de forma general y empírica. Para llegar a una mejor concepción de las necesidades existentes en el proceso de desarrollo de software y de las tendencias actuales en el soporte al mismo, se entrevistó personal capacitado, al Ing. Yunier Saborit (Jefe de la Dirección de Informatización de la UCI) que ofreció luz para delimitar finalmente los requisitos más relevantes y que cumplen las necesidades vigentes de la gestión del conocimiento en el proceso de producción de software. Por último se crearon prototipos de interfaz de usuario que muestran una vista más acertada de lo que se desea desarrollar, aportando precisión en cuanto a los requisitos funcionales rectores de la aplicación.

2.5.2 Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. El sistema propuesto ha de satisfacer las siguientes exigencias:

RF1 Autenticar usuario.

1.1. Permitir al usuario introducir su usuario y contraseña del dominio UCI.

1.2. Validar datos introducidos usando los directorios activos UCI.

1.3. Obtener los datos particulares del usuario autenticado.

RF2 Gestionar artículos.

2.1. Adicionar artículos.

2.2. Eliminar artículos.

2.3. Publicar artículos.

RF3 Gestionar foros.

3.1. Agregar foros.

3.2. Eliminar foros.

3.3. Comentar en foro.

3.4. Adicionar temas al foro.

RF4 Gestionar notas técnicas.

4.1. Adicionar notas técnicas.

4.2. Clasificar notas técnicas.

RF5 Gestionar solicitud de contrato de soporte.

5.1. Insertar solicitud de contrato de soporte.

5.2. Establecer contrato de soporte.

5.3. Disolver contrato de soporte.

RF6 Gestionar solicitud de servicio.

6.1. Realizar solicitud de servicio.

6.2. Responder solicitud de servicio.

RF7 Gestionar noticias.

7.1. Publicar noticia.

7.2. Eliminar noticia

RF8 Gestionar listas de chequeo.

8.1. Adicionar lista de chequeo.

8.2. Eliminar lista de chequeo.

8.3. Modificar lista de chequeo.

RF9 Buscar contenidos según criterios o categorías.

9.1. Insertar datos de búsquedas específico.

9.2. Brindar una búsqueda avanzada.

9.3. Brindar búsqueda solo de notas técnicas.

RF10 Gestionar usuarios.

10.1. Insertar usuario.

10.2. Eliminar usuario.

10.3. Modificar datos de usuario.

RF11 Asignar privilegios de accesos según roles.

RF12 Gestionar las preguntas más frecuentes con sus respuestas.

12.1. Adicionar preguntas más frecuentes.

12.2. Eliminar preguntas más frecuentes.

12.3. Clasificar preguntas más frecuentes por categorías.

RF13 Brindar acceso a repositorio de software.

RF14 Ofrecer estadísticas.

2.5.3. Requisitos no funcionales.

Los requisitos no funcionales especifican propiedades o cualidades que el sistema debe tener. Representan las características del producto.

- **Seguridad**

El sistema debe ser fiable, para ello ha de cumplir con determinados requisitos que eviten problemas de funcionamiento y comportamiento.

- Permitir autenticación obligatoria y segura por el dominio.
- Permitir el acceso a información y funcionalidades según el rol.

- Permitir la prevención, acción y recuperación en el mínimo de tiempo ante fallos como falta de conexión o alimentación.

- Permitir ocultar la información que aparece en la URL.

- Realizar periódicamente salvas de la base de datos.

- **Usabilidad**

  - Brindar una interfaz de usuario orientada a la ejecución de acciones, de una manera rápida, minimizando los pasos a dar en cada proceso.

  - Permitir de manera fácil la corrección de errores de introducción de datos.

  - Ofrecer la opción Ayuda para un mejor uso de las funcionalidades.

- **Disponibilidad**

  - Permitir una disponibilidad de 24 x 7.

  - No realizar procesos de mantenimiento en horarios laborables para evitar afectaciones en el servicio.

- **Portabilidad**

  - Garantizar que el sistema sea multiplataforma, para un funcionamiento tanto en Linux como en Windows.

- **Apariencia e interfaz gráfica.**
  - Brindar una interfaz de usuario con apariencia profesional pero a la vez amigable e intuitiva, que ofrezca seguridad, formalidad y una interacción efectiva al usuario.
- **Rendimiento**

  - Procesar la búsqueda u almacenamiento de información en un tiempo inferior de los 15 segundos.

  - Permitir la conexión concurrente de 500 usuarios activos sin afectarse la rapidez de la aplicación.

- **Software**

- Las máquinas clientes han de contar con un navegador web, sin importar la plataforma.

➢ **Hardware**

- La máquina cliente requiere como mínimo 256 MB de RAM.

- La PC que funcionará como servidor web y de base de datos ha de contar con una capacidad de disco duro no menos de 80 GB y una memoria RAM de no menos 712MB.

2.6. Descripción del sistema propuesto.

En pos de satisfacer los requerimientos planteados anteriormente se ha analizado la creación de un sistema que posibilite, a los profesionales vinculados al proceso de producción de software, la documentación de manera automatizada de los conocimientos adquiridos en tiempo real de desarrollo. Para ello, los usuarios del sistema han de solicitar un contrato de soporte, el cual le otorgará la condición de cliente del sistema y con ello los privilegios necesarios para realizar la gestión de contenidos y la solicitud de los servicios de soporte disponibles (asesoría, capacitación y entrenamiento). Los clientes podrán solicitar de manera en línea cada uno de estos servicios y recibir una respuesta rápida por parte del grupo de soporte en cuanto a la posibilidad de satisfacer la petición.

Como parte de la gestión de la información el cliente podrá insertar artículos, listas de chequeo, preguntas más frecuentes, noticias y notas técnicas. Estas últimas son anotaciones donde se registran constantemente los errores o incidencias a las que se enfrentan los desarrolladores y la solución encontrada por los mismos. Se encontrarán organizadas por temáticas y los usuarios podrán acceder a ellas mediante la búsqueda avanzada del buscador del sistema. El buscador también posibilita acceder a cualquier contenido que haya sido insertado en el sistema.

Otros servicios ofrecidos es el acceso a la descargas de software. Esta aplicación web será una herramienta de asistencia para el proceso productivo materializando toda una política de soporte, además de llegar a constituir a una óptima fuente de consulta. A través de foros podrán interactuar e intercambiar información al respecto de determinados temas, además de plantear dudas y esperar respuestas de personal capacitado o de otros usuarios conocedores del tema.

2.7. Definición de casos de uso del sistema.

Para la definición de casos de uso claros y entendibles se tuvieron en cuenta la aplicación de una serie de patrones de casos de uso que posibilitan un mejor entendimiento de las funcionalidades con que ha

de contar el sistema. Los patrones utilizados son: Completar una única meta, Alternativas exhaustivas, íntegras, El nombre revela la intención, Preciso y Legible, CRUD: Completo y Claro Conjunto de Roles.

2.7.1. Actores del sistema.

Los actores son terceros fuera del sistema que interactúan directamente con él.

Actor	Justificación
Usuario	Generaliza todos los usuarios del sistema. Se autentica.
Cliente	Persona que interactúa directamente con la aplicación para solicitar servicios inherentes al contrato establecido anteriormente.
Administrador de contenido.	Persona que interactúa directamente con la aplicación para administrar el flujo de información, contenido y accesos.
Jefe de soporte.	Persona que interactúa directamente con la aplicación para oficializar contratos y coordinar los servicios.

Tabla 2.1. Actores del sistema

2.7.2. Casos de uso del sistema.

Los casos de uso del sistema son un fragmento de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores.

CU-1	**Autenticar usuario.**
Actor	Usuario
Descripción	Comienza cuando el usuario introduce su nombre y contraseña y pulsa Iniciar Sesión. El sistema verifica la validez de los mismos y si son correctos este tiene acceso, de lo contrario se le notifica la negación de acceso. En la primera autenticación de un usuario sus datos personales serán obtenidos del LDAP y serán registrados en la base de datos del sistema.
Referencia	RF1
Prioridad	Crítico

Tabla 2.2. Resumen del Caso de uso 1

CU- 2	**Gestionar usuario**
Actor	Administrador de contenido.
Descripción	El administrador de contenido adiciona, modifica o elimina usuarios locales del sistema. A medida que los usuarios del dominio tienen acceso a la aplicación, se van adicionando también como usuarios del sistema.
Referencia	RF10
Prioridad	Crítico

Tabla 2.3. Resumen del caso de uso 2.

CU- 3	**Asignar roles**
Actor	Administrador de contenido.
Descripción	El administrador de contenido asigna entre los roles definidos (usuario, cliente, jefe de soporte, administrador de contenido) cual tendrá cada usuario. Por defecto, la persona autenticada cumplirá con el rol de usuario. Según el rol será el nivel de privilegios y accesos con que contará el usuario.
Referencia	RF11
Prioridad	Crítico

Tabla 2.4. Resumen del caso de uso 3.

CU- 4	**Solicitar contrato**
Actor	Usuario
Descripción	El usuario solicita el establecimiento de un contrato para lo cual inserta sus datos.
Referencia	RF5
Prioridad	Crítico

Tabla 2.5. Resumen del caso de uso 4.

CU- 5	**Invalidar contrato**
Actor	Jefe de soporte
Descripción	El jefe de soporte invalida un contrato debido a la violación de alguna de las políticas de contrato establecida. La persona propietaria del contrato pierde los privilegios automáticamente y no podrá ser cliente nuevamente del sistema.
Referencia	RF5
Prioridad	Secundario

Tabla 2.6. Resumen del caso de uso 5.

CU- 6	**Solicitar servicio.**
Actor	Cliente
Descripción	El cliente solicita un servicio determinado y para ello inserta los datos del servicio.
Referencia	RF6
Prioridad	Crítico

Tabla 2.7. Resumen del caso de uso 6.

CU- 7	**Responder solicitud de servicio**
Actor	Jefe de soporte
Descripción	El jefe de soporte da respuesta a las solicitudes de servicio según las capacidades, compromisos y prioridades.
Referencia	RF6
Prioridad	Crítico

Tabla 2.8. Resumen del caso de uso 7.

CU- 8	**Clasificar notas técnicas.**
Actor	Administrador de contenido.
Descripción	Clasifica las notas técnicas por categorías para viabilizar la búsqueda avanzada.
Referencia	RF4
Prioridad	Secundario

Tabla 2.9. Resumen del caso de uso 8.

CU- 9	**Participar en foro.**
Actor	Usuario
Descripción	El usuario inserta comentario en algún tema del foro o puede crear un nuevo tema.
Referencia	RF3
Prioridad	Secundario

Tabla 2.10. Resumen del caso de uso 9.

CU- 10	**Crear Contenido**
Actor	Cliente
Descripción	El usuario puede adicionar cualquier contenido a la aplicación llenando los campos obligatorios de cada uno. Los contenidos pueden ser nuevos foros, artículos, noticias, preguntas más frecuentes (FAQ), notas técnicas y listas de chequeo.
Referencia	RF2, RF3,RF4, RF7, RF8, RF13
Prioridad	Crítico

Tabla 2.11. Resumen del caso de uso 10

CU- 11	**Gestionar Contenido**

Actor	Administrador de contenido.
Descripción	El administrador podrá publicar, editar o eliminar los contenidos existentes. Los contenidos pueden ser nuevos foros, artículos, noticias, preguntas más frecuentes (FAQ) y listas de chequeo.
Referencia	RF2, RF3, RF7, RF8, RF13
Prioridad	Crítico

Tabla 2.12. Resumen del caso de uso 11.

CU- 12	**Buscar contenido**
Actor	Usuario
Descripción	El usuario busca contenido según los criterios que inserta. También podrá elegir cuando quiere solo notas técnicas.
Referencia	RF9, RF12
Prioridad	Crítico

Tabla 2.13. Resumen del caso de uso 12.

CU- 13	**Acceder a software**
Actor	Cliente
Descripción	El cliente tiene posibilidad de acceder a software a través de un buscador.
Referencia	RF14
Prioridad	Secundario

Tabla 2.14. Resumen del caso de uso 13.

CU- 14	**Obtener estadísticas**
Actor	Administrador de contenido

Descripción	El administrador solicita estadísticas sobre usuarios, contratos, servicios y notas técnicas.
Referencia	RF15
Prioridad	Opcional

Tabla 2.15. Resumen del caso de uso 14.

2.7.3. Diagrama de Casos de Uso.

Un diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores.

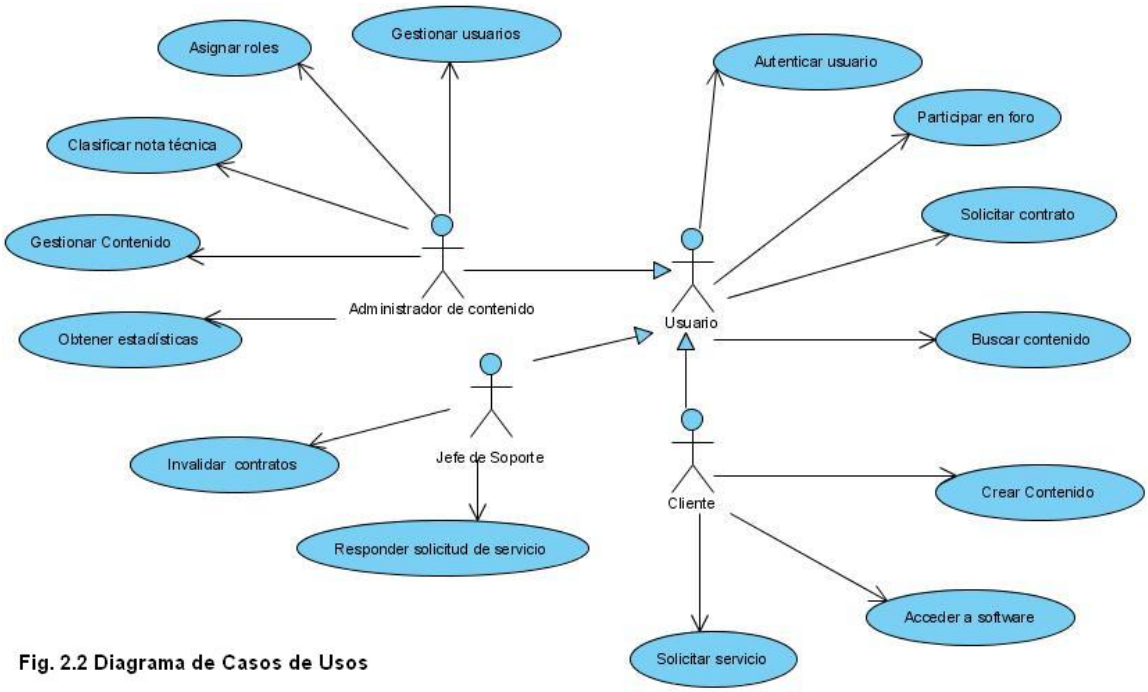


Fig. 2.2 Diagrama de Casos de Usos

2.8. Realización de los Casos de Uso.

Se muestran las descripciones de los casos de uso más importantes. Para ver los restantes remitirse a anexos I.

Tabla 2.16.Descripción del CU Autenticar Usuario.

Caso de Uso:	**Autenticar Usuario**

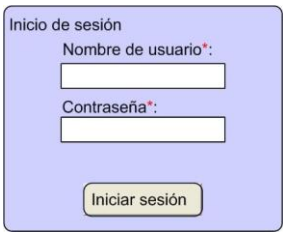
Actores:	Usuario (Inicia)
Resumen:	Mediante este caso de uso se garantiza la seguridad de la aplicación, teniendo en cuenta que el usuario se autenticará por su cuenta del dominio y se comportará de acuerdo a su rol en la aplicación. Comienza cuando el usuario introduce su nombre y contraseña y pulsa Iniciar Sesión. El sistema verifica la validez de los mismos y si son correctos este tiene acceso, de lo contrario se le notifica la negación. En la primera autenticación de un usuario sus datos personales serán obtenidos del LDAP y serán registrados en la base de datos del sistema.
Precondiciones:	El usuario debe de pertenecer al dominio.
Referencias:	RF1
Prioridad:	Crítico

Flujo Normal de Eventos

Sección "Autenticar Usuario"

Acción del Actor	Respuesta del Sistema
1. El usuario introduce nombre y contraseña y pulsa sobre el botón "Iniciar Sesión"	2. El sistema comprueba que es usuario del dominio. 3. El sistema comprueba que es usuario del sistema. 4. Muestra los menús según el rol.

Prototipo de Interfaz



Flujo Alterno (Primer Acceso)

Acción del Actor	**Respuesta del Sistema**
Acción 1	2.1. El sistema comprueba que es usuario del dominio. 3.1. El sistema comprueba que no es usuario local del sistema. 4.1. El sistema guarda los datos del usuario en la base de datos. 5. Permite el acceso al sistema por el rol usuario.
Flujo Alterno (Usuario Local)	
Acción del Actor	**Respuesta del Sistema**
Acción 1	2.2. El sistema comprueba que no es usuario del dominio. 3.2. El sistema comprueba que es usuario local del sistema. 4.2. Muestra los menús según el rol.
Flujo Alterno (No acceso)	
Acción del Actor	**Respuesta del Sistema**
Acción 1	2.3. El sistema comprueba que no es usuario del dominio. 3.3. El sistema comprueba que no es usuario local del sistema. 4.3. El sistema colorea los campos de rojo y no permite el acceso.
Prototipo de Interfaz	

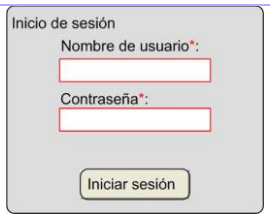
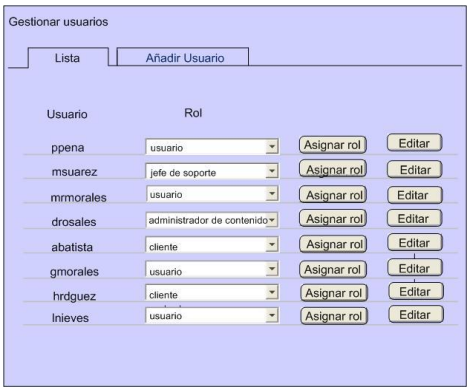
	
Poscondiciones:	Se habilitan las funcionalidades según los permisos correspondientes al tipo de usuario. Quedan guardados los datos del usuario en la base de datos para posteriores consultas.

Tabla 2.17. Descripción del CU Gestionar Usuarios.

Caso de Uso:	**Gestionar Usuarios.**
Actores:	Administrador de contenido (Inicia)
Resumen:	El administrador de contenido adiciona, modifica o elimina usuarios locales del sistema. A medida que los usuarios del dominio tienen acceso a la aplicación, se van adicionando también como usuarios del sistema.
Precondiciones:	El administrador debe tener permiso de administración
Referencias:	RF10
Prioridad:	Crítico
Flujo Normal de Eventos	
Sección "Gestionar Usuarios"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador de contenido selecciona la opción "Gestión de usuarios".	2. El sistema muestra las listas de usuarios actuales del sistema con su rol. a) Para adicionar un usuario ir a la sección "Añadir Usuario". b) Para eliminar o cambiar datos de

	usuario ir a la sección “Editar”.
Prototipo de interfaz  	
Flujo Normal de Eventos	
Sección “Añadir Usuario”	
Acción del Actor	**Respuesta del sistema**
3. El administrador de contenido selecciona la opción “Añadir Usuario”	4. El sistema muestra los campos obligatorios para adicionar a un usuario. a) Nombre y apellidos* b) Email* c) Contraseña* d) Rol del proyecto e) Proyecto Nota: El símbolo* señala los campos obligatorios. Por defecto toda persona autenticada toma el rol de usuario.
5. El administrador de contenido inserta los datos y presiona la opción “Crear cuenta nueva”.	6. El sistema valida que los datos son correctos e inserta el nuevo usuario. 7. El sistema actualiza el listado de usuarios

	publicado.
Flujo alterno	
Acción del Actor	**Respuesta del sistema**
Acción 5	1.1 El sistema valida que los datos no son correctos y muestra un mensaje de error.
Prototipo de Interfaz 	
Flujo Normal de Eventos	
Sección "Editar Usuario"	
Acción del Actor	**Respuesta del Sistema**
3. El administrador de contenido selecciona el usuario y presiona la opción "Editar".	4. El sistema muestra los campos con los datos actuales del usuario.
5. El administrador de contenido modifica los datos y presiona "Enviar".	6. El sistema salva el cambio y actualiza los datos del usuario.
Flujos Alternos	
Sección "Eliminar Usuario"	

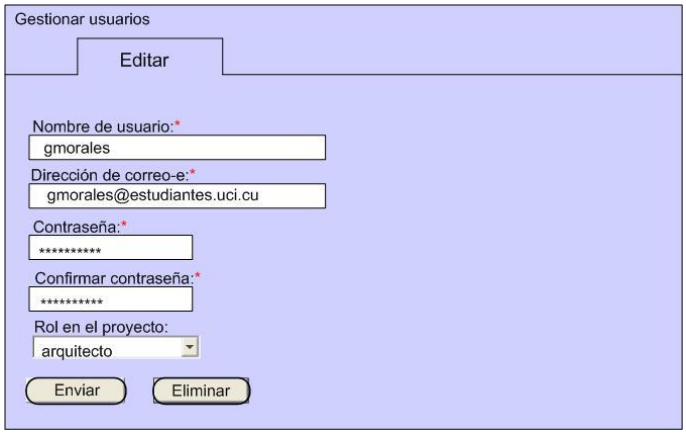
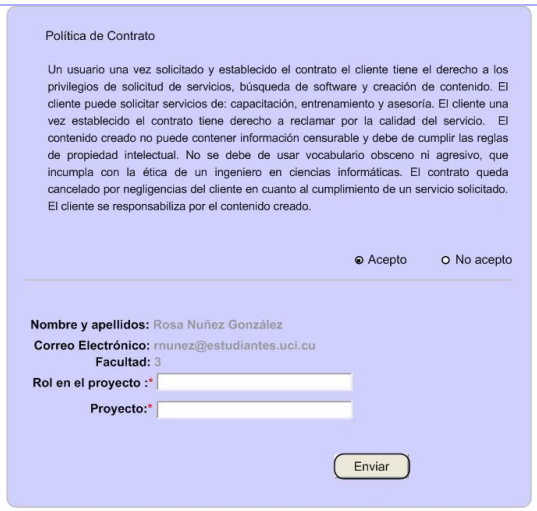
Acción del Actor	Respuesta del Sistema
3. El administrador de contenido presiona la opción “Eliminar”.	4. El sistema muestra un mensaje de confirmación de eliminado.
5. El administrador confirma presionando la opción “Eliminar”.	6. El sistema elimina el usuario y es borrado de la base de datos. 7. El sistema actualiza el listado de usuarios publicado.
Prototipo de interfaz  	
Poscondiciones:	Fue creada o modificada una instancia de un usuario.

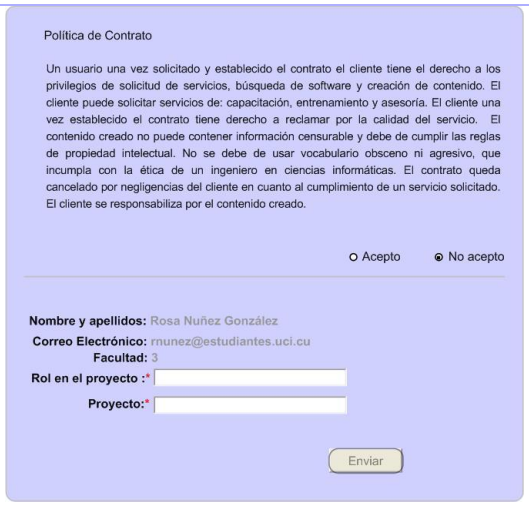
Tabla 2.18. Descripción del CU Solicitar Contrato.

Caso de Uso:	**Solicitar Contrato.**
Actores:	Usuario (Inicia)
Resumen:	El usuario solicita el establecimiento de un contrato, para el cual debe de aceptar la política de contrato y llenar los datos obligatorios de la solicitud.
Precondiciones:	El caso de uso comienza a partir de que la planilla de solicitud, correspondiente al usuario, ya se encuentra cargada con los datos del dominio.
Referencias:	**RF5**

Prioridad:	Crítico
Flujo Normal de Eventos	
Sección "Solicitar Contrato"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción "Solicitar Contrato".	2. El sistema muestra la política de contrato junto a la opción "Acepto" y No Acepto"
3. Si el usuario presiona la opción "Acepto"	4. El sistema mostrará los datos del usuario (nombre y apellidos, correo electrónico, facultad) ya tomados del LDAP y habilita otros campos a llenar por el usuario para la solicitud de un contrato. a) Rol en el proyecto * b) Proyecto* Nota: El símbolo (*) señala los campos obligatorios.
5. El usuario llena los campos y presiona la opción "Enviar".	6. El sistema valida los datos como correctos, inserta un nuevo contrato en la BD, le asigna al usuario el rol de cliente, le proporciona los permisos correspondientes y muestra el siguiente mensaje: "Ya eres uno de nuestros clientes. Gracias"
Prototipo de interfaz	



Flujo Alterno	
Acción del Actor	**Respuesta del Sistema**
3.1. Si el usuario presiona la opción “No acepto”	4.1. El sistema mantiene inhabilitado los campos obligatorios a llenar y el botón “Enviar”. Muestra el mensaje “El contrato solo será establecido bajo las políticas de contrato” retorna a la página anterior del historial.
Acción 5	6.1. El sistema valida los datos como no correctos y colorea de rojo los campos a llenar como muestra de error.
Prototipos de interfaz	



Poscondiciones:	Queda añadido un nuevo contrato y cliente a la aplicación.

Tabla 2.19. Descripción del CU Solicitar Servicio.

Caso de Uso:	**Solicitar Servicio.**
Actores:	Cliente (Inicia)
Resumen:	El cliente solicita un servicio. Para ello llena los datos del servicio necesitado en la solicitud de servicio.
Precondiciones:	El caso de uso comienza a partir de que la planilla de solicitud, correspondiente al usuario, ya se encuentra cargada con los datos del dominio.
Referencias:	RF6
Prioridad:	Crítico
Flujo Normal de Eventos	
Sección "Solicitar Servicio."	
Acción del Actor	**Respuesta del Sistema**
1. El cliente selecciona la opción "Solicitar servicio"	2. El sistema mostrará los datos del usuario (id de contrato, nombre y apellidos, correo electrónico) ya tomados del LDAP, además

	de otros campos a llenar por el usuario para la solicitud del servicio. a) Tipo de servicio.* b) Intervalo de Fechas.* c) Horario.* d) Local.* e) Características particulares. Nota: El símbolo* señala los campos obligatorios.
3. El cliente inserta los datos y selecciona la opción "Enviar."	4. El sistema valida que los datos son correctos y muestra el mensaje: "Su solicitud ha sido enviada".

Prototipo de interfaz

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Acción 3	4.1 El sistema valida que los datos nos cumplen

	con el formato correcto y colorea de rojo la casilla del error. 4.2. El sistema, hasta que todos los campos obligatorios no estén llenos, no habilita la opción “Enviar”.
Poscondiciones:	Se adiciona una nueva solicitud a la lista de solicitudes.

Tabla 2.20. Descripción del CU Responder Solicitud de Servicio.

Caso de Uso:	**Responder Solicitud de Servicio.**
Actores:	Jefe de soporte (Inicia)
Resumen:	El jefe de soporte da respuesta a las solicitudes de servicio según las capacidades, compromisos y prioridad.
Precondiciones:	Deben existir solicitudes no respondidas. El jefe de soporte ha de contar con los privilegios de su rol.
Referencias:	RF6
Prioridad:	Crítico
Flujo Normal de Eventos	
Sección “Responder Solicitud”	
Acción del Actor	**Respuesta del Sistema**
1 El jefe de soporte selecciona la opción “Ver solicitudes de servicios”	1. El sistema muestra las solicitudes.
3. El jefe de soporte selecciona una solicitud.	4. El sistema muestra la solicitud.
5. El jefe de soporte lee la solicitud y valora si está en condiciones de brindar el servicio solicitado. Para ello selecciona el botón “Eliminar” o “Aceptar”.	6. Si el jefe de soporte ha seleccionado el botón “Aceptar” el sistema permite elaborar un email con la notificación de que el servicio ha sido aceptado además de los

	términos en que lo fue. a) Lugar* b) Horario* c) Profesional a cargo. * d) Particularidades del servicio. Nota: El símbolo (*) señala los campos obligatorios.
7. El jefe de soporte inserta los datos y presiona el botón “Enviar”.	8. El sistema valida que los datos son correctos y envía el mensaje al destinatario.

Prototipo de interfaz

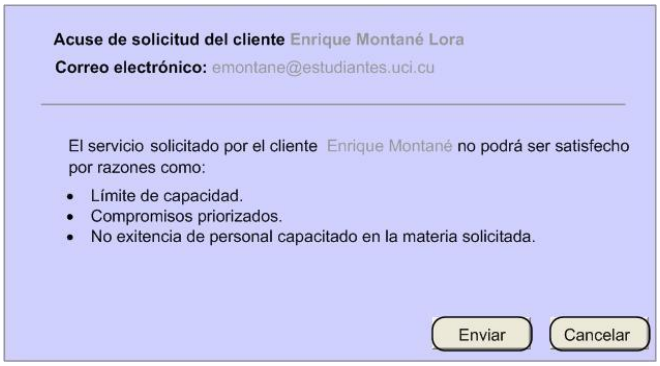
Flujos Alterno a)	
Acción del Actor	**Respuesta del Sistema**
Acción 5	6. Si el Jefe de soporte ha seleccionado el botón "Eliminar" el sistema envía un mensaje con la imposibilidad de brindar el servicio y elimina la solicitud de la base de datos.
Flujos Alterno b)	
Acción del Actor	**Respuesta del Sistema**
Acción 7	8.1. El sistema valida que los datos no son correctos y colorea de rojo la casilla del error. Vuelve a la acción 7.
Prototipo de interfaz 	
Poscondiciones:	Un servicio queda coordinado o una solicitud eliminada.

Tabla 2.21. Descripción del CU Crear Contenido.

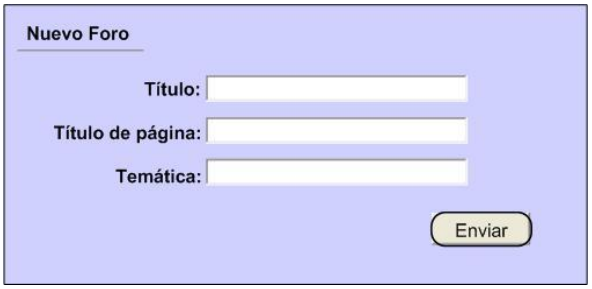
Caso de Uso:	**Crear Contenido**
Actores:	Cliente (inicia)
Resumen:	El cliente crea un nuevo contenido, un foro, un artículo, una lista de chequeo, una noticia o una FAQ, para ello inserta los campos obligatorios para cada tipo de

	contenido.
Precondiciones:	El rol de usuario debe ser cliente.
Referencias:	RF2, RF3,RF4, RF7, RF8, RF13
Prioridad:	Crítico

Flujo Normal de Eventos	
Sección "Crear Contenido"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador de contenido selecciona la opción "Crear Contenido"	2. Se despliega un menú con los tipos de contenidos posibles a crear. a) Para adicionar un foro ir a la sección "Adicionar Foro". b) Para adicionar un artículo ir a la sección "Adicionar Artículo". c) Para adicionar una noticia ir a la sección "Adicionar Noticia". d) Para adicionar una FAQ ir a la sección "Adicionar FAQ". e) Para adicionar una lista de chequeo ir a la sección "Adicionar Lista de Chequeo". f) Para adicionar una Nota Técnica ir a la sección "Insertar Nota Técnica".
Flujo Normal de Eventos	
Sección "Adicionar Foro"	
Acción del Actor	**Respuesta del Sistema**
3. El administrador de contenido selecciona la opción "Foro".	4. El sistema muestra los campos de datos obligatorios para insertar:

	• Título * • Título de página • Temática * Nota: El símbolo* señala los campos obligatorios.
5. El administrador de contenido inserta la información y selecciona la opción “Enviar”. Si desea tener una vista previa del foro, ver la sección “Vista Previa”.	6. El sistema valida que el campo obligatorio está lleno y que no existe otro foro con el mismo nombre y adiciona el nuevo foro

Prototipo de interfaz



Flujo Alterno	
Acción del Actor	**Respuesta del Sistema**
Acción 5	6.1. Si existe algún foro con el mismo nombre el sistema muestra un mensaje de error que notifique la contradicción y vuelve a la acción 5.

Flujo Normal de Eventos	
Sección “Adicionar Artículo”	
Acción del Actor	**Respuesta del Sistema**
3. El administrador de contenido selecciona la opción “Artículo”.	4. El sistema muestra los campos de datos obligatorios para insertar : a) Título *

	b) Título de la página* c) Cuerpo* Nota: El símbolo* señala los campos obligatorios.
5. El administrador de contenido inserta la información y selecciona la opción “Enviar”. Si desea ver una vista previa del artículo ver la sección “Vista Previa”.	6. El sistema valida que no hayan campos obligatorios vacíos y adiciona el nuevo artículo.

Prototipo de interfaz

Flujo alterno	
Acción del Actor	**Respuesta del Sistema**
Acción 5	6.1. Si existe algún campo vacío este se coloreará de rojo y vuelve a la acción 5.
Flujo Normal de Eventos	
Sección “Adicionar Noticia”	
Acción del Actor	**Respuesta del Sistema**
3. El administrador de contenido selecciona la opción “Noticia”.	4. El sistema muestra los campos de datos obligatorios para insertar: a) Título *

	b) Título de la página* c) Cuerpo* Nota: El símbolo* señala los campos obligatorios.
5. El administrador de contenido inserta la información y selecciona la opción “Enviar”. Si desea ver una vista previa de la noticia, ver la sección “ Vista Previa”	6. El sistema valida que no hayan campos vacíos y adiciona la nueva noticia.

Prototipo de interfaz

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Acción 5	6.1. Si existe algún campo vacío, este se coloreará de rojo y el sistema a vuelve a la acción 5.
Flujo Normal de Eventos	
Sección “Adicionar FAQ”	
Acción del Actor	**Respuesta del Sistema**
3. El administrador de contenido selecciona la opción “FAQ”.	4. El sistema muestra los campos de datos obligatorios para insertar: a) Título de la página *

	b) Categoría c) Pregunta* d) Respuesta* Nota: El símbolo* señala los campos obligatorios.
5. El administrador de contenido inserta la información y selecciona la opción “Enviar”. Si desea ver una vista previa de la FAQ, ver la sección “ Vista Previa”	6. El sistema valida que no hayan campos obligatorios vacíos y que la pregunta no coincida con la de otra FAQ y adiciona la nueva FAQ

Prototipo de interfaz

Flujo alterno	
Acción del Actor	**Respuesta del Sistema**
Acción 5	6.1. Si existe algún campo vacío, este se coloreará de rojo y el sistema a vuelve a la acción 5.

Flujo Normal de Eventos	
Sección “Adicionar Listas de Chequeo”	
Acción del Actor	**Respuesta del Sistema**
3. El administrador de contenido selecciona la opción “Listas de Chequeo”.	4. El sistema muestra los campos de datos obligatorios para insertar:

	e) Título* f) Título de la página* g) Categoría* h) Cuerpo* Nota: El símbolo* señala los campos obligatorios.
5. El administrador de contenido inserta la información y selecciona la opción “Enviar”. Si desea ver una vista previa de la Listas de chequeo, ver la sección “ Vista Previa”	6. El sistema valida que no hayan campos obligatorios vacíos y adiciona la nueva Lista de Chequeo.

Prototipo de interfaz

Flujo alterno	
Acción del Actor	**Respuesta del Sistema**
Acción 5	6.1. Si existe algún campo vacío, este se coloreará de rojo y el sistema a vuelve a la acción 5.

Flujo Normal de Eventos	
Sección “Insertar Nota técnica”	
Acción del Actor	**Respuesta del Sistema**
3. El usuario selecciona la opción “Insertar nota	4. El sistema muestra los campos obligatorios a

<table>
<tr><td>técnica”.</td><td>llenar para insertar la nota.

a) Título *

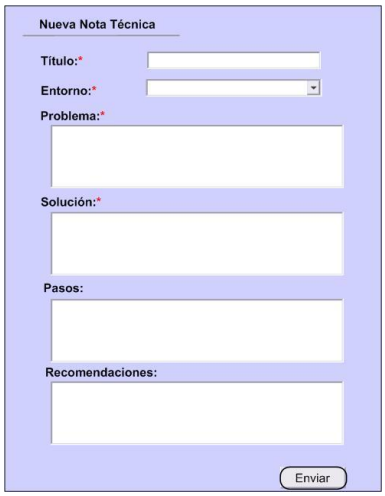
b) Entorno*

c) Problema*

d) Solución*

e) Paso a paso

f) Recomendación

Nota: El símbolo* señala los campos obligatorios.</td></tr>
<tr><td>5. El cliente llena los campos y selecciona la opción “Enviar”. Si desea ver una vista previa de la Listas de chequeo, ver la sección “ Vista Previa”</td><td>6. El sistema valida que no hayan campos obligatorios vacíos y adiciona la nueva nota técnica.</td></tr>
<tr><td colspan="2">Prototipo de interfaz
 
</td></tr>
<tr><td colspan="2">Flujo alterno</td></tr>
<tr><td>Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td>Acción 5</td><td>6.1. Si existe algún campo vacío, este se coloreará de rojo y el sistema a vuelve a la acción 5.</td></tr>
</table>

Flujo Normal de Eventos	
Sección "Vista Previa"	
Acción del Actor	**Respuesta del Sistema**
1. El cliente selecciona la opción "Vista Previa"	2. El sistema muestra una vista previa de cómo quedaría la publicación del contenido y retorna a la acción 5.
Poscondiciones:	Queda añadido un nuevo contenido a la base de datos.

Tabla 2.22. Descripción del CU Gestionar Contenido.

Caso de Uso:	**Gestionar Contenido**
Actores:	Administrador de contenido (Inicia)
Resumen:	El administrador podrá publicar, editar o eliminar los contenidos existentes. Los contenidos pueden ser nuevos foros, artículos, noticias, FAQs y listas de chequeo.
Precondiciones:	Deben existir contenidos en la lista de nuevos contenidos.
Referencias:	RF2, RF3, RF7, RF8, RF13
Prioridad:	Crítico
Flujo Normal de Eventos	
Sección "Gestionar Contenido"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador de contenido selecciona la opción "Administración de contenido".	2. El sistema muestra el listado de los nuevos contenidos. a) Para publicar contenido ir a la sección "Publicar". b) Para editar contenido ir a la sección

	"Editar". c) Para eliminar contenido a la sección "Eliminar"
Flujo Normal de Eventos	
Sección "Publicar"	
Acción del Actor	**Respuesta del Sistema**
3. El administrador de contenido selecciona el contenido (Artículo, Foro, Noticia, FAQs, Listas de Chequeo) que publicará. 4. Selecciona de la lista desplegable "Opciones de publicación" dónde desea publicar el contenido y presiona la opción "Actualizar".	5. El sistema publica el contenido donde se ha especificado.
Flujo Normal de Eventos	
Sección "Editar"	
Acción del Actor	**Respuesta del Sistema**
3. El administrador de contenido selecciona el contenido (Artículo, Foro, Noticia, FAQs, Listas de Chequeo) que desea editar. 4. El administrador de contenido selecciona la opción "Editar".	5. El sistema muestra el contenido y las opciones correspondientes de cada uno posibles a modificar o configurar.
6. El administrador de contenido modifica o configura el contenido y presiona la opción " Vista Previa"	7. El sistema valida que los campos obligatorios no estén vacíos y muestra una vista de cómo quedaría la publicación.
8. El administrador de contenido selecciona la opción "Enviar".	9. El sistema salva la modificación realizada.
Flujo Alterno	

Acción del Actor	Respuesta del Sistema
Acción 5	6.1. Si existe algún campo vacío, este se coloreará de rojo y el sistema vuelve a la acción 5.
Sección "Eliminar"	
Acción del Actor	**Respuesta del Sistema**
5. El administrador de contenido presiona la opción "Eliminar".	6. El sistema solicita una confirmación de la eliminación.
7. El administrador de contenido presiona la opción "Eliminar".	8. El contenido es borrado y el listado de contenidos actualizado.
Prototipo de interfaz 	
Flujo Alterno	
Acción del Actor	**Respuesta del Sistema**
6.1. El administrador de contenido presiona la opción "Cancelar".	7.1. El sistema vuelve a la acción 1.
Poscondiciones:	Se ha actualizado y publicado la información del contenido.

Conclusiones parciales.

A lo largo de este capítulo se hace una descripción detallada del sistema propuesto. Primeramente se desarrolló un modelo de dominio para un mejor entendimiento de los conceptos relacionados con la solución y el esclarecimiento de las funcionalidades que esta debe satisfacer. En base a ellos se llevó a cabo una modelación en términos de casos de uso, que una vez especificados garantizan una compresión exacta de lo que se desea desarrollar. Todos los artefactos generados servirán de entrada para la etapa posterior del análisis y el diseño.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA.

Introducción

En el presente capítulo se describe el flujo de trabajo Análisis y Diseño del sistema, donde se trasforma el modelo de dominio en las estructuras de datos necesarios para implementar el software. Durante el análisis se revisan los requisitos que se describieron en la captura de requisitos, para proceder a su refinamiento y estructuración. El objetivo de este flujo de trabajo es establecer una mejor comprensión de los requisitos. Mientras que en el diseño se modela el sistema, de forma tal que soporte los requisitos tanto funcionales como no funcionales y otras restricciones.

3.1. Análisis del sistema.

El análisis se basa en un modelo de objetos conceptual al cual se le llama Modelo de Análisis. El modelo de análisis ayuda a refinar los requisitos y permite razonar sobre los aspectos internos del sistema.

Ayuda también a estructurar los requisitos ya que proporciona una estructura centrada en el mantenimiento, en aspectos tales como la flexibilidad ante los cambios y la reutilización. (Jacobson, y otros)

3.1.1. Clases del Análisis.

Una clase de análisis representa una abstracción de una o varias clases y subsistemas del diseño de sistema.

Características de las clases del análisis:

- Se centran en el tratamiento de los requisitos funcionales y pospone los no funcionales.

- Definen atributos conceptuales y reconocibles en el dominio del problema. Estos atributos con frecuencia pasan a ser clases en el diseño y la implementación.

- Siempre encajan en uno de tres estereotipos básicos: de interfaz, de control o de entidad.

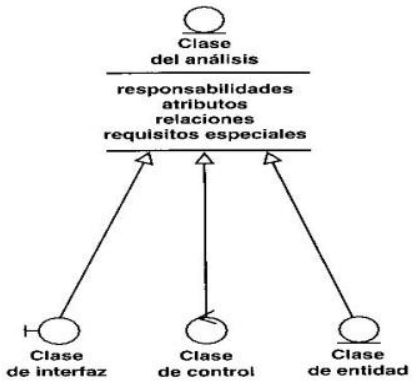


Fig 3.1. Estereotipos de clases del análisis. (Jacobson, y otros)

Estos tres estereotipos están estandarizados en UML y se utilizan para ayudar a los desarrolladores a distinguir el ámbito de las diferentes clases. (Jacobson, y otros)

Clase Interfaz

Las clases de interfaz se utilizan para modelar la interacción entre el sistema y sus actores. Esta interacción implica recibir información y peticiones de (y hacia) los usuarios y sistemas externos.

Clase Entidad

Las clases de entidad se utilizan para modelar información que posee una larga vida y que es a menudo persistente. Modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un objeto o un suceso del mundo real.

Clase Control

Las clases de control representan coordinación, secuencia, transacciones y control de otros objetos, y se usan con frecuencia para encapsular el control de un caso de uso en concreto. También se utilizan para representar derivaciones y cálculos complejos, como la lógica del negocio, que no pueden asociarse con ninguna información concreta, de larga duración, almacenada por el sistema.

3.1.2. Realización de Casos de Usos en el análisis.

La realización de los casos de usos en el análisis es una colaboración como parte del modelo de análisis en el que se muestra cómo se ejecuta un determinado caso de uso en términos de clases del análisis y de sus objetos del análisis en interacción. (Jacobson, y otros)

3.1.2.1. Diagramas de clases del análisis.

Una clase de análisis y sus objetos normalmente participan en varias realizaciones de casos de uso, y algunas de las responsabilidades, atributos y asociaciones de una clase concreta suelen ser sólo relevantes para una única realización de caso de uso. Es por esto que es importante coordinar todos los requisitos sobre una clase y sus objetos que pueden tener diferentes casos de uso (Jacobson, y otros)

Las clases del análisis representan un primer modelo conceptual de "cosas en el sistema que tienen responsabilidades y comportamientos". (Rational Software Corporation, 2003)

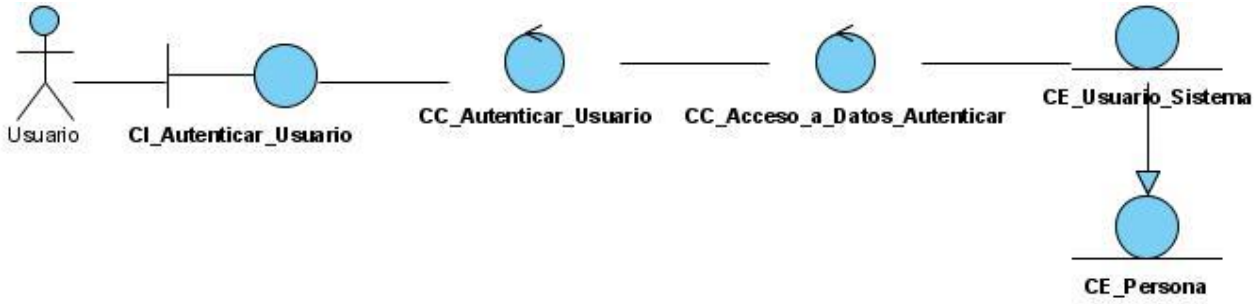


Fig 3.2 Diagrama de clases del análisis del CU Autenticar usuario.

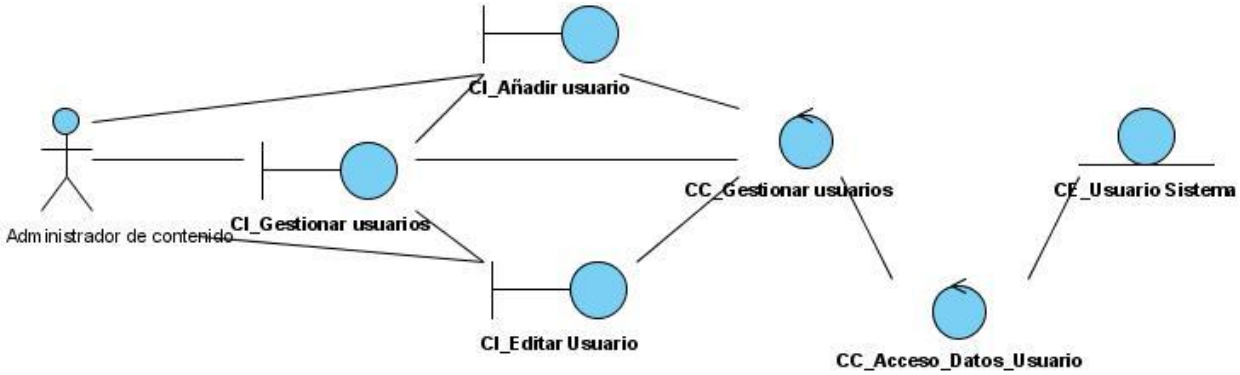


Fig 3.3. Diagrama de clases del análisis del CU Gestionar usuario.

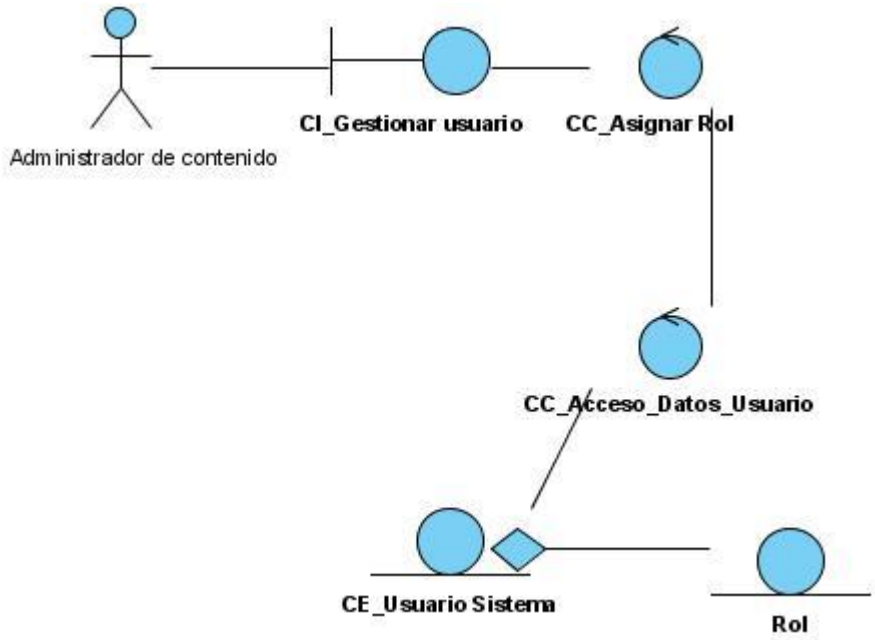


Fig 3.4 Diagrama de clases del análisis del CU Gestionar roles.

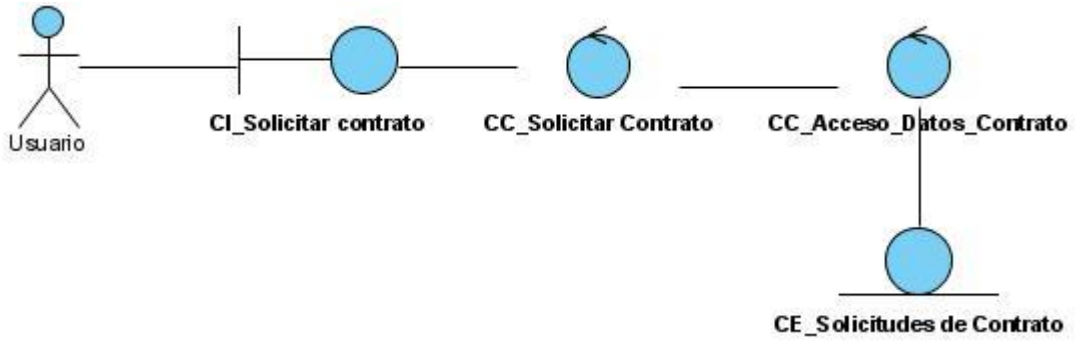


Fig 3.5 Diagrama de clases del análisis del CU Solicitar Contrato.

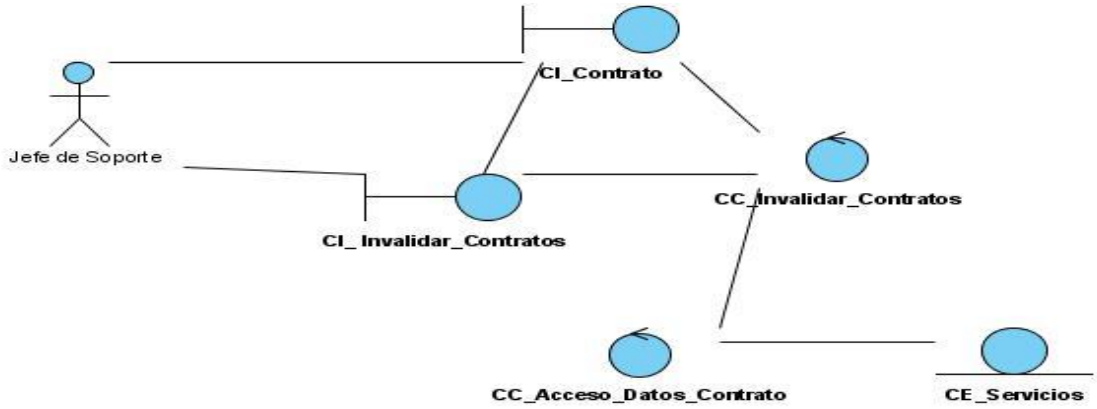


Fig 3.6 Diagrama de clases del análisis del CU Invalidar Contrato.

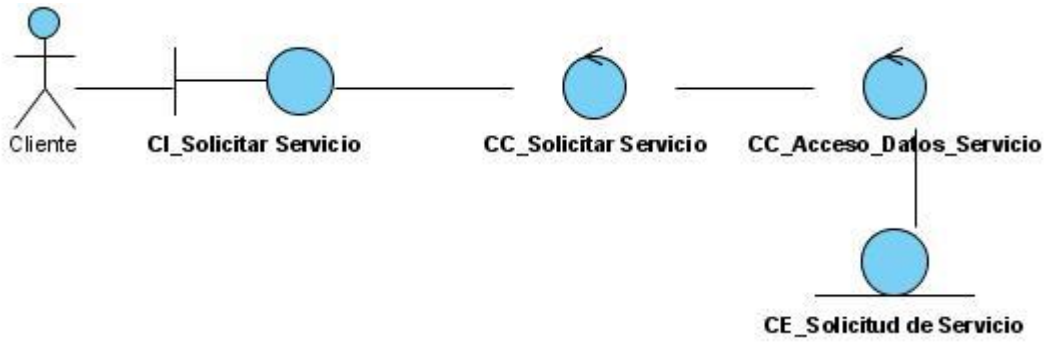


Fig 3.7 Diagrama de clases del análisis del CU Solicitar Servicio

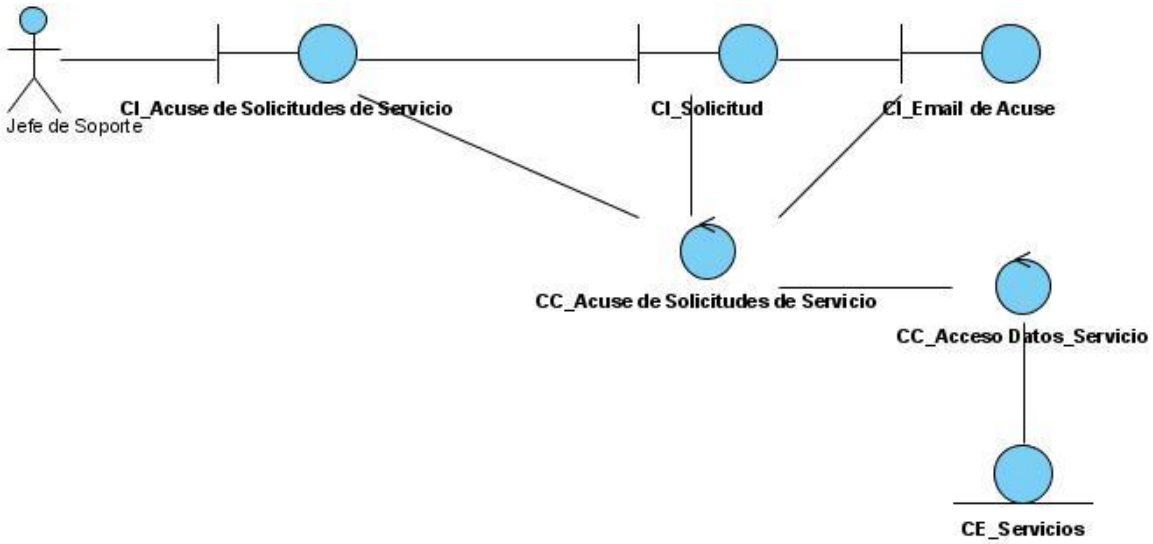


Fig 3.8 Diagrama de clases del análisis del CU Responder Solicitud de Servicio.

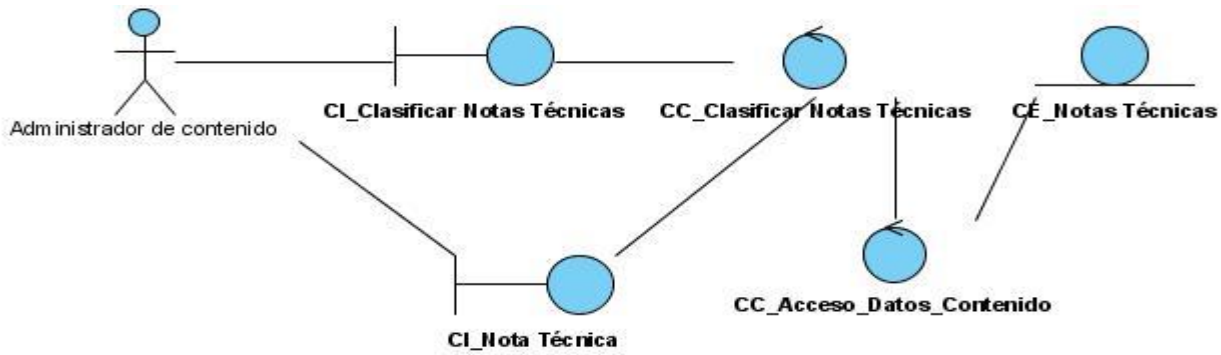


Fig 3.9 Diagrama de clases del análisis del CU Clasificar Notas Técnicas

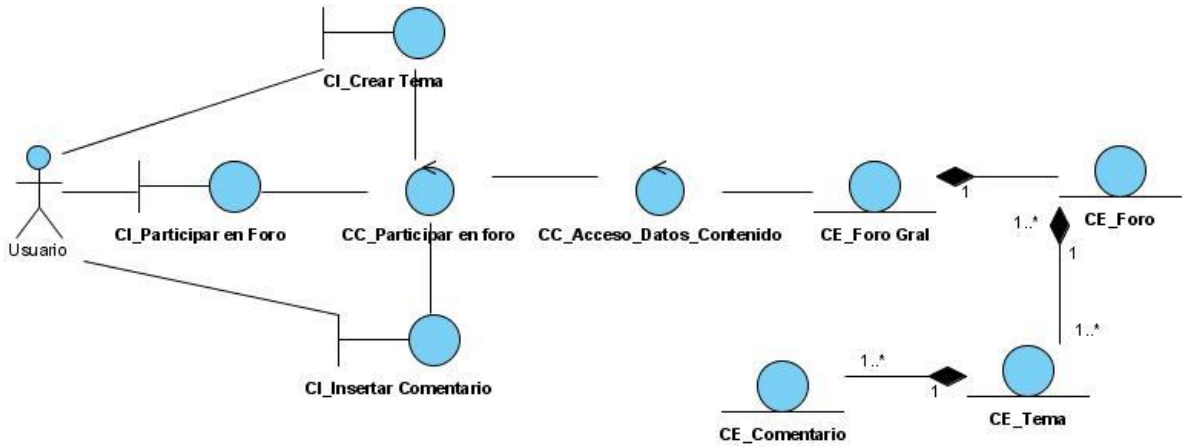


Fig 3.10 Diagrama de clases del análisis del CU Participar en foro

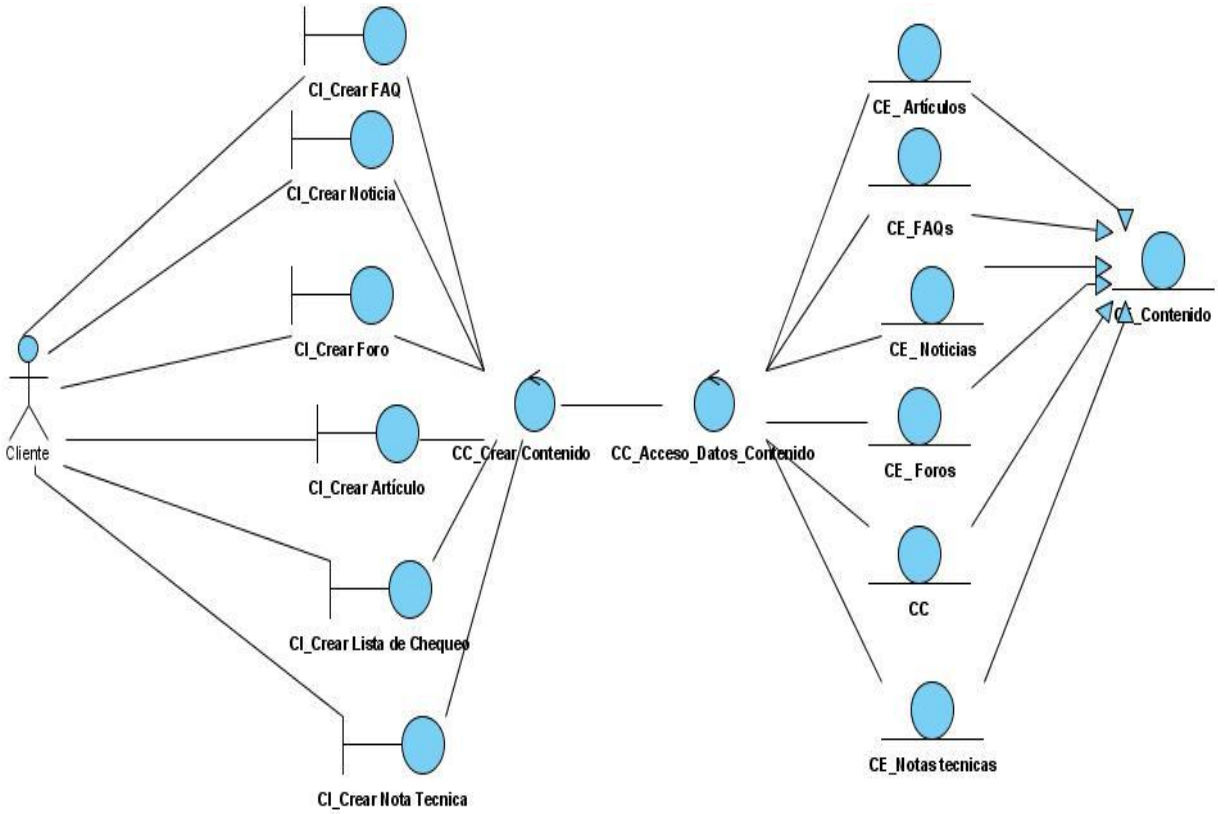


Fig 3.11 Diagrama de clases del análisis del CU Crear Contenido

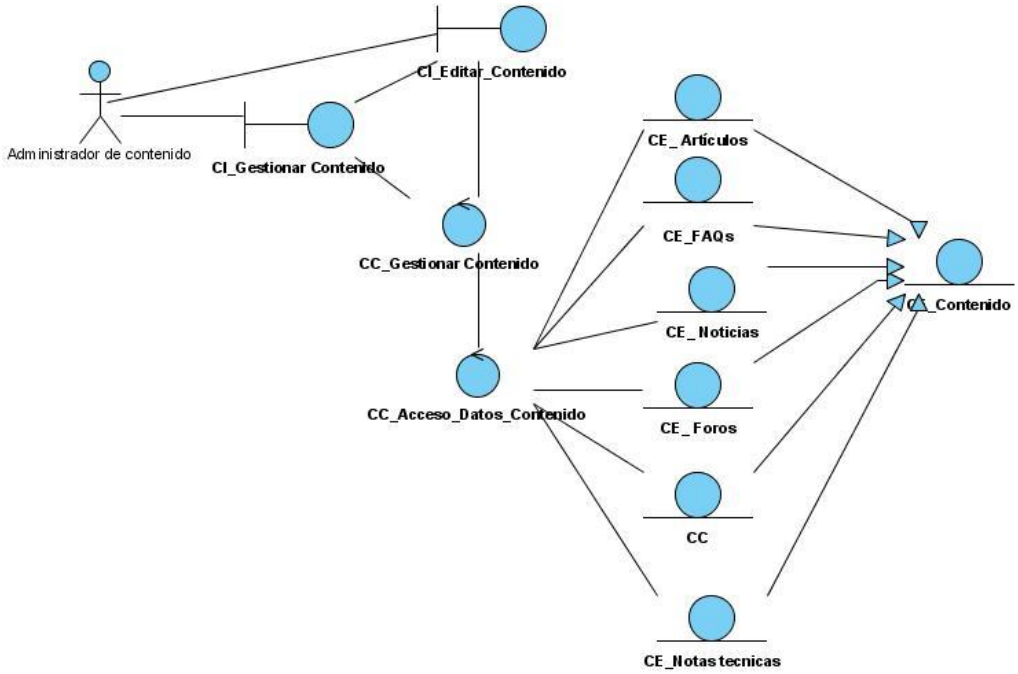


Fig 3.12 Diagrama de clases del análisis del CU Gestionar Contenido

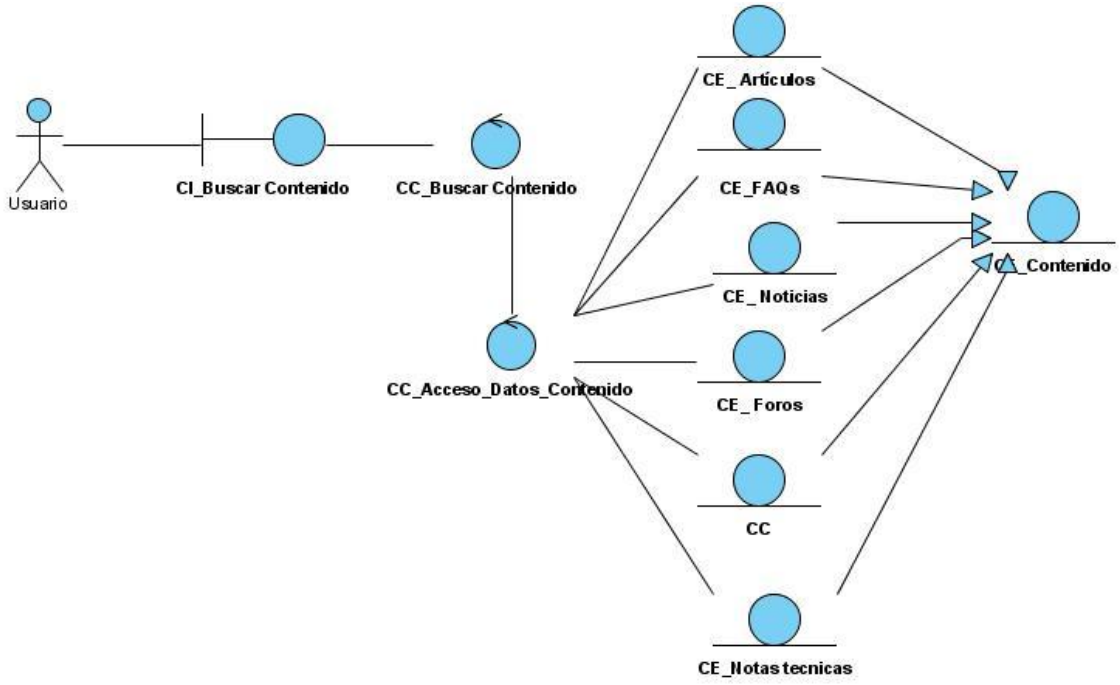


Fig 3.13 Diagrama de clases del análisis del CU Buscar Contenido

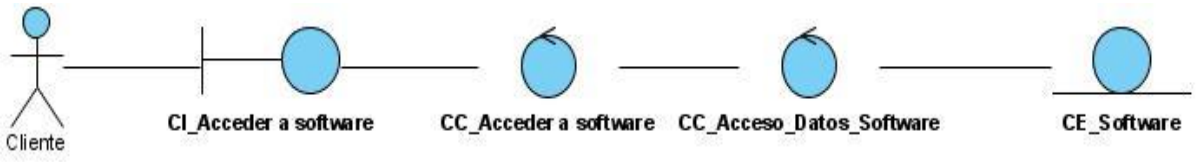


Fig 3.14 Diagrama de clases del análisis del CU Acceder software.

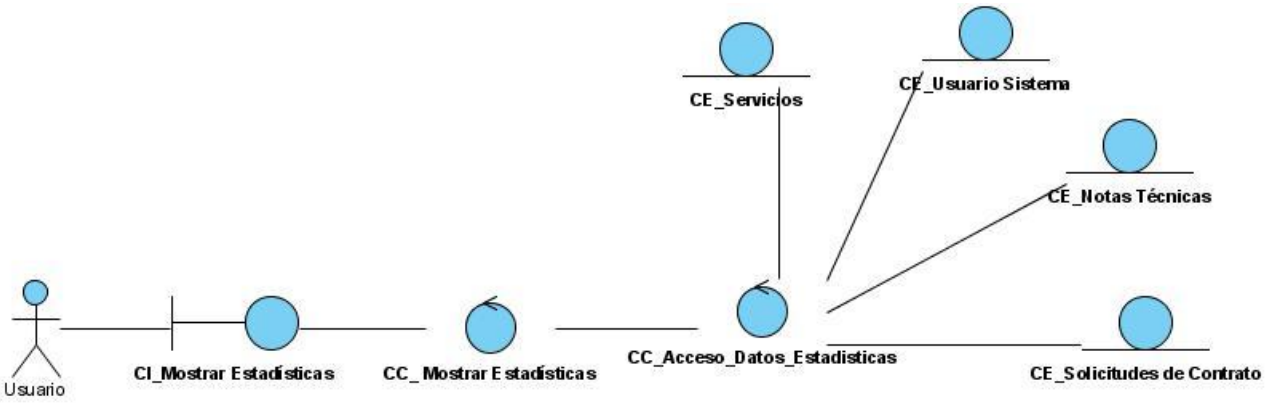


Fig 3.15 Diagrama de clases del análisis del CU Mostrar Estadísticas

3.1.1.2. Paquetes del análisis.

Los paquetes del análisis proporcionan un medio de organización para los artefactos del modelo de análisis en piezas manejables. Un paquete de análisis puede constar de clases de análisis, de realizaciones de casos de uso y de otros paquetes del análisis.

Estos paquetes deben ser cohesivos y débilmente acoplados, es decir, sus contenidos deben estar fuertemente relacionados y sus dependencias unos de otros deben minimizarse. (Jacobson, y otros)

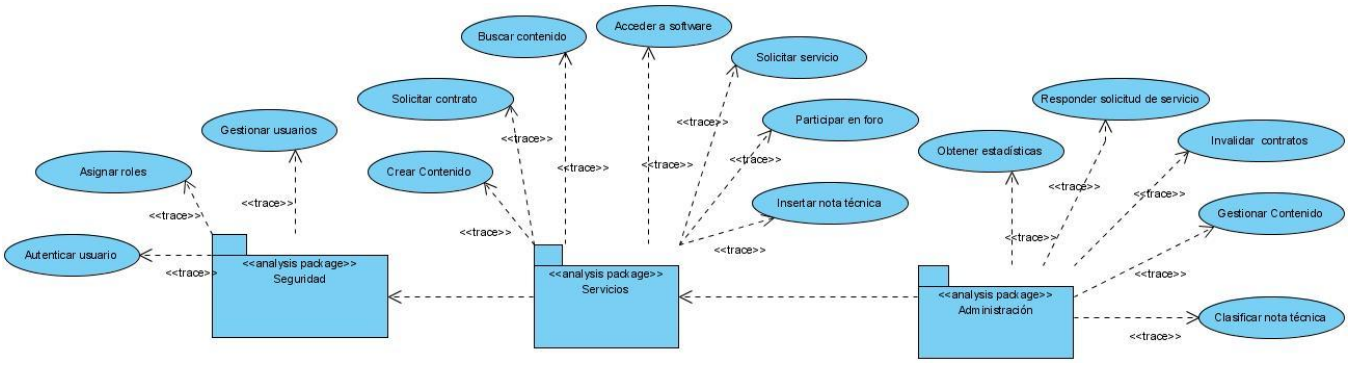


Fig 3.16 Diagrama del Paquetes del análisis

3.1.1.3. Diagramas de Interacción del Análisis (Colaboración).

Un diagrama de colaboración es una representación gráfica que refleja los objetos que participan en la realización de un caso de uso o un escenario de este, además de sus interacciones a través de enlaces y los mensajes que se envían unos a otros.

(Ver Anexos II)

3.2. Diseño del sistema.

El diseño del sistema contribuye a una arquitectura estable y sólida y en la creación del modelo de implementación. El modelo de diseño se puede utilizar para visualizar la implementación y para soportar las técnicas de programación gráfica. (Jacobson, y otros)

3.2.1. Patrones de diseño utilizados.

En búsqueda de un diseño claro, flexible y reutilizable se han tenido en cuenta la aplicación de patrones de diseño GOF:

El patrón **Singleton** (instancia única) garantiza que solo haya una única instancia de una clase y esta sea accesible desde cualquier punto de la aplicación. Es necesario cuando hay clases que tienen que gestionar de manera centralizada un recurso. Un ejemplo es el caso de las dos clases gestoras Gestionar usuario y Gestionar contenido, donde una variable global no garantiza que sólo se instancie una vez.

El patrón **Cadena de responsabilidad** permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada. Se aplica cuando más de un objeto tiene que actuar sobre una llamada y no se quiere construir el conocimiento de estas interacciones dentro del programa cliente. Este patrón se ve aplicado cuando la clase de acceso a datos para autenticar necesita llevar a cabo la autenticación mediante LDAP o por el sistema, donde si uno de estos objetos no sabe cómo responder a una solicitud en específico o no está en condiciones, la pasa a lo largo del árbol de objetos, a su sucesor. Como el nombre lo implica, cada objeto a lo largo de la ruta de la solicitud puede tomar la responsabilidad y atender la solicitud.

3.2.2. Clases del Diseño.

Una clase de diseño es una abstracción sin costuras de una clase o construcción similar en la implementación del sistema.

Fig. 3.17 Esquema clases del diseño. (Jacobson, y otros)

Extensiones para diseño web.

Debido a que la solución informática a realizar que se plantea es una aplicación web se decidió que la realización de este modelado sea a través de estereotipos web.

Estereotipos:

1.  *Server Page:* representa la página Web que tiene código que se ejecuta en el servidor.
2.  Client Page: una instancia de Página Cliente es una página Web, con formato HTML.
3.  *Form:* colección de elementos de entrada que son parte de una página cliente.
4. <<*Build*>>*:* representa una asociación especial que relaciona las páginas cliente con las páginas servidor.
5. <<*Link*>>*:* expresa las asociaciones más comunes entre las páginas, en este caso la del hipervínculo.
6. <<*Submit*>>*:* es la relación que se crea siempre entre una página servidor y un formulario.

7. <<*Redirect*>>*:* representa el redireccionamiento hacia otra página.

8. <<*Aggregated by*>>*:* representa la relación de agregación, como por ejemplo la de los formularios en las páginas clientes.

3.2.3. Realizaciones de casos de uso del diseño

3.2.3.1. Diagramas de clases del diseño.

Una clase de diseño y sus objetos participan en varias realizaciones de casos de uso. También puede suceder que algunas operaciones, atributos y asociaciones sobre una clase específica sean solo relevantes para una sola realización de caso de uso. Para manejar todo esto se utilizan los diagramas de clases conectados a una realización de caso de uso, mostrando así sus clases participantes, subsistemas y sus relaciones. (Jacobson, y otros)

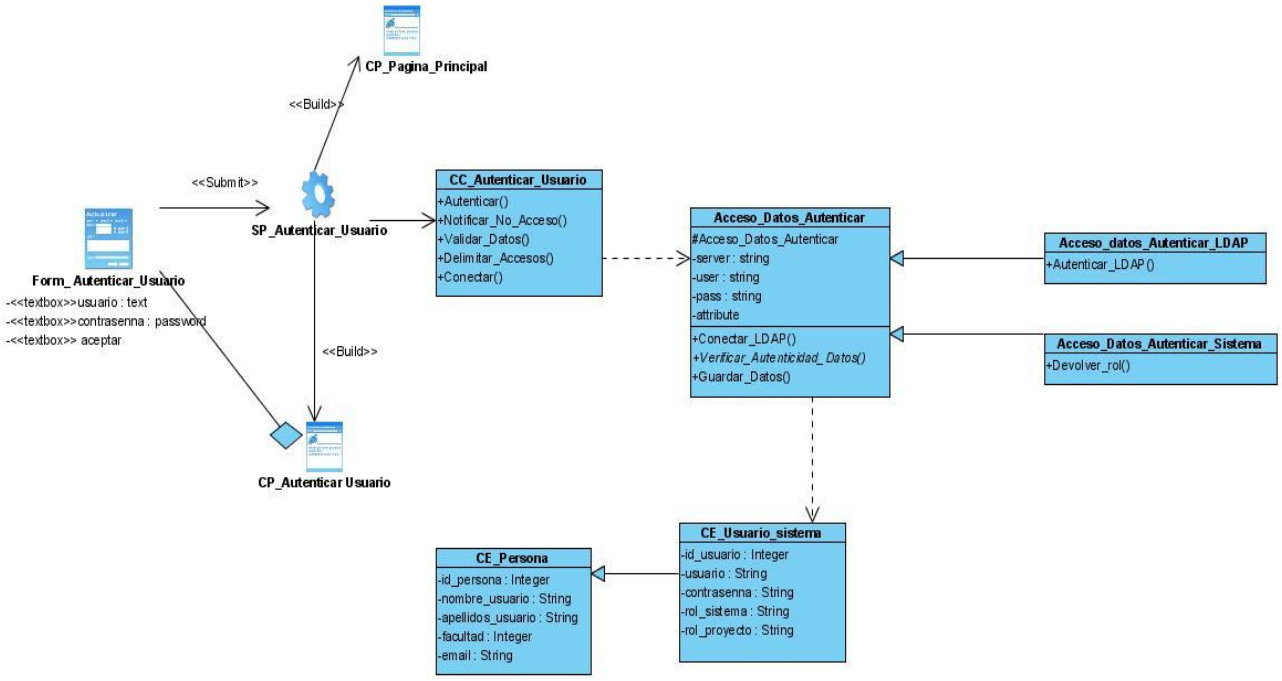


Fig 3.18 Diagrama de clases del diseño CU Autenticar Usuario

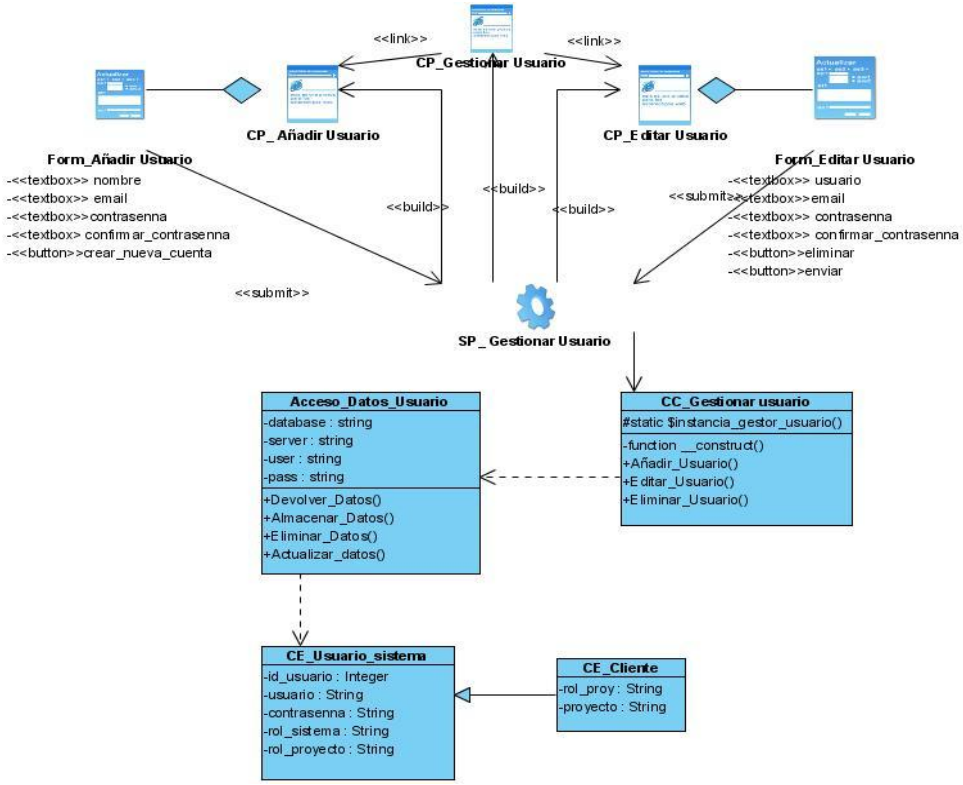


Fig 3.19 Diagrama de clases del diseño CU Gestionar Usuario

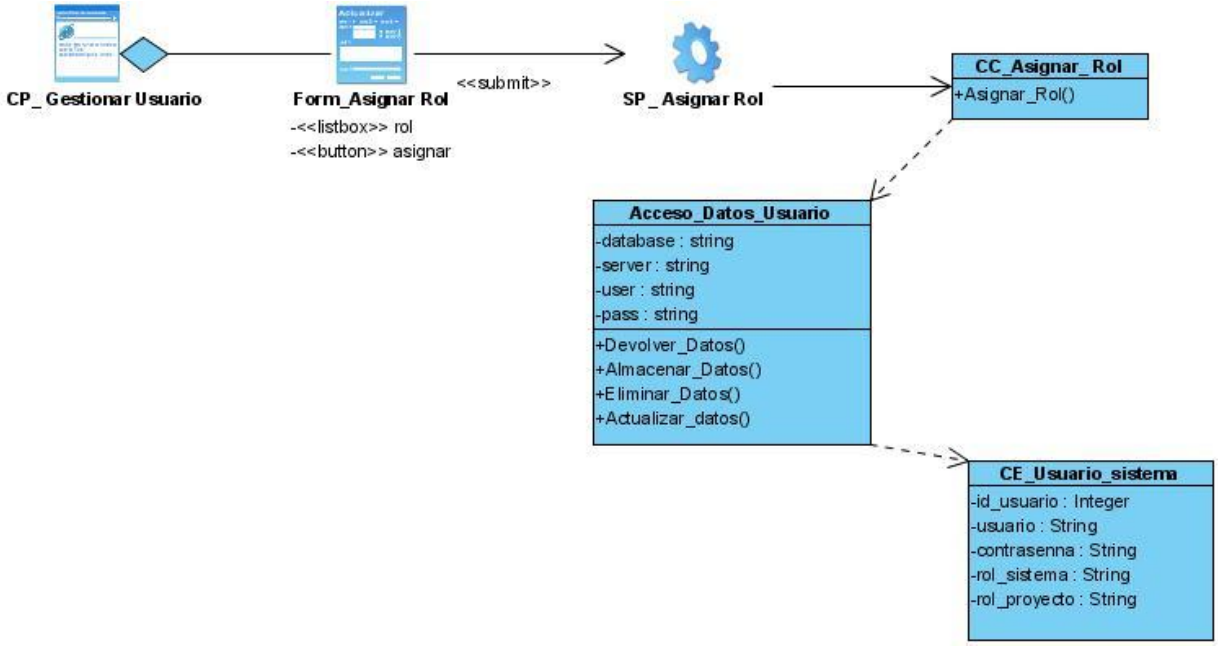


Fig 3.20 Diagrama de clases del diseño CU Asignar Rol

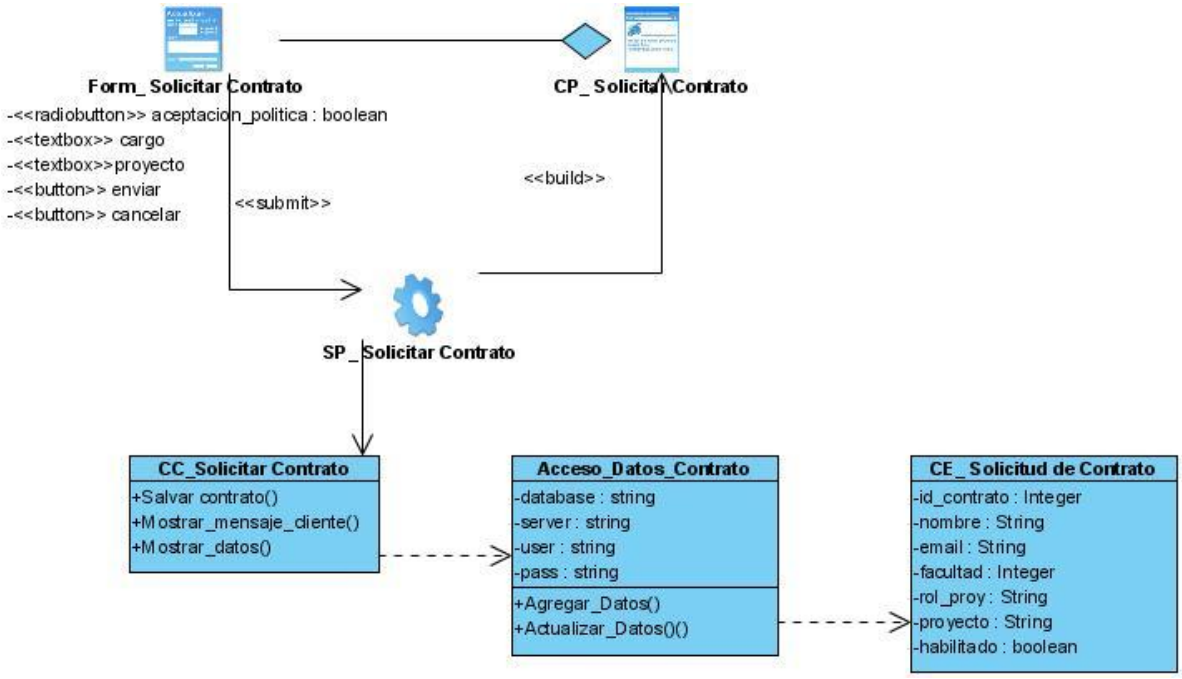


Fig 3.21 Diagrama de clases del diseño CU Solicitar Contrato

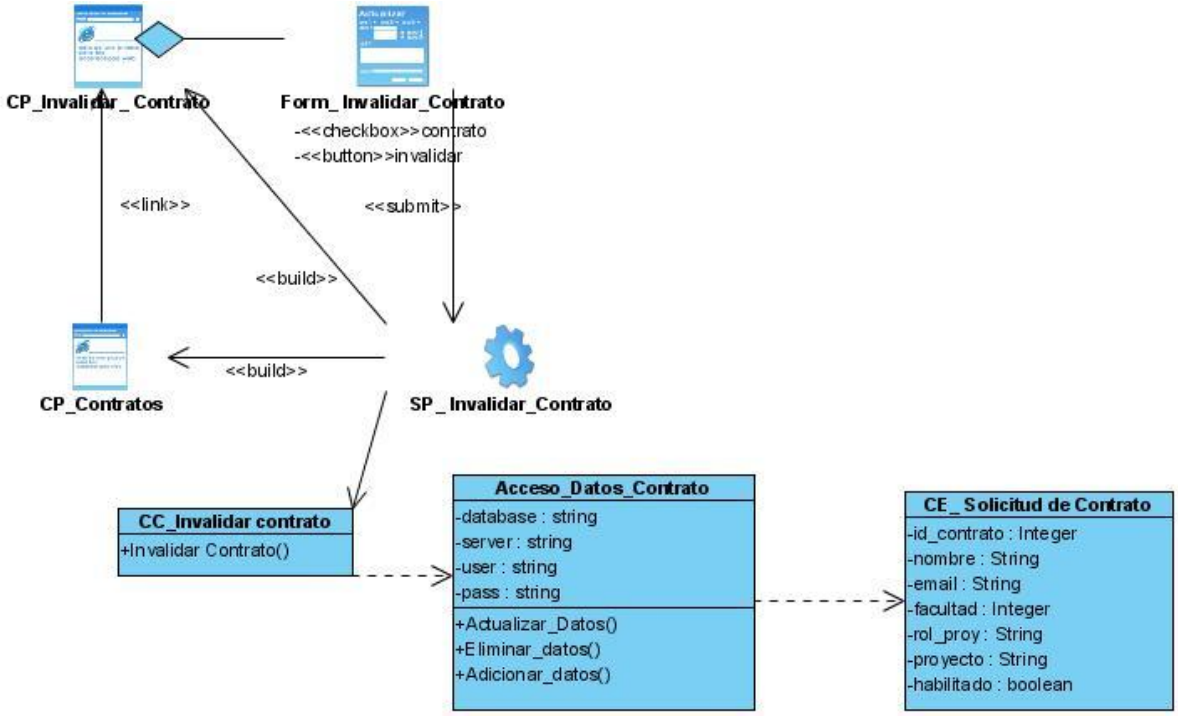


Fig 3.22 Diagrama de clases del diseño CU Invalidar Contrato

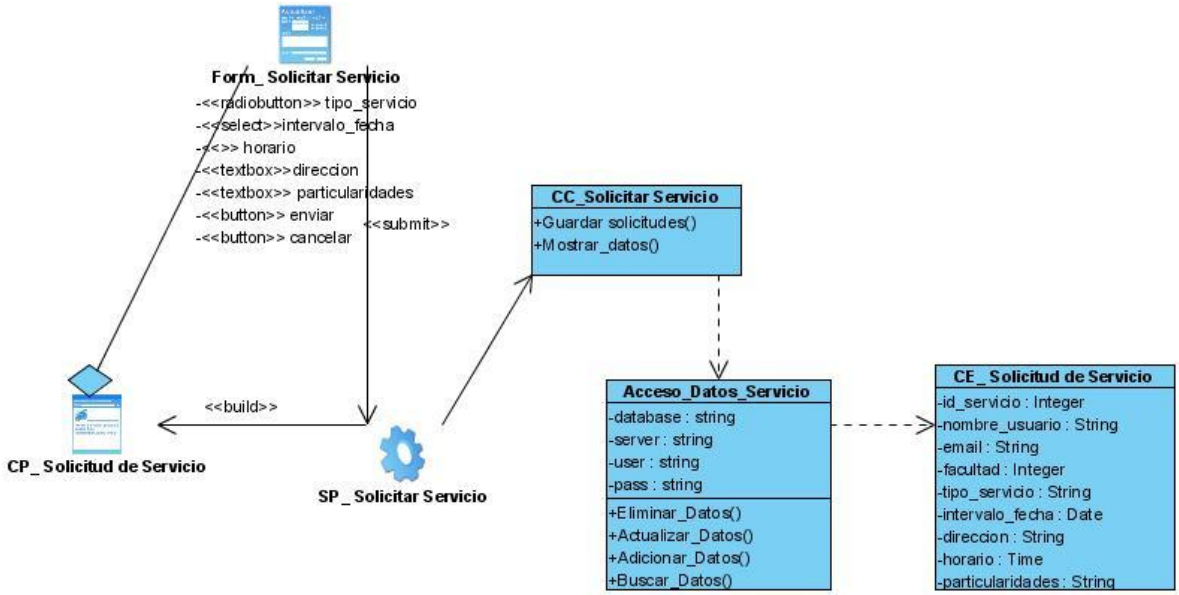


Fig 3.23 Diagrama de clases del diseño CU Solicitar Servicio

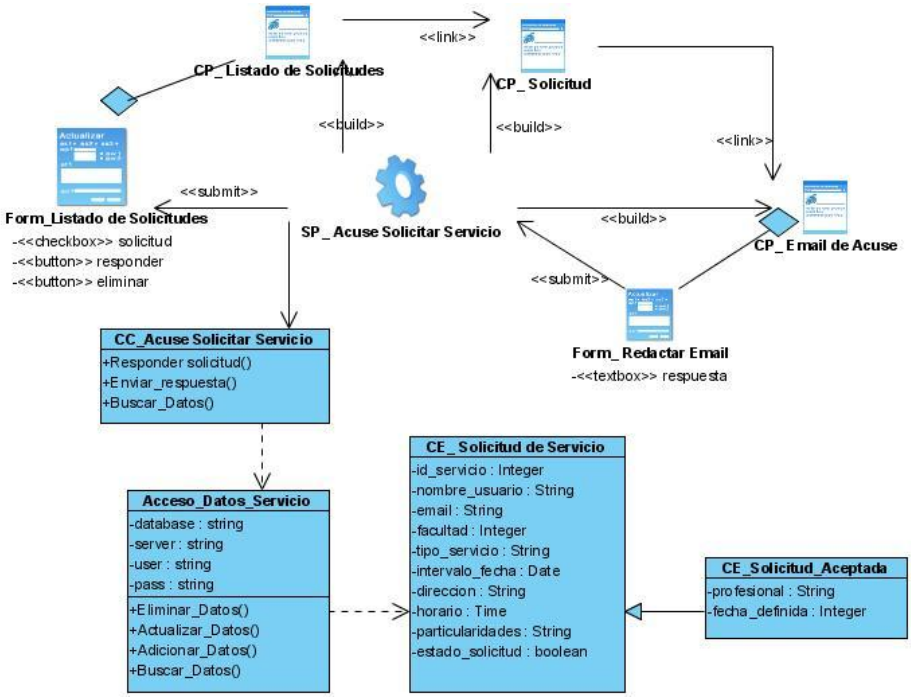


Fig 3.24 Diagrama de clases del diseño CU Responder Solicitud de Servicio

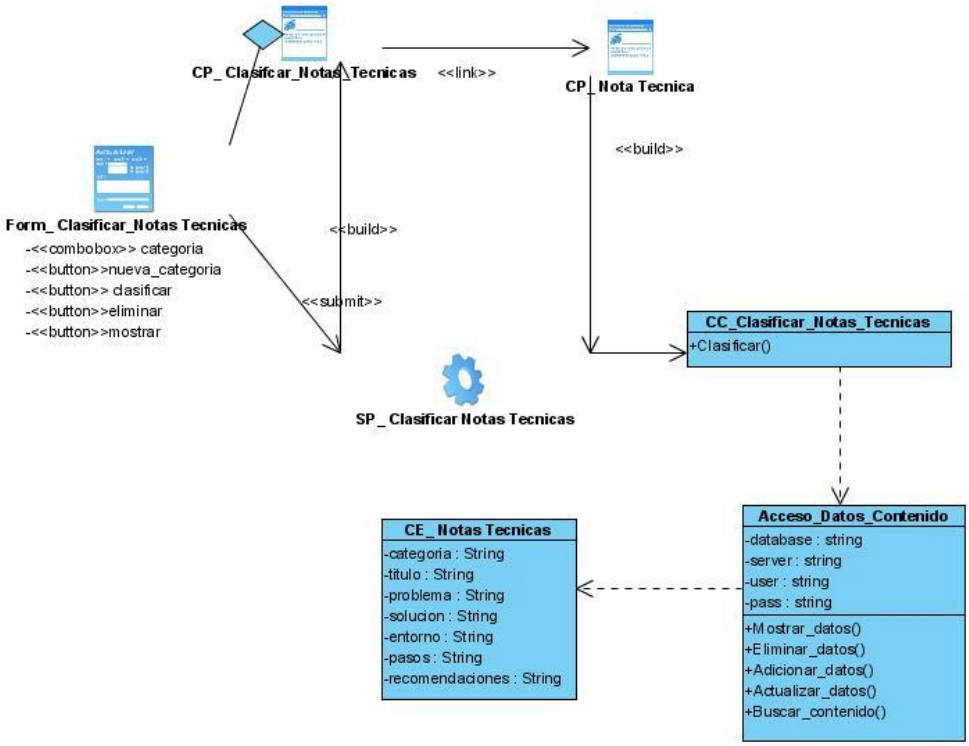


Fig 3.25 Diagrama de clases del diseño CU Clasificar Notas Técnicas

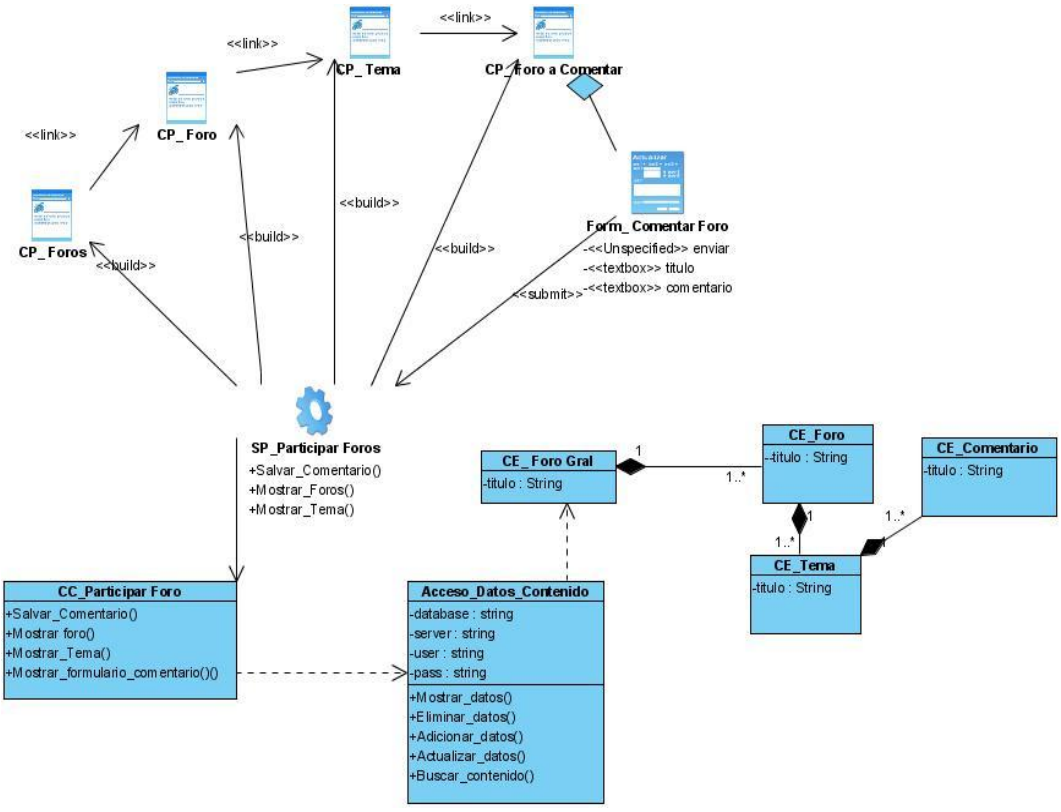


Fig 3.26 Diagrama de clases del diseño CU Participar Foro

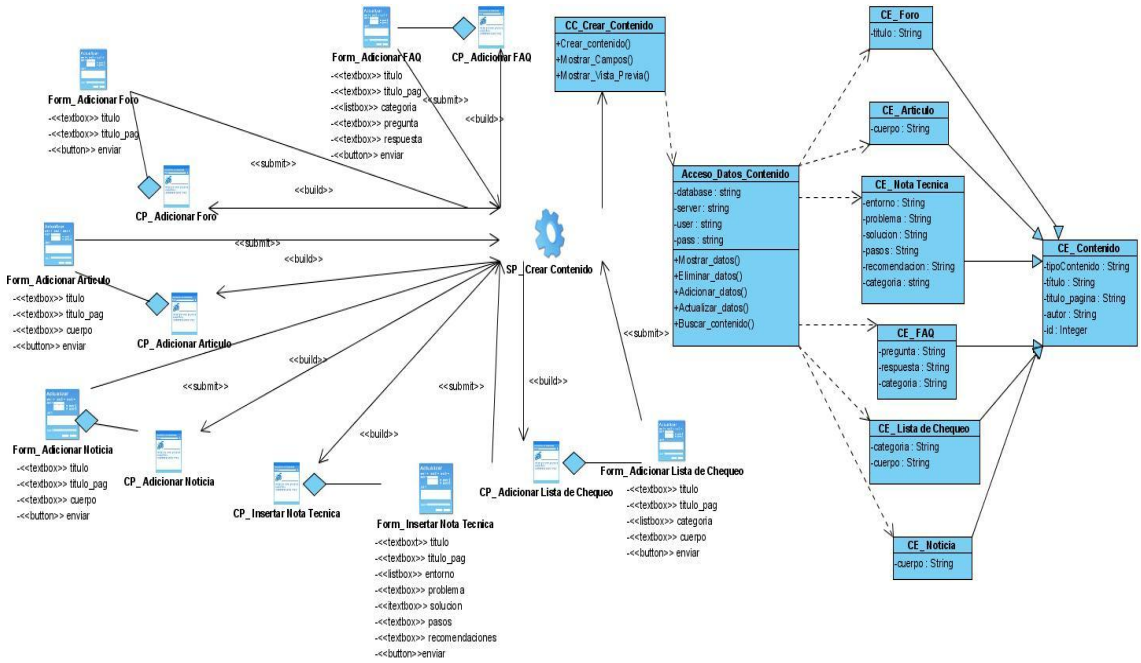


Fig 3.27 Diagrama de clases del diseño CU Crear Contenido

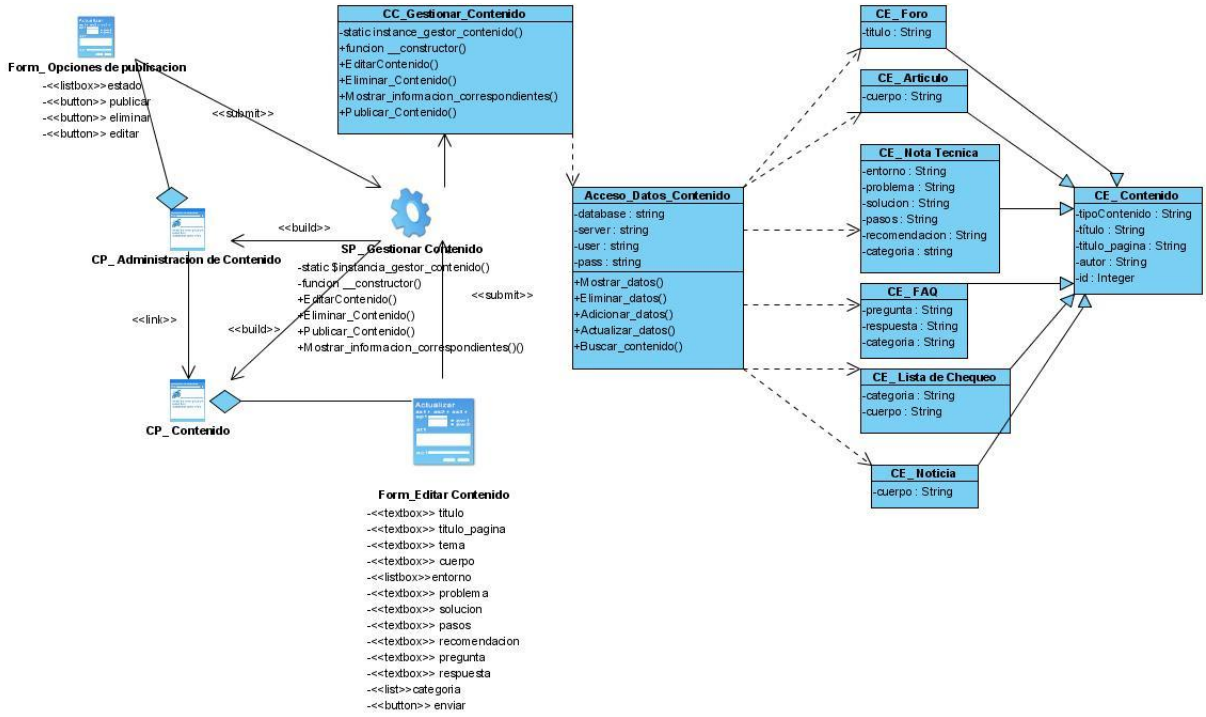


Fig 3.28 Diagrama de clases del diseño CU Gestionar Contenido

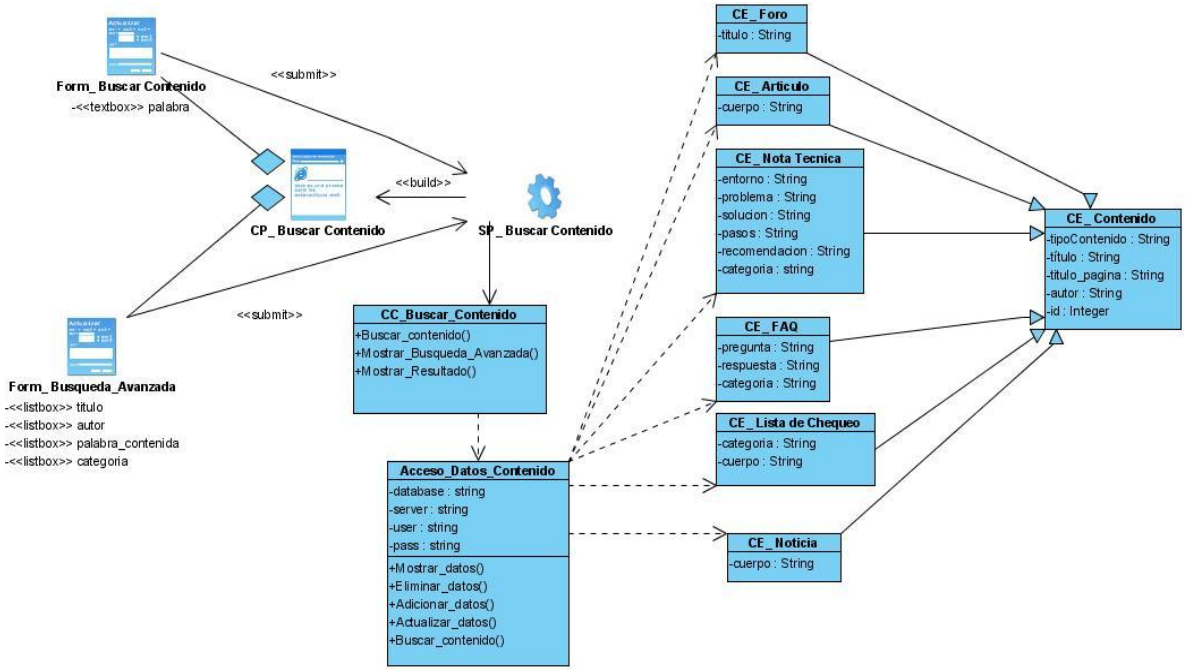


Fig 3.29 Diagrama de clases del diseño CU Buscar Contenido

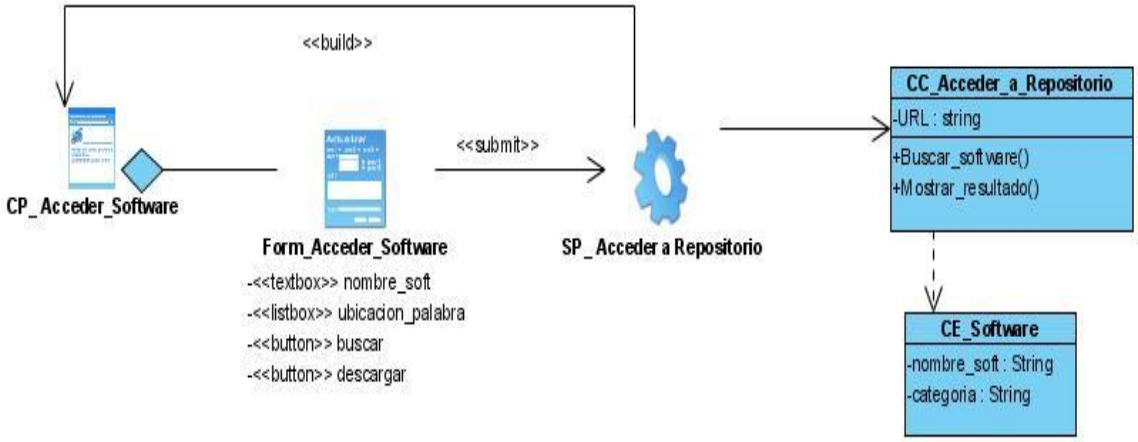


Fig 3.30 Diagrama de clases del diseño CU Buscar Software

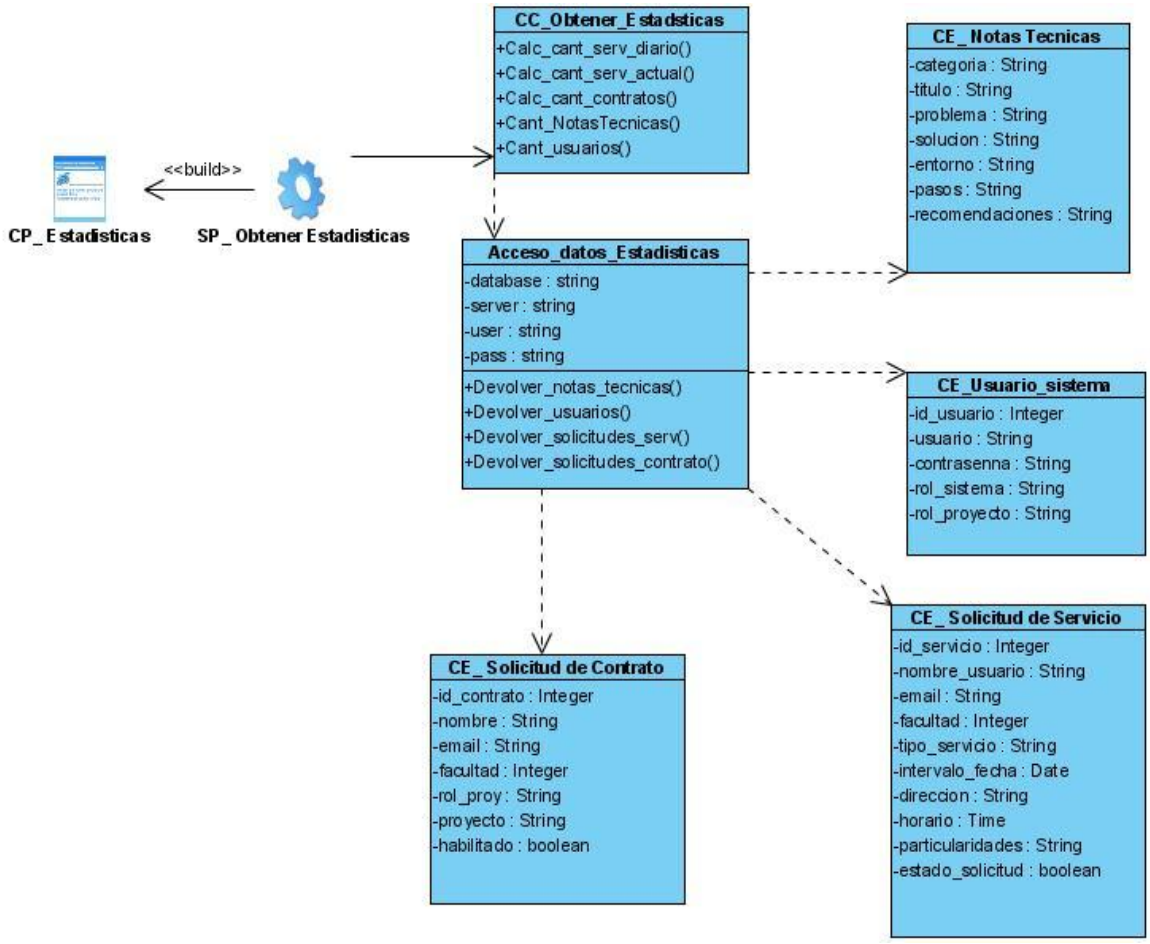


Fig 3.31 Diagrama de clases del diseño CU Obtener Estadísticas

3.2.3.2 Paquetes del diseño.

Al igual que los paquetes del análisis, los paquetes del diseño son una colección de clases, relaciones, realizaciones de casos de usos y otros paquetes. Son usados para agrupar elementos del modelo de diseño que están relacionados, con el propósito de organizarlos. (Rational Software Corporation, 2003)

Los elementos dentro del paquete pueden depender de los elementos contenidos en otros paquetes, esto da surgimiento a la dependencia entre paquetes, la cual puede ser usada como una herramienta para analizar la resistencia del modelo de diseño: un modelo con paquetes dependientes es menos resistente al cambio.

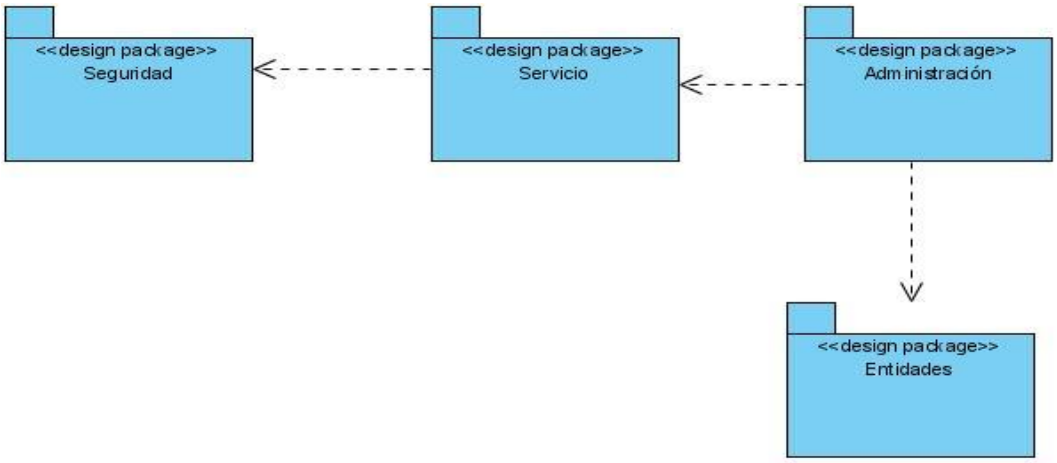


Fig. 3.32. Diagramas de paquetes de diseño.

3.2.3.3. Diagrama de dependencias entre paquetes del análisis y el diseño.

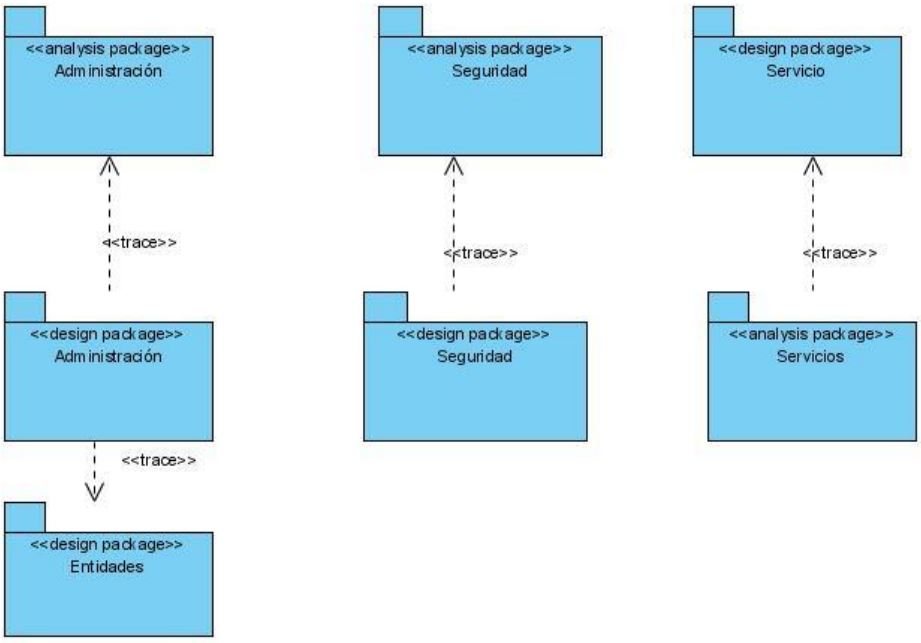


Fig. 3.33 Diagrama de dependencias entre paquetes del análisis y el diseño.

3.2.3.4. Diagramas de Interacción del Diseño (Secuencia).

La secuencia de acciones en un caso de uso comienza cuando el actor invoca el caso de uso mediante el envío de algún tipo de mensaje al sistema. Si se tiene en cuenta el interior de dicho sistema se tendrá algún objeto de diseño que reciba el mensaje del actor. Luego, el objeto de diseño

llama a otro objeto, y de esta manera los objetos implicados interactúan para realizar y llevar a cabo el caso de uso.

En el diseño se representa esto a través de diagramas de secuencia, ya que el objetivo principal es encontrar secuencias de interacciones detalladas y ordenadas en el tiempo.

En los diagramas de secuencia se muestran las interacciones entre objetos mediante transferencias de mensajes entre objetos o subsistemas.

(Ver Anexos III)

3.2.3.5. Descripciones de clases del diseño.

Nombre: Autenticar_ Usuario	
Clase Controladora	
Métodos:	
Nombre:	Autenticar()
Descripción:	Autentica al usuario a través del LDAP y lo registra en la BD.
Nombre:	Notificar_ No_ Acceso()
Descripción:	Notifica que el usuario no tiene acceso a la aplicación.
Nombre:	Validar_ Datos()
Descripción:	Valida los datos de los usuarios.
Nombre:	Delimitar_ Accesos()
Descripción:	Delimita accesos a los usuarios según su rol dentro de la aplicación.
Nombre:	Conectar()
Descripción:	Se conecta a la capa de acceso a datos.

Nombre: Acceso_ Datos_ Autenticar	
Clase de Acceso a Datos	
Atributo	**Tipo**
Database	String
Server	String
User	String

Pass	String
Métodos:	
Nombre:	Conectarse_ LDAP()
Descripción:	Se conecta al sistema LDAP de la uci.
Nombre:	Verificar_ Autenticidad_ Datos()
Descripción:	Verifica si los datos del usuario son correctos.
Nombre:	Guardar_ Datos()
Descripción:	Guarda los datos del usuario en el BD.

Nombre: Acceso_ Datos_ Autenticar_ LDAP	
Clase de Acceso a Datos	
Métodos:	
Nombre:	Autenticar_ LDAP()
Descripción:	Autentica al usuario a través del LDAP de la UCI.

Nombre: Acceso_ Datos_ Autenticar_ Sistema	
Clase de Acceso a Datos	
Métodos:	
Nombre:	Devolver_ Rol()
Descripción:	Devuelve el rol del usuario autenticado.

Nombre: Asignar_ Rol	
Clase Controladora	
Métodos:	
Nombre:	Asignar_ Rol()
Descripción:	Asigna rol a un usuario.

Nombre: Acceso_ Datos_ Usuario	
Clase Acceso_Datos_Usuario	
Atributo	**Tipo**

Database	String
Server	String
User	String
Pass	String
Métodos:	
Nombre:	Devolver_ Datos()
Descripción:	Devuelve los datos del usuario al que se le va a asignar un rol.
Nombre:	Almacenar_ Datos()
Descripción:	Almacena los datos en la BD del sistema.
Nombre:	Eliminar_ Datos()
Descripción:	Elimina los datos del usuario.
Nombre:	Actualizar_ Datos()
Descripción:	Actualiza los nuevos datos del usuario que se encuentra en la BD del sistema.

Nombre: Invalidar_ Contrato	
Clase Controladora	
Métodos:	
Nombre:	Invalidar_ Contrato()
Descripción:	Invalida los contratos solicitados por los usuarios.

Nombre: Acceso_ Datos_ Contrato	
Clase Acceso a Datos	
Atributo	**Tipo**
Database	String
Server	String
User	String
Pass	String
Métodos:	
Nombre:	Actualizar_ Datos()
Descripción:	Actualiza los nuevos datos introducidos en la BD.

Nombre:	Eliminar_ Datos()
Descripción:	Elimina los datos de la BD.
Nombre:	Adicionar_ Datos()
Descripción:	Adiciona nuevos datos a la BD.

Nombre: Acuse_ Solicitar_ Servicio	
Clase Controladora	
Métodos:	
Nombre:	Responder_ Solicitud()
Descripción:	Responde las solicitudes de servicios de los clientes.
Nombre:	Enviar_ Respuesta()
Descripción:	Envía respuesta a las solicitudes.
Nombre:	Mostrar_ Datos()
Descripción:	Muestra los datos del servicio.

Nombre: Acceso_ Datos_ Servicio	
Clase Acceso a Datos	
Atributo	**Tipo**
Database	String
Server	String
User	String
Pass	String
Métodos:	
Nombre:	Eliminar_ Datos()
Descripción:	Elimina los datos de la BD.
Nombre:	Actualizar_ Datos()
Descripción:	Actualiza los datos en la BD.
Nombre:	Adicionar_ Datos()
Descripción:	Adiciona nuevos datos en la BD.
Nombre:	Buscar_ Datos()
Descripción:	Hace una búsqueda de datos de servicio.

Nombre: Clasificar_ Notas_ Técnicas	
Clase Controladora	
Métodos:	
Nombre:	Clasificar()
Descripción:	Clasifica las notas técnicas según su categoría.

Nombre: Acceso_ Datos_ Contenido	
Clase Controladora	
Atributo	**Tipo**
Database	String
Server	String
User	String
Pass	String
Métodos:	
Nombre:	Mostrar_ Datos()
Descripción:	Muestra los datos del contenido.
Nombre:	Eliminar_ Datos()
Descripción:	Elimina los datos del contenido.
Nombre:	Adicionar_ Datos()
Descripción:	Adiciona nuevos datos del contenido.
Nombre:	Actualizar_ Datos()
Descripción:	Actualiza los nuevos datos del contenido en la BD.
Nombre:	Buscar_ Contenido()
Descripción:	Hace una búsqueda del contenido en la BD.

Nombre: Buscar_ Contenido	
Clase Controladora	
Métodos:	
Nombre:	Buscar_ Contenido()
Descripción:	Hace una búsqueda del contenido según palabras claves descritas por el usuario.
Nombre:	Mostrar_ Resultado()

Descripción:	Muestra el resultado de la búsqueda.
Nombre:	Mostrar_ Búsqueda_ Avanzada()
Descripción:	Muestra la opción de hacer búsqueda avanzada a través de criterios.

Nombre: Acceder_ a_ Repositorio	
Clase Controladora	
Atributo	**Tipo**
URL	String
Métodos:	
Nombre:	Buscar_ Software()
Descripción:	Hace una búsqueda de los softwares disponibles en la red.
Nombre:	Mostrar_ Resultado()
Descripción:	Muestra el resultado de la búsqueda.

Nombre: Obtener_ Estadísticas	
Clase Controladora	
Métodos:	
Nombre:	Calc_ cant_ serv_ diario()
Descripción:	Calcula la cantidad de servicios coordinados por día.
Nombre:	Calc_ cant_ serv_ actual()
Descripción:	Calcula la cantidad de servicios coordinados hasta la fecha.
Nombre:	Calc_ cant_ contratos()
Descripción:	Calcula la cantidad total de contratos establecidos.
Nombre:	Cant_ NotasTécnicas()
Descripción:	Cuenta la cantidad de notas técnicas existentes en la BD.
Nombre:	Cant_ Usuarios()
Descripción:	Cuenta la cantidad de usuarios registrados en la BD.

Nombre: Acceso_ Datos_ Estadísticas	
Clase Controladora	
Atributo	**Tipo**
Database	String
Server	String
User	String
Pass	String
Métodos:	
Nombre:	Devolver_ Notas_ Técnicas()
Descripción:	Devuelve los datos referentes a las notas técnicas.
Nombre:	Devolver_ Usuarios()
Descripción:	Devuelve los datos referentes a los usuarios.
Nombre:	Devolver_ Solicitudes_ Servicio()
Descripción:	Devuelve los datos referentes a las solicitudes de servicio.
Nombre:	Devolver_ Solicitudes_ Contrato()
Descripción:	Devuelve los datos referentes a las solicitudes de contrato.

Nombre: Crear_ Contenido	
Clase Controladora	
Métodos:	
Nombre:	Crear_ Contenido()
Descripción:	Crea un contenido nuevo para insertar en la BD.
Nombre:	Mostrar_ Campos()
Descripción:	Muestra los campos necesarios para insertar el nuevo contenido.
Nombre:	Mostrar_ Vista_ Previa()
Descripción:	Muestra una vista previa del contenido creado.

Nombre: Gestionar_ Contenido
Clase Controladora
Métodos:

Nombre:	Static_ $instancia_ gestor_ contenido()
Descripción:	Crea una instancia de un objeto de la clase dando respuesta al patrón Singleton.
Nombre:	function_ construct()
Descripción:	Hace la función de construir la clase.
Nombre:	Editar_ Contenido()
Descripción:	Ofrece hacer cambios a contenidos creados.
Nombre:	Eliminar_ Contenido()
Descripción:	Elimina el contenido creado que el administrador de contenido desee eliminar.
Nombre:	Publicar_ Contenido()
Descripción:	Publica el contenido que el administrador de contenido desee publicar.
Nombre:	Mostrar_ Información_ Correspondiente()
Descripción:	Muestra la información correspondiente a la solicitud del usuario.

Nombre: Gestionar_ Usuario	
Clase Controladora	
Métodos:	
Nombre:	function_ construct()
Descripción:	Hace la función de construir la clase.
Nombre:	Static_ $instancia_ gestor_ usuario()
Descripción:	Crea una instancia de un objeto de la clase dando respuesta al patrón Singleton.
Nombre:	Añadir_ Usuario()
Descripción:	Agrega un usuario nuevo al sistema.
Nombre:	Editar_ Usuario()
Descripción:	Permite hacer cambios en los datos del usuario.
Nombre:	Eliminar_ Usuario()
Descripción:	Elimina usuarios del sistema.

Nombre: Solicitar_ Contrato

Clase Controladora	
Métodos:	
Nombre:	Salvar_ Contrato()
Descripción:	Guarda el contrato en la BD.
Nombre:	Mostrar_ Mensaje_ Cliente()
Descripción:	Muestra al usuario un mensaje donde se le notifica que ya es cliente.
Nombre:	Mostrar_ Datos()
Descripción:	Muestra los datos del contrato.

Nombre: Acceso_ Datos_ Contrato	
Clase Controladora	
Atributo	**Tipo**
Database	String
Server	String
User	String
Pass	String
Métodos:	
Nombre:	Agregar_ Datos()
Descripción:	Agrega nuevos datos del contrato en la BD.
Nombre:	Actualizar_ Datos()
Descripción:	Actualiza los nuevos datos en la BD.

Nombre: Solicitar_ Servicio	
Clase Controladora	
Métodos:	
Nombre:	Guardar_ Solicitudes()
Descripción:	Guarda las solicitudes de servicios en la BD.
Nombre:	Mostrar_ Datos()
Descripción:	Muestra los datos del servicio solicitado.

Nombre: Participar_ en_ Foro	
Clase Controladora	
Métodos:	
Nombre:	Salvar_ Comentario()
Descripción:	Guarda el comentario del foro en la BD.
Nombre:	Mostrar_ Foros()
Descripción:	Muestra los foros disponibles del sistema.
Nombre:	Mostrar_ Temas()
Descripción:	Muestra los distintos temas creados en el foro.
Nombre:	Mostrar_ Formulario_ Comentario()
Descripción:	Muestra el formulario donde el cliente inserta el comentario.

3.4. Diseño de Base de datos

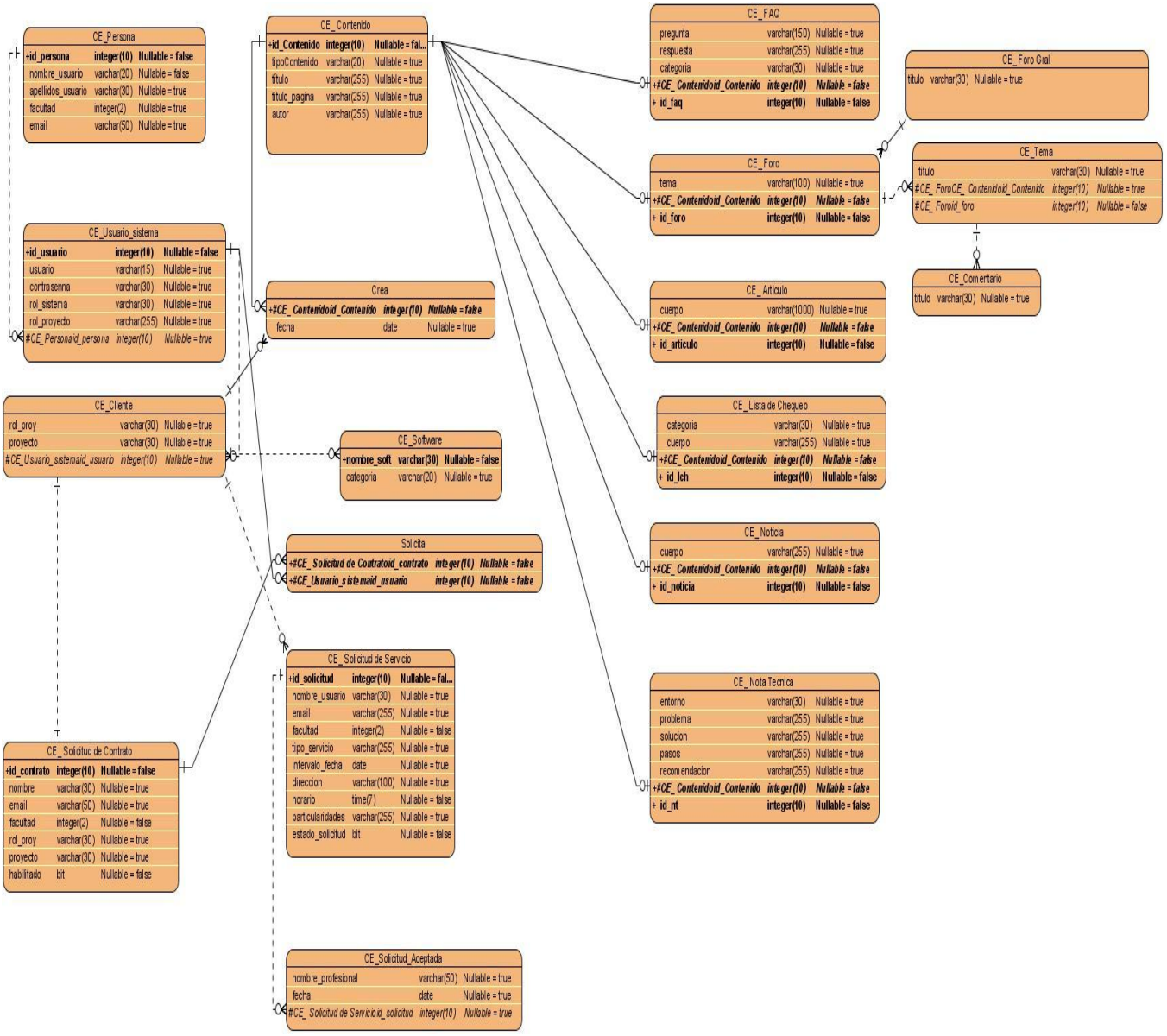


Fig. 3.34. Diagrama Entidad-Relación

Conclusiones Parciales

En el transcurso de este capítulo se obtuvieron los artefactos más significativos del análisis y el diseño de la aplicación. Estos servirán de entrada para el flujo de implementación, facilitando un mayor entendimiento de la aplicación a los desarrolladores.

Capítulo 3

Análisis y Diseño del Sistema

En el análisis se desarrollaron las clases de análisis y se llevó a cabo la realización de los Casos de Uso, generando los diferentes diagramas de clases e interacción y los paquetes del análisis.

En el diseño se realizaron las clases del diseño y los distintos diagramas de clases del diseño e interacción.

Además, en este capítulo se llevó a cabo también el diseño de la Base de Datos.

CAPÍTULO 4: ANÁLISIS DE RESULTADOS.

Introducción

En este capítulo se hace un análisis de cada uno de los resultados obtenidos a través del desarrollo de la solución propuesta, a la cual se le aplican métricas técnicas para su evaluación.

El objetivo de las pruebas es el de encontrar el mayor número de errores posibles en un tiempo real.

Los modelos de análisis y diseño no pueden probarse en el sentido convencional, ya que no pueden ejecutarse. Sin embargo, existen otras formas de evaluar estos modelos mediante la utilización de revisiones técnicas formales, las cuales se emplean para examinar la exactitud y consistencia de ambos modelos de análisis y diseño y a través, además, de la aplicación de métricas. El objetivo de las métricas en general es entender y mejorar la calidad del producto, evaluar la efectividad del proceso y mejorar la calidad del trabajo realizado a nivel de proyecto. (PRESSMAN 1998)

Las métricas internas se aplican a un producto de software no ejecutable durante todas las etapas de su desarrollo. Permiten medir la calidad de los entregables intermedios y predecir la calidad del producto final. Además, le permiten al usuario iniciar acciones correctivas temprano en el ciclo de desarrollo.

4.1 Análisis del Modelo de Dominio.

El modelado del dominio permitió conocer de manera general el funcionamiento del sistema a implementar propuesto como solución a la problemática planteada. Mediante el mismo se brindó una visión de todos los procesos que se llevan a cabo en el desarrollo del sistema, además de la identificación de todos los productos entregables de gran importancia para el dominio del sistema. Tuvo como objetivo principal el de proporcionar una comprensión común de los conceptos encontrados en el entorno del dominio.

A través de este modelo se identificaron las reglas del negocio y se llevó a cabo un glosario de términos, el cual propició el establecimiento de un lenguaje común para facilitar la comprensión entre las distintas entidades que interactúen, en un futuro, con el sistema a implementar.

Con la realización del modelo de dominio se dio paso al flujo de trabajo Levantamiento de los Requisitos.

4.2 Análisis de la Especificación de requisitos.

La etapa de levantamiento de requisitos es fundamental en el desarrollo de un software. Uno de los principales objetivos de esta etapa es el de servir de base a clientes, usuarios y desarrolladores con el propósito de llegar a un acuerdo sobre las necesidades que debe satisfacer el sistema a implementar. Es por esto que la calidad de la especificación de los requisitos es un elemento crucial para la buena marcha del proyecto de desarrollo de software y asegurar así la calidad del producto final. Por lo que se han ideado formas de evaluación para dichas especificaciones. La verificación de requisitos puede considerarse como la actividad cuyo objetivo primordial es alcanzar la calidad interna, mientras que la validación tiene como objetivo asegurar que los requisitos elicitados, documentados, analizados y verificados representen realmente las necesidades de clientes y usuarios.

Para facilitar la verificación y validación de los requisitos se hace uso de un conjunto de heurísticas, las cuales tienen como objetivo identificar los casos de uso que potencialmente puedan contener defectos, basándose para ello en los valores de ciertas métricas que pueden calcularse de forma automática mediante herramientas de gestión de requisitos. Si el valor de estas métricas esta fuera del rango habitual establecido en las heurísticas correspondientes entonces un caso de uso se considera como potencialmente defectuoso y debe ser revisado. (Una propuesta para la verificación de requisitos basada en métricas, 2004)

4.2.1 Métricas de calidad para la Especificación de Requisitos.

Esta métrica fue propuesta por Alan Davis en 1993 (PRESSMAN, 2005), corresponde a la calidad de la especificación de requerimientos. Incluye características como: especificidad, compleción, corrección, comprensión, verificación, consistencia, logro, concisión, trazabilidad, modificación, exactitud, reutilización.

Aplicación de la métrica.

Primeramente se tiene a nr que representa el total de requisitos de la especificación:

nr=nf+nnf

donde nf es el número de requisitos funcionales y nnf es el número de requisitos no funcionales. Luego se puede medir la especificidad o ausencia de ambigüedad con la fórmula:

Q=nui/nr, donde

nui es el número de requerimientos donde todos los revisores tuvieron interpretaciones idénticas. Mientras más cerca esté Q del valor 1 menor ambigüedad de la especificación.

Para la evaluación de la métrica de la especificación de los requisitos se llevó a cabo dos revisiones a diferentes versiones con el objetivo de comparar el trabajo realizado y mejorar su calidad.

Revisiones realizadas a las especificaciones de requisitos.	
Revisión 1	**Revisión 2**
$n_f = 30$, $n_{nf} = 10$;	$n_f = 32$, $n_{nf} = 20$;
$n_r = n_f + n_{nf} = 30+10$,	$n_r = n_f + n_{nf} = 32+20$
$n_r = 40$.	$n_r = 52$.
entonces $Q = n_{ui}/n_r$, donde	entonces $Q = n_{ui}/n_r$, donde
$n_{ui} = 30 - 5(F, NF)$; por lo que	$n_{ui} = 52 - 1(F)$; por lo que
$n_{ui} = 25$	$n_{ui} = 51$
$Q = 25/40 = 0.63$	$Q = 51/52 = 0.98$

Tabla 4.1 Comparación de las revisiones realizadas a las especificaciones.

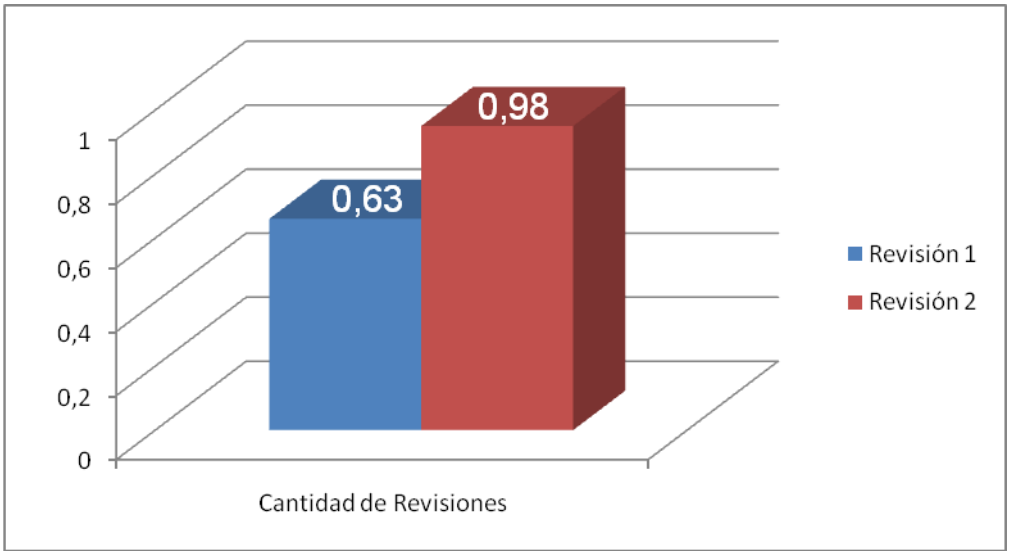


Fig. 4.1: Gráfico de Control de la Calidad de Especificación de Requisitos.

Resultado del análisis.

Tras analizar los resultados alcanzados en las distintas revisiones realizadas, se puede llegar a la conclusión de que los requisitos obtenidos en el flujo de trabajo Levantamiento de Requisitos poseen ausencia de ambigüedad prácticamente en un 100%, ya que el valor de Q se encuentra muy cercano a 1.

4.2.2 Modelo de métricas para Casos de Uso.

Heurísticas de especificación de requisitos.

En la siguiente tabla se muestran algunas de las métricas principales de las heurísticas y sus rangos habituales.

Métricas Principales	Rango Habitual	Descripción
NOS	[4,9]	Número de pasos
NOAS/NOS	[30%, 60%]	Proporción de pasos de actor

Tabla 4.2 Métricas Principales y Rango Habitual. (Una propuesta para la verificación de requisitos basada en métricas, 2004)

Heurística basada en NOAS/NOS.

Esta heurística se basa en la idea de que un caso de uso sirve básicamente para expresar una interacción actor-sistema. Por lo que el número de pasos de actor y el de pasos de sistema deben estar en torno al 50%, considerando también la posibilidad de que existan pasos de inclusión o extensión en los que se realice otro caso de uso.

Las situaciones que llevan a esta métrica fuera del rango habitual son:

- El hecho de obviar la participación del sistema, por lo que el caso de uso resulta incompleto.
- El hecho de haber desglosado demasiado las acciones de un actor determinado.
- El hecho de incluir interacciones de actores con el entorno del sistema o con otros actores.

(Una propuesta para la verificación de requisitos basada en métricas, 2004)

Aplicación de la métrica.

Nombre del CU	NOAS	NOS	Valor de la Métrica
Autenticar Usuario	1	3	33.4%
Gestionar Usuario	7	16	43.8%
Asignar Rol	2	4	50.0%
Solicitar Contrato	4	9	44.5%
Invalidar Contrato	3	7	42.8%
Solicitar Servicio	2	4	50.0%
Acuse de Solicitudes de Servicio	4	8	50.0%
Clasificar Notas Técnicas	5	10	50.0%
Participar en Foro	5	10	50.0%
Crear Contenido	14	28	50.0%
Gestionar Contenido	10	16	62.5%
Buscar Contenido	4	8	50.0%
Acceder a Software	2	4	50.0%
Obtener Estadísticas	1	2	50.0%

Resultado del análisis.

Con la aplicación de la heurística basada en NOAS/NOS se llegó a la conclusión de que las especificaciones de los casos de uso se encuentran desarrollados de forma correcta para su uso en la etapa de Análisis y Diseño, ya que el flujo de pasos que los describe están dentro del rango habitual de la heurística.

Métricas Orientadas a Objetos para la calidad del diagrama de casos de uso.

Para evaluar la calidad del diagrama de casos de uso se aplicaron además un conjunto de atributos fundamentales:

- **Completitud:** grado en que se ha logrado detallar todos los casos de uso relevantes.

- **Consistencia:** grado en que los casos de uso del sistema describen las interacciones adecuadas entre usuario y sistema.

- **Correctitud:** grado en que las interacciones actor/sistema soportan adecuadamente el proceso del negocio.

- **Complejidad:** grado de claridad en la presentación de los elementos que describen el contexto y la claridad del sistema.

Aplicación de la métrica.

Atributo	Factores	Valor
Completitud	1. ¿Han sido definidos todos los roles relevantes de usuario encargados de generar/ modificar o consultar información?	Cantidad de roles relevantes definidos: 4 Número de roles relevantes omitidos: 0 Representa un 0%
	2. ¿Se presenta una descripción resumida de todos los conceptos del dominio?	Número de conceptos del dominio: 20 Número de conceptos que no presentan una descripción detallada: 0 Representa un 0%

	3. ¿Están definidos todos los requisitos que justifican la funcionalidad del caso de uso?	Cantidad de requisitos: 32 Número de requisitos omitidos por caso de uso: 0 Representa un 0% Número de casos de uso: 14 Cantidad de casos de uso que tienen requisitos omitidos: 0 Representa un 0%
	4. ¿Todos los casos de uso han sido clasificados de acuerdo a su relevancia en (crítico, secundario, auxiliar, opcional)?	Total de casos de uso: 14 Cantidad de casos de uso que no han sido clasificados: 0 Representa un 0%
	5. ¿Se presenta una descripción detallada (descripción extendida esencial) de todos los casos de uso del sistema?	Total de casos de uso: 14 Cantidad de casos de uso que no han sido clasificados: 0 Representa un 0%
	6. ¿Se presenta una descripción resumida (descripción de alto nivel) de todos los casos de uso?	Total de casos de uso: 14 Cantidad que no tienen una descripción resumida: 0 Representa un 0%
		Valor Total: 100%
Consistencia	1. ¿Está adecuadamente redactado (en el lenguaje del usuario) el flujo de eventos?	Total de casos de uso: 14 Cantidad que no tienen el flujo de eventos redactado en el lenguaje del usuario: 0

		Representa un 0%
	2. ¿Representa el caso de uso una interacción observable por un actor?	Total de casos de uso: 14 Cantidad que no representan una interacción observable por un actor: 0 Representa un 0%
	3. ¿El nombre dado a los casos de uso es una expresión verbal que describe alguna funcionalidad relevante en el contexto del usuario?	Total de casos de uso: 14 Cantidad que tienen un nombre incorrecto: 0 Representa un 0%
	4. ¿Existe una adecuada separación entre el flujo básico de eventos y los flujos alternos y/o flujos subordinados?	Total de casos de uso: 14 Cantidad de casos de uso complejos que no tienen una separación del flujo básico y de flujos alternos: 1 Representa un 7.14%
		Valor Total: 98.22%
Correctitud	1. ¿Representa el caso de uso requisitos comprensibles por el usuario?	Total de requisitos: 32 Cantidad de requisitos que no son comprensibles por el usuario: 0 Representa: 0% Total de casos de Uso: 14 Número de casos de uso en que los requisitos representados no son

		comprensibles por el usuario: 0 Representa: 0%
Complejidad	1. ¿Los elementos dentro del diagrama están adecuadamente ubicados de manera que facilitan su interpretación?	Total de casos de Uso: 14. Número de casos de uso que requieren reubicación de manera que faciliten su interpretación: 0 Representa: 0%
		Valor Total: 100%

Resultado del análisis.

En el factor de consistencia existe un caso de uso complejo que no tiene separación entre flujo básico de eventos y el flujo alterno, por lo que este factor se ve afectado en un 98.22%.

En los demás factores las métricas aplicadas resultaron satisfactorias en un 100%.

4.3 Métricas para el Modelo de Diseño.

Las métricas para diseño proporcionan al diseñador una mejor visión interna, ayudan a que el diseño evolucione a un nivel superior de calidad.

Para la aplicación de estas métricas Berard [Laranjeira ´90] define cinco características fundamentales.

1. Localización.
2. Encapsulamiento.
3. Ocultamiento de la información.
4. Herencia.
5. Técnicas de abstracción de objetos.

4.3.1 Métricas para el Modelo de Diseño Orientado a Objetos (OO).

A medida que los modelos de diseño OO van creciendo en tamaño y complejidad es aconsejable y beneficioso tener una visión más objetiva de las características del diseño. Esto se puede lograr a través del uso de las métricas OO, el cual se puede aplicar tanto al modelo de diseño como al modelo de análisis.

Métricas orientadas a Clases.

Una clase es la unidad principal de todo sistema OO. Por lo que las medidas y métricas para una clase individual, la jerarquía de clases, y las colaboraciones de clases resultan sumamente valiosas para un ingeniero de software que necesite estimar la calidad de un diseño.

Las características mencionadas anteriormente se pueden utilizar como bases para el conjunto de métricas CK.

Métricas CK (Chidamber y Kemerer 94).

Las métricas de Chidamber y Kemerer consisten originalmente en seis métricas calculadas por cada clase: **MPC**[5], **APH**[6], **NDD**[7], **ACO**[8], **RPC**[9] y **CCM**[10] (Laing, 2001). En este trabajo se hace uso solo de algunas de ellas para la validación de las clases del diseño.

1. **Árbol de Profundidad de Herencia (APH)**

El objetivo de esta métrica es medir la distancia desde una clase a la clase raíz del árbol de herencia. Una clase con un número de herencia muy alto tiende a incrementar los errores, por lo que se recomienda un APH de 5 o menos clases herederas.

2. **Número de Descendientes (NDD)**

Cuenta el número de hijos inmediatos del árbol de herencia. Un alto grado de NDD indica una alta reutilización de los métodos, ya que la herencia es una forma de reutilización. Una clase con muchos hijos requiere de mayor tiempo para las pruebas, pero también indica menos errores, esto se debe al alto grado de reutilización.

[5] MPC: Métodos Ponderados por Clases
[6] APH: Árbol de Profundidad de Herencia
[7] NDD: Número de Descendientes
[8] ACO: Acoplamiento entre Clases Objeto
[9] RPC: Respuesta para una Clase
[10] CCM: Carencia de Cohesión de los Métodos

Aplicación de la métrica.

Métricas CK	Resultado del análisis
APH	La longitud máxima desde hasta la raíz tiene un valor de 2, por lo que el diseño de las clases se puede clasificar como sencillo y fácil de implementar.
NDD	El número mayor de clases hijas es 1, por lo que se refleja un gran manejo de la reutilización de los métodos de una clase a otra.

Conclusiones parciales

En el presente trabajo se logró el objetivo propuesto, ya que se llevó a cabo el proceso de análisis y diseño de una plataforma que permita brindar servicios a la comunidad de desarrollo de software de la Universidad de las Ciencias Informáticas, así como la gestión del conocimiento generado durante el mismo en los proyectos productivos.

La validación de los distintos artefactos que se obtuvieron como resultado en los capítulos anteriores mostró una gran aceptación de los mismos. Esta validación se llevó a cabo a través de la aplicación de métricas para la calidad de la especificación de los requisitos y métricas OO para el diseño.

Tras el análisis de estos resultados se puede llegar a la conclusión de que el presente trabajo está apto para su uso posterior en la posible implementación del sistema propuesto.

CONCLUSIONES

Una vez concluido el presente trabajo se llegó a la conclusión de que los objetivos trazados para el mismo fueron logrados exitosamente, ya que:

- Se evidenció que la gestión del conocimiento es un proceso fundamental dentro del proceso de desarrollo de software, como parte de la materialización del capital intelectual. Gracias a ello se hacen vigentes ventajas como la reutilización y socialización del conocimiento generado en un marco de trabajo que puede ser útil para otras personas. Cualquier entidad que desee obtener un mayor logro en su desempeño laboral debe proporcionar los mecanismos de gestionar conocimientos adquiridos por sus trabajadores. De esta forma se garantizaría un gran avance sobre la información de la misma.

- Se postularon las ventajas de contar con una herramienta que sea capaz de brindar soporte en línea al proceso productivo de software,

- Quedaron constatados los beneficios de utilizar la metodología RUP, el lenguaje de modelado UML y la herramienta Visual Paradigm para la obtención de artefactos claros y entendibles generados a través del análisis y diseño, los cuales sirven de entradas para la implementación del sistema.

- Se llevó a cabo la validación de la calidad de los artefactos obtenidos a través del uso de un conjunto de métricas aplicadas a las especificaciones de casos de uso y al modelo de diseño, lo que arrojó un resultado satisfactorio.

RECOMENDACIONES

Se exhorta proseguir con el flujo de implementación del sistema teniendo como entrada todos los artefactos generados en el presente trabajo. Para ello, se recomienda la utilización de un CMS, ya sea Drupal, Joomla u otro, por sus características gestoras de contenido y de usuarios. Para las nuevas funcionalidades se pudiera utilizar PHP como lenguaje de programación y como gestor de base de datos MySQL o PostgreSQL.

Bibliografía

Aja Quiroga., L. L. (2002). Gestión de información, gestión del conocimiento y gestión de la calidad en las organizaciones. *10 05 2002*.

Albacete. (2002). *Diseño y Programación Orientada a Objetos.* Retrieved 04 30, 2008, from Diseño y Programación Orientada a Objetos: http://www.info-ab.uclm.es/asignaturas/42579/cap4/Creacion.htm

Arenas, M. I. (2000). *Geneura.* Retrieved febrero 06, 2008, from Geneura: http://geneura.ugr.es/~maribel/php/index.html

Bernárdez, B., Durán, A., & Toro, M. (2004). Una propuesta para la verificación de requisitos basada en métricas. *1* (2).

Casañola Trujillo, I. Y. (2006). *Propuesta de modelo de producción de software para la Universidad de las CienciasInformáticas.* Ciudad Habana.

Claudio, M. (2001). *Proyecto Informático: Una metodoligía Simplificada.* Buenos Aires, Argentina.

Escalona, M. J., & Koch, N. (2002). *Ingeniería de Requisitos en aplicaciones para la web.* Sevilla.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns CD: Elements of Reusable Object -Oriented Software.* Addison Wesley Longman .Inc.

García Arenas, M. I. (2000). *Geneura.* Retrieved febrero 06, 2008, from http://geneura.ujr.es

García Robles, R. (2001). El nuevo paradigma de la gestión del conocimiento y su aplicación en el ámbito educativo.

Gracia, J. (2005, mayo 27). *Ingenieros Software.* Retrieved from http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php

Gunnar Övergaard, K. P. (2004). Use Cases Patterns and Blueprints. Addison Wesley Professional.

IBM Company. (2008). *IBM*. Retrieved 1 16, 2008, from http://www.ibm.com/support/es/es/

Jacobson, I., Booch, G., & Rumbaugh, J. (2000). *El proceso unificado de desarrollo de software.* Madrid: Addison Wesley Object Technology Series.

Kendall & Kendall . (2005). *Análisis Y Diseño De Sistemas.*

Knowledge Master Corporation. (2007). *Knowledge Master* . Retrieved 01 15, 2008, from http://www.conceptmaps.it/KM-ConceptualKnowledgeBases-esp.htm

Larman, C. (2008). *El Mundo Informatico*. (wordpress) Retrieved abril 22, 2008, from El Mundo Informatico: http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/

Marquetti, H., & Otros. (1997). *Marquetti Asociados.* Retrieved febrero 08, 2008, from Marquetti Asociados: http://www.marquetti-asociados.com.ar/empresa.php

Microsoft Corporation. (2008, 1 14). *Ayuda y Soporte Técnico*. (Microsoft Corporation) Retrieved 1 16, 2008, from http://support.microsoft.com/gp/services/

Oracle Company. (2008). *Oracle* . Retrieved 1 16, 2008, from http://www.oracle.com/support/index.html

Pressman, R. S. (2005). *Ingeniería del Software: Un enfoque práctico.* Mc Graw Hill.

Rational Software Corporation. (2003). Rational Unified Process 1.0. *2003.06.00* .

Saborit Ramírez, I. Y. (2007, Noviembre 30). El soporte en la Universidad de las Ciencias Informáticas. Ciudad Habana.

Schumller, J. (2000). *Aprendiendo UML en 24 horas.* México: Pearson Educación .

Sierra, A. A. (2002, Octubre). *Programación Extrema y Software Libre.* Retrieved febrero 08, 2008, from Programación Extrema y Software Libre: http://www.seguridad.unam.mx/eventos/datos/ev11/semi18/mat.7.pon19.semi18.pdf

Soporte Bankoi S.L. (2007). *Servicios de HelpDesk Bankoi*. (Bankoi) Retrieved 1 16, 2008, from http://www.soportehelpdesk.com/web.htm

Unión Internacional de las Telecomunicaciones. (2003). *Primera Fase de la CMSI (10-12 de diciembre de 2003, Ginebra), Declaración de Principios de Ginebra.* Ginebra.

Vieyra, M. E. (2001, 10 27). *Base de Conocimiento*. Retrieved 01 16, 2008, from http://www.fismat.umich.mx/~anta/tesis/node13.html

Visual Paradigm International. (1997-2007). *Visual Paradigm*. Retrieved febrero 08, 2008, from Visual Paradigm: http://www.visual-paradigm.com