Universidad de las Ciencias Informáticas

Facultad 10

Título: Análisis y Diseño de un Repositorio de Requisitos Reutilizables

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

Autores

Annies Carrasco Padilla

Leticia Ramos Martínez

Tutores

MSc. David Leyva Leyva

MSc. Daymy Tamayo Ávila

Ing. Zamira Segoviano Martínez

Ciudad de La Habana, junio 2008

Hombre es algo más que ser torpemente vivo: es entender una misión, ennoblecerla y cumplirla.

José Martí

Declaración de autoría

Por este medio declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los __ días del mes de __ del 2008.

Autores

Annies Carrasco Padilla

Leticia Ramos Martínez

Tutores

MSc. David Leyva Leyva

MSc. Daymy Tamayo Avila

Ing. Zamira Segoviano Martínez

Agradecimientos

Hoy ha llegado el gran día y nos convencemos de que no ha sido en vano tanto esfuerzo en estos años de estudio. Pero miramos hacia atrás y vemos a esas personas que nos han ayudado a llegar hasta aquí y pensamos que tal vez no hemos dedicado un momentico de nuestro tiempo para agradecerles. Hemos querido agradecer en este trabajo a quienes, de alguna manera, también son dueños de este triunfo.

Especialmente a Fidel y a la Revolución Cubana por permitirnos ser parte de este proyecto.

A nuestros amigos. A los de siempre y a los que descubrimos en estas horas de trabajo exhaustivo y que nunca nos fallaron.

A nuestros tutores por guiarnos ejemplarmente durante estos meses de trabajo.

A mis padres Odalys y Angel, porque si hay razones por las que hoy estoy aquí, esas razones son ellos.

A mi hermana Anisleydis, porque tratando de ser un buen ejemplo para ella he logrado ser mejor persona.

A mi tía Mayra que es mi otra mami, a Rigo, a mi abuelita Yaya que adoro, en fin, a todos mis familiares porque siempre han estado al tanto de mis estudios, mi futuro y mi vida.

A mi novio Dunel por entrar con cariño en mi vida y darme la confianza en mí misma que necesitaba.

A Daniar, por estar siempre.

A Leticia, por permitirme ser parte de esta importante etapa de su vida y ser una magnifica compañera.

Annies

A mis padres Pilar y Gume por apoyarme y guiarme siempre por el buen camino.

A mi hermano Alexander por ser mí mejor amigo, a su niña Meliza y su esposa Damaris por ayudarme en todo lo que fue posible.

A mis abuelitas Tita y Conga por ser la alegría de mi vida, a mis tíos Nieve, Joseito, Berena, a mi prima Noilda, a todos mis familiares por apoyarme en todo.

A mi novio Lester por estar siempre a mi lado, a su familia por ayudarme en todo lo que necesitaba

A mi compañera de tesis Annies con quien ha sido más que un placer trabajar.

A todos los que con su apoyo me han ayudado y han confiado en mí en todo momento.

A todos gracias.

Leticia

Dedicatoria

Dedico este gran esfuerzo a los mejores padres del mundo, mis padres…

… a mi hermana, la persona más importante en mi vida, aunque no se lo digo muy a menudo.

A todos aquellos que desde pequeña han ayudado a formar el espíritu y el carácter que han permitido que hoy esté escribiendo estas letras.

Annies

A mi mamá y mi papá
A mi hermanito
A mis abuelitas
A mi novio

Por ser lo más valioso que tengo.
Por enseñarme a luchar por mis sueños hasta hacerlos realidad.
Por todo el amor y el cariño que me han dedicado en cada minuto de mi vida.

Leticia

Resumen

Los analistas de los diferentes proyectos de la Universidad de las Ciencias Informáticas comienzan de cero una y otra vez en cada levantamiento de requisitos. Esto trae consigo que dediquen mucho tiempo a esta tarea, por lo que en muchas ocasiones el producto final no se entrega en el tiempo ni con la calidad requerida. El trabajo que se presenta enfoca la reutilización de requisitos como una vía para ayudar a minimizar esos problemas con los que se enfrentan los equipos de desarrollo de software de la universidad.

Se realiza el análisis y el diseño de un repositorio donde serán almacenados paquetes de requisitos de diferentes categorías, de los cuales el analista de software podrá reutilizar total o parcialmente. Esto permitirá una reducción en el tiempo de concepción del software, además de trabajar con requisitos que tienen la calidad necesaria.

Para desarrollar el Repositorio de Requisitos Reutilizables se propone utilizar herramientas libres, como son PostgreSQL como gestor de base de datos relacional, Apache como servidor Web, PHP y HTML como lenguajes de programación. También se utiliza la metodología de desarrollo RUP, el lenguaje de modelado UML y la herramienta CASE Visual Paradigm para el diseño del mismo.

Palabras Claves:

Requisitos, Reutilización, Repositorio, Paquetes de requisitos.

Índice

Índice de Tablas y Figuras

Introducción

A medida que la Ciencia y la Técnica han ido evolucionando y la exigencia de procesos automatizados se han extendido hacia diferentes áreas de la sociedad, la producción de software se ha hecho cada vez más necesaria, por lo que el tiempo y la calidad constituyen un elemento imprescindible.

La continua evolución de la información y las crecientes presiones de las instituciones para construir software en corto tiempo y con eficacia, acentúan la necesidad de aprovechar óptimamente los esfuerzos implicados. Dentro del proceso de desarrollo de software es imprescindible traducir eficientemente las necesidades del usuario en requerimientos de software como primer paso dentro del ciclo de vida de dicho sistema, que incluye otros procesos para lograr los propósitos propuestos como diseño, análisis, implementación, documentación y certificación.

Los requisitos son la base para el desarrollo de cualquier software, pero en muchas ocasiones en el levantamiento de requisitos se invierte gran cantidad de tiempo ya que se debe comenzar de cero cada vez que se crea un nuevo requerimiento. La obtención de una especificación de requisitos de alta calidad es fundamental para asegurar que el software se corresponda con las necesidades del cliente, por lo que rapidez no puede ser un motivo para demeritar el proceso ni el producto.

Por esta razón se ha propuesto la reutilización como una respuesta para incrementar significativamente la productividad a la vez que se mantiene o se incrementa la calidad del software. Las ventajas potenciales de la misma estimulan nuevos planteamientos dentro de la ingeniería del software, aunque la reutilización sistemática requiere una organización apropiada y una cultura idónea.

La reutilización responde al principio de aprovechar esfuerzos previos y exitosos para completar un nuevo desarrollo.

Intuitivamente, la reutilización se asocia de manera directa con la utilización de código fuente para construir un nuevo producto. Sin embargo, un elemento reutilizable es cualquier elemento del ciclo de vida del software que pueda, potencialmente, utilizarse para contribuir en el desarrollo de un nuevo producto: modelos y arquitecturas de dominio, elementos del diseño, el propio código, componentes de bases de datos, documentación, casos de pruebas e incluso los requisitos. Es recomendable, abordar la reutilización lo más temprano posible en el ciclo de vida del software, ya que ofrece una serie de

ventajas, como son estimular la reutilización a lo largo del resto del ciclo y permitir un mejor aprovechamiento del esfuerzo de desarrollo.

Las investigaciones en reutilización han mostrado que se requiere un enfoque particular, propio y adaptado a la naturaleza del software, para disponer de la capacidad de reutilizar artefactos de desarrollos previos. Este enfoque se basa en la selección, especialización e integración de elementos del software que hayan sido intencionalmente diseñados, desarrollados y documentados para servir como materia prima para nuevos desarrollos. El espectro de elementos reutilizables del software abarca diferentes productos del ciclo de vida dentro de los diferentes niveles de abstracción, incluyendo productos obtenidos en la fase de requisitos.

En el contexto de la producción empresarial del software, la reutilización de requisitos permite completar la especificación de nuevas aplicaciones para así mejorar la calidad y productividad del proceso de ingeniería de requisitos.

Existen varios métodos basados en la reutilización de requisitos, los cuales aportan mayores beneficios a sus proyectos. Ejemplo de esto es el AMIR-ST (Aproximación Metodológica para la Ingeniería de Requisitos de Sistemas Telemáticos)[SOLARTE 2003] este es un conjunto articulado y bien balanceado de métodos para el flujo de trabajo de Ingeniería de Requisitos. Otro método que existe es el SIREN (SImple REuse of software requiremeNts)[TOVAL A. and B 2001], es un método de Ingeniería de Requisitos basado en reutilización y estándares de Ingeniería del Software para obtener documentos de requisitos de calidad en un proyecto de desarrollo de software.

En la Universidad de las Ciencias Informáticas (UCI), los proyectos productivos en ocasiones no realizan el proceso de análisis de requisitos con todos los pasos requeridos, ya que no existe una herramienta o método que gestione este proceso. Esta deficiencia provoca una serie de inconvenientes como: falta de organización de los proyectos, así como retrasos en os mismos debido al excesivo tiempo invertido en la recogida de requisitos, la utilización de prácticas pobres de Ingeniería, baja calidad de los productos, además de que el índice de productividad no resulte ser el esperado. La Gestión de Requisitos en los proyectos de la UCI ha sido objeto de investigaciones anteriores[HECHEVARRÍA 2007], que incluyen las características más importantes para llevar a cabo este proceso y las herramientas más utilizadas para su realización, además de exponer la necesidad de crear una herramienta propia para garantizar la reutilización de requisitos en los Proyectos Productivos.

Teniendo en cuenta la situación antes expuesta se plantea el siguiente **problema**: ¿Cómo garantizar la reutilización de requisitos de software en los nuevos proyectos de la Universidad de Ciencias Informáticas (UCI)?

A partir de la problemática anterior, se definió como **objeto de estudio** de la presente investigación la Gestión de Requisitos. También se precisó el **campo de acción** la Reutilización de requisitos de software.

El **objetivo general** para el desarrollo de la investigación es modelar un repositorio de requisitos de software utilizando los estándares correspondientes que permitan la reutilización en nuevos proyectos.

La **idea a defender** es que la reutilización de requisitos de software en nuevos proyectos puede alcanzarse mediante un repositorio que los gestione utilizando estándares en su representación.

Como **tareas** investigativas se proponen:

- Identificar conceptos asociados a la Ingeniería de Requisitos trascendentes para la investigación.
- Realizar un estudio acerca de los estándares y especificaciones asociados a requisitos.
- Analizar métodos basados en la reutilización de requisitos.
- Analizar modelos de repositorios de requisitos.
- Realizar el análisis y diseño de un repositorio de requisitos.

Para la realización de las tareas propuestas se utilizan diferentes **métodos de investigación**.

Histórico - lógico

Se utiliza para estudiar cómo ha evolucionado el proceso de desarrollo de software y cómo las aplicaciones se han extendido a diferentes áreas del conocimiento. También se utiliza para estudiar la evolución de los estándares asociados a requisitos.

Analítico - Sintético

Se utiliza para realizar un estudio de los diferentes métodos basados en reutilización para poder arribar a una serie de conclusiones que ayuden a brindar la solución adecuada al problema científico.

Revisión de documentos

Para identificar los términos más importantes sobre la reutilización de requisitos, incluyendo investigaciones anteriores para llevar a cabo esta actividad encontrados en la bibliografía consultada.

El documento está estructurado en introducción, tres capítulos, conclusiones, recomendaciones, anexos y glosario. Cada uno de ellos abordan diferentes temáticas y logran una mejor distribución del contenido y claridad en las ideas. A continuación se presenta el nombre de los capítulos y sus objetivos en un contexto global.

En el Capítulo 1, **Fundamentación teórica**, se abordan de forma general los aspectos teóricos necesarios para la investigación, se explica qué es la Ingeniería de Requisitos (IR), qué son los requisitos, metadatos, las herramientas de la Gestión de Requisitos, enfocándose en la reutilización de los mismos, planteando sus características, ventajas y desventajas. Además se definen los estándares y especificaciones asociados a los requisitos, las herramientas utilizadas para el desarrollo de la aplicación y la metodología a seguir.

En el Capítulo 2, **Características del sistema**, se describen las características del sistema dando a conocer el flujo de trabajo con el que se ofrece la solución al problema de investigación planteado, además se realiza un modelo de dominio en el cual se plantean los principales conceptos de la aplicación. Se determina la captura de requisitos, así como los principales casos de uso de los cuales se ofrece una descripción textual.

En el Capítulo 3, **Análisis y Diseño del sistema**, se describe el sistema a través del flujo de trabajo Análisis y Diseño que propone la metodología RUP para su desarrollo.

Capítulo 1. Fundamentación teórica

En este capítulo se expresan diferentes aspectos teóricos relacionados con los temas tratados en el presente trabajo. A continuación se introducen una serie de definiciones relacionadas con la Ingeniería de Requisitos como disciplina dentro proceso de desarrollo de software, así como diferentes conceptos necesarios para desarrollar la investigación, por ejemplo: requisito, repositorio, metadatos, entre otros. Además se hace referencia a métodos y herramientas que existen actualmente, que servirán de base para el desarrollo de la presente investigación. Por otra parte se exponen los estándares y especificaciones asociados a los requisitos, así como los estándares de interoperabilidad, metadatos y aplicaciones Web. También se describen las herramientas y lenguajes a utilizar para el desarrollo de la aplicación, la metodología a seguir, entre otros elementos que se abordarán en el transcurso del trabajo.

1.1 Ingeniería de Requisitos (IR)

Como muchos de los temas relacionados con la Ingeniería de Software, la Ingeniería de Requisitos es un concepto que tiene varias acepciones, entre las que podemos encontrar:

“La disciplina de la Ingeniería de Software que trata con actividades e intenta comprender las necesidades exactas de los usuarios del sistema software, para traducir tales necesidades en instrucciones precisas y no ambiguas las cuales podrían ser posteriormente utilizadas en el desarrollo del sistema.” [LOUCOPOULOS and KARAKOSTAS 1995]

“Es el proceso mediante el cual se intercambian diferentes puntos de vista para recopilar y modelar lo que el sistema va a realizar”. [RICHARD 1997]

El proceso de Ingeniería de Requisitos tiene como objetivos descubrir, modelar, validar y mantener un documento de requisitos, utilizando una combinación de métodos, herramientas y actores.

En la Ingeniería de Requisitos se identifican diferentes aspectos como el propósito del sistema, la dirección y el alcance del mismo. Abarca un conjunto de actividades, transformaciones que pretenden comprender las necesidades de un software y convertir la declaración de estas necesidades en una descripción completa, precisa y documentada siguiendo un determinado estándar.

Cumple un papel primordial en el proceso de producción de software, ya que enfoca un área fundamental relacionada con la definición de lo que se desea producir. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento del sistema; de esta manera, se pretende minimizar los problemas relacionados con el desarrollo de sistemas.

1.2 Requerimientos/Requisitos

1.2.1 ¿Qué es un requisito?

Existen diferentes definiciones relacionadas con los requisitos de software como elemento de la Ingeniería de Software, entre las que encontramos:

Definición que aparece en el glosario de la IEEE.

1- Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.

2- Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.

3- Una representación documentada de una condición o capacidad como en 1 o 2. [RICHARD 1997]

Por requisito se entiende también como aquella propiedad que un sistema (software-hardware) debería tener para ser exitoso en el entorno en el cual se usará. [SOLARTE 2003]

Los requerimientos deben especificarse antes de intentar comenzar la construcción del producto, sin ellos no podría ser posible llevar a cabo las etapas de diseño y construcción correctamente. Los mismos pueden verse como una declaración abstracta de alto nivel de un servicio que el sistema debe proporcionar, como una definición matemática detallada y formal. Los requisitos cumplen una doble función ya que son la base para una oferta de contrato, por lo tanto deben estar abiertos a la interpretación. Además son la base para redactar el contrato en sí mismo.

1.2.2 Características de los requisitos

Un conjunto de requerimientos en estado de madurez, deben presentar una serie de características tanto individuales como en grupo. A continuación se presentan las más significativas.

Necesario: Un requerimiento es necesario si su omisión provoca una deficiencia en el sistema a construir, y además su capacidad, características físicas o factor de calidad no pueden ser reemplazados por otras capacidades del producto o del proceso.

Conciso: Un requerimiento es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en un futuro.

Completo: Un requerimiento está completo si no necesita ampliar detalles en su redacción, es decir, si se proporciona la información suficiente para su comprensión.

Consistente: Un requerimiento es consistente si no es contradictorio con otro requerimiento.

No ambiguo: Un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.

Verificable: Un requerimiento es verificable cuando puede ser cuantificado de manera que permita hacer uso de los métodos de verificación, inspección, demostración o pruebas.

1.2.3 Tipos de requisitos

Los requerimientos se pueden clasificar en:

- Requisitos funcionales
- Requisitos no funcionales

1.2.3.1 Requisitos funcionales

Son capacidades o condiciones que el sistema debe cumplir. Declaración de los servicios que el sistema debe proporcionar, cómo debe reaccionar a una entrada particular y cómo se debe comportar ante situaciones particulares.

En la realización de los casos de uso del negocio, se obtienen las actividades que serán objeto de automatización. Estas actividades no son exactamente los requerimientos funcionales, pero sí son el punto de partida para identificar qué debe hacer el sistema. Por lo tanto los requerimientos funcionales especifican el comportamiento de entrada y salida del sistema y surgen de la razón fundamental de la existencia del producto.

Los requerimientos funcionales se mantienen invariables sin importar con que propiedades o cualidades se relacionen.

1.2.3.2 Requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

Normalmente están vinculados a requerimientos funcionales, es decir, una vez que se conozca lo que el sistema debe hacer podemos determinar cómo ha de comportarse y qué cualidades debe tener.

Existen múltiples categorías para clasificar los requerimientos no funcionales, entre los cuales se encuentran Requerimientos de Software, de Hardware, Requerimientos de apariencia o interfaz externa, de Seguridad, entre otros.

Otro punto muy importante dentro de las definiciones antes expuestas es la posibilidad que tienen los requisitos de volver a utilizarse. La reutilización de los mismos aporta numerosas ventajas en el desarrollo de cualquier software permitiendo que el producto final se obtenga en menor tiempo y con mayor calidad.

1.3 Reutilización de requisitos

La obtención de sistemas de software de manera eficiente y en el tiempo adecuado está dada por la puesta en práctica de métodos y metodologías de trabajo que permitan obtener resultados altamente favorables. Para lograr estas metas los desarrolladores han recurrido al empleo de la reutilización, factor que favorece en gran medida la construcción de sistemas de mayor calidad. Entonces, ¿qué se entiende por reutilización? Existen varias definiciones sobre este término:

"Se define la reutilización de software como el uso de cualquier tipo de artefacto (también llamado activo), o parte del mismo, creado con anterioridad, en un nuevo proyecto." [SOLARTE 2003]

"Una estrategia alternativa para el proceso de construcción de sistemas de software a partir de la utilización de componentes reutilizables desarrollados con anterioridad y en los que se han garantizado factores de calidad tanto externos como internos." [SEGOVIANO and VIDIAUX 2007]

Aquella necesidad de un usuario, la cual debe ser propiedad del sistema para ser exitoso en el entorno en el cual se usará también puede haber sido una condición de otro sistema desarrollado anteriormente. La reutilización de la representación documentada de la misma es a lo que llamamos reutilización de requisitos.

Así como se puede reutilizar código, diseños, arquitecturas, también es posible la reutilización de requisitos especificados en otros proyectos y no necesariamente por el mismo equipo de desarrollo, para que dicho proceso sea más eficiente deben tenerse en cuenta los siguientes factores:

- *Unificación de los formatos para descripción de requisitos.*
- *Disminución de ambigüedad, de preferencia usando un lenguaje de especificación formal.*
- *Contar con un repositorio de requisitos de fácil acceso y gestión.*
- *Validación de requisitos.* [SOLARTE 2003]

Es primordial explotar al máximo los beneficios que ofrece la reutilización con el objetivo de aumentar la calidad durante el desarrollo de un software, teniendo siempre en cuenta sus inconvenientes para que no afecten el mismo. Para favorecer esta característica es necesario que estos recursos estén acompañados de cierta descripción, que puede lograrse mediante el empleo de metadatos.

1.4 Metadatos

Ante el volumen y heterogeneidad de información existente en la World-Wide Web, las bibliotecas y unidades de información necesitan crear sistemas de organización y recuperación de información electrónica que proporcionen, a los usuarios profesionales, una alternativa ante los motores de búsqueda y que permitan garantizar la calidad y fiabilidad del contenido. Distintas comunidades y dominios informativos que utilizan la Web como soporte de sus conocimientos están desarrollando servicios de información de calidad basados en la estructuración de la información sobre la información (metadatos).

Un metadato es toda aquella información descriptiva sobre el contexto, calidad, condición o característica de un recurso, que tiene la finalidad de facilitar su recuperación, autentificación, evaluación, preservación o interoperabilidad.[MARTINEZ and LARA. 2006].

En el campo de la informática "los metadatos son datos altamente estructurados que describen información, el contenido, la calidad, la condición y otras características de los datos. Es información sobre información o datos sobre los datos".[METADATOS 2007]

Existen muchos factores que influyen en la calidad de los registros de metadatos, entre los que se encuentran:

- La información proporcionada en los metadatos depende de las facilidades que ofrezca el creador(es) del registro y del tiempo necesario para añadir dicha información.
- Las capacidades de edición o herramientas proporcionadas por el repositorio.
- Los metadatos asociados a los paquetes de requisitos correspondientes al Repositorio de Requisitos serán descritos a través de documentos XML.

1.4.1 ¿Qué es XML?

XML (Extensible Markup Language) es un metalenguaje extensible de etiquetas desarrollado por el W3C (World Wide Web Consortium). Es un subconjunto simplificado del SGML (Standard Generalized Markup Language o Lenguaje de Marcación Generalizado) que fue diseñado principalmente para la Web. Proporciona a los diseñadores crear sus propias etiquetas, habilitando la definición, transmisión,

validación y la interpretación de datos entre aplicaciones y entre organizaciones. XML tiene múltiples aplicaciones, entre las que se destaca su uso como estándar para el intercambio de datos entre diversos software.

A partir de documentos XML, se obtienen mejores resultados de búsqueda, así como la facilidad de poder ver la información de un determinado paquete de requisitos sin tener que consultarlo, debido a que la información en dichos documentos está etiquetada por su significado de manera precisa, y es posible localizarla de forma mucho más clara que en documentos HTML (HyperText Markup Language).

En la actualidad se ha planteado el uso de metadatos en diferentes campos y en este caso XML se utiliza para describir requisitos, contribuyendo así a facilitar su intercambio y reutilización. Pero esto es sólo un aspecto, también se necesita que los requisitos, los metadatos, entre otros, sean conocidos e interpretados de la misma forma tanto por sus creadores como por los usuarios finales, esto sólo se logra mediante el empleo de estándares.

1.5 Estándares

Se define un estándar como “un documento, establecido por consenso y aprobado por un cuerpo reconocido, que proporciona, para el uso común y repetido, reglas, guías o características para actividades o sus resultados, dirigidas a alcanzar el grado de orden óptimo en un contexto dado.” [JACOBSON I and BOOCH G. 1999]

También se plantea que es un patrón, una tipificación o una norma de cómo realizar algo. [AULAGLOBAL 2005]

Frecuentemente los términos estándar y especificación se utilizan indistintamente, no obstante, es importante puntualizar su diferencia. Si una tecnología, formato o método ha sido ratificado por algún organismo oficial de estandarización, se trata de un estándar, si no es aprobado por ninguna de esas entidades entonces nos estamos refiriendo a una especificación. Aunque, en algunos casos, una especificación puede considerarse un estándar por defecto si su uso es extendido y entretanto se ratifica como estándar.

La utilización de estándares y especificaciones se deben tener en cuenta sobre todo cuando queremos garantizar la reutilización ya que delimitan un ambiente de seguridad y organización, y proporcionan un único formato a la información utilizada. Existen diferentes tipos de estándares según la información que será recogida en ellos, en los próximos sub-epígrafes se hará referencia a los que serán de gran utilidad en la presente investigación.

1.5.1 Estándares asociados a requisitos

El Grupo de Investigación de Ingeniería del Software del Departamento de Informática y Sistemas de la Universidad de Murcia propone una organización de los documentos de requisitos, que es a su vez utilizada por diferentes autores del área que consideran que cada documento se corresponde con un nivel de especificación diferente y por tanto, tendrá objetivos y destinatarios distintos. La siguiente figura muestra la jerarquía de documentos que se especifican en dicha organización. [TOVAL A. and B 2001]

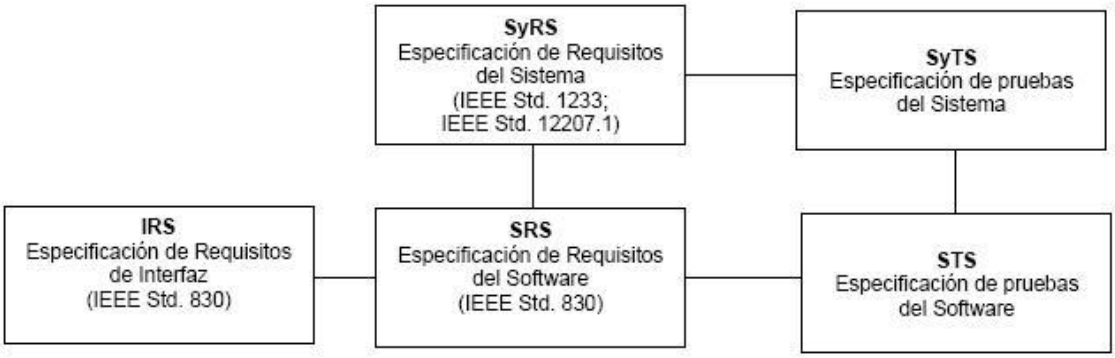


Figura 1 Jerarquía de documentos de requisitos.

La aplicación de dichos documentos en un proyecto dado y su obligatoriedad está determinada por las decisiones que se tomen en el mismo o las especificaciones de cada uno de ellos. La decisión sobre la jerarquía a utilizar corresponde al jefe de proyecto (junto con el analista) y depende en gran medida de las dimensiones del proyecto y la complejidad del sistema que se va a desarrollar:

Los documentos de especificación de requisitos del sistema (**SyRS**) y pruebas del sistema (**SyTS**) son únicos, pero no son necesarios cuando el problema es simple y está claro desde el principio que todos los requisitos van a ser soportados por el software.

El documento de especificación de requisitos del software (**SRS**) es obligatorio y puede dividirse en subdocumentos.

El documento de especificación de requisitos de interfaz (**IRS**) es opcional y se utiliza cuando los requisitos de interfaz son muy numerosos, con el objetivo de evitar que el SRS sea muy extenso; en otro caso, sólo se incluirán en el SRS.

Finalmente, se debe definir un documento de especificación de pruebas del software (**STS**) por cada SRS de la jerarquía.

La especificación de requisitos es el resultado final de las actividades de análisis y evaluación de requerimientos; este documento resultante será utilizado como fuente básica de comunicación entre los clientes, usuarios finales, analistas de sistema, personal de pruebas, y todo aquel involucrado en la implementación del sistema. A continuación se describen algunas de las características más importantes de las diferentes especificaciones que se toman en cuenta en esta jerarquía:

Especificación de requisitos del sistema. SyRS

De acuerdo con el estándar IEEE 12207.0, el documento de Especificación de requisitos del sistema (SyRS) debe incluir: funciones y capacidades del sistema; requisitos del negocio, organizativos y de usuario; requisitos de seguridad, seguridad a terceros y privacidad; requisitos de ingeniería de factores humanos; requisitos de operaciones y mantenimiento; y restricciones de diseño. A partir de estas características, el estándar IEEE 12207.1 detalla el contenido específico de la plantilla para dicha especificación haciendo referencia al estándar IEEE Std 1233, *Guide for Developing System Requirements Specifications*, donde se pueden encontrar guías para especificar los requisitos del sistema y un esquema del SyRS.

Especificación de requisitos del software. SRS

La mayoría de los requisitos del software se obtienen directamente a partir de los requisitos del sistema; de ahí la visión jerárquica. La Especificación de requisitos del software (SRS) se basa en el

estándar IEEE Std 830-1998 y la plantilla VOLERE, en las que se recogen requisitos relacionados con las funcionalidades del sistema, interfaces externas, rendimiento, restricciones de diseño y atributos del software (portabilidad, mantenimiento, seguridad, disponibilidad y fiabilidad).

Especificación de pruebas de sistema y software. SyTS y STS

Para poder validar los requisitos, estos deben cuantificarse en los documentos antes mencionados SyRS y SRS. En los documentos de Especificación de pruebas de sistema y de software (SyTS o STS) se especificarán criterios de prueba para asegurar que el sistema o el software cumpla con los requisitos especificados.

Especificación de requisitos de interfaz. IRS

Los requisitos relacionados con las interfaces entre los elementos del software y entre el usuario y el sistema podrían incluirse en el documento SRS. No obstante, con el objeto de no producir documentos muy extensos, en algunas ocasiones será conveniente hacer uso de un documento separado denominado Especificación de requisitos de Interfaz (IRS). Por tanto, se deben establecer las relaciones de trazabilidad adecuadas entre el SyRS, el SRS y las interfaces descritas. La plantilla del IRS tiene la misma estructura que el apartado del SRS dedicado a la especificación de requisitos de interfaz.

Las especificaciones de documentos de requisitos anteriormente expuestas se construyen sobre la base de los algunos estándares de IEEE:

IEEE 830-1998: Estándar para especificación de Requisitos del Software (**SRS**):

Este estándar describe las estructuras posibles, contenido deseable, y calidades de una especificación de requisitos del software. Formulan recomendaciones sobre prácticas a seguir para la especificación de requerimientos de software mediante una serie de plantillas para documentar la especificación de los requerimientos relevados, así como criterios para asegurar la correcta redacción de los mismos. [IEEE 1993] (Ver Anexo 1)

IEEE 1233-1998: Estándar para especificación de Requisitos del Sistema (**SyRS**)

Este estándar proporciona una guía para desarrollar la especificación de requisitos del sistema, que contempla las características y cualidades necesarias tanto para los requisitos individuales como para todos los requisitos del sistema. El desarrollo de un SyRS incluye la identificación, organización, presentación, y la modificación de los requisitos. [IEEE 1998] (Ver Anexo 2)

IEEE 12207.0: Estándar para Tecnología de la Información - Procesos de Ciclo de Vida del Software.

Este estándar establece un marco común para los procesos del ciclo de vida del software, con una terminología bien definida, que puede ser de referencia en la industria del software. Contiene procesos, actividades y tareas que se aplican durante la adquisición de un sistema, un producto de software independiente, un software de servicio y durante el suministro, desarrollo, operación y mantenimiento de productos de software.

Por su parte la Plantilla de Especificación de Requisitos VOLERE está creada para ser utilizada como base para especificaciones de requisitos. La plantilla provee secciones por cada tipo de los requisitos apropiados para los actuales sistemas de software.[J. ROBERTSON and ROBERTSON 2006] (Ver Anexo 4)

Los requisitos que serán almacenados en el repositorio deberán ser creados bajo estos estándares para que puedan ser reutilizados en otros sistemas.

1.5.2 Estándares de interoperabilidad

Se encuentran, además, estándares que permiten, en gran medida, que los diferentes sistemas puedan llevar a cabo la interoperabilidad y la reutilización pues permiten el intercambio directo de requisitos entre distintas plataformas y herramientas de gestión de los mismos.

La interoperabilidad es la “capacidad de diferentes sistemas informáticos, aplicaciones y servicios, para comunicar, compartir e intercambiar datos, información y conocimientos, de una forma precisa, efectiva y consistente; para funcionar e integrarse con otros sistemas, aplicaciones y servicios, así como ofrecer nuevos productos electrónicos” [MARTINEZ and LARA. 2006]

Dentro de los estándares de interoperabilidad que existen actualmente algunos de los más usados son el **SQI** y **OKI**.

SQI (Simple Query Interface) es una especificación que pretende ser una capa que garantice la interoperabilidad entre redes o entornos educacionales heterogéneos. Define los servicios que un repositorio puede tener disponibles para recibir y responder consultas de otros repositorios. SQI es neutral en términos de lenguaje de consultas y modelo semántico usado.

El objetivo es ser una parte del sistema que sea capaz de buscar en los distintos repositorios (heterogéneos) de objetos educativos existentes en las redes a pesar de que posean interfaces propietarias de búsqueda de cada fabricante.

OKI (*Open Knowledge Initiative***)**, por su parte, es una organización responsable de la especificación de interfaces de software que comprende una Arquitectura Orientada a Servicios (SOA) que se basa en las definiciones de alto nivel de servicio y describe cómo los componentes de un entorno de software se comunican entre sí y con otros sistemas. Se expone como un servicio Web, por lo que la comunicación es simplemente en XML.[M. GARCÍA and LÓPEZ 2007]

La base que garantiza la interoperabilidad e integración de los componentes que conforman el sistema gestor del aprendizaje se basa en servicios; dichos servicios deben disponer de un conjunto de interfaces de software conocido como Open Service Interface Definiciones (OSIDs). Cada interfaz detalla cuál es la funcionalidad que cada servicio ofrece. [J. CASAMAJÓ and ALIER]

Por estas características se propone utilizar OKI como estándar de interoperabilidad en futuras versiones del repositorio para que pueda tener disposición para la comunicación y transmisión de información con otros sistemas.

1.5.3 Estándares para aplicaciones web

El repositorio deberá desarrollarse teniendo en cuenta estándares para aplicaciones web que son básicamente las reglas de comportamiento de los “browsers”[1] definidas en las recomendaciones de la

[1] Buscadores

W3C (World Wide Web Consortium) y las cuales se han comprometido seguir los principales desarrolladores de “browsers”.

Un sitio basado en estándares web mostrará una mayor consistencia visual. Mediante el uso de XHTML (Lenguaje de Marcado de Hipertexto Extensible) para el contenido y CSS (Hojas de Estilo en Cascada) para la apariencia, se puede transformar rápidamente un sitio, sin importar que se trate de una página Web o miles, realizando cambios en un solo lugar.

No se trata simplemente de diseño gráfico, se habla de crear sitios Web fáciles de usar y mantener. Al separar estructura y contenido se obtienen grandes beneficios. Se pueden diseñar para todo tipo de navegador y dispositivo, sin crear múltiples versiones, y además tener páginas más usables y rápidas en la descarga.

Es importante tener en cuenta que los Servicios Web no son los responsables del comportamiento de un servicio, sino que se centran en cómo se puede acceder a dicho comportamiento que ofrece un servicio.

1.5.4 Estándares para metadatos

Los metadatos, y más concretamente la **DCMI** *(*Dublin Core Metadata Initiative*)* y su conjunto de elementos ISO 15836-2003, constituyen una de las infraestructuras operacionales de la Web Semántica y una de las claves de la interoperabilidad de la información electrónica. Estos esquemas de metadatos, junto a la codificación sintáctica XML/RDF (Lenguaje de Marcado de Hipertexto Extensible / marco de descripción de recursos), estándares de descripción de contenido y una serie de protocolos y normas para el intercambio de información, protagonizarán la segunda generación de la Web. [EVA MÉNDEZ and SENSO 2004]

Actualmente están aprobadas 7 normas ISO sobre este campo:

* ISO/TR 19033 “Documentación técnica del producto – Metadatos para la documentación de construcción”

* ISO 19115 “Información geográfica- Metadatos”

* ISO 8459-5:2002 “Información y Documentación- Directorio del elemento del dato bibliográfico – Parte 5: Elementos de datos para el intercambio de la catalogación y los metadatos”

* ISO/IEC 11179-3:2003 “Tecnología de información – Registros de metadatos (MDR) – Parte 3: Registro del metamodelo y atributos básicos”

* ISO/IEC 13818-1 “Tecnología de información – Codificación genérica de películas en movimiento y la información de audio asociada: Sistemas”

* ISO/IEC /TR 20943-1 “Tecnología de información – Procedimientos para archivar registros de metadatos con un contenido consistente – Parte 1: Elementos de los datos”

*** ISO 15836:2003 “Conjunto de elementos de datos Dublín Core”**

A partir de La Iniciativa de Metadatos Dublín Core (DCMI), posiblemente la iniciativa de metadatos más conocida, surgida en 1995 como resultado de un taller celebrado en Dublín, Ohio, en el año 2001 la organización de normalización de Estados Unidos, aprobó como norma estatal el conjunto de elementos del Dublín Core “Z39-85:2001 Dublin Core Metadata Element Set”.

Con el tiempo se ha incrementado la utilización del Dublín Core en Internet, lo que fue una de las razones por las cuales se estudió la posibilidad de aprobarlo como norma ISO, lo que sucedió en el año 2003, cuando se aprobó como la norma ISO 15836, esta norma está basada en su totalidad en la norma americana Z39-85. [SENÉ 2004]

Este estándar consta de quince elementos, cada uno de los cuales puede extenderse de acuerdo al uso del calificador de esquema y de tipo. Un calificador de esquema se usa para interpretar el valor en el contenido y se basa generalmente en normas externas. Un calificador de tipo refina la definición del elemento en sí mismo.[MARTÍNEZ P and SANTAMARÍA B 2002] (Ver Anexo 3)

Los elementos poseen nombres descriptivos que pretenden transmitir un significado semántico a los mismos. Cada elemento es opcional y puede repetirse. Además, los elementos pueden aparecer en cualquier orden.

Se seleccionó el DCMI (Dublin Core Metadata Initiative) como estándar para la descripción de los metadatos, los cuales son datos sobre los paquetes que estarán almacenados en el repositorio y que

contienen los requisitos y facilitarán la búsqueda dentro de dicho repositorio y el trabajo con el mismo por parte de los usuarios.

Este trabajo no es el primero ni el único que se realiza pensando en la reutilización de requisitos como herramienta para mejorar el desarrollo de un software y la calidad del producto a entregar, ya otros se han encaminado en esta línea de investigación y han obtenido sus propios resultados.

1.6 Métodos de Ingeniería de Requisitos para la reutilización

Con el auge de la producción de software alrededor del mundo diferentes organizaciones se han planteado producir más en menos tiempo y con mayor calidad por lo que han centrado su estudio en los factores que influyen en estos procesos de producción y han encontrado ideas muy parecidas a las que se defienden en este trabajo por lo que es necesario estudiarlas con el fin de utilizarlas como base para el desarrollo de la presente investigación.

1.6.1 SIREN

SIREN (SImple REuse of software requiremeNts), es el método de Ingeniería de Requisitos basado en la reutilización de requisitos desarrollado con el Grupo de Investigación de Ingeniería del Software del Departamento de Informática y Sistemas de la Universidad de Murcia. El propósito de este método es aumentar el desarrollo a través de la reutilización de requisitos, identificar descripciones de sistemas que puedan ser reutilizadas (en su totalidad o en parte) con un número mínimo de modificaciones, de manera que se reduzca el esfuerzo total que se invierte en el desarrollo del producto.

Para este método la reutilización aporta mayores beneficios, puesto que se establecen relaciones de trazabilidad entre los requisitos de alto nivel del sistema y la arquitectura, implementación y pruebas que se construyen a partir de ellos. De esta manera, si un requisito del sistema es reutilizado, se puede acelerar el proceso de desarrollo posterior. [TOVAL A. and B 2001]

Aunque existen muchas técnicas para abordar la reutilización de requisitos, en SIREN centran su trabajo en la reutilización de requisitos desde un punto de vista más práctico, intentando que el uso de este método sea lo más sencillo posible. Se sitúa en el marco de los estándares de Ingeniería del

Software más extendidos. En principio, los requisitos tienen formato textual, pero pueden incluir todo tipo de objetos como información complementaria del propio requisito, por ejemplo: tablas, diagramas o esquemas de cualquier tipo.

Como método, SIREN incorpora un conjunto de guías y técnicas, un repositorio de requisitos reutilizables, un modelo de proceso, y una herramienta que soporta el repositorio y que actualmente es Requisite Pro. Las guías que proporciona este modelo consisten en una jerarquía de documentos de especificación de requisitos, junto con las plantillas para cada uno de ellos, que determinan la estructura de un repositorio de requisitos reutilizables, que contiene requisitos de diferentes dominios verticales (como contabilidad o finanzas) o de lo que llamamos perfiles (dominios horizontales), conjunto de requisitos aplicables a distintos dominios verticales, como la seguridad de los SI. Estos perfiles no son necesariamente excluyentes entre sí.

En relación con la automatización del proceso, según investigaciones realizadas, indican que las herramientas actuales de gestión de requisitos comparten un mismo problema: no soportan adecuadamente el concepto de reutilización [TOVAL A. and B 2001]. Actualmente utilizan Requisite Pro, lo cual es una de las deficiencias de este método pues es software privativo. Como trabajo futuro se considera la conveniencia de desarrollar su propio repositorio de requisitos.

1.6.2 AMIR-ST

AMIR-ST (Aproximación Metodológica para la Ingeniería de Requisitos de Sistemas Telemáticos), es un conjunto articulado y bien balanceado de métodos para el flujo de trabajo de Ingeniería de Requisitos, que constituye un proceso de desarrollo fundamentado en la interacción continua con los usuarios, la obtención del Modelo de Negocio, el desarrollo de escenarios y el uso de una notación consistente con UML; aspectos que en conjunto facilitan los procesos de comunicación entre clientes, analistas y desarrolladores optimizando los procesos de identificación, especificación, gestión, reutilización, escalabilidad y rastreabilidad de requisitos durante todo el ciclo de vida de construcción de una solución.[SOLARTE 2003]

Este modelo propone varios métodos, como la construcción de un repositorio de requisitos de sistemas telemáticos para una posible reutilización posterior. Estos métodos prevén una reducción importante tanto en el esfuerzo (horas/hombre) como en el tiempo empleado para terminar el flujo de trabajo, un

grado de satisfacción mayor por parte de analistas y desarrolladores con respecto a especificaciones obtenidas con otras referencias metodológicas y una reducción del número de iteraciones necesarias para obtener una especificación de requisitos suficientemente validada tanto por clientes, usuarios, como ingenieros de requisitos y desarrolladores.

Aunque estas no son las únicas investigaciones que incursionan en el tema de la reutilización de requisitos, se considera que son las más relevantes cuyas ideas generales servirán de base para el desarrollo de la presente investigación.

1.7 Herramientas para el manejo de requisitos

Dentro de la Ingeniería de software existen también herramientas que se encargan de la gestión de requisitos. Es necesario referirse a dichas herramientas ya que del repositorio de requisitos reutilizables se podrán importar y exportar paquetes de requisitos elaborados por estas herramientas.

1.7.1 Integral Requisite Analyzer (IRqA)

IRqA es una herramienta de ingeniería de requisitos especialmente diseñada para soportar los procesos de ingeniería de requisitos (básicamente obtención y análisis de requisitos). Contiene un entorno gráfico y textual que permite generar una documentación sencilla que los clientes pueden revisar en las primeras fases de construcción de una solución. IRqA además, posee módulos para la definición y gestión de pruebas de aceptación, estimación de costos basados en casos de uso, gestión de versiones y navegación por los dominios del negocio.

El inconveniente con esta herramienta es su escasa flexibilidad para adaptarse a metodologías operativamente diferentes para la cual fue desarrollada.

1.7.2 RequisitePro

RequisitePro, es una herramienta centrada en documentos, que almacena los requisitos asociándolos a documentos (aunque también permite guardarlos directamente en la base de datos), mientras que

las otras herramientas están orientadas a requisitos. Auxilia especialmente en el control de cambio de requisitos, con trazabilidad para especificaciones de software y pruebas. Está muy unido a MS Word ya que es partner de Microsoft Development. La herramienta permite el uso de Oracle sobre Unix o Windows como “back-end database” y también soporta SQL Server sobre Windows.

Permite la creación y gestión de una base de datos de requisitos a través de un procesador de palabras, acceso a través de interfaces Web, plantillas para la descripción de requisitos y rastreo de requisitos en fases posteriores del proceso de desarrollo de software.

1.7.3 OSRMT

Actualmente la UCI carece de una herramienta para plataforma libre que permita la Gestión de Requisitos. Los proyectos son cada vez más complejos y para hacer la ingeniería de requisitos la Universidad utiliza la herramienta Requisite-Pro aunque es software propietario.

Se está llevando a cabo una investigación sobre una herramienta libre, **OSRMT** (Open Source Requirements Management Tool) v1.5 a la cual se le están haciendo las transformaciones necesarias para ponerla a funcionar de forma óptima. Se espera que sea el producto que se necesita para gestionar de forma ágil los requisitos.[R. SOCARRÁS and RAMOS 2008]

Hasta el momento OSRMT se muestra como la herramienta más indicada para la gestión de requisitos de software por todas las ventajas que presenta, entre otras cosas, como herramienta libre. Pero este trabajo se centra más en el análisis y el diseño de un repositorio, el cual almacenará paquetes de requisitos los cuales se podrán importar desde estas herramientas de gestión así como exportarlos para reutilizarlos en otros proyectos.

1.8 Repositorios

1.8.1 ¿Qué es un repositorio?

Un repositorio es un recurso centralizado que almacena y manipula un gran cúmulo de información. Es fundamental para guardar y compartir colecciones de código, rutinas y fragmentos que se necesitan de forma recurrente.

Cuando un usuario busca información en un repositorio, describe de manera breve el documento que quiere obtener como respuesta. El resultado es que el usuario “navega” en el repositorio realizando consultas y refinando resultados hasta que queda satisfecho.

Los repositorios se pueden clasificar en dos tipos, bibliotecas de documentos y en carpetas, aunque también se pueden identificar según su contenido, repositorios de documentación, de bases de datos, actualizaciones de sistemas operativos como Ubuntu, Debian, etc. Estos pueden contener componentes como: gráficos, imágenes textos, videos, documentos e integración de ellos como capítulos de un curso o hasta cursos completos.

1.8.2 Repositorio de requisitos

Un repositorio de requisitos es un lugar donde se almacenan requisitos con el objetivo de que puedan ser reutilizados por otros usuarios en nuevos proyectos, ya sea con algunas modificaciones o del mismo modo en que se encuentran.

Muchos de los proyectos de la UCI tienen definidos repositorios propios en los que gestionan la información. Los repositorios de requisitos permiten almacenamiento de requisitos de manera que permita potenciar su reutilización. Es un elemento muy importante que podría resolver muchos de los problemas que hemos planteado anteriormente, permitiendo a los proyectos disponer de un medio donde encontrar de manera directa la información necesaria para realizar el levantamiento de requisitos y así obtener cuantiosas mejoras en sus productos finales.

1.8.2.1 Repositorio semántico

Repositorio semántico es un término que potenciará aún más la reutilización de requisitos, el cual, al proporcionar un soporte para albergar los metadatos desempeña un importante papel de cara al futuro. No sólo los usuarios pueden consultar y buscar información, sino también agentes software o sistemas LMS (Learning Management Systems) externos. Se basa principalmente en la idea de añadir metadatos semánticos. Esas informaciones adicionales que describen el contenido, el significado y la relación de los datos se deben proporcionar de manera formal, para que así sea posible evaluarlas automáticamente por máquinas de procesamiento.

Se define el prototipo SLOR (Semantic Learning Object Repository), basado en una ontología, ha sido específicamente diseñado para la creación y administración de metadatos con propósitos de integración e intercambio con otros sistemas. Esta propuesta aporta mejores y nuevas funcionalidades sobre los repositorios actuales, gracias a la posibilidad que la ontología subyacente ofrece para ejecutar inferencias sobre el conocimiento albergado en los registros del repositorio. [HECHEVARRÍA 2007]

Pero este término es una vertiente más amplia de esta investigación, actualmente se está llevando a cabo la inserción del mismo en otros repositorios y el análisis de factibilidad de los mismos ha disparado resultados positivos, por lo que como trabajo futuro se piensa abordar en próximas versiones del presente trabajo.

1.9 Tecnologías propuestas para el desarrollo de la aplicación

Para el desarrollo de la aplicación se propone la utilización de herramientas libres ya que nuestro país ha comenzando la migración hacia el software libre por las facilidades y libertades que le brinda a los usuarios.

El software libre trae consigo numerosas ventajas, muchas de ellas exageradas (o falseadas) por la competencia propietaria. Dentro de estas ventajas que han impulsado a la utilización de GNU/Linux se encuentran el hecho de que dada la libertad de modificar el programa para uso propio, el usuario puede personalizarlo o adaptarlo a sus necesidades, corrigiendo errores si los tuviera, además la adecuación a estándares la cual es unas característica notable del software libre que no es tan

respetada por el software propietario, ávido en muchos casos notables de crear mercados cautivos. Otra de las ventajas que presenta el software libre es que es multiusuario, multiplataforma, sus productos se pueden redistribuir gratuitamente, etc. [GONZÁLEZ J 2003]

Debido a todas estas libertades y ventajas que presenta el software libre se ha decidido utilizar herramientas libres para desarrollar nuestra aplicación.

1.9.1 Lenguajes de programación

Uno de los aspectos que hay que tener en cuenta para el desarrollo de la aplicación es el lenguaje de programación en que se va a implementar, lo que vino a hacer realidad el dinamismo de las aplicaciones.

Existe una multitud de lenguajes y cada uno de ellos explota más a fondo ciertas características que lo hacen más o menos útiles para desarrollar distintas aplicaciones. La versatilidad de un lenguaje está íntimamente relacionada con su complejidad. Un lenguaje complicado en su aprendizaje permite en general el realizar un espectro de tareas más amplio y más profundo. Es por ello que a la hora de elegir el lenguaje que queremos utilizar tenemos que saber claramente qué es lo que debemos hacer y si el lenguaje en cuestión nos lo permite o no.

Estos lenguajes se pueden dividir a fin de entender mejor su clasificación en dos partes fundamentales: los lenguajes del lado del Servidor y los lenguajes del lado del Cliente, apuntando que ambos reconocen la filosofía de la arquitectura Cliente/Servidor para las plataformas Web.

Los lenguajes del lado del cliente: **HTML** (HyperText Markup Language), **JavaScript**, son aquellos que pueden se les puede dar formato a través de CSS (Cascading Style Sheets), que constituyen las tecnologías más usadas y son las encargadas de la representación visual de los componentes.

El lenguaje llamado **HTML** indica al navegador donde colocar cada texto, cada imagen o cada video y la forma que tendrán estos al ser colocados en la página. No es más que una serie de etiquetas que se utilizan para definir la forma o estilo que queremos aplicar a nuestro documento.

Por su parte **Javascript** es un lenguaje de programación utilizado para crear pequeños programitas encargados de realizar acciones dentro del ámbito de una página web. Se trata de un lenguaje de

programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Su uso se basa fundamentalmente en la creación de efectos especiales en las páginas y la definición de interactividades con el usuario. [TORRE 2006]

Los lenguajes del lado del servidor más ampliamente utilizados para el desarrollo de páginas dinámicas son el **ASP**, **PHP** y **PERL**. [ALVAREZ 2006]

ASP (Active Server Pages) es un lenguaje derivado del Visual Basic desarrollado por Microsoft. Evidentemente su empleo se realiza sobre plataformas funcionando bajo sistemas Windows NT. Es una tecnología del lado servidor de Microsoft para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS).

PHP (Hypertext Preprocessor) podría ser considerado como el lenguaje análogo al ASP utilizado en plataformas Unix y Linux. Fue creado originalmente en 1994 por Rasmus Lerdorf, inicialmente llamado PHP/FI. Es un lenguaje de alto nivel que se ejecuta en el lado del servidor, es libre e independiente de plataforma, o sea que es soportado por casi todos los sistemas operativos, rápido, con una gran librería de funciones y mucha documentación disponible. Es también un lenguaje interpretado en el HTML. Como PHP es un lenguaje de programación dentro de la corriente de “Open Source” (Código abierto), a lo largo de su historia ha tenido muchas contribuciones de otros desarrolladores. [COLECTIVO DE AUTORES 2004]

Por su parte el **PERL** (Practical Extracting and Reporting Language) es un lenguaje más rápido y potente que requiere obviamente un aprendizaje más largo y resulta más reservado para personas ya familiarizadas con la verdadera programación.

Cualquiera de ellos podría considerarse la opción ideal para hacer evolucionar un sitio Web realizado en HTML. Se decidió optar por PHP el cual se ejecuta del lado del servidor, por sus potencialidades y las ventajas que presenta como software libre y reúne las mejores capacidades funcionales para la creación de la aplicación Web. También se hará uso del lenguaje del lado del cliente Javascript para validar la entrada de datos por parte del usuario ya que evita que se sobrecargue el servidor.

1.9.2 Servidores Web

Los lenguajes de programación sobre la Web necesitan de un programa que atienda las peticiones de los sitios por los que navegamos, por lo que con esta finalidad fueron creados los Servidores Web.

En informática, un servidor es un tipo de software que realiza ciertas tareas en nombre de los usuarios. El término servidor ahora también se utiliza para referirse al ordenador físico en el cual funciona ese software, una máquina cuyo propósito es proveer datos de modo que otras máquinas puedan utilizar esos datos.

Este uso dual puede llevar a confusión. Por ejemplo, en el caso de un Servidor Web, este término podría referirse a la máquina que almacena y maneja los sitios Web, y en este sentido es utilizada por las compañías que ofrecen hospedaje. Alternativamente, el Servidor Web podría referirse al software que funciona en la máquina y maneja la entrega de los componentes de las páginas Web como respuesta a peticiones de los navegadores de los clientes.

En este epígrafe se hará referencia al Servidor Web como programa que implementa el protocolo HTTP (Hypertext Transfer Protocol). Este protocolo está diseñado para transferir lo que llamamos hipertextos, páginas Web o páginas HTML.

Entre los Servidores Web más utilizados en la actualidad se encuentran el **ISS** y **Apache**.

Internet Information Services (o Server), **IIS**, es una serie de servicios para los ordenadores que funcionan con Windows. Originalmente era parte del *Option Pack* para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS.

Como principal desventaja se encontró que es propiedad de Microsoft Corporation, por lo tanto solo funciona en sistemas operativos Windows, además de sus grandes problemas de seguridad.

Apache por otra parte es un servidor Web de código abierto. Tiene capacidad para servir páginas tanto de contenido estático, como de contenido dinámico a través de otras herramientas soportadas que facilitan la actualización de los contenidos mediante bases de datos, ficheros u otras fuentes de información.

Es un servidor de Web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos. Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo. Incentiva la retroalimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para solución de los mismos.

Apache presenta soporte para servidores virtuales y un sistema de registro muy configurable, además de ser libre por lo que permite la modificación del código fuente lo convierten en la elección de los grandes sitios de Internet y en el ideal para utilizar en esta aplicación.

1.9.3 Sistemas de Gestión de Bases de Datos

Un Sistema de Gestión de Bases de Datos (SGBD) es una herramienta que permite, mediante procedimientos o lenguajes, utilizar o actualizar datos almacenados más o menos permanentemente en una computadora, los que, organizados y relacionados entre sí, constituyen una base de datos. A esta organización de datos se le incorporan una serie de funciones y operaciones como definir los registros, sus campos, atributos, relaciones y también insertar, modificar, eliminar y consultar los datos.[CID and COSS 2006]

Para realizar la interconexión entre un SGBD y una aplicación Web es necesario utilizar un lenguaje de programación en el lado del servidor y para definir los datos y las estructuras, así como para hacer las consultas sobre los datos, el SQL (Structured Query Language), el cual es algo así como un lenguaje estructurado de consultas que su uso es estándar para la comunicación entre las aplicaciones y los SGBD.

Las consultas son una combinación de instrucciones que son transferidas desde el código de la aplicación Web hasta el SGBD utilizado para actualizar y manipular las bases de datos.

Entre los SGBD más utilizados tenemos **Oracle**, **Microsoft SQL Server**, **MySQL**, **PostgreSQL**, entre otros. Todos estos presentan un enfoque relacional con un buen basamento matemático centrado en el Álgebra Relacional.

Oracle es un sistema de gestión de base de datos relacional fabricado por Oracle Corporation. Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando su soporte de transacciones, estabilidad, escalabilidad, además de ser multiplataforma.

Ha sido criticado por algunos especialistas en cuanto a la seguridad de la plataforma, y las políticas de suministro de parches de seguridad. Aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otro SGBD con licencia libre como PostgreSQL. Tiene licencia privativa, más, las últimas versiones de Oracle han sido certificadas para poder trabajar bajo Linux.

Microsoft SQL Server con el ASP, este gestor de bases de datos es un sistema propietario, o sea, hay que pagar por usarlo pero es un producto muy bien logrado que en sus fuertes están los procedimientos almacenados que lo hace ganar en rendimiento y la replicación que no es más que mantener dos o más servidores de bases de datos al mismo tiempo actualizándose simultáneamente, sin embargo tiene sus problemas, el más relevante es el de la seguridad.

Otro de los SGBD más populares para las aplicaciones Web es el **MySQL**, el cual es un gestor de bases de datos relacionales, muy rápido, fiable, fácil de usar y robusto. El modelo relacional se caracteriza por disponer que toda la información esté contenida en tablas, y las relaciones entre datos deben ser representadas explícitamente en esos mismos datos, esto añade velocidad y flexibilidad. Este gestor de base de datos actualmente posee licencia privativa por lo que no es factible usarlo en este trabajo.

El Sistema Gestor de Bases de Datos Relacionales Orientados a Objetos conocido como **PostgreSQL** está derivado del paquete Postgres escrito en Berkeley. Con cerca de una década de desarrollo tras él, es el gestor de bases de datos de código abierto más avanzado hoy en día, ofreciendo control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, y tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, perl, tcl y python).

Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta. Además de esto tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos.

Se escogió **PostgreSQL,** como sistema de administración de bases de datos relacional pues es libre y posee gran escalabilidad. Tiene más de 15 años de desarrollo activo, por lo que se ha ganado la reputación de ser altamente confiable.

1.9.4 Arquitectura

La arquitectura asocia las capacidades del sistema especificadas en el requerimiento con los componentes del sistema que habrán de implementarla. La descripción arquitectónica incluye componentes y conectores (en términos de estilos) y la definición de operadores que crean sistemas a partir de subsistemas o, en otros términos, componen estilos complejos a partir de estilos simples.[FLEITAS and SABINA 2007]

La Arquitectura Orientada a Servicios (en inglés Service-Oriented Architecture o **SOA**), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario.

Es un estilo arquitectónico para la construcción de aplicaciones empresariales, donde se alinea la Infraestructura Tecnológica con los procesos de negocio. Se basa en servicios que representan funcionalidades de negocio y que se combinan entre sí para ofrecer soluciones adecuadas a las diferentes necesidades.[GIL 2008]

Se decidió usar SOA pues el sistema que se desarrollará está basado en servicios y este es un estilo que permite la reutilización e integración de componentes mucho más fácil, ayudando a reducir el tiempo de desarrollo. Además se podrá reemplazar un servicio sin tener que preocuparse por la tecnología fundamental; dándole mayor peso a la interfaz, la cual está definida bajo un estándar universal, en Servicios Web y XML. A través de protocolos de comunicación bien definidos, los servicios pueden ser invocados de manera que se hace hincapié en la interoperabilidad y en la transparencia de localización.

Debido a la necesidad de la interoperabilidad entre aplicaciones es que surge la arquitectura SOA, la cual está definida para dar soporte a servicios, por lo que, aunque no es lo mismo que los Web services, si es cierto que están íntimamente relacionados y que el auge de dichos servicios Web ha supuesto también el de este tipo de arquitectura. Proporciona independencia total de la plataforma y

del lenguaje, además de que permite generar servicios reutilizables y compartidos mediante el uso de lenguajes y protocolos totalmente estandarizados como **HTTP**, **SOAP**, **XML**, **UDDI** o **WSDL**.

Un **Servicio Web** (en inglés Web service) es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios Web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. XML es el formato estándar para los datos que se vayan a intercambiar. Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.

WSDL (*Web Services Description Language*), está basado en **XML** y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Contiene: especificación de la finalidad, funcionalidad, forma de uso y restricciones del servicio. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL.

SOAP (*Simple Object Access Protocol*) es un protocolo estándar que está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

SOA consta de cuatro elementos fundamentales: **consumidores**, **servidores**, **repositorio de servicios** y un **bus de servicio**.

Todos estos elementos aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.

1.10 Metodología de desarrollo del software.

En la actualidad de los softwares se exige mayor calidad y funcionalidad, además se requiere la entrega en el menor tiempo posible. En ocasiones muchos software fracasan porque los desarrolladores no adquieren una guía o método para desarrollar su trabajo. Cualquier camino que se escoja para obtener buena calidad implica que tiene que mejorar el proceso de desarrollo de software.

Para mejorar este proceso se proponen diferentes metodologías a seguir. En la actualidad existen varias metodologías, algunas de ellas son:

- *eXtreme Programming (XP)*
- *Rational Unified Process (RUP)*

Estas metodologías estructuran un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más eficiente y confiable.

1.10.1 eXtreme Programming (XP)

eXtreme Programming es una metodología ágil de desarrollo de software, esta se nutre de la comunicación directa entre las personas y se basa en la simplicidad, la comunicación y el reciclado continuo de código.

Los objetivos de XP son muy simples: la satisfacción del cliente, se trata de dar al cliente el software que él necesita y cuando lo necesita. El segundo objetivo es potenciar al máximo el trabajo en grupo.

Fases de XP:

Planificación: XP plantea la planificación como un permanente diálogo entre las partes la empresarial (deseable) y la técnica (posible). Las personas del negocio necesitan determinar:

Diseño: El objetivo de esta fase es hacer el diseño lo más sencillo posible.

Refactorizar: Es mejorar y modificar la estructura y codificación de códigos ya creados sin alterar su funcionalidad.

Desarrollo: Desarrollar el programa lo más simple posible.

Pruebas: Su objetivo es probar todo lo que se desarrolle para así obtener un programa más seguro.

1.10.2 El Proceso Unificado de Desarrollo (RUP)

El Proceso Unificado es un proceso de desarrollo de software, es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software.

El Proceso Unificado propone un modelamiento visual de la estructura y los componentes, para ello utiliza el Lenguaje Unificado de Modelado (UML).

UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos, permite visualizar, especificar, construir y documentar los artefactos de un sistema.

Características fundamentales del RUP:

Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Una Interacción involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada Interacción se realiza de forma planificada es por eso que se dice que son miniproyectos.

Fases del RUP:

La vida de un sistema transcurre a través de ciclos de desarrollo, desde su nacimiento hasta su muerte, en cada ciclo se repite el proceso unificado de desarrollo. Cada ciclo consta de cuatro fases:

Conceptualización (Concepción o Inicio): El objetivo de esta etapa es determinar la visión del proyecto.

Elaboración: Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen.

Construcción: Se obtiene un producto listo pata su utilización que está documentado y tiene un manual de usuario. Se obtiene 1 o varios release del producto que han pasado las pruebas.

Transición: El objetivo es llegar a obtener el release del proyecto.[SOFTWARE 2007]

Se decidió utilizar RUP como metodología de desarrollo de software pues traza una mejor y completa línea de trabajo. Es uno de los procesos más generales de los existentes en la actualidad, ya que se adapta a cualquier proyecto y proporciona una guía en el orden de las actividades de un equipo.

1.11 Herramienta CASE de modelado con UML

Las Herramientas para el modelado visual son también un aspecto importante a tener en cuenta en el desarrollo del presente trabajo. Existen varias y cada una de ellas con una serie de características que le permiten destacarse o no en dependencia de la línea de trabajo en la que se quiera insertar este tipo de herramientas.

1.11.1 Rational Rose

Es una herramienta propietaria para el modelado visual, que forma parte de un conjunto más amplio de herramientas que juntas cubren todo el ciclo de vida de desarrollo de software. Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software.

Características:

- Mantiene la consistencia de los modelos del sistema software.
- Chequeo de la sintaxis UML.
- Genera la documentación automáticamente.
- Generación de código a partir de los modelos.
- Ingeniería inversa (crear modelo a partir de código).

1.11.2 Visual Paradigm

Es una herramienta CASE que da soporte al modelado visual con UML 2.0, que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este software ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Visual Paradigm ofrece:

- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de integrarse en los principales IDEs.
- Disponibilidad en múltiples plataformas.

Para el modelado de la aplicación se ha decidido utilizar Visual Paradigm debido a que es una herramienta multiplataforma que presenta numerosas ventajas y una serie de libertades.

1.12 Conclusiones del capítulo

En este capítulo se abordaron una serie de conceptos que fundamentan todo lo referente a la parte teórica de la investigación, proporcionando a los lectores un mayor entendimiento del mismo. Además se han descrito una serie de métodos y herramientas de la Ingeniería de Requisitos así como los lenguajes de programación, los servidores web, los sistemas de gestión de bases de datos y la metodología a utilizar para guiar el desarrollo de la aplicación. Todo esto unido a una descripción de los estándares necesarios para desarrollar un Repositorio de Requisitos Reutilizables.

Capítulo 2. Características del sistema

En este capítulo se realiza la descripción de las características del sistema. Además se describe el Modelo de Dominio, definiendo conceptos útiles para entender el contexto del problema. Se enumeran los requisitos no funcionales y funcionales, estos últimos se estructuran mediante los Casos de Uso del sistema, de los cuales se ofrece una descripción textual.

2.1 Propuesta del sistema

El REpositorio de REquisitos REutilizables (RЭ) almacenará requisitos los cuales podrán ser utilizados por otros desarrolladores, que a su vez podrán subir nuevos requisitos para el uso de toda la comunidad, lo que garantizaría la reutilización de requisitos de software en los nuevos proyectos de la UCI. Para ello dichos requisitos deberán ser gestionados teniendo en cuenta los estándares asociados a los mismos.

En RЭ serán almacenados paquetes, los cuales contendrán una serie de funcionalidades y características de un producto determinado. Dichos paquetes de requisitos serán importados por herramientas encargadas de gestionar requisitos.

El objetivo de almacenar dichos paquetes es que otros usuarios puedan acceder a ellos y utilizarlos en parte o en su totalidad en dependencia de hasta que punto guardan relación con el nuevo software que vayan a desarrollar, por lo que también debe brindar la opción de exportar esos paquetes. Esto evitará realizar estos procesos manualmente, lo que conlleva a tener que dedicarle menos tiempo a este flujo de trabajo.

La búsqueda de requisitos es otro de los principales servicios que brindará el repositorio. Una característica a resaltar es que estos usuarios no tendrán necesidad de abrir los paquetes hasta estar seguros que dentro están los requisitos que pueden reutilizar en su software, ya que cuentan con metadatos que contendrán toda la información necesaria sobre requisitos contenidos en ese paquete.

Esta es una idea general de las principales características del repositorio, que aportarán ventajas al desarrollo de un software pues el levantamiento de requisitos es uno de los primeros y más importantes momentos en la vida del mismo.

Asegurando un correcto levantamiento de requisitos se construyen bases lo suficientemente fuertes como para no tener problemas en futuras etapas del proceso de desarrollo del software.

2.2 Modelo de Dominio

Para desarrollar un sistema de calidad se necesita dividirlo en piezas para así comprenderlo y gestionar mejor su complejidad. Estas piezas se pueden representar a través de modelos que permiten organizar y captar las características fundamentales de dicho sistema. Estos modelos pueden ser el Modelo del Dominio o el Modelo del Negocio.

Debido a que los procesos de negocio no son visibles, o sea, la información no fluye con la calidad requerida y además las reglas para cumplir su función u objetivo son difíciles de establecer, no es recomendable hacer un Modelo de Negocio de ahí que se propone realizar un modelo de dominio.

Un modelo del dominio captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema, mostrando al usuario los principales conceptos que se manejan. Mediante este modelo los usuarios, clientes y desarrolladores utilizan un vocabulario común para poder entender el contexto en que se muestra el sistema. Además permite capturar los requisitos necesarios para construir el sistema de acuerdo a las características que el cliente solicite y desee.

2.2.1 Descripción del Problema de Dominio

El sistema propuesto es un Repositorio de Requisitos Reutilizables el cual permite la reutilización de requisitos en diferentes proyectos. Este repositorio es un lugar donde profesores y estudiantes pueden almacenar requisitos de diferentes tipos ya sean de software, sistema, prueba, entre otros, propiciando así su reutilización.

El sistema presenta en su pantalla principal un mensaje de bienvenida, describiendo los servicios que se ofrecen al usuario junto con la opción de registrarse, o si ya está registrado, poder utilizar el sistema.

Para que un usuario pueda registrarse debe insertar su nombre de usuario y su contraseña previamente escogidos los cuales deben validarse.

Una vez registrado el usuario, y después de haberse validado su acceso en el sistema, puede seleccionar las siguientes actividades:

Realizar búsqueda de requisitos:

Todos los usuarios pueden realizar búsquedas de paquetes de requisitos, para esto se debe especificar una serie de criterios con los que deben cumplir la información que espera encontrar.

Consultar metadatos:

Permite a los usuarios ver los datos relacionados con los paquetes de requisitos.

Exportar paquetes de requisitos:

Permite a los analistas exportar paquetes de requisitos para su utilización previa.

Importar paquetes de requisitos:

Permite a los analistas importar paquetes de requisitos especificando el tipo de requisito a que pertenece, además de su metadato correspondiente.

Para que un paquete de requisitos pueda ser visible dentro del repositorio este debe ser revisado antes por un analista jefe.

Revisar requisitos:

Permite a los usuarios con privilegios como el analista jefe la revisión de los paquetes de requisitos importados por diferentes analistas.

El sistema permite además la opción de realizar comentarios en la cual los analistas tienen la posibilidad de emitir sus opiniones y sugerencias acerca de los paquetes de requisitos.

2.2.2 Definición de los conceptos fundamentales

Requisito: Necesidad o condición de un sistema.

Paquete de requisitos: Conjunto de requisitos reutilizables.

Metadato: Descripción de los requisitos que se encuentran dentro de los paquetes.

Categoría: Sección donde se guardan determinados paquetes de requisitos.

Usuario: Persona que interactúa con la aplicación.

Comentario: Articulo de opinión.

2.2.3 Diagrama del Modelo de Dominio

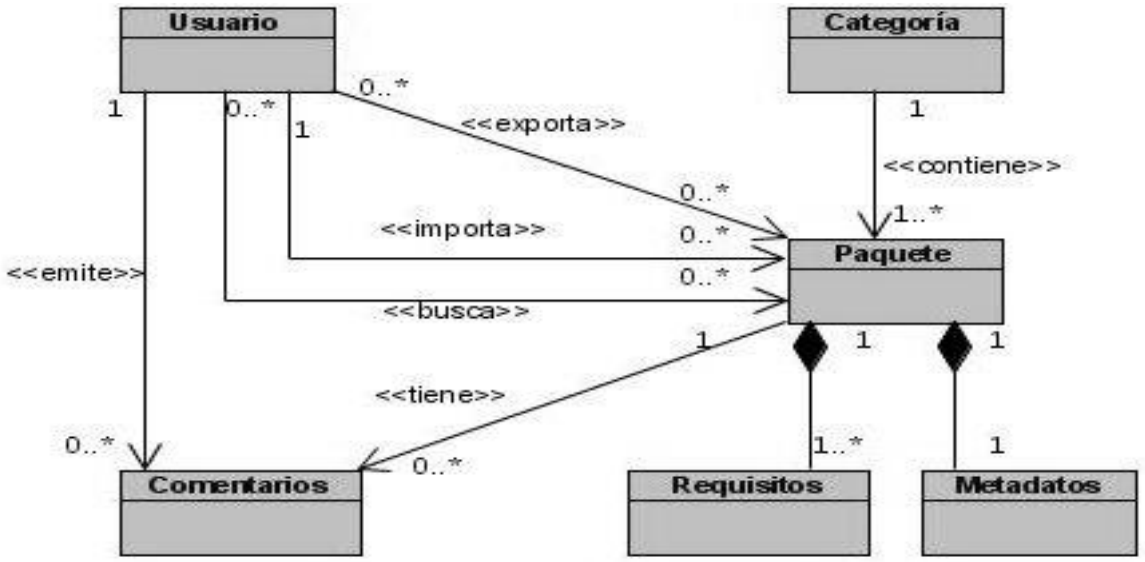


Figura 2 Diagrama del Modelo de Dominio

2.3 Levantamiento de Requisitos

Los requisitos definen lo que el sistema debe hacer, para lo cual se identifica las funcionalidades requeridas y las restricciones que se imponen. Los requisitos se pueden clasificar en: funcionales y no funcionales.

2.3.1 Requisitos funcionales

El sistema debe ser capaz de:

R1: Autenticar usuario.

R2: Editar perfil.

R 2.1: Visualizar perfil.

R3: Buscar paquetes de requisitos.

R4: Visualizar metadatos.

R5: Visualizar paquetes de requisitos.

R6: Exportar paquetes de requisitos.

R7: Actualizar paquetes de requisitos.

R7.1: Incluir paquetes de requisitos en su área de trabajo.

R7.2: Excluir paquetes de requisitos de su área de trabajo.

R8: Comentar paquetes de requisitos.

R9: Importar paquetes de requisitos.

R10: Gestionar metadatos.

R 10.1: Adicionar metadato a paquete de requisitos.

R 10.2: Modificar metadato de paquete de requisitos.

R 10.3: Eliminar metadato de paquete de requisitos.

R11: Revisar paquete de requisitos.

R 11.1: Autorizar paquete de requisitos.

R 11.2: Denegar paquete de requisitos.

R12: Eliminar paquete de requisitos.

R13: Gestionar usuarios.

R13.1: Crear cuenta de usuario.

R13.2: Eliminar cuenta de usuario.

R13.3: Modificar cuenta de usuario.

R14: Gestionar categoría.

R14.1: Adicionar categoría.

R14.2: Eliminar categoría.

R14.3: Modificar categoría.

R15: Ver reporte

R15.1: Ver lista de usuarios online.

R15.2: Ver lista de paquetes recientes.

R15.3: Ver lista de paquetes con el id del analista que lo creó y la fecha en que fue creado.

R16: Configurar HTTPs.

R17: Administrar interfaz gráfica.

R17.1: Configurar bloques.

R17.2: Configurar texto de inicio.

2.3.2 Requisitos no funcionales

Usabilidad:

Acceso rápido a los principales servicios.

Rendimiento:

Facilitar a los usuarios el rápido acceso a las páginas.

Garantizar la menor velocidad de respuesta posible.

Permitir numerosas conexiones simultáneas.

Portabilidad:

La aplicación debe funcionar en plataformas Unix, Windows y Macintosh, siendo posible el acceso a través de cualquier navegador web.

Seguridad:

Garantizar la seguridad del sistema, controlando acceso y funciones realizadas por los usuarios.

Confiabilidad:

La información manejada por el sistema está protegida de acceso no autorizado.

Legales:

La plataforma escogida para el desarrollo de la aplicación, debe estar basada en la licencia GNU/GPL.

Interfaz:

Se debe tener en cuenta algunos elementos de diseño como gráficos de encabezamiento, estilos y formatos de texto, paletas de color de los gráficos y colores o patrones del fondo adecuados.

Navegación sencilla y construcción de enlaces rápidos.

Ayuda y documentación en Línea.

Ayuda incluida en el sistema administrativo.

Contar con manual de usuario.

Software:

Servidor Web Apache.

Servidor de base de datos PostgreSQL.

El sistema se implementará con tecnología PHP.

Sistema Operativo Linux o Windows.

Hardware:

Microprocesador 200 MHz.

32 MB de memoria RAM.

2 GB de disco duro.

Respaldo eléctrico (UPS).

Mantenibilidad:

Utilización de estándares para el desarrollo de aplicaciones Web.

2.4 Modelo de Casos de Uso del Sistema

Se representan los requisitos funcionales del sistema mediante un diagrama de casos de uso utilizando UML como lenguaje de modelado visual, donde se definen los actores y los casos de uso del mismo.

2.4.1 Actores del sistema

Actores	Justificación
Usuario	Representa a una persona que interactúa con el sistema, la cual puede realizar una serie de opciones básicas.
Analista	Representa a un usuario con privilegios que tiene la opción de importar paquetes de requisitos al sistema, gestionar sus metadatos correspondientes, entre otros.
Analista Jefe	Representa a un usuario con privilegios que revisa los paquetes de requisitos importados por los analistas, también puede eliminar los que considere inútiles.
Administrador	Representa a un usuario avanzado con privilegios el cual actualiza las categorías en las cuales se guardan los paquetes de requisitos y gestiona todo lo referente a los usuarios (crear/eliminar cuenta de usuario), entre otras cosas.

Tabla 1 Actores del sistema

2.4.2 Casos de Uso del Sistema

CU-1	Autenticar usuario
Actor	Usuario
Descripción	Permite al usuario autenticarse y acceder al sistema.
Referencia	R1

Tabla 2 Caso de uso: Autenticar Usuario

CU-2	Editar perfil
Actor	Analista
Descripción	Posibilita al analista editar sus datos personales en el sistema.
Referencia	R2

Tabla 3 Caso de uso: Editar perfil

CU-3	Buscar requisitos
Actor	Usuario
Descripción	Permite a los usuarios realizar búsquedas de paquetes de requisitos.
Referencia	R3

Tabla 4 Caso de uso: Buscar requisitos

CU-4	Consultar metadato
Actor	Usuario
Descripción	Muestra los metadatos asociados a los paquetes de requisitos, esto permite obtener una visión general de los paquetes de requisitos.
Referencia	R4

Tabla 5 Caso de uso: Consultar metadato

CU-5	Consultar paquete de requisitos
Actor	Analista
Descripción	Posibilita a los analistas consultar los paquetes de requisitos para ver su contenido y valorar si los puede usar o no.
Referencia	R5

Tabla 6 Caso de uso: Consultar paquete de requisitos

CU-6	Exportar paquete de requisitos
Actor	Analista
Descripción	Permite a los analistas exportar paquetes de requisitos para su utilización previa.
Referencia	R6

Tabla 7 Caso de uso: Exportar paquete de requisitos

CU-7	Actualizar paquete de requisitos
Actor	Analista
Descripción	Posibilita a los analistas la opción de incluir o excluir paquetes de requisitos a su área de trabajo.
Referencia	R7

Tabla 8 Caso de uso: Actualizar paquete de requisitos

CU-8	Comentar paquete de requisitos
Actor	Analista
Descripción	Permite a los analistas la opción de realizar comentarios acerca de los paquetes de requisitos.
Referencia	R8

Tabla 9 Caso de uso: Comentar paquete de requisitos

CU-9	Importar paquete de requisitos
Actor	Analista
Descripción	Posibilita a los analistas importar los paquetes de requisitos al sistema.
Referencia	R9

Tabla 10 Caso de uso: Importar paquete de requisitos

CU-10	Gestionar metadato
Actor	Analista
Descripción	Posibilita a los usuarios adicionar, modificar o eliminar los metadatos de los paquetes de requisitos importados.
Referencia	R10

Tabla 11 Caso de uso: Gestionar metadato

CU-11	Revisar paquete de requisitos
Actor	Analista Jefe
Descripción	Posibilita al Analista Jefe realizar un análisis sobre los paquetes de requisitos importados por los analistas, este decide si debe dejarlos en el sistema o eliminarlos.
Referencia	R11

Tabla 12 Caso de uso: Revisar paquete de requisitos

CU-12	Eliminar paquete de requisitos
Actor	Analista Jefe
Descripción	Permite al Analista Jefe eliminar paquetes de requisitos que no sean útiles en el sistema o estén mal confeccionados.
Referencia	R12

Tabla 13 Caso de uso: Eliminar paquete de requisitos

CU-13	Gestionar usuario
Actor	Administrador
Descripción	Permite al administrador gestionar todo lo referente a los usuarios ya sea crear, modificar o eliminar cuentas de usuarios de acceso al sistema.
Referencia	R13

Tabla 14 Caso de uso: Gestionar usuario

CU-14	Gestionar categoría
Actor	Administrador
Descripción	Posibilita al administrador gestionar las categorías del sistema, (adicionar, eliminar o modificar).
Referencia	R14

Tabla 15 Caso de uso: Gestionar categoría

CU-15	Ver reporte
Actor	Administrador
Descripción	Posibilita al administrador llevar el control de los diferentes reportes como por ejemplo los usuarios en línea, los paquetes de requisitos más recientes, entre otros.
Referencia	R15

Tabla 16 Caso de uso: Ver reporte

CU-16	Configurar HTTPs
Actor	Administrador
Descripción	Posibilita al administrador acceder al sistema para realizar configuraciones respecto al protocolo de seguridad.
Referencia	R16

Tabla 17 Caso de uso: Configurar HTTPs

CU-17	Administrar interfaz gráfica
Actor	Administrador
Descripción	Posibilita al administrador llevar el control sobre la interfaz gráfica de la aplicación como añadir/eliminar bloques o configurar el texto de inicio.
Referencia	R17

Tabla 18 Caso de uso: Administrar interfaz gráfica

2.4.3 Diagrama de Casos de Uso del Sistema

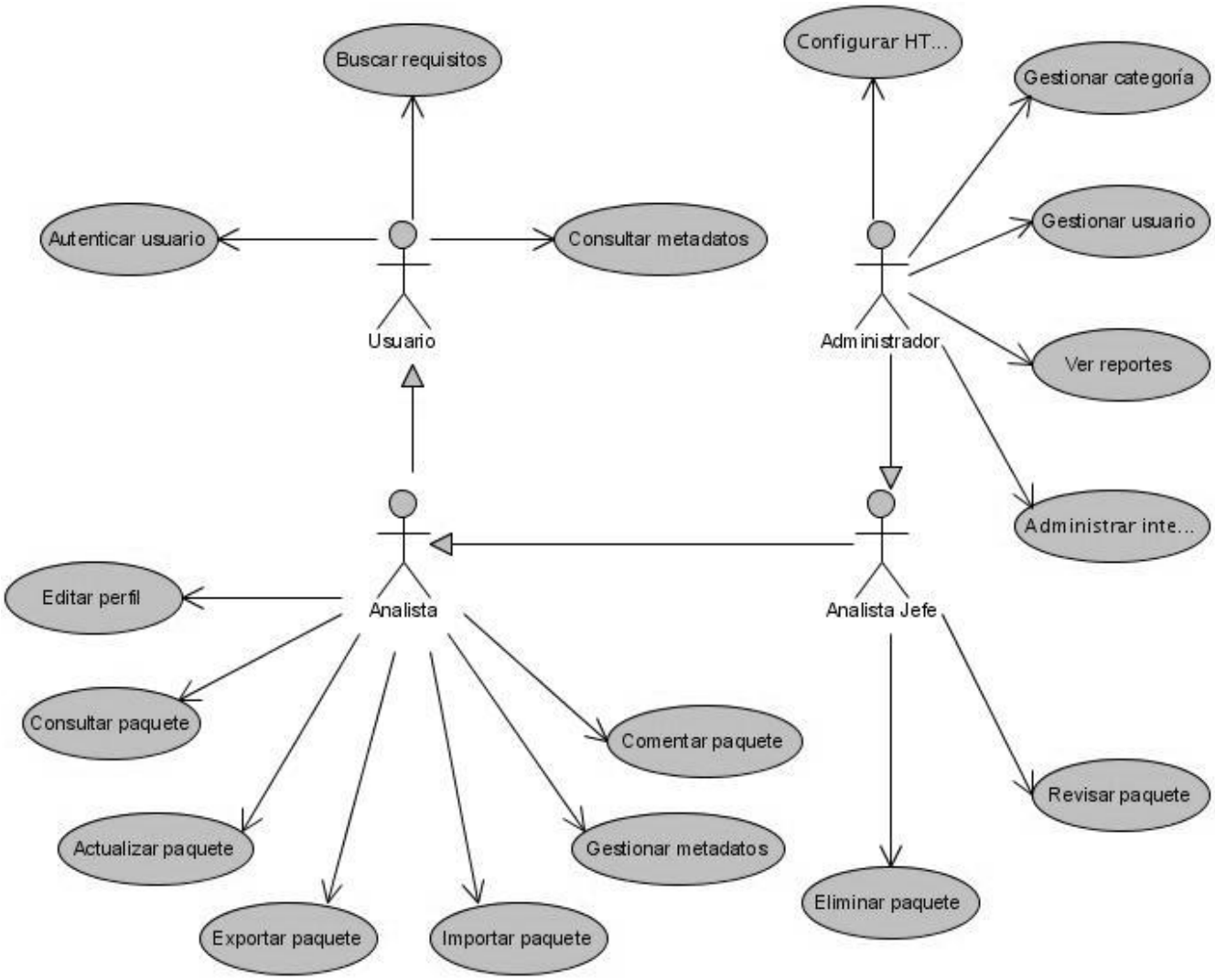


Figura 3 Diagrama de casos de uso del sistema

2.4.4 Descripción textual de los casos de uso

Para un mayor entendimiento de los casos de uso es recomendable observar las descripciones textuales de los mismos, para ello se muestran las de los cuatro primeros casos de uso, los restantes se realizan siguiendo la misma línea de trabajo. (Ver Anexo 5)

2.5 Conclusiones del capítulo

En este capítulo se han descrito las características del sistema, precisando cómo se desarrollará y cada uno de los elementos que serán utilizados. Además, se obtuvo un modelo de dominio, donde se muestra el comportamiento del sistema. Se precisaron los requisitos funcionales y no funcionales, así como los actores y los casos de uso correspondientes al sistema. También se ilustraron las relaciones entre los actores y casos de uso del sistema mediante un diagrama de casos de uso a partir del cual ya se tiene una mejor visión para realizar el análisis y el diseño del sistema.

Capítulo 3. Análisis y Diseño del sistema

En el proceso de desarrollo de RЭ, la fase de elaboración tiene una gran importancia, de ahí que el presente capítulo se centre en el análisis y diseño del mismo. Se representan un grupo de artefactos que describen como implementar el sistema, por ejemplo: los diagramas de clases de análisis y de clases de diseño por extensiones Web, los diagramas de colaboración y secuencia. Además se presenta el modelo de datos que es la base para construir la base de datos que soportará toda la información del sistema, y se termina con la descripción de sus tablas.

3.1 Análisis del sistema

El análisis del sistema es uno de los flujos de trabajos realizados durante la fase de elaboración. Este permite a través del modelo de análisis refinar la funcionalidad del sistema. El modelo de análisis contiene clases de análisis y sus objetos organizados en paquetes que colaboran. A continuación, se describen las clases que serán utilizadas en la realización de los diagramas de clases de análisis de cada caso de uso.

3.1.1 Clases de Análisis

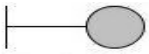
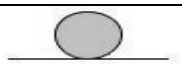
 **Clase Interfaz**	Modelan la interacción entre el sistema y sus actores.
 **Clase Control**	Coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.
 **Clase Entidad**	Modelan información que posee larga vida y que es a menudo persistente.

Figura 4 Clases de análisis

Una aplicación cliente/servidor está conformada por tres capas, en la capa de usuario aparecen fundamentalmente las clases interfaz ya que aquí se ejecutan las aplicaciones del cliente. En la capa intermedia están las clases de control debido a que en ellas se agrupan los servicios que son

compartidos por múltiples aplicaciones y en la tercera capa estarían las clases entidad porque aquí se tiene toda la información o sea a los que se denomina base de datos.

3.1.2 Diagramas de Clases de Análisis

El Modelo de Clases de Análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa las cosas del mundo real, no de la implementación automatizada de estas cosas. A continuación los diagramas correspondientes al análisis del sistema en cuestión:

Diagrama de clases de Análisis: Autenticar Usuario

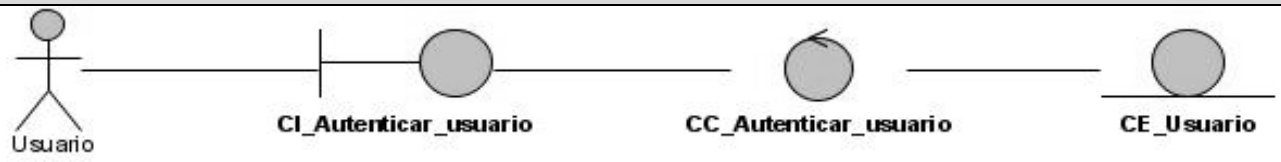


Figura 5 Diagrama de clases de análisis (CU Autenticar usuario)

Diagrama de clases de Análisis: Editar Perfil

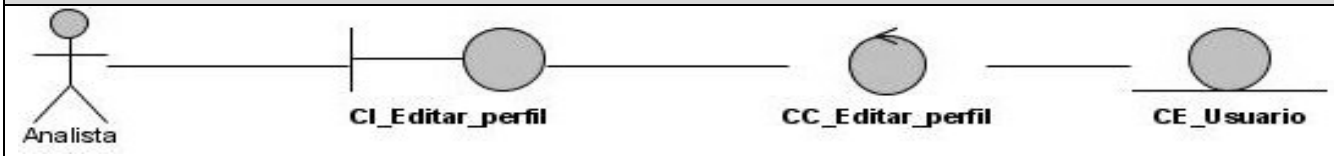


Figura 6 Diagrama de clases de análisis (CU Editar Perfil)

Diagrama de clases de Análisis: Buscar Requisitos

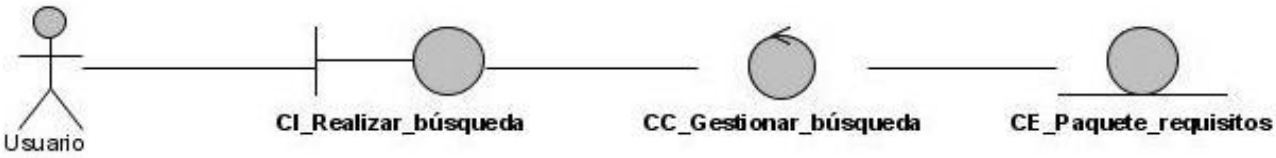


Figura 7 Diagrama de clases de análisis (CU Buscar requisitos)

Diagrama de clases de Análisis: Consultar Metadato

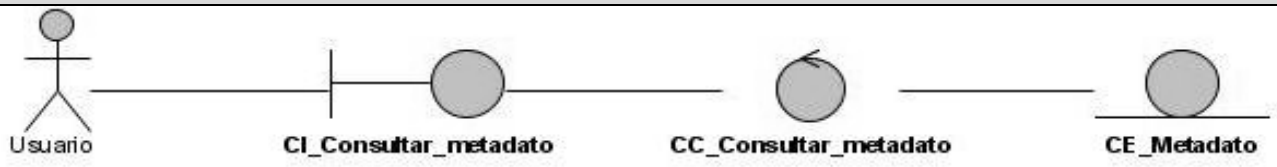


Figura 8 Diagrama de clases de análisis (CU Consultar metadato)

Diagrama de clases de Análisis: Consultar Paquete de Requisitos

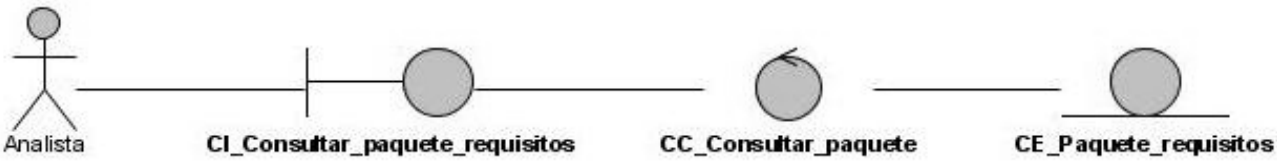


Figura 9 Diagrama de clases de análisis (CU Consultar paquete de requisitos)

Diagrama de clases de Análisis: Exportar Paquete de Requisitos

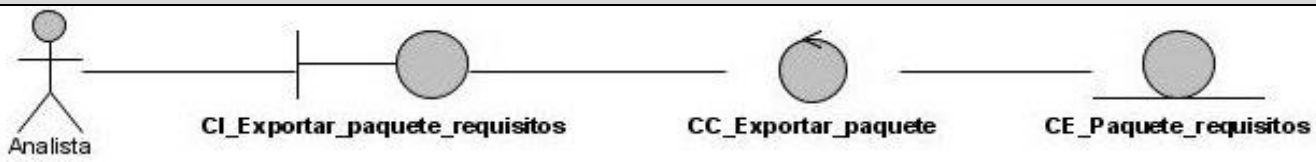


Figura 10 Diagrama de clases de análisis (CU Exportar paquete de requisitos)

Diagrama de clases de Análisis: Actualizar Paquete de Requisitos

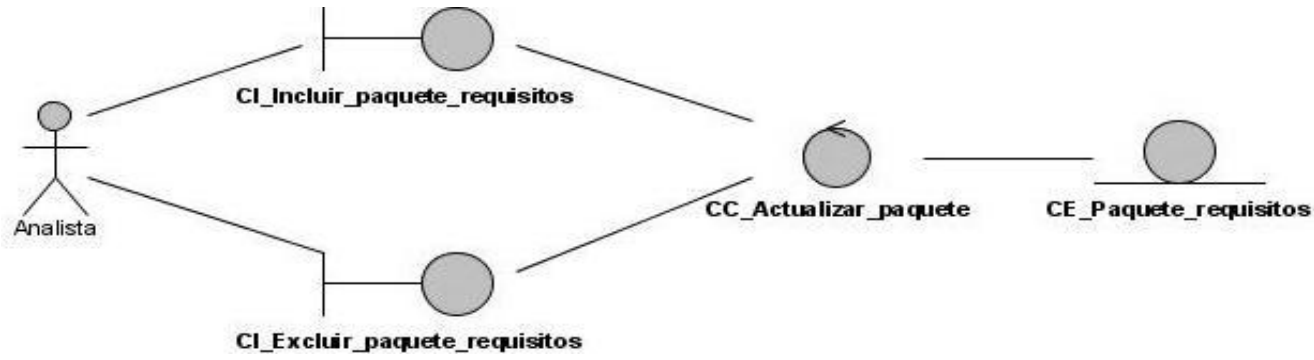


Figura 11 Diagrama de clases de análisis (CU Actualizar paquete de requisitos)

Diagrama de clases de Análisis: Comentar Paquete de Requisitos

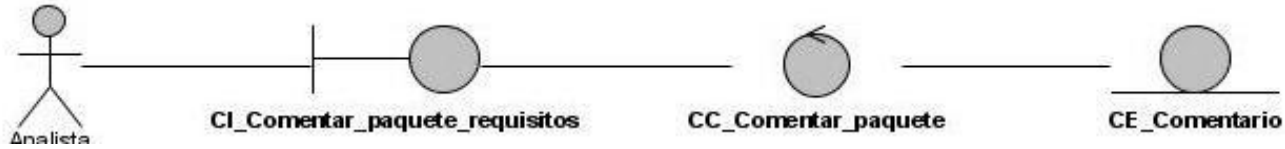


Figura 12 Diagrama de clases de análisis (CU Comentar paquete de requisitos)

Diagrama de clases de Análisis: Importar Paquete de Requisitos

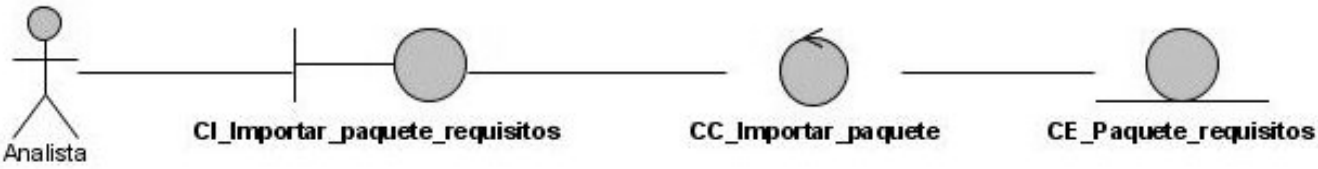


Figura 13 Diagrama de clases de análisis (CU Importar paquetes de requisitos)

Diagrama de clases de Análisis: Gestionar Metadato

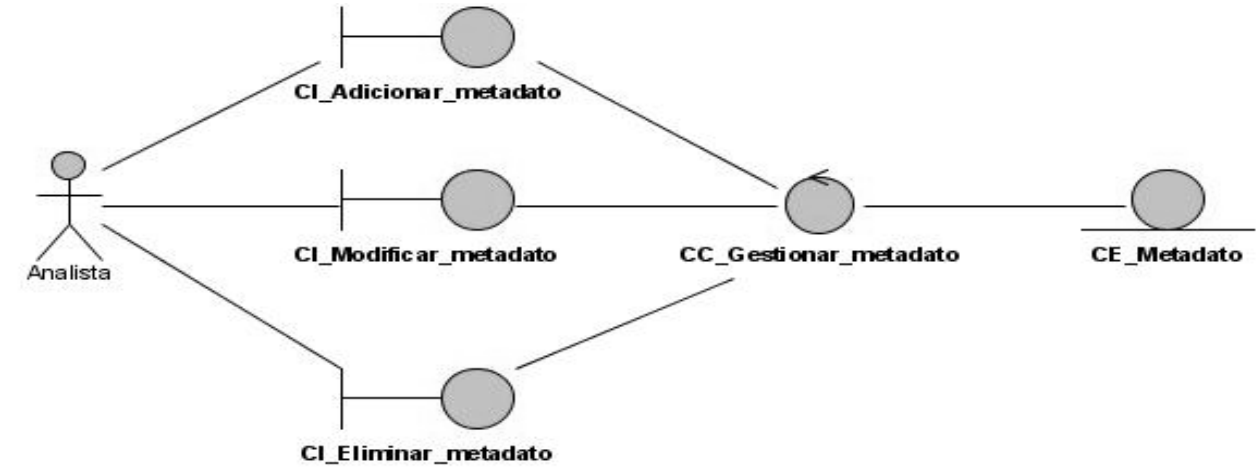


Figura 14 Diagrama de clases de análisis (CU Gestionar metadato)

Diagrama de clases de Análisis: Revisar Paquete de Requisitos

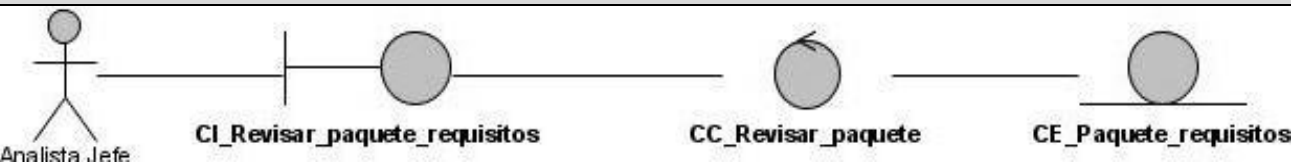


Figura 15 Diagrama de clases de análisis (CU Revisar paquete de requisitos)

Diagrama de clases de Análisis: Eliminar Paquete de Requisitos

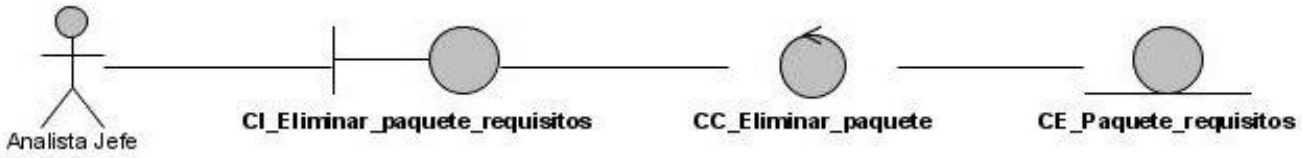


Figura 16 Diagrama de clases de análisis (CU Eliminar paquete de requisitos)

Diagrama de clases de Análisis: Gestionar Usuario

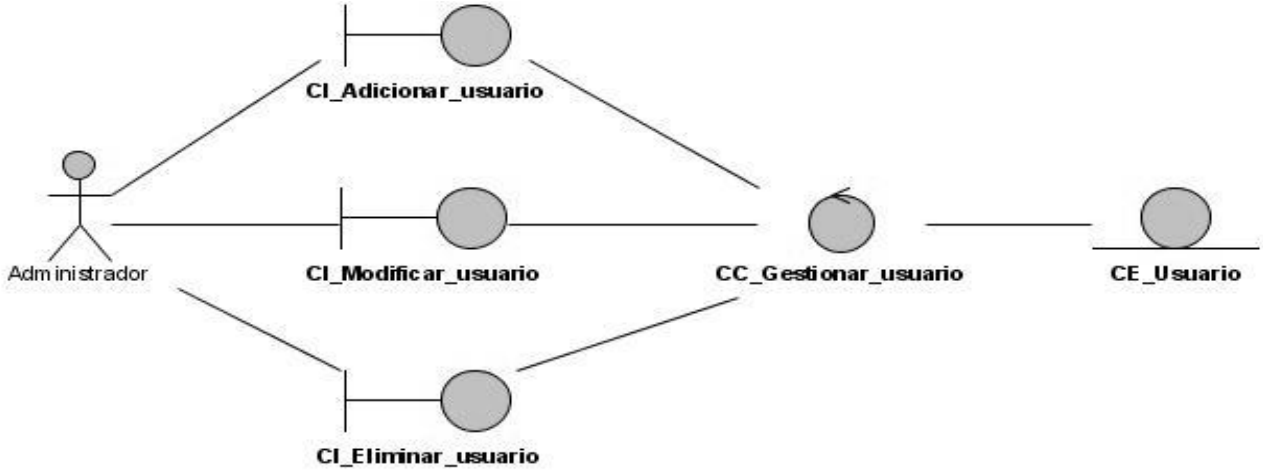


Figura 17 Diagrama de clases de análisis (CU Gestionar usuario)

Diagrama de clases de Análisis: Gestionar Categoría

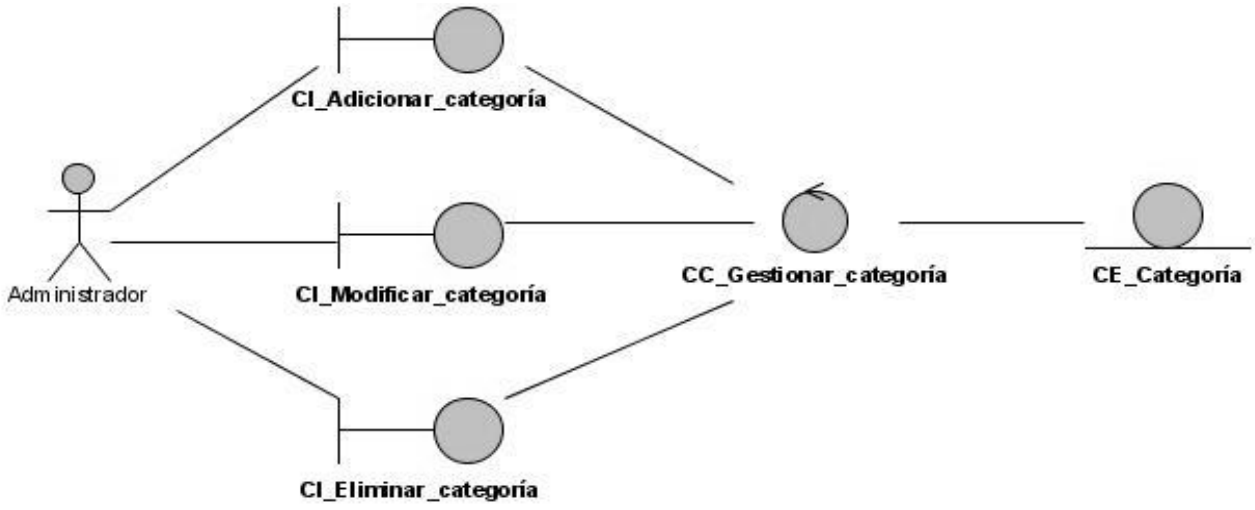


Figura 18 Diagrama de clases de análisis (CU Gestionar categoría)

Diagrama de clases de Análisis: Ver Reporte

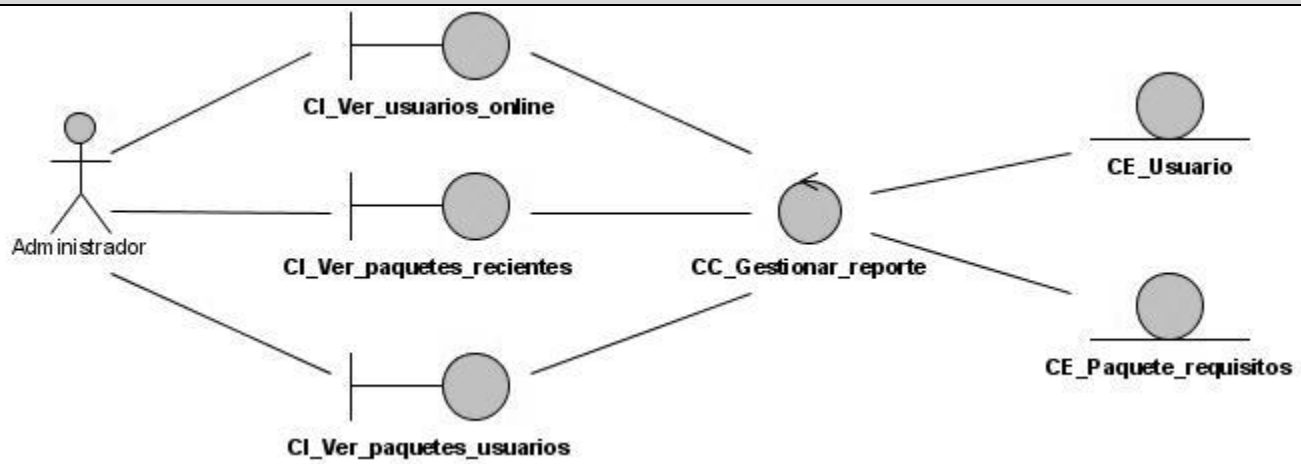


Figura 19 Diagrama de clases de análisis (CU Ver reporte)

Diagrama de clases de Análisis: Configurar HTTPs

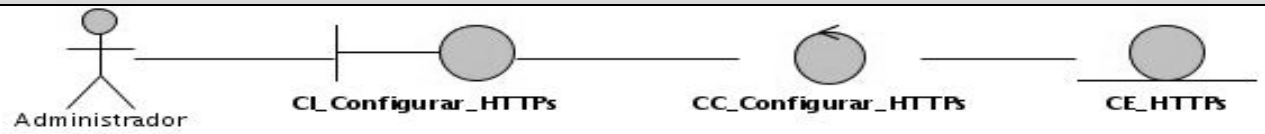


Figura 20 Diagrama de clases de análisis (CU Configurar HTTPs)

Diagrama de clases de Análisis: Administrar interfaz gráfica

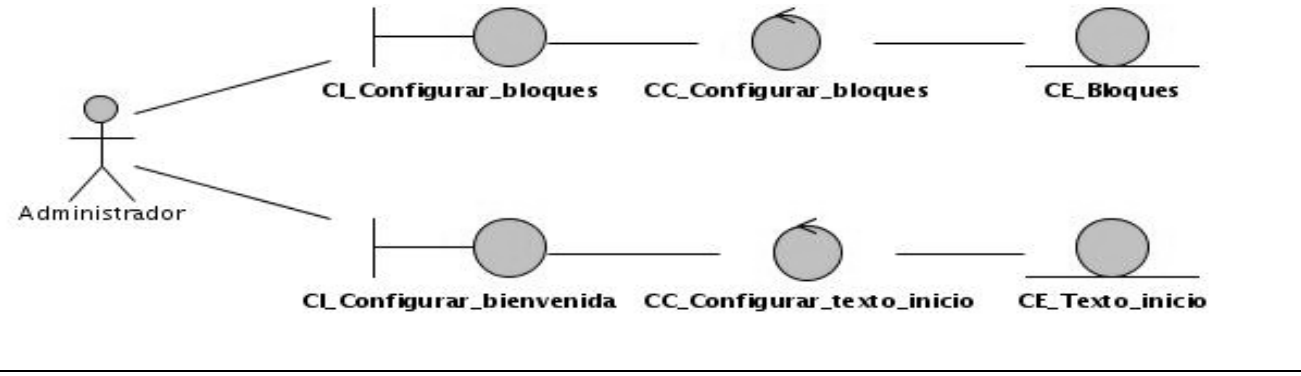


Figura 21 Diagrama de clases de análisis (CU Administrar interfaz gráfica)

3.1.3 Diagramas de Interacción de Análisis

Los diagramas de interacción explican gráficamente las interacciones existentes entre las instancias y las clases del modelo de éstas. El tipo de diagrama seleccionado para construir los diagramas de interacción de análisis fue el de colaboración.

A continuación se muestran los diagramas de colaboración de los cuatro primeros casos de uso, los restantes se realizan siguiendo la misma línea de trabajo (Ver Anexo 6)

3.2 Diseño del Sistema

El diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva cómo cumple el sistema sus objetivos. Además impone una estructura del sistema que debemos esforzarnos por conservar lo más fielmente posible cuando demos forma al mismo. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades.

3.2.1 Clases de Diseño

A continuación se nombran cada una de los tipos de clases que son utilizadas en el presente modelo de diseño:

Clases de la capa Presentación

CP_<Nombre de la página>: Son las páginas que van a funcionar como interfaz a los usuarios. Se construirán dinámicamente para ser visualizadas en el explorador de los usuarios.

SP_<Nombre de la página>: Son las páginas servidoras que construyen a las páginas clientes y tienen toda la lógica de presentación. Invocan todos los métodos necesarios de la capa lógica a través de las clases de servicio.

fr_<Nombre del formulario>: Son los formularios que se utilizan para obtener los datos introducidos por el usuario en cada una de las actividades que se realiza durante el procesamiento de un documento.

Clases de la capa Servicio

cs_<Nombre de la clase>: Son las clases que brindan los servicios necesarios a la capa de presentación. Contienen la lógica de la aplicación, en forma de transacciones de negocio. Realiza una interfaz de servicio donde se exponen todos los servicios necesarios para la capa de presentación referidos a la lógica, lo que permite separar físicamente ambas capas.

Clases de la capa Dominio

bc_<Nombre de la clase>: Son las clases que contienen las reglas del negocio de la aplicación relacionadas con una entidad del negocio.

be_<Nombre de la clase>: Son las entidades del negocio, clases que contienen todos los datos, no tienen lógica de negocio. Pueden ser objetos (individuales o colecciones).

Clases de la capa de Datos

Como no es objetivo describir las clases que están presentes en la capa de datos, la misma se representará a través de un paquete Acceso a Datos, el cual realiza una interfaz de persistencia.

3.2.2 Diagrama de Clases de Diseño

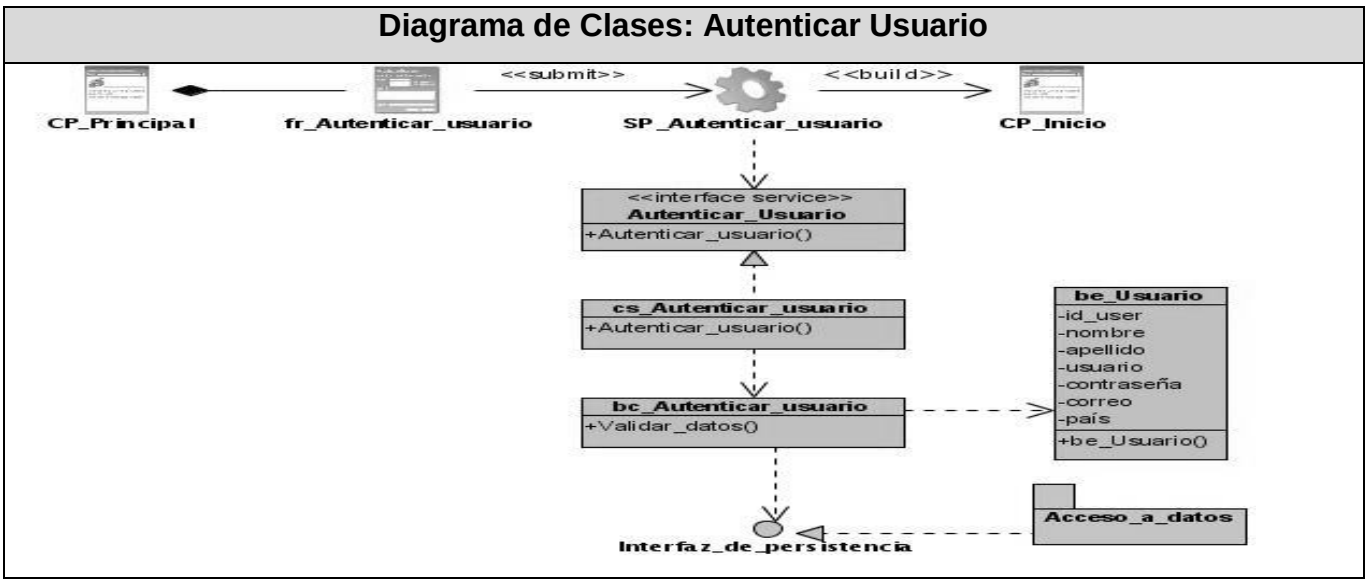


Figura 22 Diagrama de clases de diseño (CU Autenticar usuario)

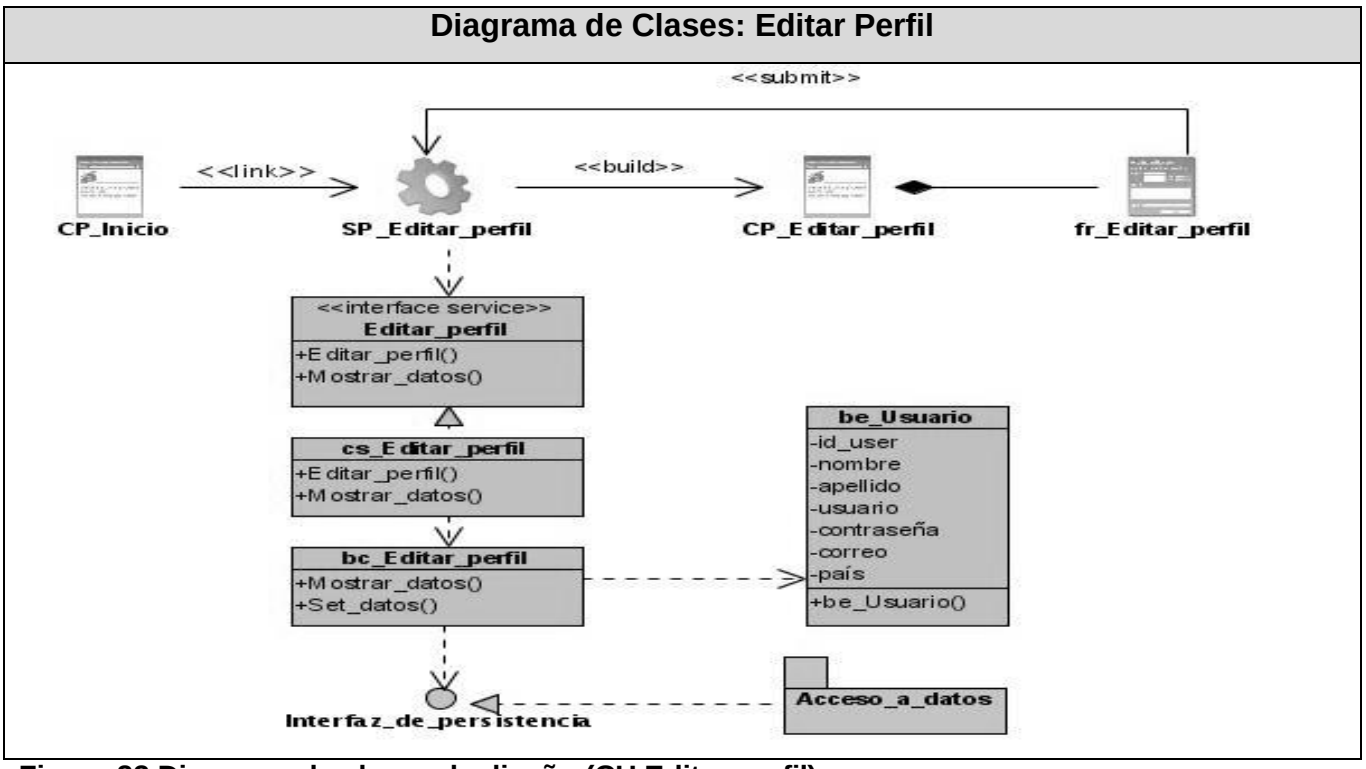


Figura 23 Diagrama de clases de diseño (CU Editar perfil)

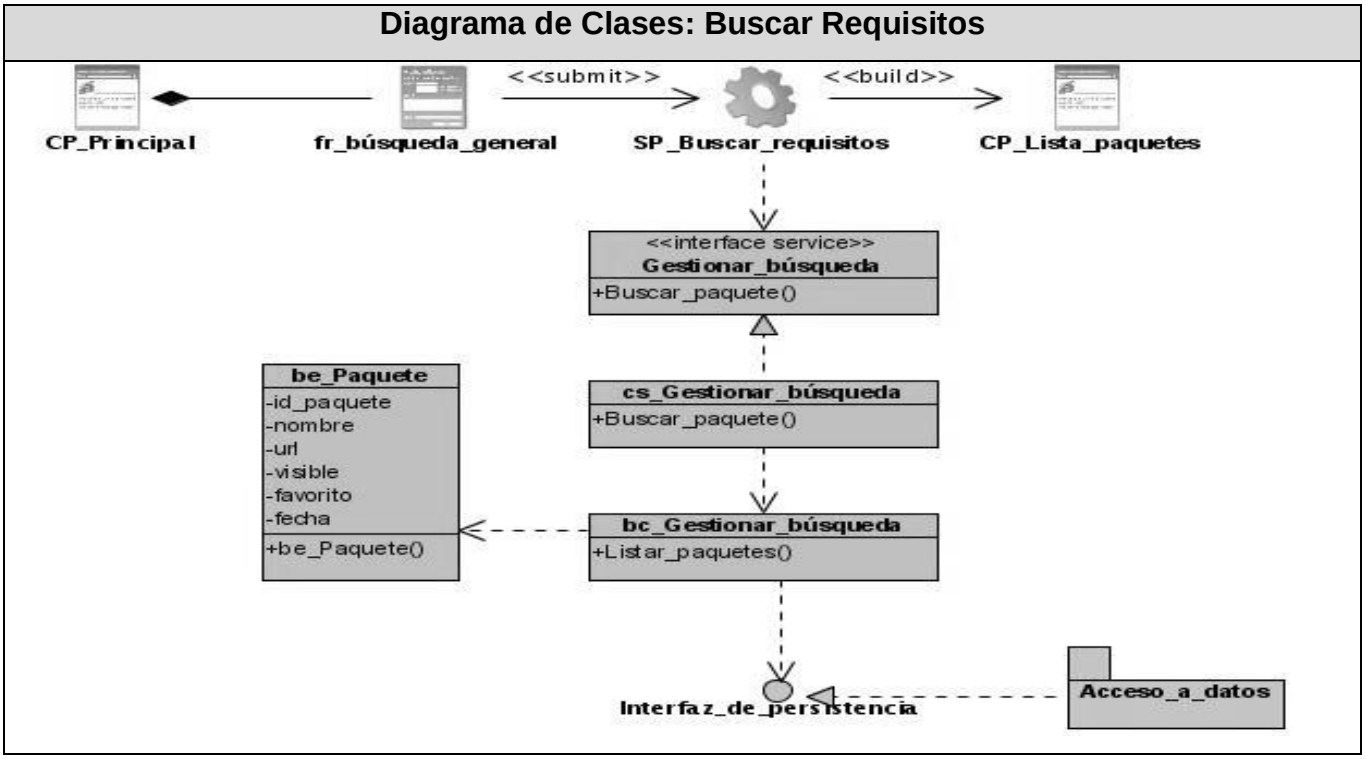


Figura 24 Diagrama de clases de diseño (CU Buscar requisitos)

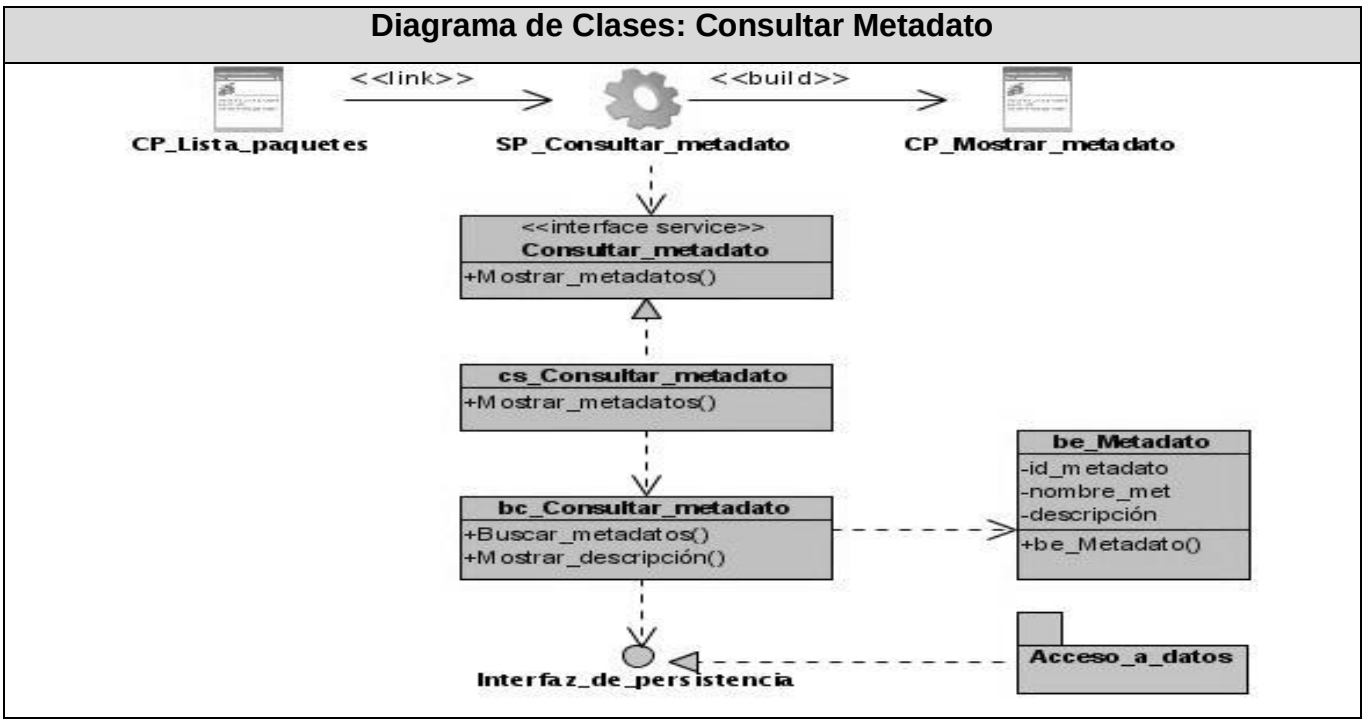


Figura 25 Diagrama de clases de diseño (CU Consultar metadato)

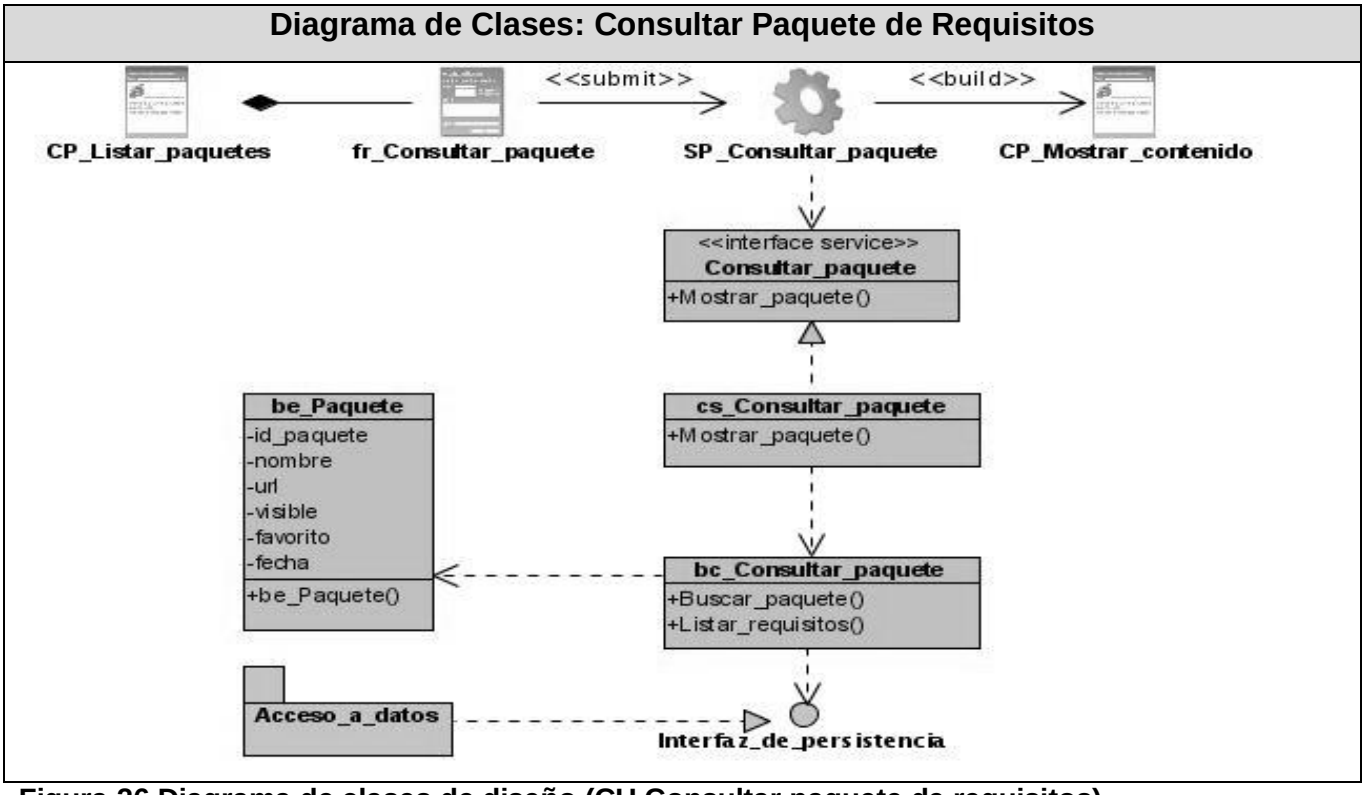


Figura 26 Diagrama de clases de diseño (CU Consultar paquete de requisitos)

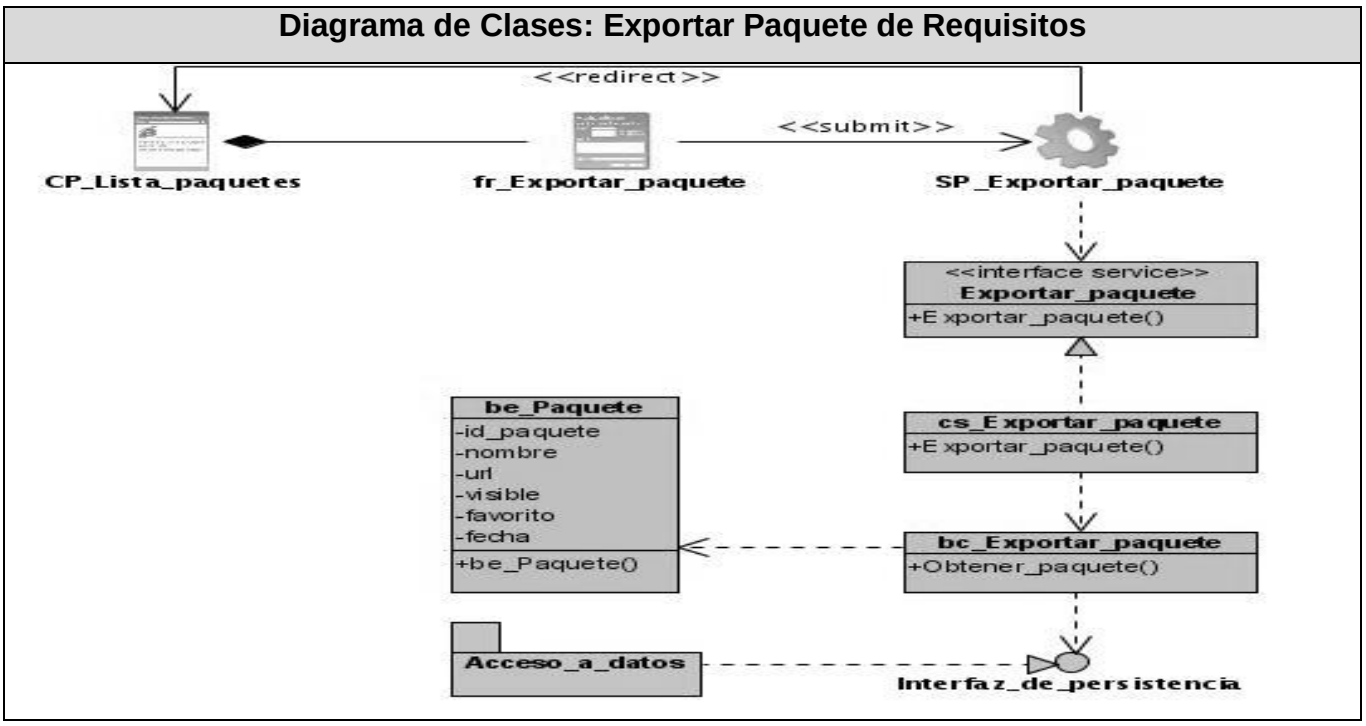


Figura 27 Diagrama de clases de diseño (CU Exportar paquete de requisitos)

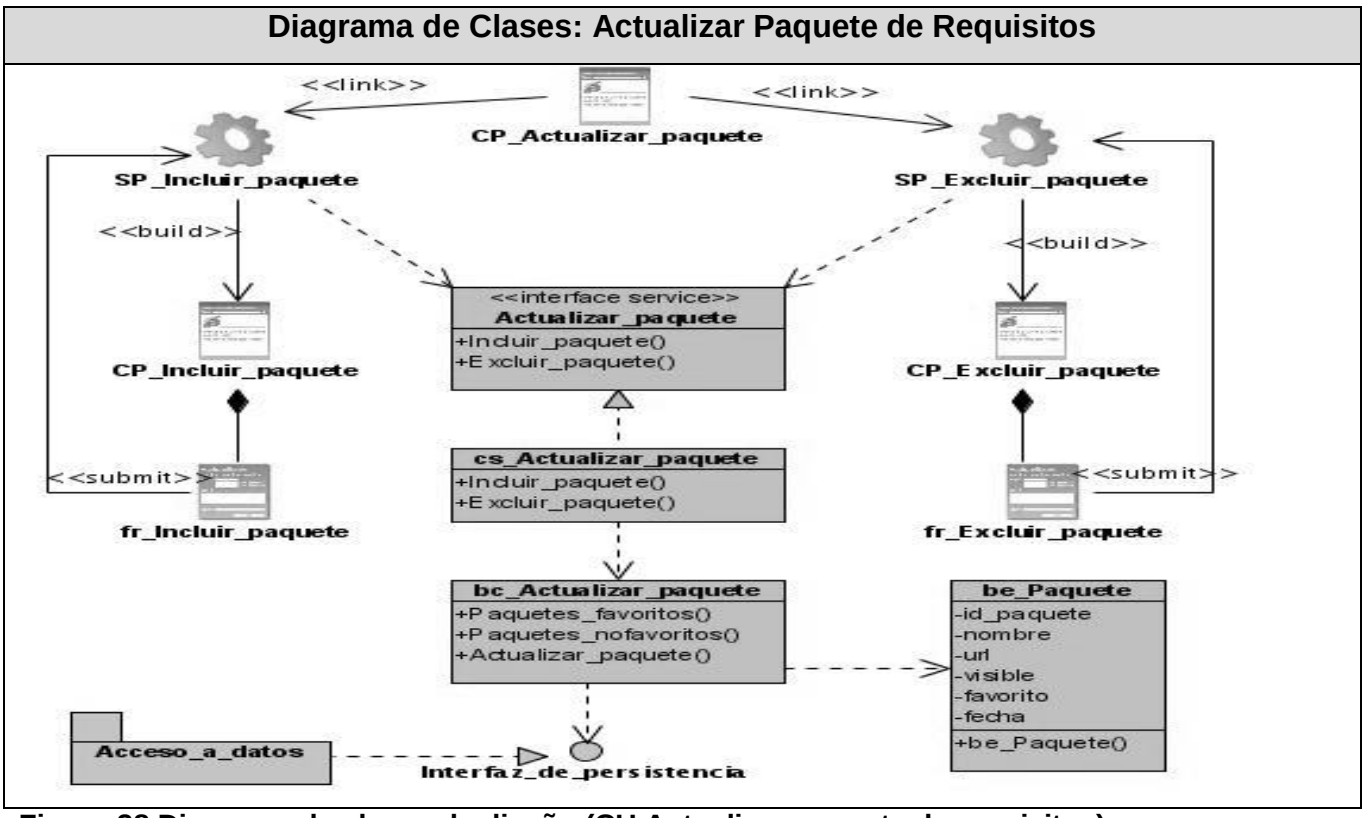


Figura 28 Diagrama de clases de diseño (CU Actualizar paquete de requisitos)

Diagrama de Clases: Comentar Paquete de Requisitos

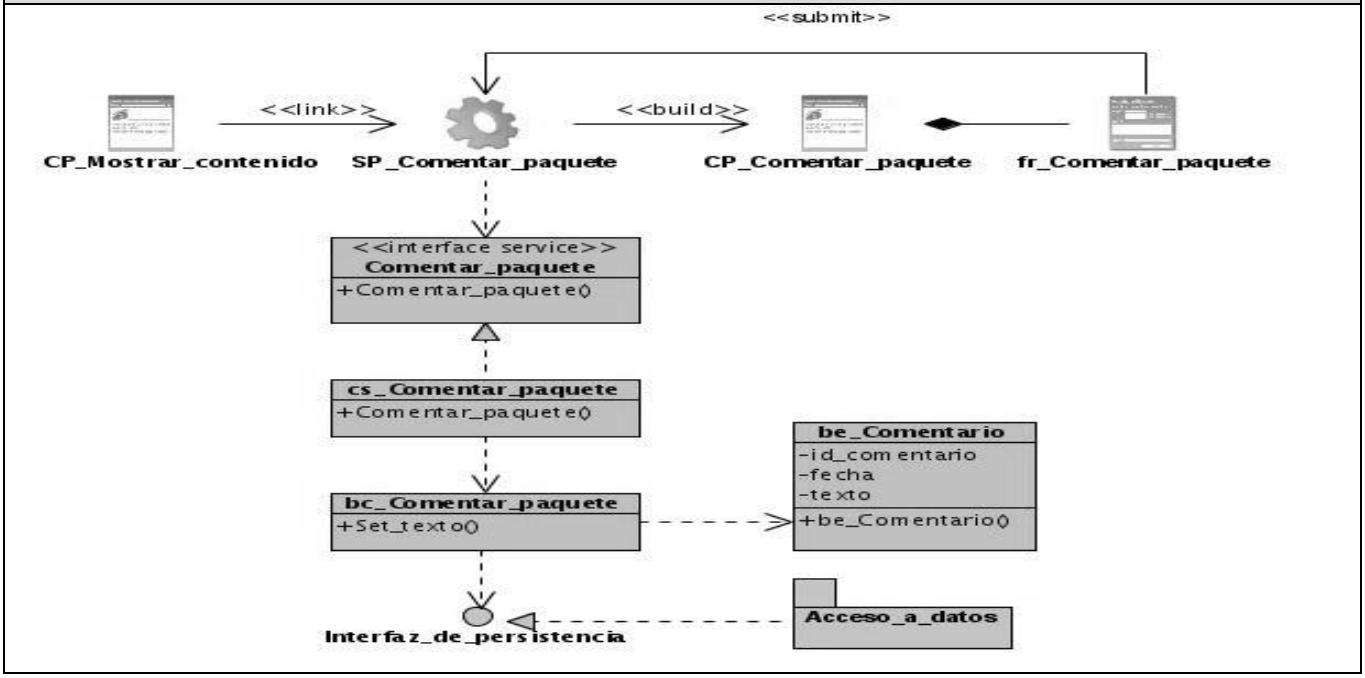


Figura 29 Diagrama de clases de diseño (CU Comentar paquete de requisitos)

Diagrama de Clases: Importar Paquete de Requisitos

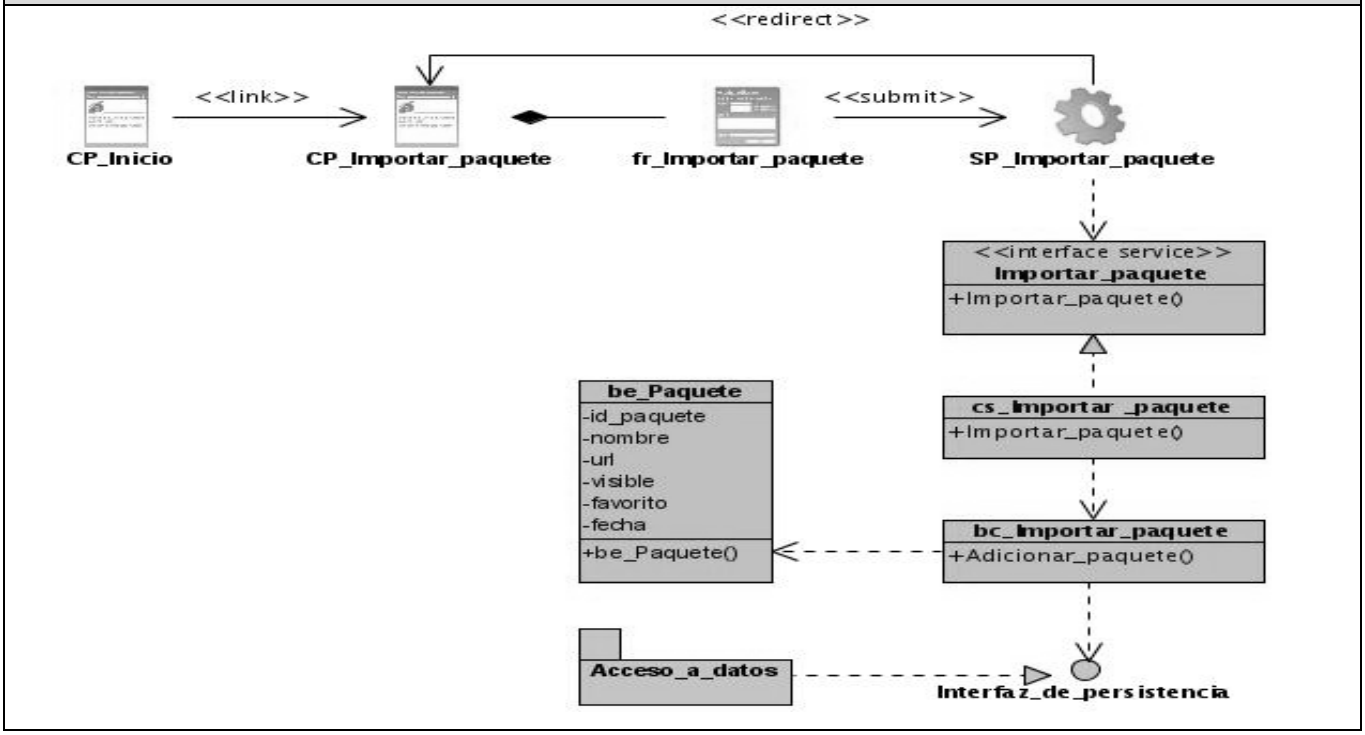


Figura 30 Diagrama de clases de diseño (CU Importar paquete de requisitos)

Diagrama de Clases: Gestionar Metadato

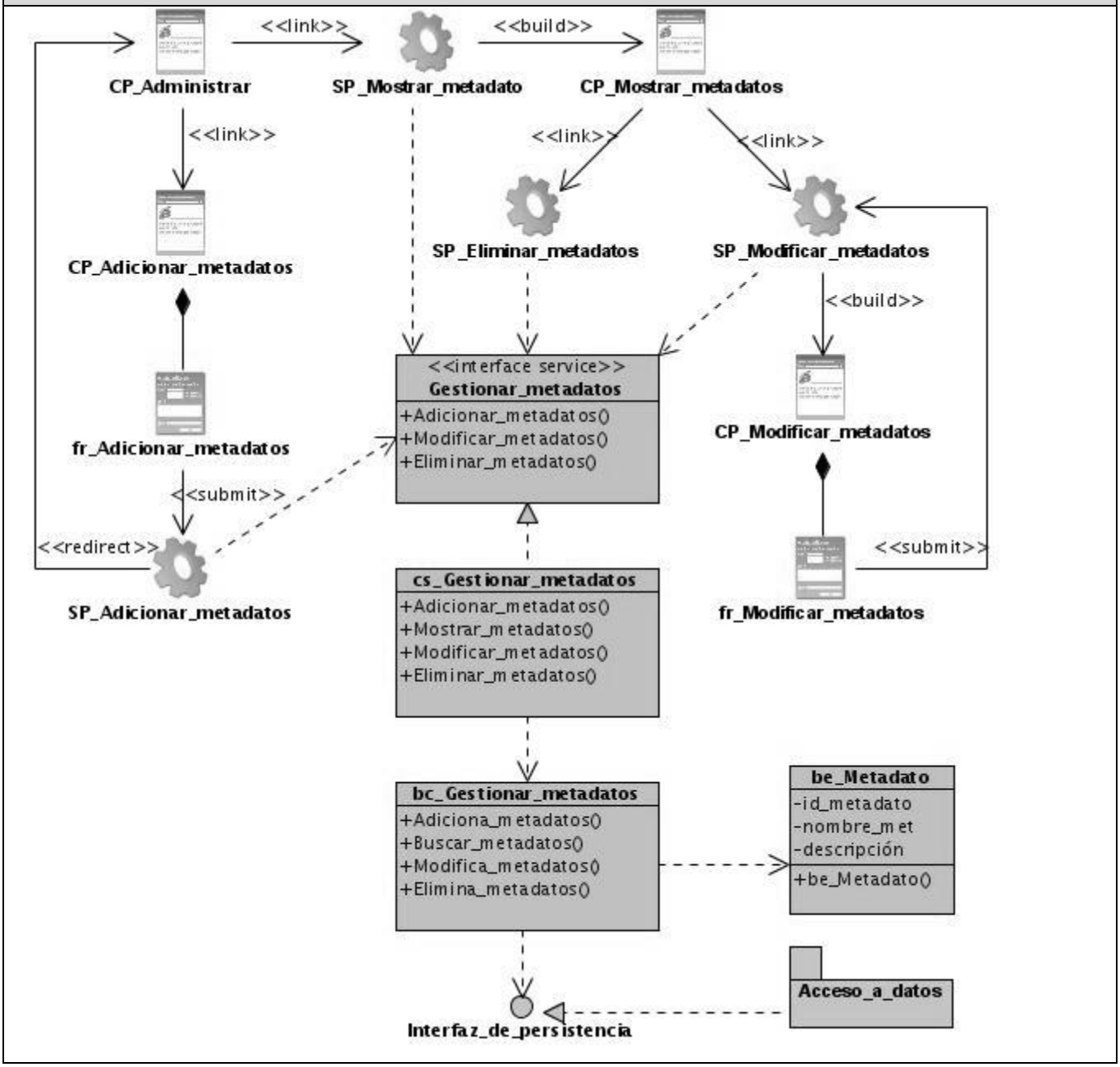


Figura 31 Diagrama de clases de diseño (CU Gestionar metadato)

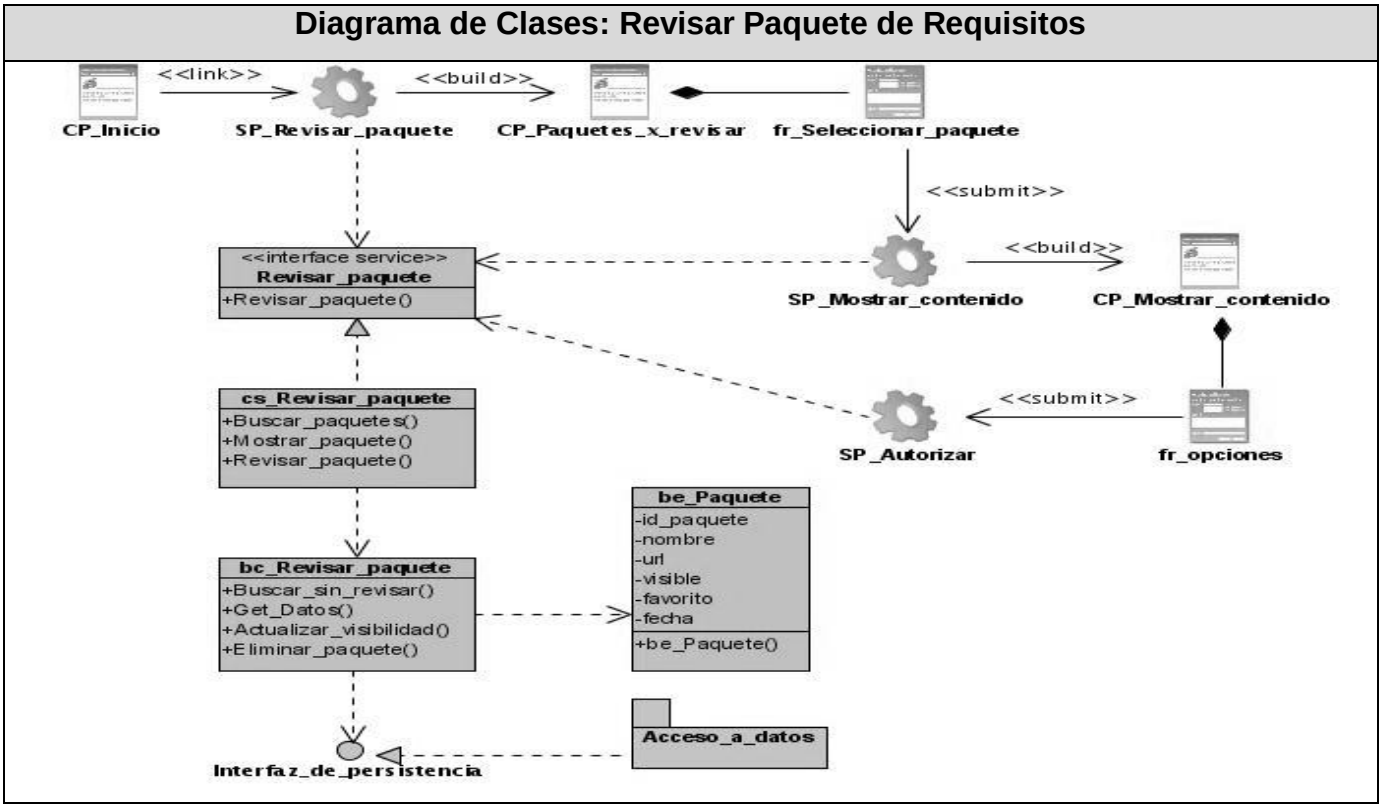


Figura 32 Diagrama de clases de diseño (CU Revisar paquete de requisitos)

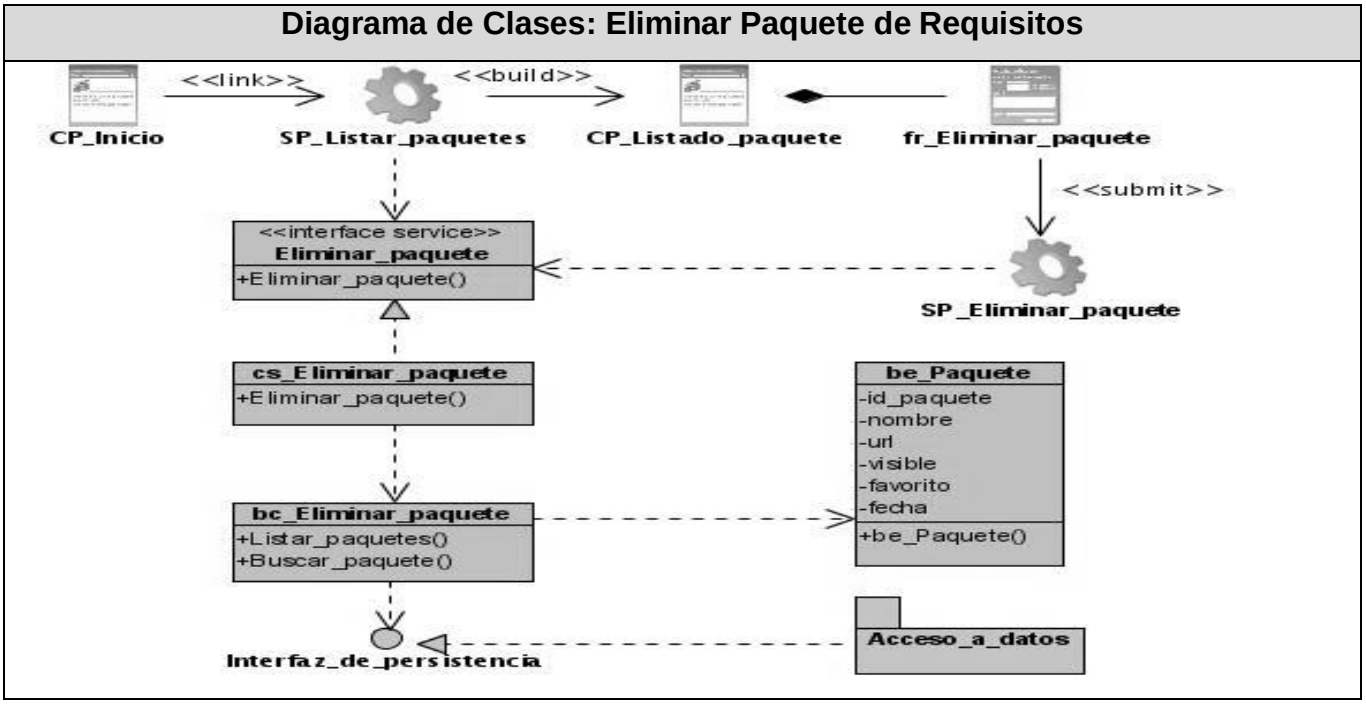


Figura 33 Diagrama de clases de diseño (CU Eliminar paquete de requisitos)

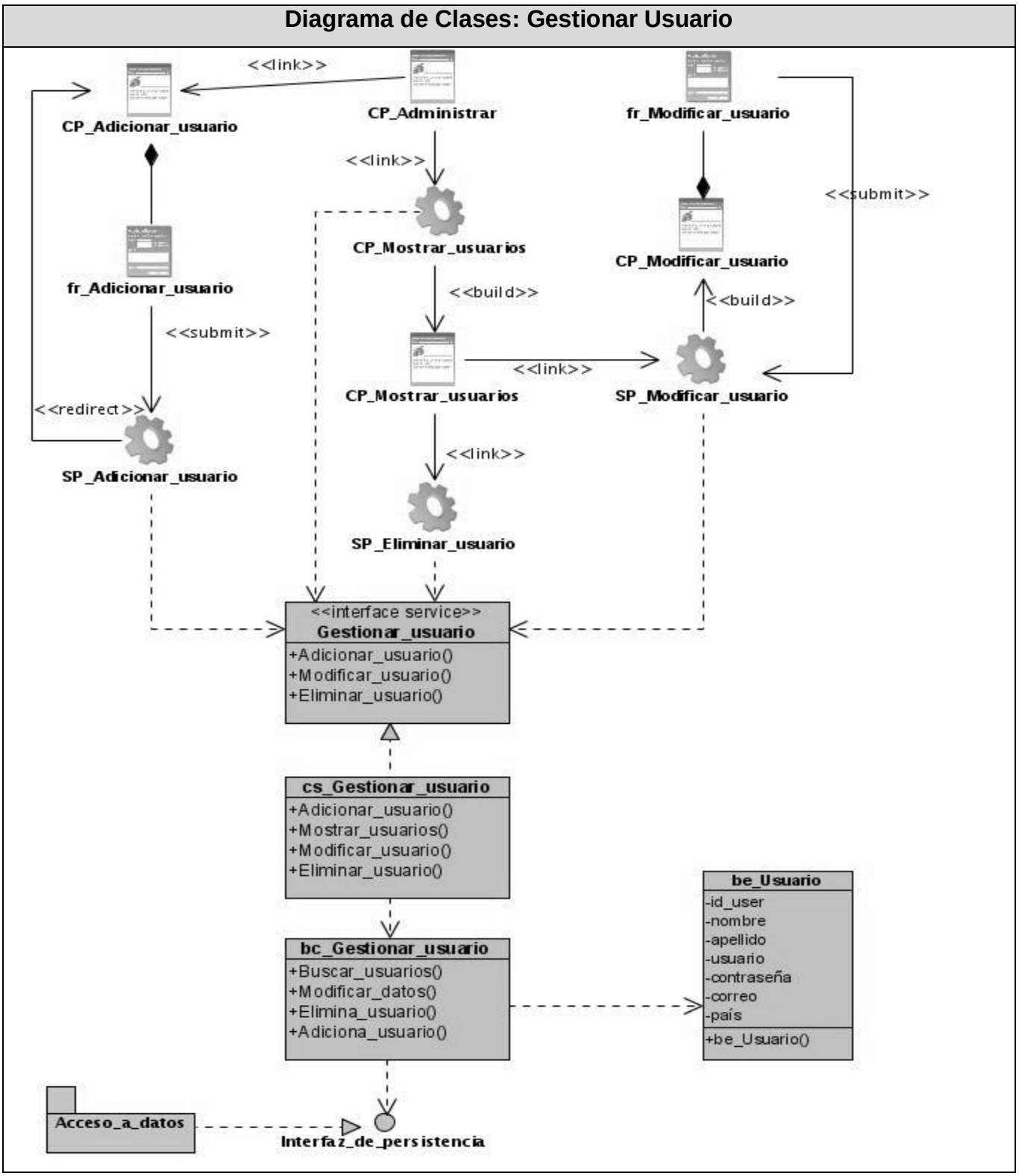


Figura 34 Diagrama de clases de diseño (CU Gestionar usuario)

Diagrama de Clases: Gestionar Categoría

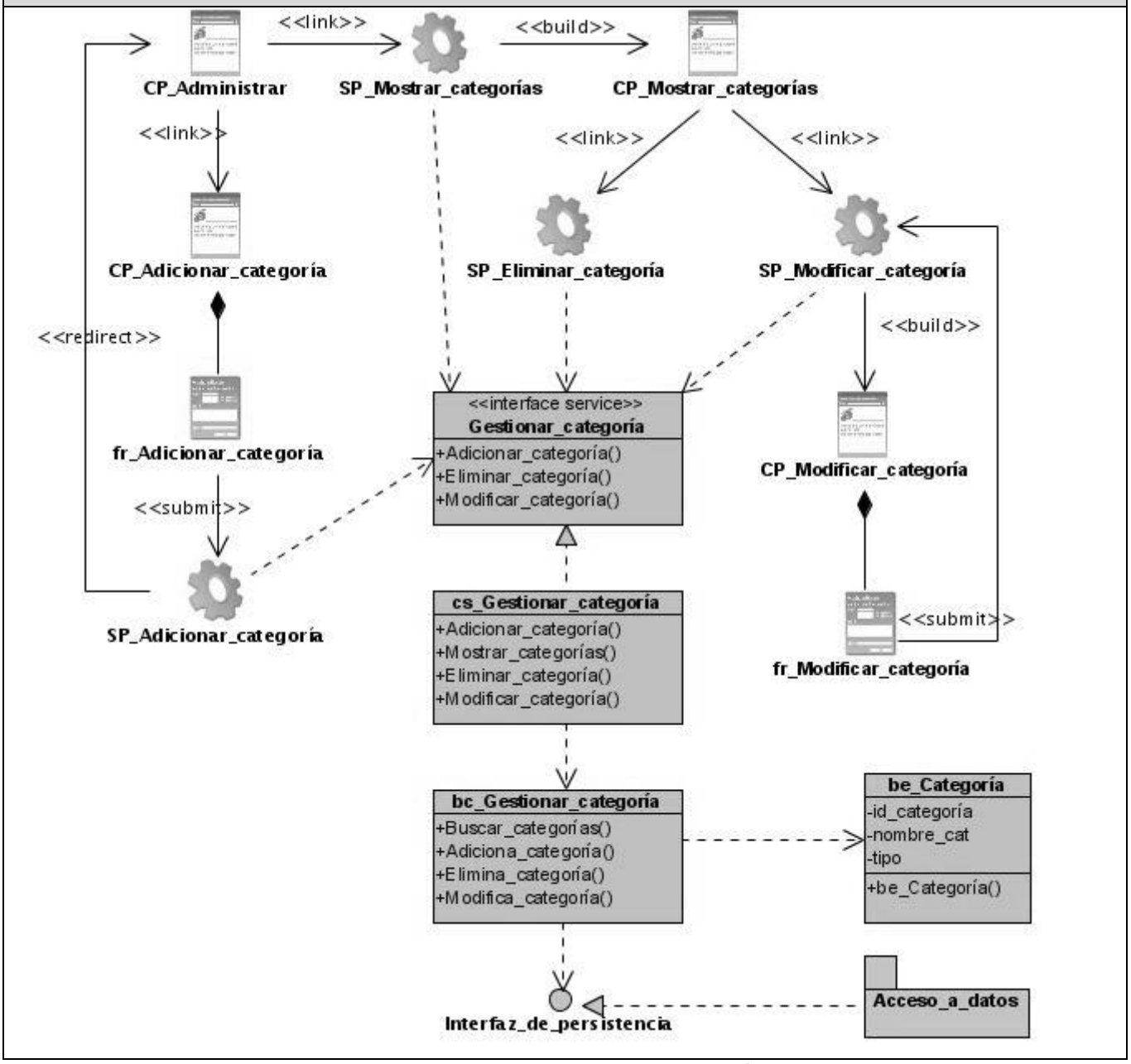


Figura 35 Diagrama de clases de diseño (CU Gestionar categoría)

Diagrama de Clases: Ver Reporte

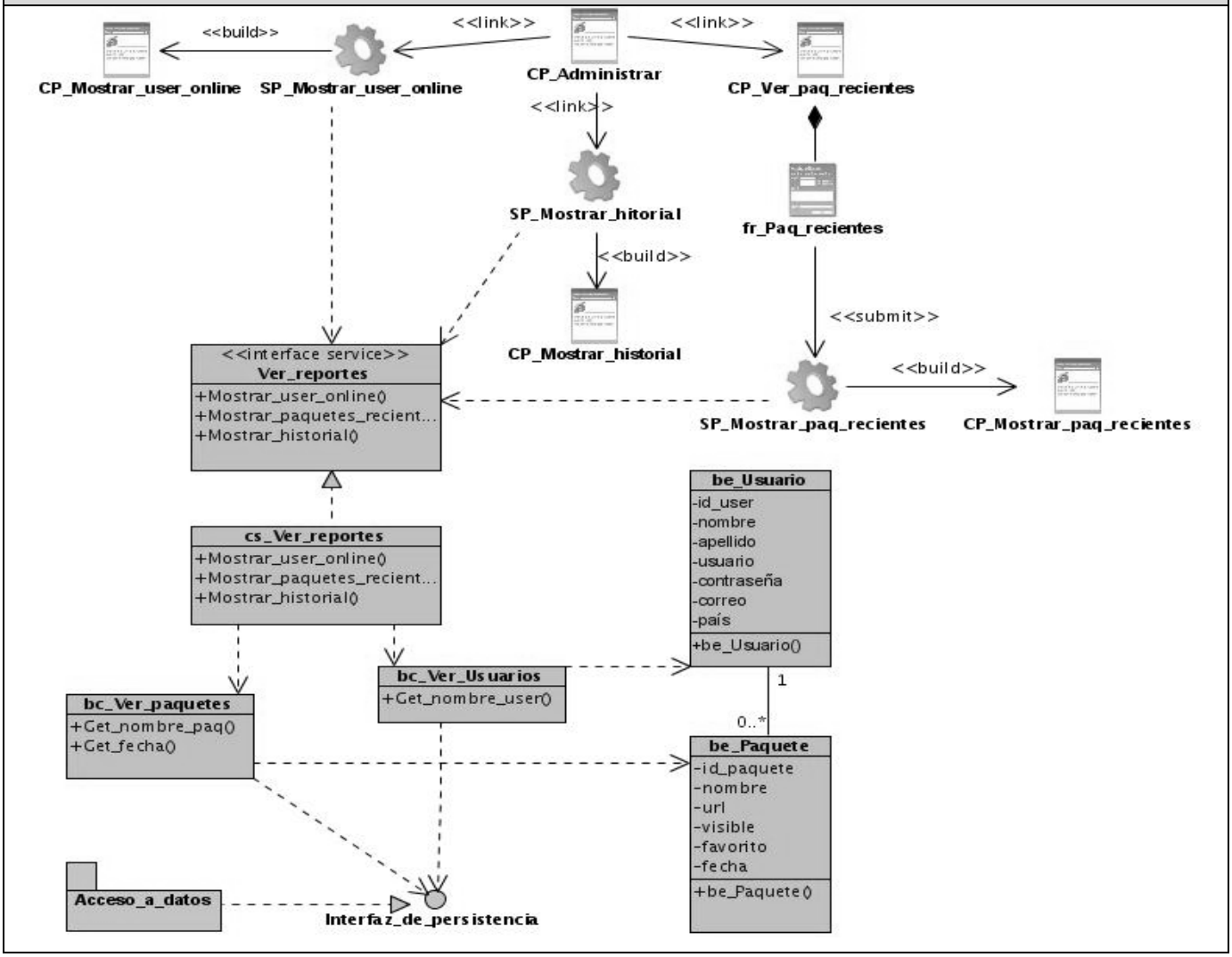


Figura 36 Diagrama de clases de diseño (CU Ver reporte)

Diagrama de Clases: Configurar HTTPs

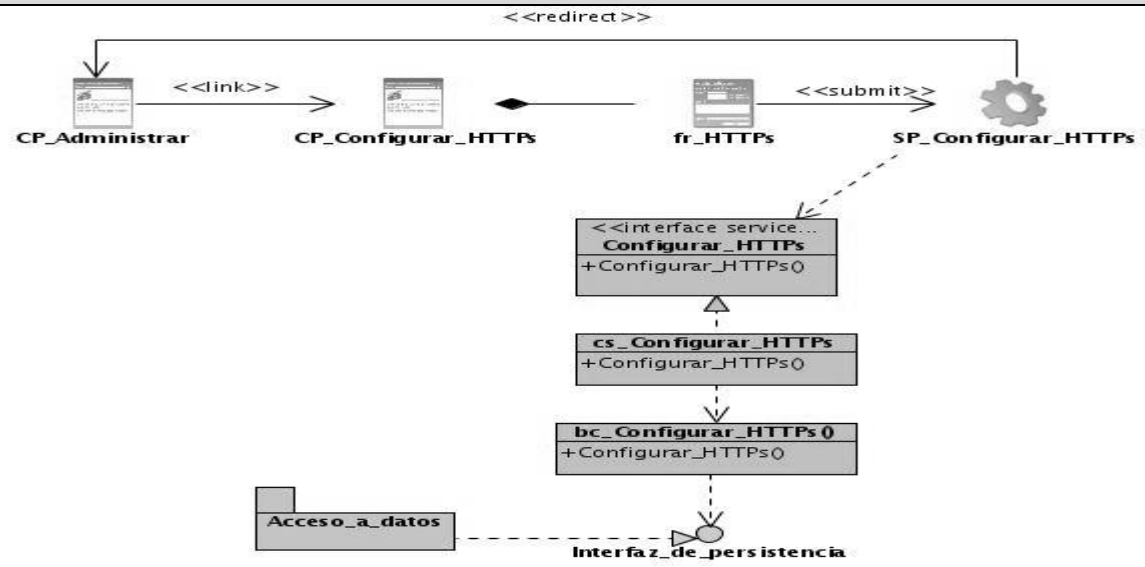


Figura 37 Diagrama de clases de diseño (CU Configurar HTTPs)

Diagrama de Clases: Administrar interfaz gráfica

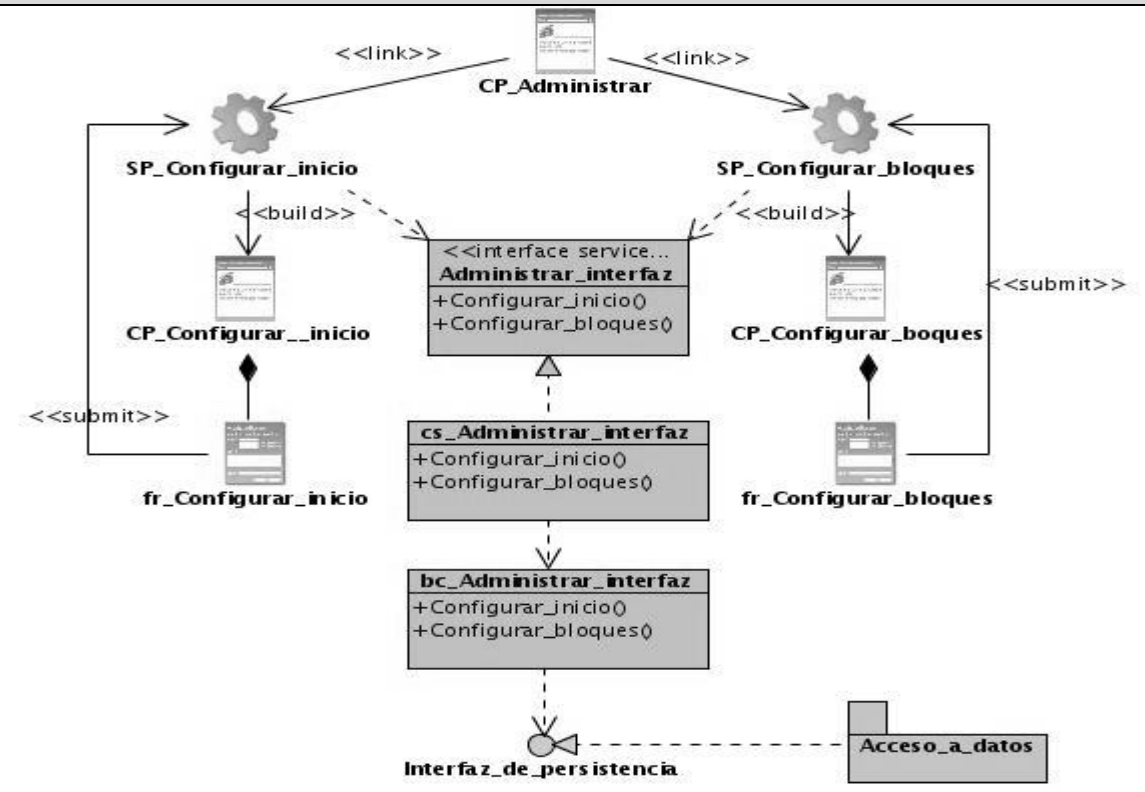


Figura 38 Diagrama de clases de diseño (CU Administrar interfaz gráfica)

3.2.3 Diagramas de Interacción de Diseño

El tipo de diagrama seleccionado para construir los diagramas de interacción de diseño fue el de secuencia. A continuación se muestran los diagramas de secuencias de los cuatro primeros casos de uso, los restantes se realizan siguiendo la misma línea de trabajo.

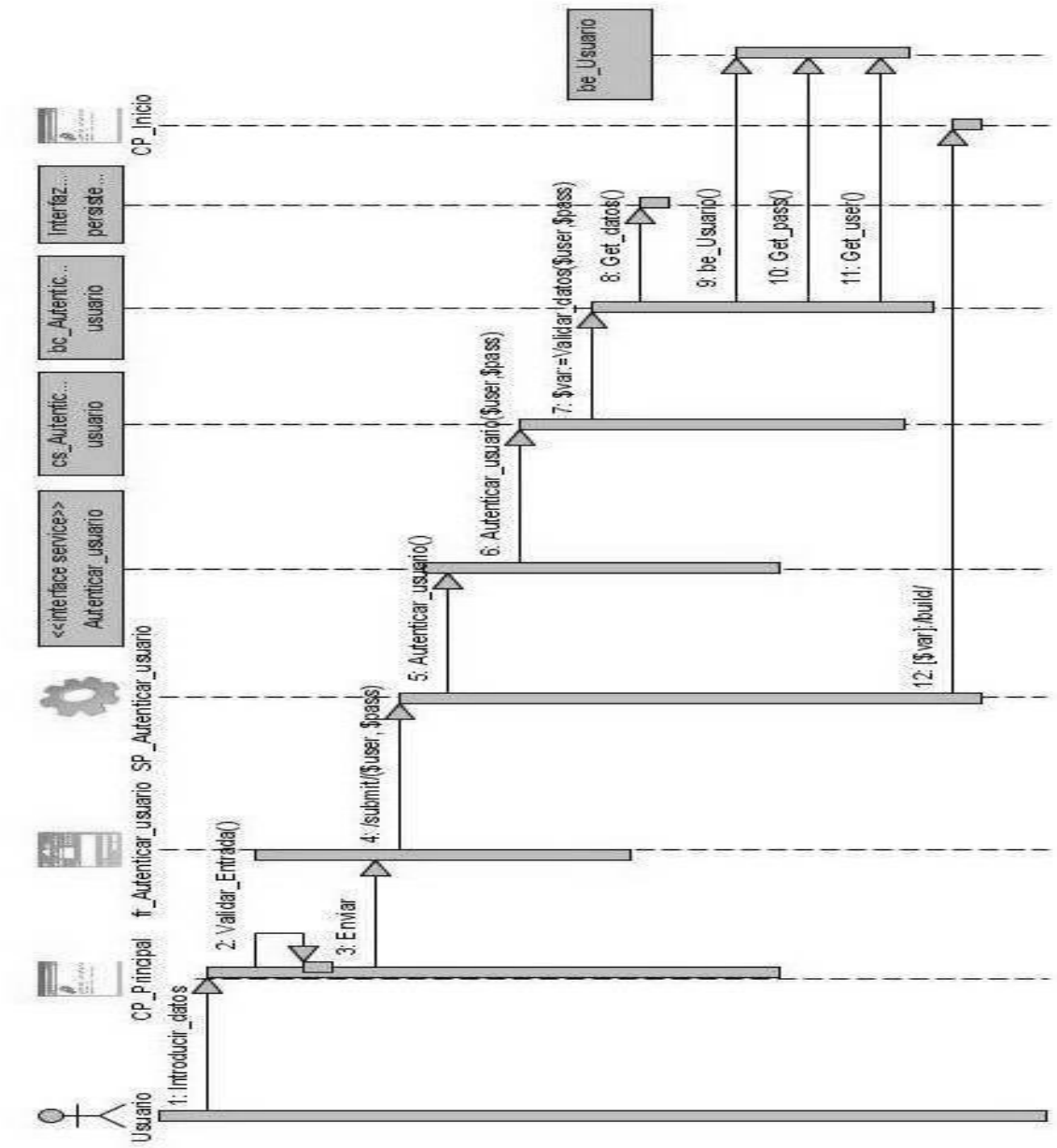


Figura 39 Diagrama de secuencia (Autenticar usuario)

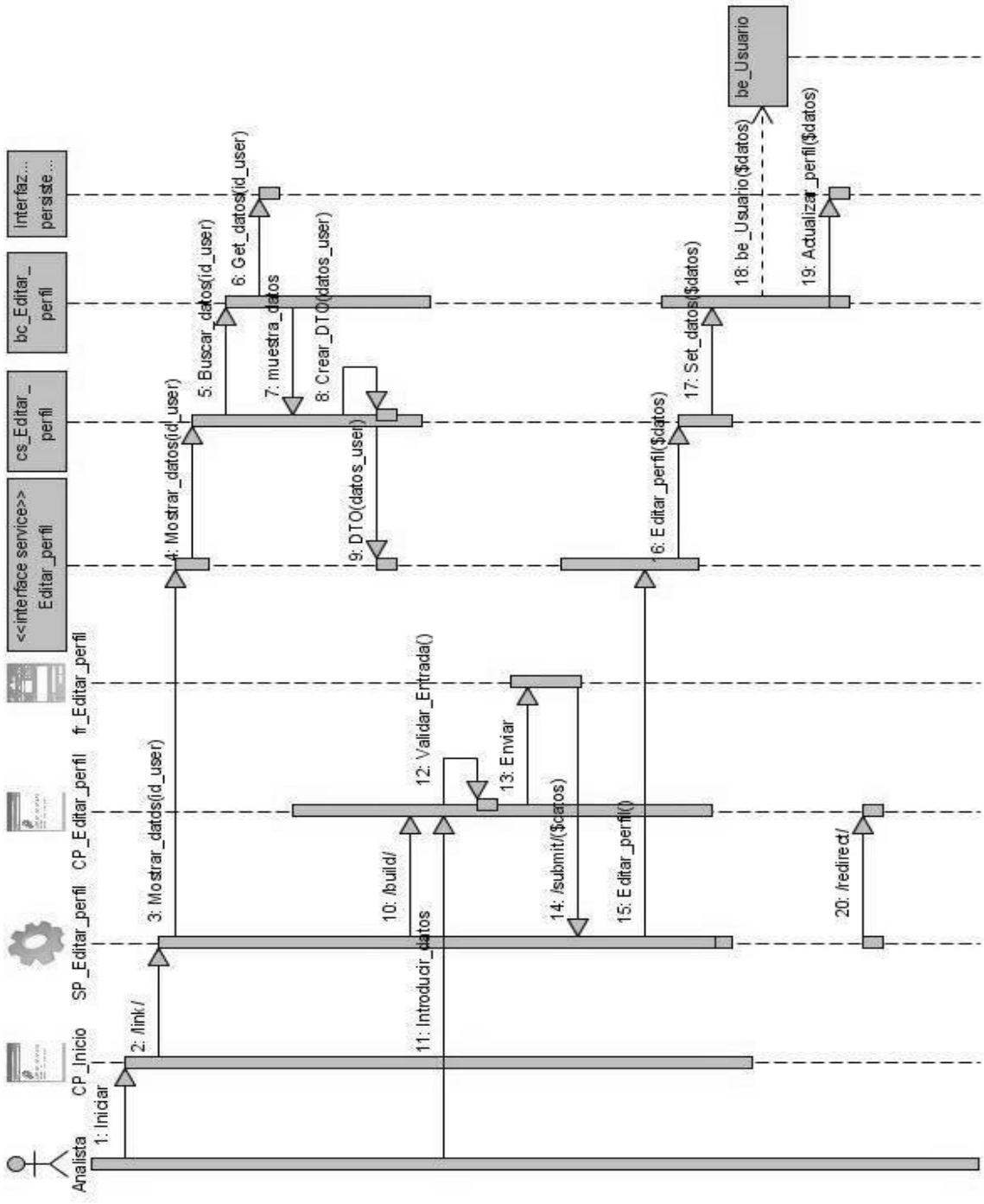


Figura 40 Diagrama de secuencia (Editar perfil)

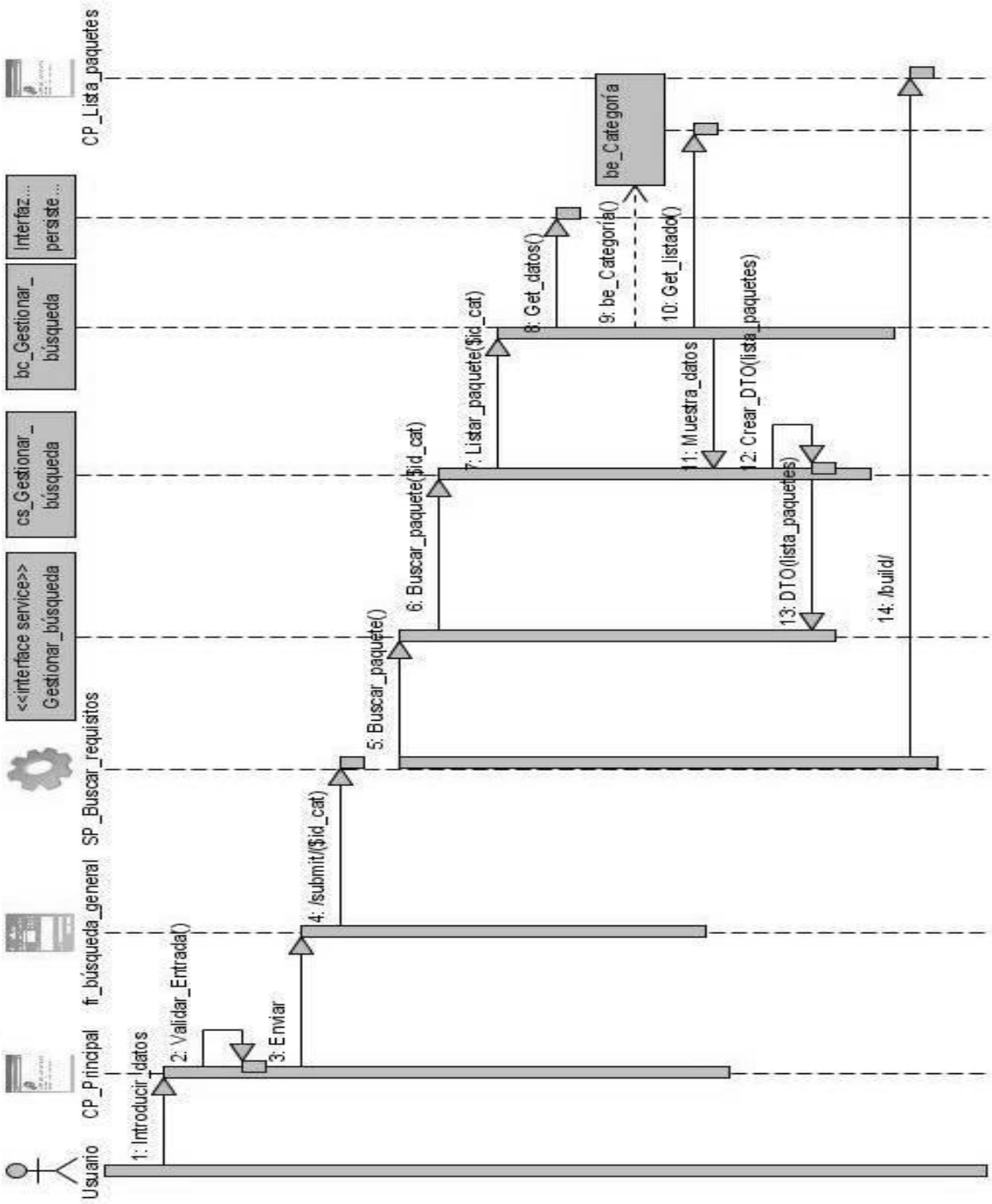


Figura 41 Diagrama de secuencia (Buscar requisitos)

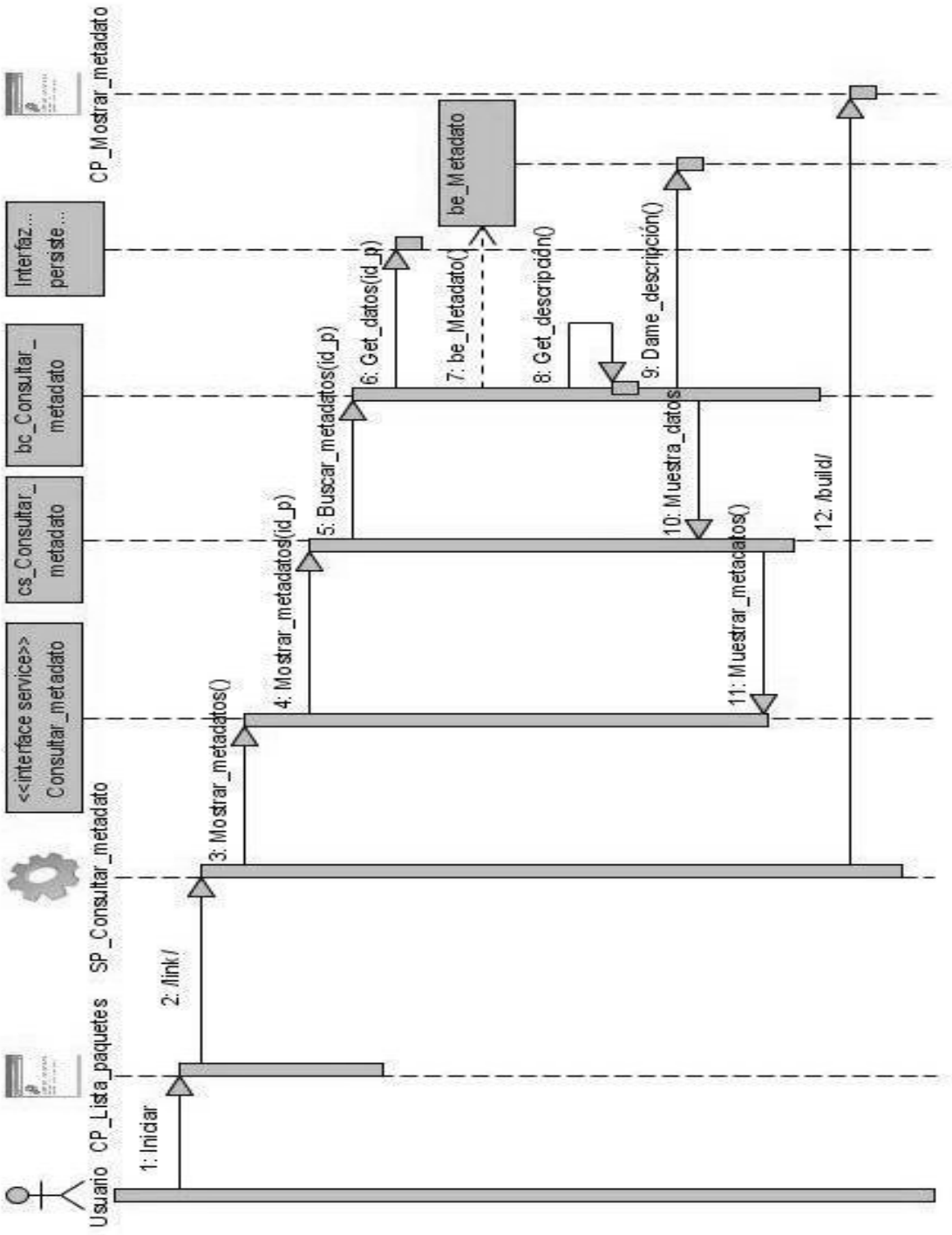


Figura 42 Diagrama de secuencia (Consultar metadatos)

3.2.4 Descripción de las clases

A continuación se describen las clases utilizadas en los diagramas de diseño por extensiones Web de aquellos casos de uso fundamentales con todos los atributos y métodos que estas contienen.

Nombre: cs_Autenticar_usuario	
Tipo de clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Autenticar_usuario(usuario, contraseña)
Descripción:	Permite obtener los datos del usuario y verificar si existe para así asignarle los permisos pertinentes.

Tabla 19 Descripción de las clases de diseño: cs_Autenticar_usuario

Nombre: cs_Editar_perfil	
Tipo de clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Mostrar_datos(id_usuario)
Descripción:	Permite mostrar todos los datos del usuario
Nombre:	Editar_perfil(nombre, apellido, usuario, contraseña, correo, país)
Descripción:	Permite editar el perfil del usuario.

Tabla 20 Descripción de las clases de diseño: cs_Editar_perfil

Nombre: cs_Gestionar_búsqueda	
Tipo de clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Buscar_paquete(id_cat)
Descripción:	Permite listar todos los paquetes que pertenezcan a la categoría seleccionada.

Tabla 21 Descripción de las clases de diseño: cs_Gestionar_búsqueda

Nombre: cs_Consultar_metadato	
Tipo de clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Mostrar_metadatos(id_paquete)
Descripción:	Permite mostrar el contenido de los metadatos.

Tabla 22 Descripción de las clases de diseño: cs_Consultar_metadato

Nombre: cs_Consultar_paquete	
Tipo de clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Mostrar_paquete(id_paquete)
Descripción:	Permite mostrar los requisitos existentes en el paquete.
Nombre:	Listar_requisitos(id_paquete)
Descripción:	Lista todos los requisitos del paquete.

Tabla 23 Descripción de las clases de diseño: cs_Consultar_paquete

Nombre: cs_Exportar_paquete	
Tipo de clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Exportar_paquete(id_paquete)
Descripción:	Permite exportar un paquete de requisitos.

Tabla 24 Descripción de las clases de diseño: cs_Exportar_paquete

Nombre: cs_Actualizar_paquete	
Tipo de clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Incluir_paquete(id_paquete)
Descripción:	Permite incluir un paquete de requisitos a su área de trabajo.
Nombre:	Excluir_paquete(id_paquete)
Descripción:	Permite excluir un paquete de requisitos de su área de trabajo.

Tabla 25 Descripción de las clases de diseño: cs_Actualizar_paquete

Nombre: cs_Comentar_paquete	
Tipo de clase: Controladora	
Por cada responsabilidad:	
Nombre:	Comentar_paquete(id_paquete)
Descripción:	Permite realizar un comentario sobre determinado paquete de requisitos.

Tabla 26 Descripción de las clases de diseño: cs_Comentar_paquete

Nombre: cs_Importar_paquete	
Tipo de clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Importar(id_paquete, nombre, url, visible, favorito)
Descripción:	Permite importar un paquete de requisitos al servidor y actualizar la base de datos.

Tabla 27 Descripción de las clases de diseño: cs_Importar_paquete

Nombre: cs_Eliminar_paquete	
Tipo de clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Listar_paquetes()
Descripción:	Permite listar todos los paquetes de requisitos que hallan sido denegados.
Nombre:	Eliminar_paquete(id_paquete)
Descripción:	Permite eliminar un paquete de requisitos.

Tabla 28 Descripción de las clases de diseño: cs_Eliminar_paquete

Nombre: cs_Revisar_paquete	
Tipo de clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Buscar_paquetes(visible)
Descripción:	Permite buscar todos los paquetes de requisitos que aún no se han revisado.
Nombre:	Mostrar_paquete(id_paquete)
Descripción:	Permite mostrar el contenido del paquete de requisitos.
Nombre:	Revisar_paquete()
Descripción:	Permite que un paquete de requisitos pueda ser aceptado o eliminado del sistema.

Tabla 29 Descripción de las clases de diseño: cs_Revisar_paquete

Nombre: cs_Gestionar_usuario	
Tipo de clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Adicionar_usuario(id_usuario, nombre, apellido, usuario, contraseña, correo, país)
Descripción:	Permite adicionar un nuevo usuario al sistema.
Nombre:	Mostrar_usuario()
Descripción:	Permite listar todos los usuarios del sistema.
Nombre:	Modificar_usuario(nombre, apellido, usuario, contraseña, correo, país)
Descripción:	Permite modificar los datos de un usuario.
Nombre:	Eliminar_usuario(id_usuario)
Descripción:	Permite eliminar un usuario y actualizar la base de datos.

Tabla 30 Descripción de las clases de diseño: cs_Gestionar_usuario

Nombre: cs_Configurar_HTTPs	
Tipo de clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Configurar_HTTPs()
Descripción:	Permite configurar el protocolo seguro.

Tabla 31 Descripción de las clases de diseño: cs_Configurar_HTTPs

Nombre: cs_Gestionar_categoría	
Tipo de clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Adicionar_categoría(id_ categoría, nombre, tipo)
Descripción:	Permite adicionar una nueva categoría al sistema.
Nombre:	Mostrar_ categoría()
Descripción:	Permite listar todas las categorías del sistema.
Nombre:	Modificar_ categoría(nombre, tipo)
Descripción:	Permite modificar los datos correspondientes a una categoría.
Nombre:	Eliminar_ categoría(id_ categoría)
Descripción:	Permite eliminar una categoría y actualizar la base de datos.

Tabla 32 Descripción de las clases de diseño: cs_Gestionar_categoría

Nombre: cs_Administrar_interfaz_gráfica	
Tipo de Clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Configurar_inicio()
Descripción:	Permite configurar la página principal de la herramienta, poniéndole imagen y texto.
Nombre:	Configurar_bloques()
Descripción:	Permite configurar los bloques del sistema según desee.

Tabla 33 Descripción de las clases de diseño: cs_Configurar_HTTPs

Nombre: cs_Ver_reportes	
Tipo de clase: Controladora	
Por cada responsabilidad:	
Nombre:	Mostrar_usuarios_online()
Descripción:	Permite mostrar la lista de usuarios online del sistema.
Nombre:	Mostrar_paquetes_recientes()
Descripción:	Permite mostrar una lista con los últimos paquetes de requisitos insertados.
Nombre:	Mostrar_historial()
Descripción:	Permite mostrar una lista con los paquetes de requisitos y todos sus datos.

Tabla 34 Descripción de las clases de diseño: cs_Ver_reportes

Nombre: cs_Gestionar_metadatos	
Tipo de clase: Controladora	
Por cada responsabilidad:	
Nombre:	Adicionar_metadato(id_ metadato, nombre, descripción)
Descripción:	Permite adicionar un metadato a un paquete de requisitos.
Nombre:	Mostrar_ metadato()
Descripción:	Permite listar todos los metadatos del sistema.
Nombre:	Modificar_ metadato (nombre, descripción)
Descripción:	Permite modificar un determinado metadato.
Nombre:	Eliminar_ metadato(id_ metadato)
Descripción:	Permite eliminar un metadato y actualizar la base de datos.

Tabla 35 Descripción de las clases de diseño: cs_Gestionar_metadatos

Nombre: bc_Autenticar_usuario	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Validar_datos()
Descripción:	Permite validar los datos del usuario.

Tabla 36 Descripción de las clases de diseño: bc_Autenticar_usuario

Nombre: bc_Editar_perfil	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Set_datos()
Descripción:	Permite modificar los datos del usuario.

Tabla 37 Descripción de las clases de diseño: bc_Editar_perfil

Nombre: bc_Gestionar_búsqueda	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Listar_paquete()
Descripción:	Permite obtener la lista de paquetes según la categoría entrada previamente.

Tabla 38 Descripción de las clases de diseño: bc_Gestionar_búsqueda

Nombre: bc_Consultar_metadatos	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Buscar_metadatos()
Descripción:	Permite buscar el metadato seleccionado.
Nombre:	Mostrar_descripción()
Descripción:	Permite mostrar el metadato seleccionado.

Tabla 39 Descripción de las clases de diseño: bc_Consultar_metadato

Nombre: bc_Consultar_paquete	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Buscar_paquete()
Descripción:	Permite buscar el paquete seleccionado.
Nombre:	Listar_requisitos()
Descripción:	Permite listar el contenido del paquete seleccionado.

Tabla 40 Descripción de las clases de diseño: bc_Consultar_paquete

Nombre: bc_Exportar_paquete	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Obtener_paquete()
Descripción:	Permite obtener el paquete de requisitos seleccionado con todos sus datos.

Tabla 41 Descripción de las clases de diseño: bc_Exportar_paquete

Nombre: bc_Actualizar_paquete	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Paquetes_favoritos()
Descripción:	Permite listar los paquetes que tiene el usuario en su área de trabajo.
Nombre:	Paquetes_nofavoritos()
Descripción:	Permite listar los paquetes que no tiene el usuario en su área de trabajo.
Nombre:	Actualizar_paquete()
Descripción:	Permite incluir o excluir el paquete del área de trabajo del usuario.

Tabla 42 Descripción de las clases de diseño: bc_Actualizar_paquete

Nombre: bc_Comentar_paquete	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Set_texto()
Descripción:	Permite adicionar un comentario a un paquete.

Tabla 43 Descripción de las clases de diseño: bc_Comentar_paquete

Nombre: bc_Importar_paquete	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Adicionar_paquete()
Descripción:	Permite adicionar un paquete al sistema.

Tabla 44 Descripción de las clases de diseño: bc_Importar_paquete

Nombre: bc_Eliminar_paquete	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Listar_paquete()
Descripción:	Permite listar todos los paquetes del sistema.
Nombre:	Buscar_paquete()
Descripción:	Permite buscar el paquete seleccionado.

Tabla 45 Descripción de las clases de diseño: bc_Eliminar_paquete

Nombre: bc_Revisar_paquete	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Buscar_sin_revisar()
Descripción:	Permite buscar todos los paquetes sin revisar.
Nombre:	Get_datos()
Descripción:	Permite obtener todos los datos del paquete seleccionado.
Nombre:	Actualizar_visibilidad()
Descripción:	Permite poner el paquete visible en el sistema.
Nombre:	Eliminar_paquete()
Descripción:	Permite eliminar los paquetes que sean denegados por el revisor.

Tabla 46 Descripción de las clases de diseño: bc_Revisar_paquete

Nombre: bc_Gestionar_usuario	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Adiciona_usuario()
Descripción:	Permite adicionar un usuario al sistema.
Nombre:	Buscar_usuarios()
Descripción:	Permite buscar todos los usuarios del sistema.
Nombre:	Modificar_datos()
Descripción:	Permite modificar los datos del usuario seleccionado.
Nombre:	Elimina_usuario()
Descripción:	Permite eliminar el usuario seleccionado.

Tabla 47 Descripción de las clases de diseño: bc_Gestionar_usuario

Nombre: bc_Gestionar_categoría	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Adiciona_categoría()
Descripción:	Permite adicionar una categoría al sistema.
Nombre:	Buscar_categorías()
Descripción:	Permite buscar todas las categorías del sistema.
Nombre:	Modifica_categoría()
Descripción:	Permite modificar los datos de la categoría seleccionada.
Nombre:	Elimina_categoría()
Descripción:	Permite eliminar una categoría seleccionado.

Tabla 48 Descripción de las clases de diseño: bc_Gestionar_categoría

Nombre: bc_Gestionar_metadato	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Adiciona_metadatos()
Descripción:	Permite adicionar un metadato al sistema.
Nombre:	Buscar_metadatos()
Descripción:	Permite buscar todos los metadatos del sistema.
Nombre:	Modifica_metadatos()
Descripción:	Permite modificar el metadato seleccionado.
Nombre:	Elimina_metadatos()
Descripción:	Permite eliminar el metadato seleccionado.

Tabla 49 Descripción de las clases de diseño: bc_Gestionar_metadato

Nombre: bc_Ver_paquetes	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Get_nombre_paq()
Descripción:	Permite obtener el nombre del paquete.
Nombre:	Get_fecha()
Descripción:	Permite obtener la fecha de importación del paquete.

Tabla 50 Descripción de las clases de diseño: bc_Ver_paquetes

Nombre: bc_Ver_Usuarios	
Tipo de Clase: Componente del Negocio o clase controladora (capa de dominio)	
Por cada responsabilidad:	
Nombre:	Get_nombre_user()
Descripción:	Permite obtener el nombre del usuario.

Tabla 51 Descripción de las clases de diseño: bc_Ver_Usuarios

Nombre: bc_Configurar_HTTPs	
Tipo de clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Configurar_HTTPs()
Descripción:	Permite configurar el protocolo seguro.

Tabla 52 Descripción de las clases de diseño: bc_Configurar_HTTPs

Nombre: bc_Administrar_interfaz_gráfica	
Tipo de Clase: Clase de servicio o clase controladora (capa de servicio)	
Por cada responsabilidad:	
Nombre:	Configurar_inicio()
Descripción:	Permite configurar la página principal de la herramienta, poniéndole imagen y texto.
Nombre:	Configurar_bloques()
Descripción:	Permite configurar los bloques del sistema según desee.

Tabla 53 Descripción de las clases de diseño: bc_Administrar_interfaz_gráfica

Nombre: be_Usuario	
Tipo de Clase: Entidad del negocio o clase entidad (capa de dominio)	
Atributo:	**Tipo**
id_user	int
nombre	string
apellido	String
usuario	string
contraseña	string
correo	string
país	string
Por cada responsabilidad:	
Nombre:	be_Usuario()
Descripción:	Este es el constructor de un usuario.
Nombre:	Get_id_user()
Descripción:	Permite obtener el identificador del usuario.
Nombre:	Get_user()
Descripción:	Permite obtener usuario.
Nombre:	Get_pass()
Descripción:	Permite obtener la contraseña del usuario.
Nombre:	Get_nombre()
Descripción:	Permite obtener el nombre del usuario.
Nombre:	Get_apellido()
Descripción:	Permite obtener el apellido del usuario.
Nombre:	Get_correo()
Descripción:	Permite obtener el correo del usuario.
Nombre:	Get_país()
Descripción:	Permite obtener el país del usuario.
Nombre:	Actualizar()
Descripción:	Permite actualizar los datos del usuario.

Tabla 54 Descripción de las clases de diseño: be_Usuario

Nombre: be_Categoría	
Tipo de Clase: Entidad del negocio o clase entidad (capa de dominio)	
Atributo:	**Tipo**
id_categoría	int
nombre_cat	string
tipo	String
Por cada responsabilidad:	
Nombre:	be_Categoría()
Descripción:	Este es el constructor de una categoría.
Nombre:	Get_id_cat()
Descripción:	Permite obtener el identificador de la categoría.
Nombre:	Get_nomb_cat()
Descripción:	Permite obtener el nombre de la categoría.
Nombre:	Get_tipo()
Descripción:	Permite obtener el tipo de categoría.

Tabla 55 Descripción de las clases de diseño: be_Categoría

Nombre: be_Comentario	
Tipo de Clase: Entidad del negocio o clase entidad (capa de dominio)	
Atributo:	**Tipo**
id_comentario	int
fecha	date
texto	String
Por cada responsabilidad:	
Nombre:	be_Comentario()
Descripción:	Este es el constructor de un comentario.

Tabla 56 Descripción de las clases de diseño: be_Comentario

Nombre: be_Paquete	
Tipo de Clase: Entidad del negocio o clase entidad (capa de dominio)	
Atributo:	**Tipo**
id_paquete	int
nombre_paq	string
fecha	date
url	string
visible	bool
favorito	bool
Por cada responsabilidad:	
Nombre:	be_Paquete()
Descripción:	Este es el constructor de un paquete.
Nombre:	Get_id_paq()
Descripción:	Permite obtener el identificador del paquete.
Nombre:	Get_nom_paq()
Descripción:	Permite obtener el nombre del paquete.
Nombre:	Get_fecha()
Descripción:	Permite obtener la fecha en que se importa el paquete.
Nombre:	Get_url()
Descripción:	Permite obtener la dirección física de los requisitos.
Nombre:	Get_visible()
Descripción:	Permite comprobar si el paquete está visible o no.
Nombre:	Get_favorito()
Descripción:	Permite comprobar si el paquete es favorito o no.
Nombre:	Set_visible()
Descripción:	Permite modificar el estado de ese atributo
Nombre:	Set_favorito()
Descripción:	Permite modificar el estado de ese atributo

Tabla 57 Descripción de las clases de diseño: be_Paquete

Nombre: be_Metadato	
Tipo de Clase: Entidad del negocio o clase entidad (capa de dominio)	
Atributo:	**Tipo**
id_metadato	int
nombre_met	string
descripción	String
Por cada responsabilidad:	
Nombre:	be_Metadato()
Descripción:	Este es el constructor de un metadato.
Nombre:	Get_id_met()
Descripción:	Permite obtener el identificador del metadato.
Nombre:	Get_nomb_met()
Descripción:	Permite obtener el nombre del metadato.
Nombre:	Get_descripción()
Descripción:	Permite obtener la descripción del metadato.

Tabla 58 Descripción de las clases de diseño: be_Metadato

3.2.5 Diagrama Entidad Relación de la Base de Datos

Diagrama Entidad Relación

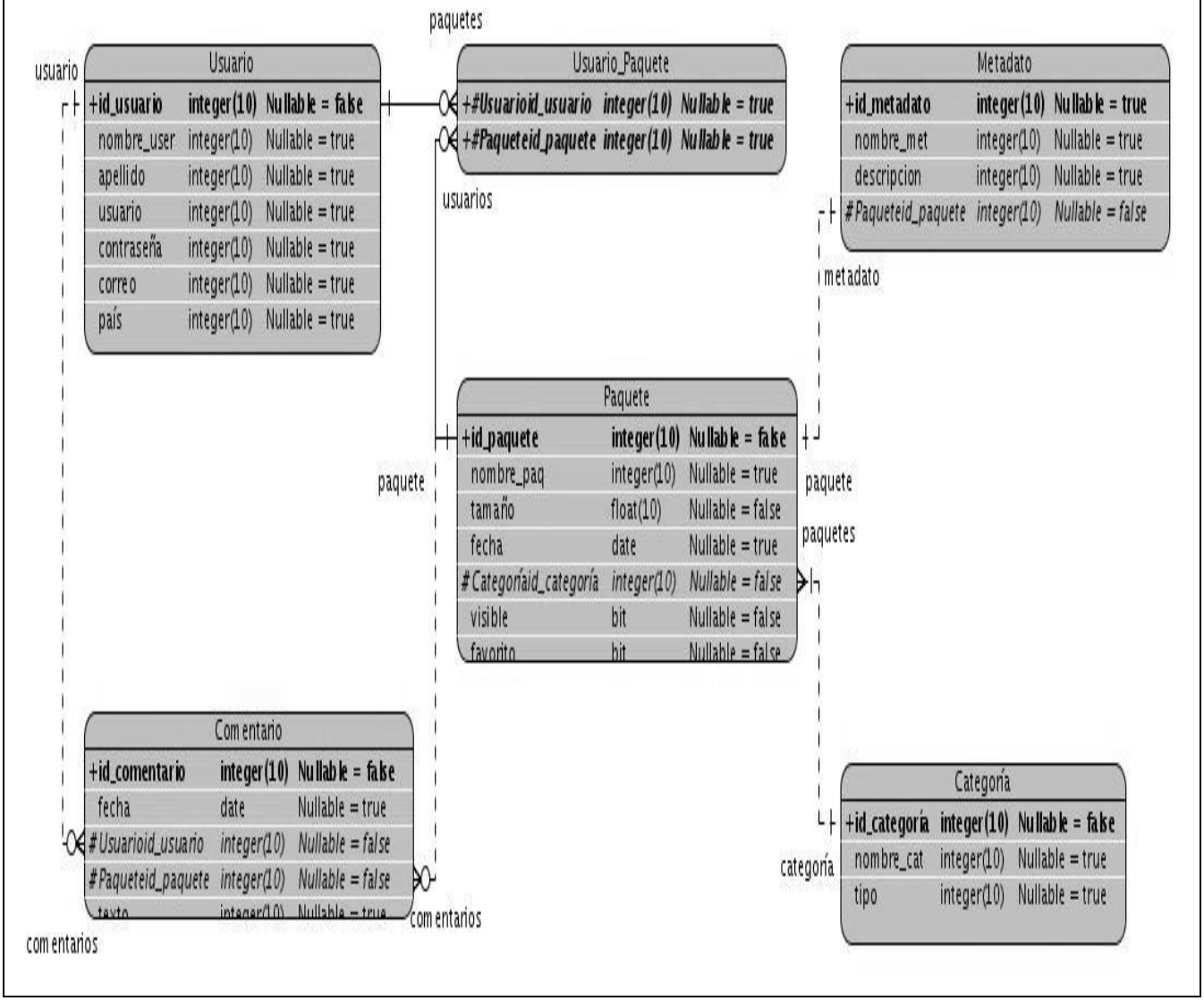


Figura 43 Diagrama Entidad Relación de la Base de Datos

3.2.6 Descripción de las tablas de la Base de Datos

Nombre: Usuario_Paquete		
Descripción: En esta tabla se almacenan los datos de los usuarios y paquetes.		
Atributo	**Tipo**	**Descripción**
id_usuario	int	Identificador del comentario.
id_paquete	int	Identificador del paquete.

Tabla 59 Descripción de las tablas de la base de datos. Usuario_Paquete

Nombre: Usuario		
Descripción: En esta tabla se almacenan los datos personales de los usuarios.		
Atributo	**Tipo**	**Descripción**
id_usuario	int	Identificador del usuario.
nombre_user	string	Nombre del usuario.
apellido	string	Apellido del usuario.
usuario	string	Usuario único para acceder al sistema.
contraseña	string	Contraseña del usuario para acceder al sistema.
país	string	Lugar de procedencia.
correo	string	Dirección de correo electrónico.

Tabla 60 Descripción de las tablas de la base de datos. Usuario

Nombre: Paquete		
Descripción: En esta tabla se almacenan los datos de los paquetes de requisitos.		
Atributo	**Tipo**	**Descripción**
id_paquete	int	Identificador del paquete.
nombre_paq	string	Nombre del paquete.
tamaño	float	Tamaño del paquete.
visible	bool	Visibilidad del paquete dentro del sistema.
favorito	bool	Visibilidad del paquete en su área de trabajo.
fecha	date	Fecha en que se importó el paquete.
id_categoría	int	Identificador de la categoría donde se encuentra.

Tabla 61 Descripción de las tablas de la base de datos. Paquete

Nombre: Categoría		
Descripción: En esta tabla se almacenan los datos de las categorías del sistema.		
Atributo	**Tipo**	**Descripción**
Id_categoría	int	Identificador de la categoría.
nombre_cat	string	Nombre de la categoría.
tipo	string	Tipo de categoría.

Tabla 62 Descripción de las tablas de la base de datos. Categoría

Nombre: Metadato		
Descripción: En esta tabla se almacenan los datos de los metadatos.		
Atributo	**Tipo**	**Descripción**
id_metadato	int	Identificador del metadato.
nombre_met	string	Nombre del metadato.
descripción	string	Descripción del metadato.
id_paquete	int	Identificador del paquete al cual pertenece dicho metadato.

Tabla 63 Descripción de las tablas de la base de datos. Metadato

Nombre: Comentario		
Descripción: En esta tabla se almacenan los datos de los comentarios realizados.		
Atributo	**Tipo**	**Descripción**
id_comentario	int	Identificador del comentario.
fecha	date	Fecha en que se realiza el comentario.
id_usuario	int	Identificador del usuario que realiza el comentario.
id_paquete	int	Identificador del paquete al que se le adiciona el comentario.

Tabla 64 Descripción de las tablas de la base de datos. Comentario

3.2.7 Principios de Diseño

Para lograr una aplicación que responda a las necesidades del cliente se debe tener en cuenta el conocimiento y el ambiente que influyen sobre los usuarios a los que va dirigido el producto, por tanto el diseño de la misma debe estar orientado al cumplimiento de este objetivo.

La aplicación que se va a desarrollar va dirigida a informáticos los cuales deben tener un amplio conocimiento en el trabajo con computadoras y más específicamente con aplicaciones similares, pero el diseño se centra en lograr una interfaz clara, de fácil uso, pues las actividades que realizan durante la reutilización de requisitos no son complejas, además que el objetivo es facilitar la realización de la fase de inicio del proceso de desarrollo del software y no complicarla aún más.

3.2.8 Interfaz

El Repositorio de Requisitos Reutilizables cuenta con una página principal desde la cual pueden navegar los usuarios sin privilegios, los cuales tienen acceso a buscar paquetes de requisitos según las categorías existentes, además de consultar los metadatos de dichos paquetes. El usuario autentificado podrá acceder a las otras funcionalidades que brinda el sistema en dependencia de los privilegios que tenga.

El menú principal estará ubicado verticalmente y a la izquierda, el elemento del menú donde se encuentra el usuario en ese momento se encontrará destacado con un color secundario. En el sistema aparecerá la fecha y el nombre del usuario que está autenticado.

La aplicación cuenta además con una ayuda que debe estar accesible como parte del menú en todas las páginas de la aplicación, con el objetivo de que el usuario vea la información que necesita en ese momento. Esta cuenta con informaciones generales de la aplicación, explicaciones de la funcionalidad del sistema. Cada página muestra cómo realizar aquellas operaciones que estén relacionadas con la posición donde se encuentre el usuario en dicho momento.

Teniendo en cuenta que el sistema es una aplicación de trabajo, el diseño se ha definido de forma que se utilicen colores básicos, el azul, el blanco, con pocas imágenes, con el objetivo de lograr un mayor

rendimiento cuando se carguen las páginas. La resolución óptima para la cual está diseñada la aplicación es de 1024 x 768 px y los elementos de la pantalla serán siempre de los colores definidos.

La tipografía será siempre Tahoma, por su amplia legibilidad y por las facilidades conocidas que brinda para la lectura digital. El menú principal será a 10 ptos y los submenús a 8 ptos. Las ventanas de error, validación de datos y otras mantendrán el mismo diseño.

3.2.9 Patrones de diseño

Con el objetivo de lograr un sistema robusto y reusable el diseño del mismo se realizó utilizando patrones de diseño, ya que estos son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Los patrones que se utilizaron responden a la arquitectura definida para el desarrollo del sistema y son los siguientes:

Domain Model: es el diseño orientado a objetos de cada componente del sistema, además permite el aislamiento tanto con sus capas superiores como inferiores. Así se dice que provee un modelado transparente de la presentación y la persistencia. Se utiliza principalmente cuando la lógica de negocio es compleja, y depende de muchos factores, los objetos son los indicados para modelarlo. Un Modelo de Dominio crea una red de objetos interconectados, donde cada objeto representa algo significativo e individual.

Service Layer: utiliza los objetos del modelo del dominio para resolver los casos de uso de la aplicación. Se utiliza para definir con una capa de servicios un límite de la aplicación que expone un conjunto de operaciones a través de una interfaz de servicio. Una Service Layer encapsula toda la lógica de negocio, control de transacciones y coordinación de la respuesta en la implementación de las operaciones que expone.

DataTransferObject (DTO): Estos objetos DTO cumplen el papel de contenedores de datos y se utiliza con el objetivo de facilitar el transporte de información entre las capas de presentación/servicio, agrupando los datos necesarios de presentación que se encuentran en atributos de varios objetos del dominio. Están orientados a cada caso de uso y su vista asociada en la capa presentación.

3.3 Conclusiones de capítulo

En este capitulo se ha desarrollado el flujo de trabajo análisis y diseño cuya importancia radica en mostrar una idea completa de lo que será realmente el software. Se obtuvieron los artefactos fundamentales que propone la metodología RUP como son los diagramas de clases de análisis y de diseño, diagrama de interacción, modelo de datos y descripción de las clases. Además de plantear los principios de diseño incluyendo en él aquellos patrones utilizados para modelar el diseño teniendo en cuenta la arquitectura propuesta. Esto servirá de guía para los programadores que implementarán el sistema.

Conclusiones

Una vez culminada la investigación se puede afirmar que se cumplieron los objetivos propuestos, arribando a las siguientes conclusiones:

- No existen estándares de metadatos para la descripción de requisitos.
- El uso de los estándares propuestos por la IEEE para la representación de requisitos, permite la reutilización de los mismos en nuevos proyectos.
- El uso de estándares correspondientes para la construcción del repositorio garantiza la interoperabilidad con otros sistemas.
- El modelo del Repositorio de Requisitos Reutilizables que se propone ofrece un conjunto de servicios que potenciará en gran medida la reutilización de requisitos y con ello la calidad del software.

Recomendaciones

Para dar continuidad a la presente investigación se recomienda:

- Promover el tema de reutilización de requisitos en los Proyectos Productivos.
- Continuar con el proceso de desarrollo del repositorio en sus siguientes fases.
- Proponer un esquema de metadatos para la descripción de requisitos.
- Agregar nuevas funcionalidades que fomenten aún más la reutilización de requisitos a partir de las experiencias durante su explotación.
- Insertar las ideas del enfoque semántico en el Repositorio de Requisitos Reutilizables en futuras versiones.
- Utilizar el estándar OKI para lograr mecanismos de interoperabilidad en próximas versiones del repositorio.

Referencias Bibliográficas

ALVAREZ, R. *Lenguajes de lado servidor*, 2006. [Disponible en: http://www.desarrolloweb.com/articulos/243.php.

AULAGLOBAL. *Observatorio de e-Learning*, 2005. [Disponible en: http://www.aulaglobal.net.ve/observatorio/mobile/articles.php?lng=es&pg=86

CID, A. and J. E. COSS. *Herramienta de autor para la creación y gestión de objetos de aprendizaje reutilizables* Ciudad de la Habana, Universidad de las Ciencias Informáticas, 2006. p.

COLECTIVO DE AUTORES. *Historia de PHP y proyectos relacionados*

2004. [Disponible en: http://elgaizka.com/manualphp/history.html

EVA MÉNDEZ and J. A. SENSO. *Metadatos*, 2004. [Disponible en: http://www.sedic.es/autoformacion/metadatos/introduccion.htm

FLEITAS, R. A. and Y. SABINA. *Arquitectura General del Proyecto Centro Rector de*

Universidad para Todos. Ciudad de La Habana, Universidad de las Ciencias Informáticas, 2007. p.

GIL, M. A. *Arquitectura Orientada a Servicios*. Universidad de las Ciencias Informáticas, 2008.

GONZÁLEZ J, S. J., ROBLES G. *Introducción al software libre* ESCET, 2003.

HECHEVARRÍA, D. *Modelos Semánticos para la Gestión de Requisitos*. Ciudad de la Habana, Universidad de las Ciencias Informaticas, 2007. p.

IEEE. *Standard IEEE 830-1993: Recommended Practice for Software Requirements Specifications*, 1993.

---. *Std 1233-1998. Guide for Developing System Requirements Specifications*, 1998.

J. CASAMAJÓ, P. C. and M. ALIER. *Una plataforma de integración*. Proyecto CAMPUS. Barcelona, Universitat Oberta de Catalunya. p.

J. ROBERTSON and S. ROBERTSON. *Plantilla de Especificación de Requisitos* 2006. [Disponible en: http://www.volere.co.uk/template_ESP.doc.

JACOBSON I and BOOCH G. *El Proceso Unificado de Desarrollo de Software*. 1999. p.

LOUCOPOULOS, P. and V. KARAKOSTAS. *System Requirements Engineering McGrawHill*. 1995. p.

M. GARCÍA and S. LÓPEZ. *SOA: para los sistemas de educación en línea*, 2007.

MARTINEZ, J. A. and P. LARA. *Interoperabilidad de los contenidos en las plataformas de e-Learning: normalización, bibliotecas digitales y gestión del conocimiento*. *Revista de Universidad y Sociedad del Conocimiento*, 2006. 3.

MARTÍNEZ P and SANTAMARÍA B. *LOS METADATOS Y LA INFORMACIÓN DIGITAL* México, 2002. p.

METADATOS. *¿Qué son los Metadatos?*, 2007. [Disponible en: http://antares.inegi.gob.mx/metadatos/metadat1.htm

R. SOCARRÁS and L. RAMOS. *Propuesta de una herramienta libre para la Gestión de Requisitos en proyectos productivos de la facultad 10.* Ciudad de La Habana, Universidad de las Ciencias Informáticas, 2008. p.

RICHARD, H. *IEEE Software Requirement Engineering*. New York 1997. p.

SEGOVIANO, Z. and Y. VIDIAUX. *La reutilización como parte de la calidad del software. Su aplicación en la UCI.* Ciudada de La Habana, Universidad de las Ciencias Informaticas, 2007. p.

SENÉ, M. L. *LOS METADATOS Y SU LUGAR EN LA ARENA INTERNACIONAL*. Ciudad de La Habana, 2004. p.

SOFTWARE, D. D. I. D. *Introducción a la Ingeniería de Software*. Universidad de las Ciencias Informáticas, 2007.

SOLARTE, M. F. *Propuesta de una Aproximación Metodológica para la Ingeniería de Requisitos de Sistemas Telemáticos* 2003.

TORRE, A. *Lenguajes_del_lado_servidor_o_cliente*, 2006. [Disponible en: http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

TOVAL A., N. J. and M. B. *SIREN: Un Proceso de Ingeniería de Requisitos Basado en Reutilización.* Murcia, 2001. p.

Bibliografía

O.K.I.™ Architectural Concepts. Massachusetts Institute of Technology, 2003.

ALFARO, D. *Diseño del Módulo de Adquisición para el Departamento Desarrollo de Colecciones de la Biblioteca Nacional José Martí*. Ciudad de La Habana, Universidad de las Ciencias Informáticas, 2007. p.

COMPANY, T. R. *Presente y Futuro de la Reutilización de Software*, 2006.

GONZALEZ, P. *Tema 6 Reutilización de Software*, 2006. [Disponible en: http://wwwdi.ujaen.es/asignaturas/isii/isii0506/documentos/ISII_0506_06_1p.pdf

LARMAN, C. *UML y Patrones*. p.

POSTGRESQL, E. D. D. D. *Manual de usuario de PostgreSQL*, 1999.

PRESSMAN, R. S. *Ingeniería del Software: un enfoque práctico. Parte I y II* quinta edición. La Habana, Editorial Félix Varela, 2005. p.

SAMETINGER, J. *Software Engineering with Reusable Components*. Springer Verlag, 1997. p. 3-540-62695-6

TAMAYO, D. *Herramientas para la reutilización de contenidos educativos*, 2007. P.

Anexos

Anexo 1: Contorno de SRS

IEEE 830-1998: Estándar para especificación de Requisitos del Software (SRS)

* Calidad -> No ambigua, Completa, Concisa, Fácil de Verificar, Consistente, Fácil de Modificar, Fácilmente Trazable, Clasificada por prioridades.

1. Introducción
    - Propósito
    - Alcance
    - Definiciones, siglas, y abreviaciones
    - Referencias
    - Apreciación global
2. Descripción global
    - Perspectiva del producto
    - Funciones del producto
    - Características del usuario
    - Restricciones
    - Atención y dependencias
3. Los requisitos específicos
4. Apéndices
5. Índice

Anexo 2: Contorno de SyRS

IEEE 1233-1998: Estándar para especificación de Requisitos del Sistema (SyRS)

1. Introducción
    - Sistema propósito
    - Sistema de alcance
    - Definiciones, acrónimos y abreviaturas
    - Referencias
    - Sistema de visión general
2. Descripción general del sistema
    - Sistema de contexto
    - Sistema de modos y de los Estados
    - Las principales capacidades de sistema
    - Las principales condiciones del sistema
    - Las principales limitaciones del sistema
    - Características del usuario
    - Supuestos y dependencias
    - Escenarios de operaciones
3. Sistema de las capacidades, condiciones y limitaciones
4. Sistema de interfaces

Anexo 3: Conjunto de elementos de datos Dublín Core: ISO 15836:2003

NOMBRE	DESCRIPCIÓN	ESQUEMA	TIPO	ETIQUETA
Título	Nombre dado a un recurso u objeto.	• Interno * • AACR2	• Principal • Alternativo	DC.Title
Autor o creador	Persona(s) reponsable(s) del contenido intelectual del recurso u objeto.	• Interno * • USMARC	• Nombre • Correo electrónico	DC.Author
Encabezamiento	Direcciona hacia los tópicos que describen el título o contenido del recurso u objeto.	• Interno * • LCSH • UDC. • DDC • NLM • Etc.	• Palabras o frases claves separadas por comas. • Notas acerca del recurso.	DC.Subject
Descripción	Descripción textual del recurso.	• Interno * • URL • URN.	• Texto Libre. • Resumen.	DC.Description
Editor	Entidad responsable de hacer que el producto se encuentre disponible en la red.	• Interno. * • USMARC	• Nombre • Correo electrónico	DC.Publisher
Otros Colaboradores **	Persona(s)que hayan hecho alguna contribución intelectual adicional significativa al trabajo.	• Interno. * • USMARC	• Nombre • Correo electrónico	DC.Contributor
Fecha de publicación	Fecha en que se puso el recurso a disposición del usuario.	• IETF.RFC-822 ANSI.X3.30-1985 • ISO • FGDC, etc.	• Fecha de creación	DC. Date
Tipo de Recurso u Objeto	Categoría o género del recurso.	• DC Objetcts. * • Texto libre.		DC.Type
Formato	Formato de los datos	• IMT • MIME. • Texto libre.		DC.Format
Identificador	Identificador único del recurso u objeto.	• URL • URN • ISBN • ISSN • FPI • Número de versión.	• Primera versión del recurso. • Copia de la versión original del recurso.	DC.Identifier
Fuente	Identificador único de un trabajo a partir del cual proviene el recurso actual.	• Texto libre. • URL • URN. • ISBN • ISSN • FPI		DC.Source
Idioma	Idioma del contenido intelectual.	• Texto libre. * • Z3953 • ISO.639 • Lenguaje cómputo.		DC.Language

Relación	Un identificador de un segundo recurso y su relación con el recurso actual.	• URL* • URN • ISBN • ISSN • FPI	• Texto libre	DC.Relation
Cobertura	Característica de cobertura espacial y temporal del recurso.	• Latitud y longitud. • OSGB • ANSI X3.30-1985 • Texto libre	• Espacial • Temporal	DC.Coverage
Derechos	Nota sobre derechos de autor. Su uso se considera experimental.	• Texto libre* • URL • URN		DC.Rights

Anexo 4: Plantilla de Especificación de Requisitos: VOLERE

1. Guías del Proyecto

    El Propósito del Proyecto

    El Cliente, el Comprador otras partes involucradas

    Usuarios del Producto

2. Restricciones del Proyecto

    Restricciones Obligatorias

    Denominación de Acuerdos y Definiciones

    Hechos Relevantes y Suposiciones

3. Requisitos Funcionales

    El Alcance del Trabajo

    El Alcance del Producto

    Requisitos Funcionales y Data

4. Requisitos No Funcionales

    Requisitos de Percepción (Tocar y Sentir)

    Requisitos Capacidad de Uso y Humanidad

Requisitos de Desempeño

Requisitos Operacionales y Ambientales

Requisitos de Preservación y Soporte

Requisitos de Seguridad

Requisitos Culturales y Políticos

Requisitos Legales

5. Aspectos del Proyecto

Aspectos Abiertos

Soluciones Disponibles (Off-the-Shelf)

Nuevos Problemas

Tareas

Migración al Nuevo Producto

Riesgos

Costos

Documentación del Usuario y Entrenamiento

Sala de Espera

Ideas para Soluciones

Anexo 5: Descripción textual de los casos de uso

Caso de Uso	**Autenticar usuario**
Actores	Usuario
Propósito	Permitir a un usuario autenticarse y acceder al sistema.
Resumen	El caso de uso inicia cuando el usuario decide acceder al sistema, este introduce los datos que le solicitan para acceder al mismo y finaliza cuando el sistema verifica los datos y le habilita la entrada.
Referencias	R1
Precondiciones	El usuario debe estar registrado.
Poscondiciones	Se habilitan las funcionalidades según lo privilegios.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción de autenticarse.	1.1 El sistema solicita (usuario y contraseña) al usuario.
2. El usuario introduce los datos.	2.1 El sistema comprueba la validez de los datos del usuario. 2.2 El sistema verifica que el usuario este registrado. 2.3 El sistema muestra la interfaz de bienvenida y las opciones a las que tiene acceso según sus privilegios.
Flujo Alternativo	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si los datos no son válidos el sistema envía un mensaje de aviso. 2.3 Si el usuario no está registrado el sistema envía un mensaje de aviso.
Prioridad	Crítico

Descripción textual del CU Autenticar usuario

Caso de Uso	**Editar perfil**
Actores	Analista
Propósito	Permitir a un analista editar su perfil.
Resumen	El caso de uso inicia cuando el analista selecciona la opción de editar su perfil, este inserta los datos que el sistema le pide y finaliza cuando el sistema actualiza dichos datos.
Referencias	R2
Precondiciones	El analista debe estar autenticado.
Poscondiciones	Los datos del analista quedan actualizados.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El analista selecciona la opción de editar su perfil.	1.1 El sistema muestra los datos personales del analista que puede insertar o modificar.
2. El analista inserta los datos.	2.1 El sistema verifica los datos. 2.2 El sistema actualiza los datos
Flujo Alternativo	
Acción del Actor	**Respuesta del Sistema**
	2.1 Si los datos no son válidos el sistema envía un mensaje de aviso.
Prioridad	Auxiliar

Descripción textual del CU Editar Perfil

Caso de Uso	**Buscar requisitos**
Actores	Usuario
Propósito	Permitir a un usuario realizar búsquedas de paquetes de requisitos.
Resumen	El caso de uso inicia cuando un usuario selecciona la opción de buscar paquetes de requisitos. El sistema muestra una serie de criterios que el usuario puede utilizar para especificar las características de contenido que busca. El caso de uso finaliza cuando el sistema muestra los resultados de la búsqueda.
Referencias	R3

Precondiciones	El usuario debe insertar al menos un criterio para la búsqueda.
Poscondiciones	Se muestran los resultados de la búsqueda.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción de buscar paquetes de requisitos.	1.1 El sistema muestra un formulario de búsqueda.
2. El usuario introduce al menos un dato de lo que desea buscar.	2.1 El sistema muestra la información que describe el usuario.
Flujo Alternativo	
Acción del Actor	**Respuesta del Sistema**
	2.1 En caso de no existir ningún resultado el sistema envía un aviso.
Prioridad	Crítico

Descripción textual del CU Buscar requisitos

Caso de Uso	**Consultar metadatos**
Actores	Usuario
Propósito	Permitir a un usuario consultar metadatos.
Resumen	El caso de uso se inicia cuando el usuario selecciona la opción de visualizar los metadatos asociados a los paquetes de requisitos y finaliza cuando el sistema muestra dichos metadatos.
Referencias	R4
Precondiciones	--
Poscondiciones	Se muestra el contenido de los metadatos consultados.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona el metadato a consultar.	1.1 El sistema muestra el contenido del metadato.
Flujo Alternativo	
Acción del Actor	**Respuesta del Sistema**
Prioridad	Secundario

Descripción textual del CU Consultar metadato.

Anexo 6: Diagramas de colaboración de análisis

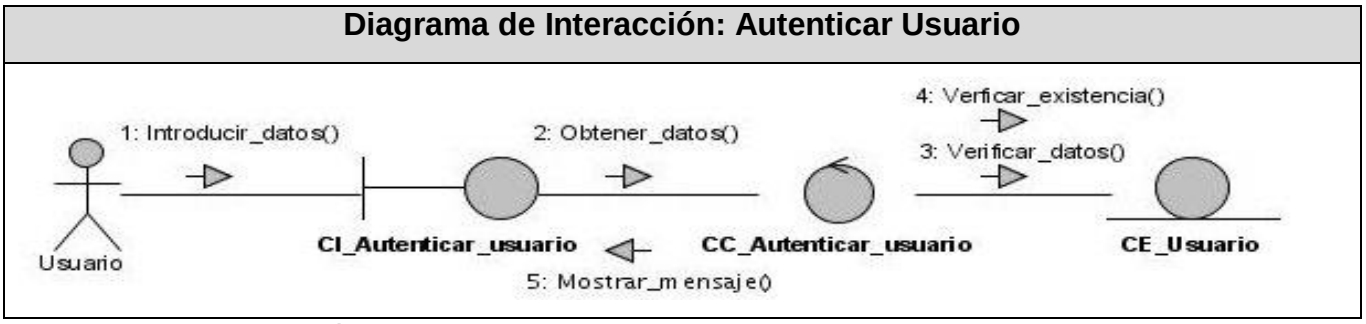


Diagrama de colaboración (CU Autenticar Usuario)

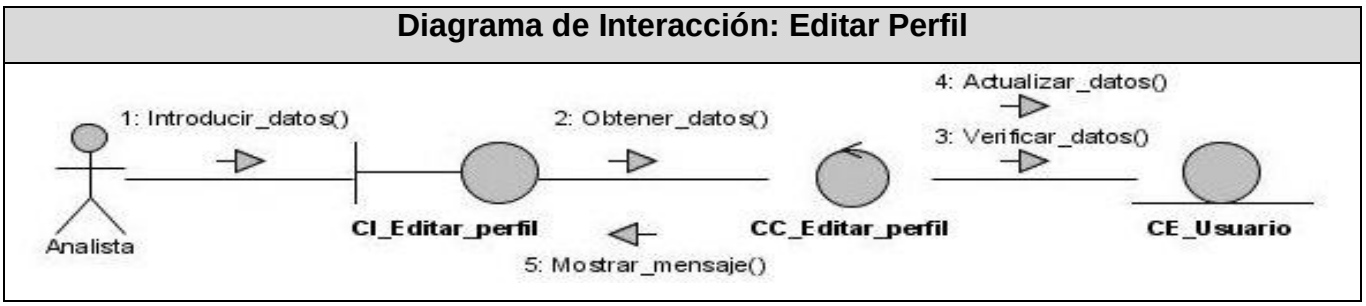


Diagrama de colaboración (CU Editar Perfil)

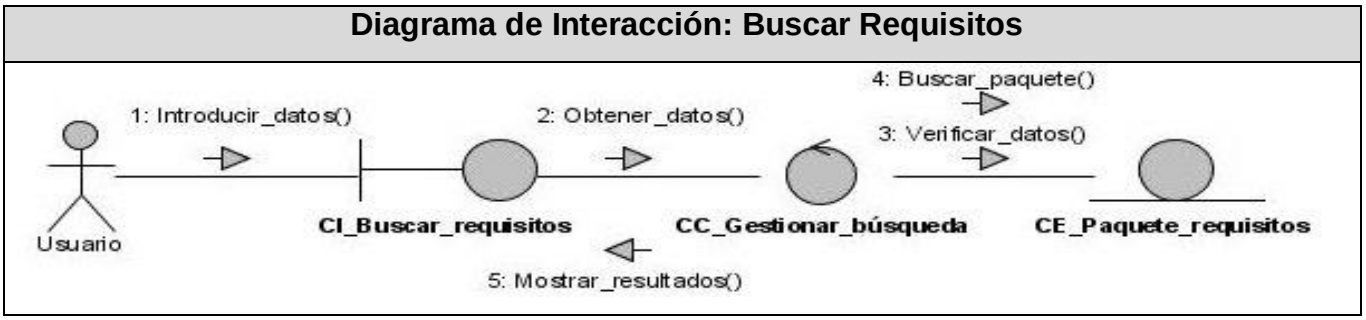


Diagrama de colaboración (CU Buscar requisitos)

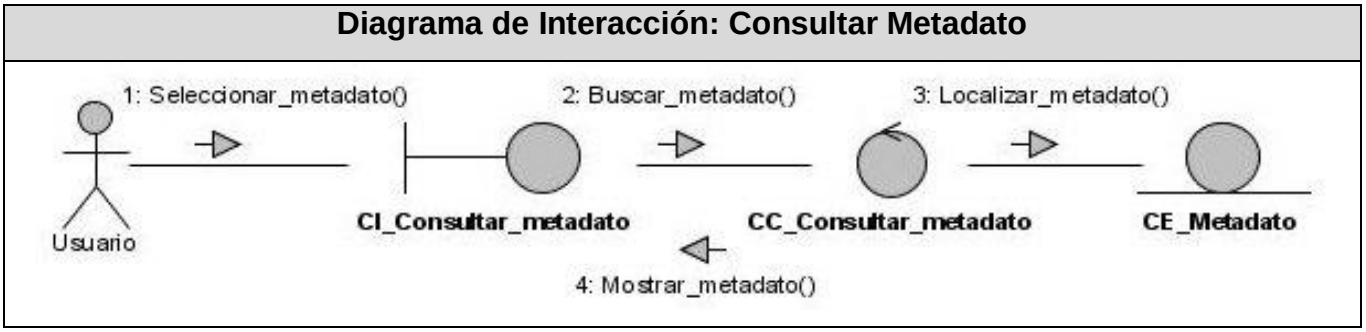


Diagrama de colaboración (CU Consultar metadato)

Glosario de términos

IEEE: Corresponde a las siglas de The Institute of Electrical and Electronics Engineers, es una asociación técnico profesional mundial dedicada a la estandarización, entre otras cosas.

Metadatos: Información acerca de las propiedades de los datos. También puede ser información sobre la estructura de los datos o información que especifique el diseño de objetos.

Norma: En el área tecnológica, una norma o estándar es una especificación que reglamenta procesos y productos para garantizar la interoperabilidad.

Interoperabilidad: Capacidad de dos o más sistemas para relacionarse e intercambiar información de manera útil y con sentido.

Metodología: La metodología se entenderá aquí como la parte del proceso de investigación que sigue a la propedéutica y permite sistematizar los métodos y las técnicas necesarios para llevarla a cabo.

Metamodelo: Es un modelo de información para describir modelos.

Artefacto: En tecnología, artefacto es un dispositivo concebido y fabricado, sea de modo artesanal o industrial, por una o más personas. La característica principal de los artefactos es que cumplen una función técnica, es decir, sirven para hacer algo.

Aplicación: Programas con los cuales el usuario interactúa a través de una interfaz.

Proceso: Es un conjunto de actividades o eventos que se realizan o suceden con un determinado fin.

Producto: Resultado obtenido de un proyecto productivo.

Interfaz: Parte de un programa que permite el flujo de información entre un usuario y la aplicación.