Universidad de las Ciencias Informáticas
Facultad 10

Título: “Modelación de un Sistema para la Gestión del Proceso de Liberación de Productos Software.”

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autores: Osmany Cosano Delgado.

Harnier Suárez Rodríguez.

Tutor: Ing. Odeimy Morales Duarte.

Ing. Geiser A. Pérez Rivas.

CIUDAD DE LA HABANA, CUBA

Junio, 2009

“Año del 50 Aniversario del Triunfo de la Revolución

DEDICATORIA

Dedicatoria

A mis padres Pedro y Caridad, a mi hermano, a mi compañera Dayana, a mi familia en general. Por todo el cariño y toda una vida de constante dedicación y sacrificios, por darme todo lo que necesitaba.

Con ellos a mi lado todo será alcanzable.

Harnier Suárez Rodríguez.

A mis padres Rosa y Jesús, a mi hermana Saskia, a mi cuñado Osmany y a su familia, a mi compañera Laura y a mis sobrinos Roxana y Yosvany a todos gracias por sus esfuerzo y porque desde el principio siempre confiaron en mí.

Osmany Cosano Delgado.

AGRADECIMIENTOS

Quiero expresar mis más sinceros agradecimientos a las dos personas que más amo: mis padres, se me tendría que conceder la eternidad y no bastaría para tratar de retribuir todo el amor y el cariño que me han brindado, todos los esfuerzos y sacrificios que han hecho para que hoy sea un hombre correcto. De ellos he recibido la mayor riqueza que jamás pueda tener: sus impecables ejemplos.

A mi prima Damilsy y a su esposo Antonio, por haberme abierto las puertas de sus corazones como a un hijo, por brindarme su cariño y apoyo, gracias les doy por haber hecho que mi vida fuera de casa fuera tan amena y por qué no más fácil también.

A mi compañera gracias por ser la inspiración de mi vida, por ser el paradigma de persona a seguir, por todo su amor, comprensión, dedicación y ayuda.

A mis compañeros, gracias por permitirme haber sido parte de sus vidas durante estos 4 años, para mi ha sido un placer ese privilegio concedido.

A todos aquellas personas que de una forma u otra han hecho posible la realización de este trabajo, Gracias.

Harnier Suárez Rodríguez.

AGRADECIMIENTOS

Primero agradecer a la Revolución por haberme dado esta linda oportunidad de estudiar en la UCI.

Agradecer de todo corazón a mis padres quienes desde pequeño supieron guiarme hacia el camino correcto y que sin su ayuda y dedicación no hubiera sido posible convertirme en lo que soy hoy, a ustedes muchas gracias de todo corazón.

A mi hermana Saskia quien ha sido para mi más que una hermana mi segunda madre y a su esposo Osmany que ha sido un verdadero hermano a ustedes gracias por su amor y apoyo.

A todo la familia de mi cuñado Aida, Esmildo, Yanet y Ronney a ustedes gracias por su apoyo y todo su amor.

A mi tía Migdalia le doy muchas gracias por haberme acogido como a un hijo durante estos 5 años y darme todo su cariño, a mi prima Zudelis por darme su amor y confianza y a su esposo Ovidio gracias por su confianza y por sus consejos que a lo largo de estos 5 años me sirvieron de mucho y me prepararon para la vida.

A todo la familia de Ovidio El Masa, Dania, Limay, Omar y Omarito gracias por su confianza y cariño

A mi compañera Laura gracias por haber confiando en mi y por darme todo tu amor.

A mis amistades de siempre Amalia, Lisbeth, Daylian, Elsita, Lisett, Ronny, Víctor, Soto, a ustedes gracias por estar siempre en el momento indicado.

A mis amistades de la UCI si mencionar nombres ustedes siempre estarán en mi corazón, a todas mis amigas y a mis hermanos no de sangre pero si de corazón, ustedes saben quienes son, nunca los olvidaré, a todos mis compañeros de grupo gracias por compartir estos 5 años de mi vida gracias.

Osmany Cosano Delgado.

Declaración de Autoría

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

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

Harnier Suárez Rodríguez.

_____

Firma del Autor

Osmany Cosano Delgado.

_____

Firma del Autor

Ing. Geiser A. Pérez Rivas.

_____

Firma del Tutor

Datos de Contacto

Odeimy Morales Duarte (**omorales@uci.cu**).

Profesión: Profesor Facultad 1.

Título: Graduado de Ingeniero en Ciencias Informáticas. Habana 2008.

Categoría Docente: Adiestrado.

Años de graduado: 1

Geiser Arecio Pérez Rivas (**gperez@uci.cu**).

Profesión: Especialista de Calisoft.

Título: Graduado de Ingeniero en Ciencias de la Informática. Habana 2007.

Categoría Docente: Instructor.

Años de graduado: 2

RESUMEN

Resumen

Entre los principales flujos de trabajo que se realizan en la Dirección Central de Calidad se encuentra el de comprobar la calidad de los productos software que se crean tanto en la Universidad de las Ciencias Informáticas (UCI), como en otras Empresas del país; al mismo se le denomina Proceso de Liberación de Productos Software (PLPS), actualmente este proceso se lleva a cabo de forma manual causando numerosos inconvenientes en la gestión del mismo. Este trabajo pretende modelar un Sistema Informático que solucione los problemas relacionados con la planificación, gestión y control del cúmulo de actividades, recursos e información, necesarios para garantizar el Proceso de Liberación de todos los productos software. Para alcanzar este objetivo se utilizó la metodología de desarrollo RUP, el lenguaje de modelado UML, y la herramienta CASE Visual Paradigm.

Palabras Claves: Proceso de Liberación de Productos Software, RUP, UML, Visual Paradigm.

Abstract

Checking the quality of software products created at the Information Sciences University (UCI) and in other enterprises throughout the country is among the key workflows carried out at the Central Management of Quality. It is known as Software Release Process (PLPS). Today this process is done manually, bringing about numerous drawbacks in its management. This paper is aimed at modeling a system able to solve the problems associated with planning, management and control of a series of activities, resources and information necessary to ensure the release process of all software products. In order to achieve this goal, we used the RUP development methodology; the UML modeling language and the CASE Visual Paradigm tool.

Keywords: RUP, UML, Visual Paradigm.

Índice

ÍNDICE

ÍNDICE

Figuras

ÍNDICE

Tablas

Introducción

Desde que el hombre comenzó a dejar pruebas de su paso a través de los años, la información se volvió imprescindible para su vida, más ahora en el siglo XXI, con el establecimiento de una sociedad donde la información está presente en todos los ámbitos en que se desarrolla, especialmente en los entornos estudiantiles, laborales e institucionales; por tal motivo se vio en la necesidad de gestionar este valioso recurso de forma eficiente y efectiva, para garantizarlo ha acudido a la utilización de sistemas de información computarizados, en particular el software[1].

La producción de software hoy, constituye un sector de enorme importancia mundial, se encuentra en el centro de todas las grandes transformaciones; sobre todo si se considera que los temas de primer orden, como lo son la economía digital, la evolución de las empresas y la administración del conocimiento, se gestionan con software.

Los proyectos informáticos se imponen en la "nueva era del conocimiento". Por esta razón el número de Empresas Desarrolladoras de Software (EDS) ha aumentado a nivel mundial, y con ello los niveles de competencia en la industria, establecidos, al principio por producir más, ahora desde otra perspectiva, producir con calidad, este término es ahora la nueva preocupante de dichas empresas, pues debe tenerse en cuenta en todas las etapas del desarrollo del mismo, para poder satisfacer las necesidades de los clientes. Es por ello que comprobar la calidad de estos productos es una de las actividades más importante que deben garantizar.

Al calor de la batalla de ideas surge en Cuba la Universidad de las Ciencias Informáticas (UCI), expresión del más novedoso esfuerzo del país en aras de desarrollar la industria del software, la cual está destinada a ser la mayor EDS de la nación, las líneas de trabajo fundamentales de este centro estarán destinadas a la informatización de la sociedad y a la inserción de la misma en el mercado mundial de software, para así elevar la economía nacional, pues el sustento de la nación depende en gran parte de ello. Por estas razones la UCI no podía quedar exenta de la necesidad de producir con calidad.

[1]Software: es un conjunto de programas, instrucciones y reglas informáticas que permiten ejecutar distintas tareas en una computadora.

INTRODUCCIÓN

La comprobación de la calidad de los productos software al final del desarrollo, en la UCI, se le conoce con el nombre de Proceso de Liberación de Productos Software (PLPS), definido por la Dirección Central de Calidad de la propia universidad como: flujo de trabajo que comprende desde la gestión de Solicitud de Pruebas hasta la liberación del producto apto para ser entregado al cliente; este proceso, en su fase intermedia comprende entre otras cosas la gestión de Reuniones, Planificación de Recursos, Pruebas, etc.

En sus inicios, el PLPS estaba concebido para que se realizara a nivel de facultad, es decir, cada una tenía la responsabilidad de comprobar la calidad sólo a los productos de exportación que creaban.

Hoy la UCI precisa extender este proceso a todos los productos que desarrolla, incluyendo los que no van a ser exportados, unido a esto, existen demandas de otras EDS en el país que desean comprobar la calidad de sus productos en la universidad, como consecuencia, el número de proyectos a procesar aumenta aproximadamente en seis veces.

Bajo las circunstancias anteriormente expuestas la Dirección Central de Calidad toma la decisión de crear lo que es hoy el Laboratorio Industrial de Pruebas de Software (LIPS) cuyo propósito principal es lograr que todo producto elaborado, tanto en la UCI como fuera de ella, y presentado al laboratorio fuera comprobado y evaluado según normas y estándares de calidad, antes de ser entregado al cliente, siendo esta evaluación confiable para los equipos de desarrollo y para los clientes de la UCI.

La creación del LIPS, como se aprecia, sucede con el objetivo de satisfacer la necesidad de centralizar el PLPS, dicho cambio estructural provoca el surgimiento de una nueva ***situación problémica***: los especialistas no contaban con un método óptimo y eficiente para planificar, gestionar y controlar el cúmulo de actividades, recursos e información generada, necesarios para garantizar el Proceso de Liberación de todos los productos existentes, que sean llevados al laboratorio, contándose así con procesos poco controlados.

A continuación se expone un análisis realizado por los especialistas de calidad, después de un estudio exhaustivo, basado en la información generada por los distintos productos, que han sido procesados por el Grupo de Liberación y Pruebas de Software. El estudio se centró en, estimar el tamaño de dicha información en números de páginas, para que se comprenda en cierta medida cuán compleja y difícil sería la tarea de gestionar y controlar dicha información.

INTRODUCCIÓN

Básicamente un proyecto de software consta de aplicación y su respectiva documentación. Durante su estadía en el LIPS, se generan las siguientes informaciones: **Casos de Prueba, Escenarios[2], Listas de Chequeos, No Conformidades.**

Aplicación: Una aplicación cuenta con **Casos de Uso**, y por cada caso de uso se genera un **Caso de Prueba**, cada caso de prueba se traduce en **Escenarios**, y a su vez genera **No Conformidades.**

Documentación: Por cada documento elaborado durante el desarrollo del producto se genera una **Lista de Chequeo,** y por cada una se generan **No conformidades**.

A continuación se hará la estimación del número de páginas correspondientes a la información generada por un producto:

- Durante el desarrollo del software se crean aproximadamente 10 documentos.
- Cada documento es comprobado a través de 1 Lista de Chequeo.
- Una Lista de Chequeo es redactada aproximadamente en 15 páginas.
- Por cada Lista de Chequeo se generan aproximadamente 30 No Conformidades.
- En una página son redactadas aproximadamente 8 No Conformidades.
- Un proyecto de software consta como mínimo de 1 aplicación informática.
- Cada aplicación consta aproximadamente de 150 casos de uso.
- Por cada caso de uso se genera al menos uno de prueba.
- Un caso de prueba genera aproximadamente 10 escenarios.
- En 6 páginas pueden ser archivados 10 escenarios.
- Por cada 10 casos de prueba se generan 30 No Conformidades.

[2]Escenarios: Flujo específico de eventos para realizárseles pruebas.

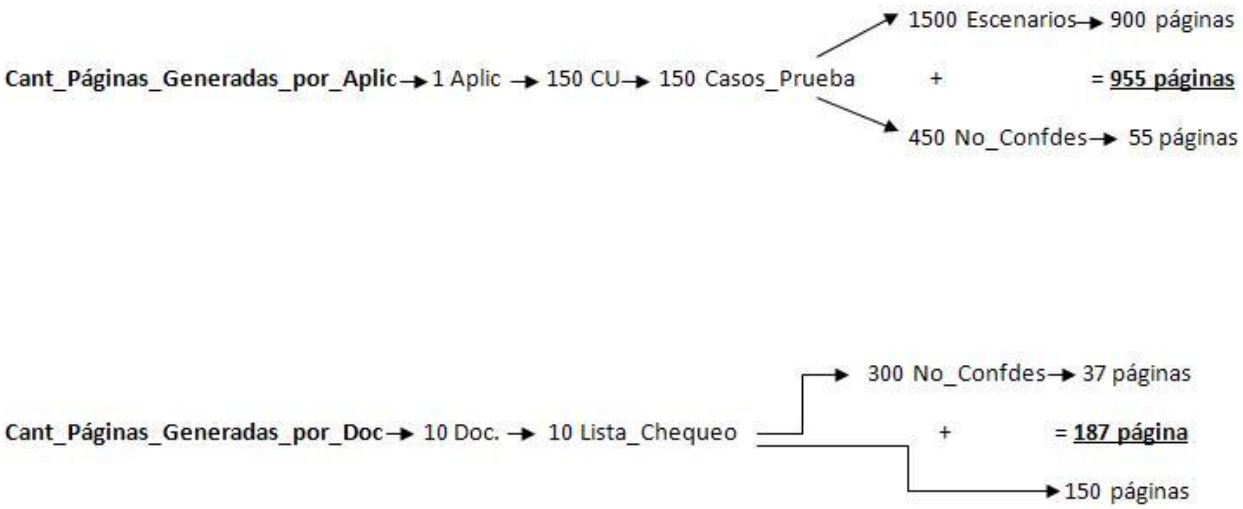


Figura 1: Análisis Realizado por los Especialistas. Cantidad de Páginas Generadas por Documento.

Cabe destacar además que a un producto de software se le realizan tantas **iteraciones**[3] de pruebas como hagan falta con el objetivo de que el mismo salga del LIPS con el mínimo de errores permisibles. Generalmente a todos los productos que se encuentran hoy en el laboratorio se le realizan 3. Por lo tanto el número de páginas resultantes correspondiente a la información que se genera por producto va a estar determinada por:

$$Total\_Páginas\_Generadas = 3 * (Cant\_Páginas\_Generadas\_Doc. + Cant\_Páginas\_Generadas\_Aplic)$$

Figura 2: Análisis Realizado por los Especialistas. Total de Páginas Generadas.

Retomando el análisis anterior, el número total de páginas generadas por proyecto serían:

[3] Iteraciones: Es un ciclo completo de prueba que abarca hasta las respuestas de las No Conformidades por parte del equipo de desarrollo.

Total_Páginas_Generadas = 3* (Cant_Páginas_Generadas_ Doc. + Cant_Páginas_Generadas_ Aplic)

Total_Páginas_Generadas = 3* (187 + 955)

Total_Páginas_Generadas = 3426 páginas

Figura 3: Análisis Realizado por los Especialistas. Total de Páginas Generadas.

Cabe destacar que sólo se enfatizó en la cantidad de información que puede generar un producto, sin tener en cuenta que en el laboratorio pueden coexistir simultáneamente varios de ellos, actualmente entran al LIPS aproximadamente 10 productos, estos generan alrededor de **34 260 páginas en un mes**, constituyendo un volumen considerable de información, donde su gestión y control se hace cada vez más difícil por parte de los especialistas.

Lo anterior constituye sólo una parte del gran problema que afecta al LIPS, pues en este caso no se ha tenido en cuenta los problemas existentes en cuanto a la planificación, gestión y control de otros aspectos importantes para llevar a cabo el PLPS, como por ejemplo la gran cantidad de actividades que el mismo comprende, así como los recursos que estas implican.

Por todo lo anteriormente expuesto se plantea como ***problema científico*** ¿Cómo transformar el Proceso de Liberación de Productos de Software, definido por la Dirección Central de Calidad, en un lenguaje entendible por los desarrolladores, que facilite su posterior automatización?

Como **Objeto de estudio** Proceso de Desarrollo de Software y como **Campo de Acción** Análisis y Diseño del sistema para automatizar el Proceso de Liberación de Productos de Software en la UCI.

Objetivo general

Modelar un sistema informático que facilite la Planificación, Gestión y Control del Proceso de Liberación de Productos de Software.

El desarrollo de este trabajo permitirá establecer las bases para la construcción de una herramienta informática para la gestión del Proceso de Liberación de Productos de Software, facilitando así, la

obtención de un producto que minimizaría los problemas de planificación, gestión y control de las actividades y recursos que comprende y dispone este proceso.

Para alcanzar el objetivo anteriormente planteado y dar solución a la situación problémica descrita anteriormente es imprescindible la realización de las siguientes **tareas:**

1. Estudio del estado del arte del Proceso de Comprobación de la Calidad de Productos Software a nivel internacional, nacional y en la UCI.
2. Estudio de las metodologías, herramientas y arquitecturas a utilizar para el desarrollo del sistema informático.
3. Entrevistas con los clientes para identificar sus necesidades y que deben garantizarse en un sistema de gestión de Proceso de Liberación de Productos Software.
4. Elaboración de la lista de requisitos funcionales y no funcionales que debe cumplir y tener respectivamente el sistema de gestión de Proceso de Liberación de Productos Software.
5. Realizar el análisis y diseño del sistema de gestión de Proceso de Liberación de Productos Software.

Para el desarrollo de estas tareas se implementaron como Métodos Científicos los siguientes:

Métodos Teóricos:

Analítico-Sintético: Mediante este método se realizó el procesamiento de la información, analizándose documentos y conceptos, haciéndose énfasis en la profundización de los mismos con el objetivo de caracterizar el Objeto de Estudio. Una vez entendido este, la situación problémica y la teoría, se pasó entonces a la unión de todos estos conocimientos para dar solución al problema.

Histórico-Lógico: Este método brindó la posibilidad de conocer la evolución del Proceso de Liberación de los Productos Software llevado a cabo en la UCI.

Métodos Empíricos:

Entrevistas: Constituyó un medio para el conocimiento de los problemas existentes al realizar el Proceso de Liberación de Productos Software en sus inicios, realizándose de forma manual, adquiriendo experiencias sobre el proceso para solucionar el problema de forma eficiente.

Encuestas: Constituyó un medio para conocer información referente a la usabilidad del sistema, es decir, detalles de cómo se espera que funcione el mismo.

Estructura del Documento

El documento está estructurado en 3 capítulos, que contiene todo lo relacionado con el trabajo realizado.

Capítulo 1: "Fundamentación Teórica", se abordan los principales conceptos a utilizar, se hace un estudio de los sistemas que automatizan la Planificación, Gestión y Control de aspectos relacionados con el Proceso de Liberación de Productos de Software y se hacen mención de las tecnologías, herramientas y metodologías utilizadas.

Capítulo 2: "Características del sistema", describe los procesos mediante un modelo del dominio, se identifican actores del sistema. Se definen además las funcionalidades del mismo, a través de los requerimientos funcionales y no funcionales, y se describen detalladamente. Se realiza la modelación de los casos de usos del sistema, así como una descripción detallada de cada uno de ellos.

Capítulo 3: "Análisis y Diseño del sistema", se explica todo lo relacionado con el análisis y diseño del sistema que se propone, además de los diagramas que se utilizan para su modelado, diagramas de clases del análisis, de interacción y el diseño de clases.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA.

1.1. Introducción.

Proporcionar software con calidad, es la premisa que debe garantizar toda EDS[4] que aspire incluirse en el mercado mundial, de ahí el esfuerzo mancomunado para alcanzarlo. Este indicador tiende a verse afectado, como consecuencia, entre otras cosas, por la gran demanda existente, devenida de la informatización masiva de empresas, instituciones a nivel mundial, también como consecuencia del alto grado de complejidad que exigen actualmente los sistemas informáticos, este último obstaculiza el proceso de desarrollo provocando grandes retrasos en la programación, aumento del tiempo y costos empleados para su confección. Es evidente la gran repercusión que tiene el indicador calidad, en el proceso de desarrollo del software, no por gusto este tema ha sido, por más de 30 años objeto de preocupación para especialistas, ingenieros, investigadores y comercializadores de software, pues son conscientes de cuán nefasto y perjudicial podría ser para las EDS no garantizarla.

Como solución a esta arraigada preocupante ya existen disímiles empresas u organizaciones certificadas mundialmente para comprobar la calidad de productos software a través del testing, cabe destacar que cada cual definen y estructuran este proceso de manera muy específica, dependiendo entre otras cosas, de las características propias de dichas empresas, entiéndase métodos de trabajo, recursos disponibles, técnicas de pruebas de software que aplican, por solo citar algunos. A pesar de la heterogeneidad planteada se puede establecer un punto común: es que la gran mayoría se apoyan en el uso de las TIC[5] para llevarlo a cabo, tal es el caso del uso evidente de herramientas automáticas en la ejecución de diferentes técnicas de pruebas, tales como: pruebas de funcionamiento, de stress o sobrecarga, etc.; sin embargo lo anterior no sustenta la afirmación de que el proceso se encuentre completamente automatizado. No se tiene conocimiento de que alguna lo haya conseguido, en caso contrario, tanto esta

[4] Empresas Desarrolladoras de Software.

[5] TIC: Es el acrónimo de: Tecnologías de la Información y las Comunicaciones. Conjunto de servicios, redes, software y dispositivos que tienen como fin la mejora de la calidad de vida de las personas dentro de un entorno, y que se integran a un sistema de información interconectado y complementario.

información, como la solución que crearon para conseguirlo son patrimonio propio de la empresa y por tanto confidencial.

El planteamiento anterior se corrobora con la siguiente afirmación:

En Uruguay, existe una empresa que se dedica a estas labores, el Centro de Ensayo de Software (CES), de la cual una de sus funcionarias Raquel Abella expresó "*Si bien está definida la forma de trabajo, no hay una herramienta única, para garantizar la ejecución del proceso de comprobación de la calidad del software, utilizamos varias, no obstante también hay trabajo manual con procedimientos predefinidos. Dentro de las herramientas que utilizamos, podemos mencionar un sistema que implementa el modelo Customer Relationship Management (CRM) para el seguimiento de los clientes y un repositorio común para la información de proyectos. También utilizamos un sistema de contabilidad y un sistema de registro de esfuerzo para el seguimiento de los mismos. Utilizamos además planillas de cálculo y documentos de texto.*"

De las razones anteriormente expuestas se desprenden las premisas de este trabajo, el cual está enfocado en la modelación de un sistema informático que permite la automatización del Proceso de Liberación de Productos de Software, definido por la Dirección Central de Calidad de la UCI, el mismo comprende entre otras cosas, aspectos claves como:

- La gestión de solicitudes.
- La gestión de la planificación de los recursos tangibles que intervienen durante el proceso.
- La gestión de imprescindibles documentos, que se generan durante su ejecución.

A pesar de no encontrarse publicado un sistema informático, confeccionado para garantizar la automatización del proceso de comprobación de la calidad a productos software, cabe destacar si existen varios de ellos que de cierta forma se relacionan específicamente con los aspectos anteriormente mencionados, comprendidos en el PLPS. A continuación se analizan algunos ejemplos de estas soluciones software, además se exponen las razones que determinan el por qué no constituyen una solución a aplicar en el LIPS.

1.2. Gestión de Solicitudes.

Actualmente existen Sistemas Gestores de Solicitudes que permiten al usuario realizar peticiones de solicitud de información, de cualquier naturaleza, ya sean de carácter particular o general, a su vez le informa el estado de gestión de su solicitud. Permiten además almacenar las solicitudes en la base de datos para el análisis de la existencia de solicitudes similares o previas sobre un mismo asunto. Ofrecen numerosas ventajas, donde se pueden mencionar como generales, las siguientes:

- Automatizan la gestión de solicitudes e incluso incidencias.
- Brindan información como soporte a la decisión.
- Mejora de la calidad del servicio prestado.
- Permiten la gestión y control de los cambios.
- Brindan indicadores de rendimiento, eficiencia, efectividad y mejora de la calidad, entre otras.

A continuación se exponen algunos Sistemas Gestores de Solicitud implementados en Cuba:

Sistema de Gestión de Solicitudes de Insumos del Área de Investigaciones del Centro de Inmunología Molecular.

Es un sistema gestor de solicitudes cubano, creado en la UCI, que brinda la posibilidad de realizar, por parte del Área de Investigaciones del Centro de Inmunología Molecular, solicitudes de insumos[6] a los fabricantes y proveedores, necesarios para la realización de experimentos e investigaciones, con el fin de obtener nuevos biofármacos exportables de marcada calidad internacional, para el tratamiento del cáncer y enfermedades del sistema inmune.

El objetivo fundamental de este sistema es corregir las limitaciones, inicialmente existentes en el proceso de solicitudes de insumos que se llevaba a cabo manualmente, como por ejemplo: esta solución software posibilita la centralización de la información, facilitando así el control de dichas solicitudes, permite

[6] Los insumos se conceptualizan como bienes y/o servicios que se incorporan al proceso productivo las unidades económicas y que, con el trabajo de los obreros, empleados y el apoyo de las máquinas, son transformados en otros bienes o servicios con un valor agregado mayor.

además conocer los estados en que se encuentran los pedidos realizados a los proveedores de una forma más fácil y evita cometer errores en cuanto a la nomenclatura de los insumos; es decir, permite controlar y manejar información con mayor eficiencia y rapidez involucrando mayor seguridad en la toma de decisiones y gestión de compra de insumos.

Dentro de sus principales características tecnológicas se pueden mencionar las siguientes:

- ✓ Sistema Gestor de Base de Datos: MySQL.
- ✓ Servidor de Aplicaciones: LAMP[7].
- ✓ Lenguaje en el lado del cliente: Java Script.
- ✓ Entorno de Desarrollo: Zend Studio.
- ✓ Metodología de Desarrollo: Rational Unified Process (RUP).
- ✓ Herramienta de Modelado: Rational Rose Enterprise Edition.

Sistema de Gestión de Solicitudes de Recursos Audiovisuales.

Este sistema también es cubano, fue desarrollado en la UCI, con el objetivo principal de: facilitar el trabajo de la Dirección de Comunicación Audiovisual (DCAV) de la propia universidad, en función de la producción de los recursos audiovisuales (videos, imágenes, sonido, locuciones, animaciones, digitalizaciones), que solicita el departamento de Software Educativo necesarios para el desarrollo de los proyectos multimedia que le asignan a la universidad; a través de la gestión eficiente y controlada de dichas solicitudes. Controlada porque el sistema permite que todas las operaciones realizadas a una solicitud queden registradas en un mecanismo de persistencia, y eficiente porque el mismo facilita todo el trabajo con la documentación de las mismas.

[7] El término LAMP se refiere a la combinación de las tecnologías:

- **L**inux (Sistema Operativo). **A**pache (Servidor Web). **M**ySQL (Servidor de Base de Datos). **P**HP (Lenguaje de Programación)

A continuación se hará mención a las principales características tecnológicas del sistema:

- ✓ Metodología de Desarrollo: RUP.
- ✓ Sistema Gestor de Base de Datos: MySQL.
- ✓ Herramienta de Modelado: Rational Rose Enterprise Edition.

Babel. Sistema Automatizado de Gestión de Solicitudes de Traducción e Interpretación.

Babel es un sistema de gestión de solicitudes cubano. Integra las tecnologías de la información a la gestión de solicitudes de los servicios de traducción e interpretación. Este sistema ofrece a sus usuarios la información precisa sobre el estado de su solicitud y además, las competencias del traductor, al aumentar el valor añadido de cada recurso que interviene en el proceso.

Esta herramienta de trabajo permite la organización, clasificación de la información y la recuperación de documentos con oportunas normas de seguridad.

Características tecnológicas del sistema de gestión de solicitudes Babel:

- ✓ Es una aplicación programada en PHP, propicia el acceso a bases de datos, envío de correo, creación de PDF.
- ✓ Para la gestión de los datos se utilizan las bases de datos relacionales de MySQL.

Babel v2.0 es una solución propuesta por la Facultad de Ingeniería Eléctrica de la Universidad Central de Las Villas para la Unidad de Traducción del Centro de Información.

Los objetivos perseguidos con este sistema:

- ✓ Realizar una aplicación capaz de administrar los diferentes flujos de trabajo que se generan en cada servicio solicitado.
- ✓ Lograr un sistema integrado de gestión vía Web en el que se conjugarán tres interfaces fundamentales: la de usuario, traductor y administrador.
- ✓ Permitir que la aplicación realice búsquedas dentro de los datos almacenados, de acuerdo a diferentes criterios de selección.

- ✓ Generar reportes para saber el estado actual de las solicitudes que se realizan.
- ✓ Incluir la opción que el usuario pueda ver y modificar sus datos y los documentos siempre que no hayan sido asignados a un traductor.

Este sistema aparece como una poderosa herramienta que integra y facilita la utilización de los servicios a través de la red. En la Intranet de ETECSA su uso provee un considerable ahorro de llamadas y transportación entre entidades geográficamente distantes y que requieran de los servicios de traducción e interpretación, aprovecha la infraestructura tecnológica — Intranet Corporativa de la Empresa— y, además, permite informatizar y optimizar el proceso de traducción.(BUSTELO RUESTA, Diciembre 2001)

Estos sistemas han sido confeccionados para dar solución a un determinado aspecto, con el fin de satisfacer las necesidades propias de una determinada empresa u organización, implementando los procesos de negocio concretos de dichas entidades, y como el sistema que se modela en este trabajo, precisa específicamente gestionar las Solicitudes de Pruebas: documento oficial que debe crear el Jefe del Proyecto, en el cual solicita legalmente al Jefe del LIPS el inicio del proceso de comprobación de la calidad de su(s) producto(s) software y las Solicitudes de Inclusión de Usuarios, planilla confeccionada por el Solicitante de Usuarios, en la cual queda plasmada la necesidad de incluir un determinado usuario al sistema; entonces las aplicaciones previamente analizadas no constituyen soluciones a considerar.

Además sus características tecnológicas nos son compatibles con las definidas, para el desarrollo del sistema que automatiza el PLPS, por ejemplo:

Tecnologías	**Sistemas previamente analizados**	**Sistema que automatiza el PLPS**
Sistema Gestor de Base de Datos	MySQL	PostgreSQL
Lenguaje de Programación	PHP	Java
Herramienta de Modelado	Rational Rose Enterprise Edition	Visual Paradigm Enterprise Edition

Tabla 1: Características Tecnológicas

1.3. Gestión de documentos.

Existen disímiles Sistemas de Gestión Electrónica de Documentos, los cuales controlan la producción, la circulación, el almacenamiento y recuperación de cualquier tipo de información. Aspiran a administrar y controlar de modo conjunto mediante el auxilio de la informática, todo tipo de datos, documentos y conocimientos existentes en las organizaciones en que se aplican.

Específicamente los Especialistas del Grupo de Liberación y Pruebas de Software, durante la ejecución del PLPS, necesitan gestionar documentos de vital importancia, como son aquellos en los que se plasman las Técnicas de Pruebas, los Casos de Pruebas, Listas de Chequeos, entre otros, a aplicar durante el proceso, así como los que registran las No Conformidades a medida que se comprueba la calidad de un producto software.

A continuación se analiza una herramienta que automatiza la gestión de las No Conformidades (NC):

Sistema para la Gestión de las NC durante las pruebas de calidad

Este sistema creado en la UCI, posee un módulo de Informes de NC, es una herramienta directa que le permite gestionar eficazmente la información de las NC que se generan durante el proceso de prueba de un producto software, perteneciente a un proyecto de la universidad. A través del cual se podrá crear, modificar y eliminar los informes de NC, así como la notificación por correo al responsable de resolverlas (Equipo de Desarrollo), de la existencia de NC asignadas y por ende pendientes de solución.

A continuación se exponen un conjunto de características tecnológicas que se emplearon en el desarrollo del sistema:

- ✓ Entorno de Desarrollo Integrado: Microsoft Visual Studio 2005.
- ✓ Lenguaje de Programación: C#.
- ✓ Sistema Gestor de Base de Datos: SQL Server 2000.

En la actualidad una de las premisas vigentes en las EDS cubanas es la utilización de herramientas certificadas “Open Source”, es decir, no propietarias, como pueden observar para el desarrollo del sistema

en análisis se utilizaron un conjunto de tecnologías que entran dentro del calificativo propietarias, por tanto tampoco constituye una solución a considerar.

1.4. Gestión de la Planificación de Recursos.

El éxito de cualquier actividad en la vida humana depende en gran medida de la planificación que se haga. Además una estimación acertada de una actividad ayuda a valorar, evaluar y prever el tiempo que se necesita para la misma.

Existen algunas herramientas informáticas para la planificación de recursos: humanos y tecnológicos. A continuación se mencionará una que se desarrolló en la UCI: “SoftLaQ”, en el año 2007, como uno de sus objetivos principales estaba: automatizar la planificación de los recursos tangibles (computadoras, personas y tiempo) que intervenían en las pruebas de software que se aplicaban en aquel entonces en el Laboratorio de Calidad de Software, hoy LIPS.

Desafortunadamente esta herramienta se encuentra en estado de caduquez producto a la evolución que ha experimentado el proceso de comprobación de la calidad de los productos software hoy PLPS, por lo tanto no constituye una solución a considerar.

Teniendo en cuenta las descripciones de los Sistemas Informáticos anteriormente abordados se arriban a las siguientes conclusiones: a pesar de que cumplen conceptualmente con algunas actividades que debe garantizar el sistema que se modela para automatizar el Proceso de Liberación de Productos de Software, ninguno puede ser implementado en el Laboratorio Industrial de Pruebas de Software, por las razones anteriormente expuestas que afirman el por qué no son soluciones a considerar; no obstante aún asumiendo que dichas soluciones automatizan tanto la gestión de solicitudes, como la gestión de documentos y la planificación de recursos, que intervienen específicamente en el PLPS, aún así serían entonces 3 herramientas que automatizan el proceso, lo cual no constituye una solución óptima, lo que se pretende es alcanzar una buena solución, a través del desarrollo de una herramienta que automatice completamente el mismo. Por lo tanto será objetivo de los próximos acápites abordar las tecnologías que se aplicarán durante el Proceso de Desarrollo de dicho Software, definiendo tanto las Metodologías, como el Ambiente de Desarrollo.

Proceso de Desarrollo de Software.

Con el objetivo de crear y mantener las soluciones de software de calidad, aplicando las tecnologías y prácticas computacionales, surge la Ingeniería de Software. El desarrollo y evolución constante que han tenido los procesos de esta materia han traído consigo la realización de varias tareas en este campo, como son: análisis de requisitos, especificación, diseño y arquitectura, programación, prueba, documentación y mantenimiento.(JACOBSON, 1998)

El proceso de desarrollo del software define el conjunto de actividades precisas para convertir los requisitos de los usuarios en el conjunto seguro y resistente de artefactos que componen un producto software.

Su desarrollo es un proceso exigente, por lo tanto, la búsqueda de una metodología adecuada, con el fin de hacerlo más predecible y eficiente, es una tarea de vital importancia, por tal motivo, se destina parte de este capítulo a hacer un estudio breve, pero no menos profundo de algunos aspectos que comprenden las Metodologías de Desarrollo, con el objetivo de garantizar lo anteriormente plasmado.

1.5. Metodologías de desarrollo de Software.

1.5.1. Conceptos.

Las metodologías de desarrollo de software son un conjunto de procedimientos y técnicas para el desarrollo de productos software. Este conjunto de procedimientos tienen la responsabilidad de definir cuáles serán las tareas a realizar.

Los desarrolladores a su vez, utilizarán una o varias técnicas, las cuales indican cómo deben ser realizadas estas tareas, apoyándose en las herramientas de software que automatizarán su aplicación, obteniéndose como resultado el producto deseado. Una metodología puede seguir uno o varios modelos de ciclo de vida, los cuales indican qué obtener a lo largo del desarrollo del proyecto, pero no cómo hacerlo (*Ver Figura 4).*

Una metodología debe definir con precisión cuáles van a ser los artefactos[8], roles y actividades involucradas, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo.

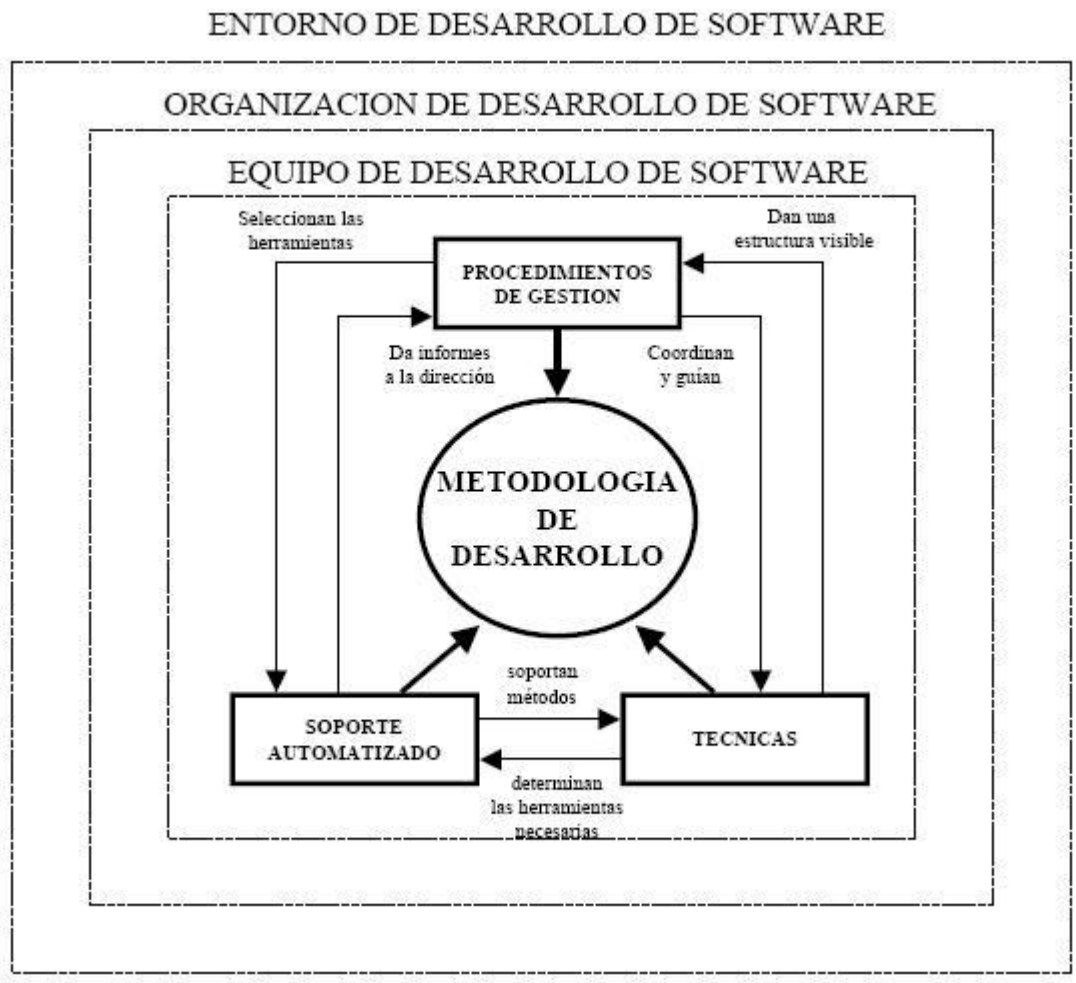


Figura 4: Impacto de las metodologías en el entorno de desarrollo

[8] Un artefacto es una pieza de información que es producida, modificada o usada por el proceso de desarrollo, define un área de responsabilidad para un rol y está sujeta a control de versiones. Un artefacto puede ser un modelo, parte de él o un documento.

Opciones a tener en cuenta para implantar una metodología en los entornos de desarrollo:

- Bien puede crear una metodología que se adecue al entorno o proceso de desarrollo del software, estableciendo Procedimientos de Gestión (PG[9]), Técnicas de Desarrollo (TD[10]) y Soporte Automatizado[11].

  ó

- Analizar y evaluar las metodologías existentes y seleccionar la que más se adapte a las necesidades del equipo de desarrollo.

En este trabajo no se va a tener en cuenta la primera opción, ya que la misma requiere de estudios avanzados del tema, y ese no es el objetivo de la investigación, por tanto se analizarán y clasificarán las metodologías para poder seleccionar la adecuada.

1.5.2. Clasificación.

Debido a la diversidad de propuestas y diferencias en cuanto al grado de detalle, información disponible y el alcance de cada una de las metodologías, la clasificación de las mismas se hace una tarea compleja. A grandes rasgos, si se toman como criterios las notaciones utilizadas para especificar los artefactos producidos por las distintas actividades que comprende el proceso de desarrollo de software se puede clasificar las metodologías en dos grupos: Metodologías Estructuradas y Metodologías Orientadas a Objetos. (PRESSMAN, 1997), (LETELIER, 2004)

Por otra parte, según la filosofía de desarrollo, aquellas metodologías con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, reciben el apelativo de Metodologías Tradicionales (o peyorativamente denominada Metodologías Pesadas). Otras metodologías, denominadas Metodologías Ágiles, están más orientadas a la generación de código con

[9] Los Procedimientos de Gestión definen la forma de llevar a cabo las actividades fundamentales del proceso de desarrollo del software. Además constituye el vínculo de comunicación entre usuarios y desarrolladores.

[10] Las Técnicas de Desarrollo se utilizan para aplicar un Procedimiento de Gestión, pueden ser gráficas y/o textuales, además determinan el formato de los Productos resultantes en cada Tarea.

[11] Conjunto de herramientas automatizadas que se aplicarán durante la creación del software.

ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen especial hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso.(CANÓS, 2004)

A continuación se hará énfasis en estas dos últimas clasificaciones, actualmente las más empleadas, realizando una comparación entre ambas, aspecto importante a tener en cuenta, durante la decisión de qué tipo metodología utilizar para el desarrollo del Sistema que ocupa este trabajo.

Metodologías Tradicionales

Las Metodologías Tradicionales son aquellas que están guiadas por una fuerte planificación, tratando de establecer procedimientos y documentar todo el proceso de desarrollo para minimizar el riesgo de cada proyecto y controlar su evolución.

Abordan estos problemas proponiendo comenzar con una fase de análisis y diseño, en la que se tomen todas las decisiones, previa al comienzo del desarrollo. Finalizada esta fase será el momento de comenzar con la implementación, que debe finalizar con una etapa de pruebas que asegure la calidad antes de implantar el sistema en producción.

Metodologías Ágiles

Por otro lado en el 2001 fue creado el Manifiesto Ágil, documento que establece las ideas principales de las metodologías ágiles:

- Los individuos y las interacciones entre ellos son más importantes que las herramientas y los procesos empleados.
- Es más importante crear un producto software que funcione que escribir documentación exhaustiva.
- La colaboración con el cliente debe prevalecer sobre la negociación de contratos.
- La capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan.

Estas metodologías ponen de relevancia que, la capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan, siendo esta flexibilidad para muchos clientes una

ventaja competitiva. Proponen además un proceso de desarrollo de software *incremental*, es decir, se entregan pequeñas partes de software con ciclos rápidos, *cooperativo* donde clientes y desarrolladores trabajan juntos con una cercana comunicación, *sencillo,* pues el método en si mismo es fácil de aprender y modificar, bien documentado.

Metodologías tradicionales vs. Metodologías ágiles

La tabla 2 expone las diferencias más relevantes que se manifiestan entre ambas metodologías, no solo desde el punto de vista del proceso en sí, sino también como el contexto de equipo y la organización más favorables a cada una de estas filosofías de desarrollo de software.

Metodologías Tradicionales	**Metodologías Ágiles**
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.	Basadas en heurísticas provenientes de prácticas de producción de código.
Se espera que no ocurran cambios de gran impacto durante el proyecto.	Preparados de manera muy especial para cambios durante el proyecto.
Impuestas externamente.	Impuestas internamente (por el equipo).
Proceso mucho más controlado, con numerosas políticas/normas.	Proceso menos controlado, con pocos principios.
Más artefactos. El modelado es esencial, mantenimiento de modelos.	Pocos artefactos. El modelado es prescindible, modelos desechables.
Más roles, más específico.	Pocos roles, más genéricos y flexibles.
Existe un contrato prefijado.	No existe un contrato tradicional, debe ser bastante flexible.

El cliente interactúa con el equipo de desarrollo mediante reuniones.	Cliente es parte del equipo de desarrollo.
Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas/usadas en proyectos grandes y con equipos posiblemente dispersos.	Orientada a proyectos pequeños. Corta duración (o entregas frecuentes), equipos pequeños (< 10 integrantes) y trabajando en el mismo sitio.
Se promueve que la arquitectura se defina tempranamente en el proyecto.	La arquitectura se va definiendo y mejorando a lo largo del proyecto.
Énfasis en la definición del proceso: roles, actividades y artefactos.	Énfasis en los aspectos humanos: el individuo y el trabajo en equipo.

Tabla 2: Diferencias entre metodologías ágiles y tradicionales.(LETELIER, 2004)

¿Qué metodología debo usar para el desarrollo del Sistema Computacional?

Teniendo en cuenta las clasificaciones previamente tratadas, se concluye que para el desarrollo de este software se precisa de una metodología tradicional, por el simple hecho de que son sumamente eficientes en proyectos de gran envergadura y tamaño (respecto a tiempo y recursos), este calificativo comprende por tanto documentar todo el proceso de desarrollo.

Como Metodología tradicional se propone utilizar Rational Unified Process (RUP), pues aplica varias de las mejores prácticas en el desarrollo moderno de software adaptándose a un amplio rango de proyectos y de organizaciones.

Provee a cada miembro del equipo, un fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas del desarrollo de software. Esta metodología permite que todos los integrantes de un equipo de trabajo, conozcan y compartan el proceso de desarrollo, una base de conocimientos y los distintos modelos de cómo desarrollar el software utilizando un lenguaje de modelado común: UML.(WESLEY, 2004)

Provee un enfoque estructurado para realizar tareas y responsabilidades en una organización de desarrollo. Su principal objetivo es asegurar la producción de software de alta calidad, que cumpla las necesidades de sus usuarios finales, que sea realizado en las fechas acordadas y con el presupuesto disponible.

Detalle de la metodología de desarrollo propuesta.

RUP por su popularidad se ha convertido en la metodología estándar “de facto” para el desarrollo de proyectos de suma importancia. Es un proceso iterativo e incremental, lo que facilita que sea un proceso planificado y gestionado que:

- Se adapta a los cambios de los requerimientos con pocas alteraciones.
- Involucra al usuario/cliente durante el proceso.
- Permite detectar y gestionar los riesgos durante todo el ciclo de vida del proyecto.
- Se basa en la construcción de prototipos ejecutables.

Además de las características propias de los procesos iterativos se caracteriza por:

- Está dirigido durante el proceso por los Casos de Uso (que modelan las interacciones entre el usuario y el sistema).
- Se enfoca en la arquitectura de los sistemas a construir (centrado en la arquitectura).
- Nos proporciona una guía efectiva de cómo utilizar el Lenguaje Unificado de Modelado (UML).
- Es un proceso configurable: RUP es una metodología muy extensa y en la mayoría de los casos en el momento de su implantación se considera un proceso costoso para la cantidad de entregables y actividades que se definen, pero hay que tener en cuenta, una de las principales funcionalidades que ofrece: no es obligatorio hacer uso de todas las actividades y entregables definidos, sino que se puede configurar el proceso, con el fin de adaptarlo a aquellas partes que se consideran necesarias.

El proceso RUP se repite en una serie de ciclos. Cada ciclo concluye con una versión del producto (release[12]) y cada ciclo está dividido por 4 fases: Inicio (Concepción), Elaboración, Construcción y Transición. Cada una de las fases está dividida a su vez por iteraciones, y en cada una se realizan 5 procesos o flujos de trabajo principales: requerimientos, análisis, diseño, implementación, pruebas y entrega o preparación del release.

RUP define por lo tanto el proceso mediante dos dimensiones:

- Dinámica o en el tiempo. Se expresa en términos de ciclos, fases, iteraciones y fechas límites o hitos.
- Estática. Se describe en términos de actividades[13], entregables (o artefactos[14]), workflow[15] (o flujos de trabajo) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

Si bien las metodologías son de vital importancia en el proceso de desarrollo del software, no menos imprescindible es el Ambiente de Desarrollo (Development Environment).

1.6. Ambiente de Desarrollo

Es donde se definen el conjunto de herramientas y tecnologías, las versiones a utilizar para diseñar, desarrollar, depurar e implementar un software.

A continuación se presentarán el conjunto de herramientas candidatas para la confección del software que ocupa este trabajo, como son un Ambiente de Desarrollo Integrado (Integrated Development Environment-IDE), las herramientas de modelado, sistema gestor de base de datos, etc.

12 Release: Versión del producto.

13 Actividades: Tareas reales que se tienen que realizar durante el proceso (cómo).

14 Artefactos: Son el resultado de las actividades e incluyen modelos, documentos, códigos fuente, etc. (qué).

15 Workflow: definen la secuencia de actividades que se tienen que realizar en un punto del proceso (cuándo).

1.6.1. Ambiente de Desarrollo Integrado.

Un Ambiente de Desarrollo Integrado es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI[16]). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Estos proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, etc.

En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto, como es el caso de Smalltalk.

Existen varios IDEs en la actualidad entre los cuáles se pueden mencionar C++Builder, Eclipse, Netbeans, Visual Studio, etc.

Para el proceso de codificación del sistema será utilizado Netbeans 6.0 por ser un producto de código abierto, proporciona herramientas para la construcción de todos los componentes J2EE, incluidas páginas web, servlets, servicios web, necesarios para la confección de aplicaciones web profesionales, provee igualmente soporte para framework web como Spring e Hibernate.

1.6.2. Sistemas Gestores de Bases de Datos (SGBD)

Debido a la necesidad de manipular datos persistentes y de larga durabilidad en el sistema se hace necesario la aplicación de un SGBD, a través del cual sea posible recuperar y almacenar información, interpretable y útil para el usuario, con facilidad y fiabilidad, de ahí que se ha convertido en el instrumento o soporte básico de mayor despliegue en la actualidad para la gestión de los sistemas informáticos por todo tipo de empresas, independientemente de su tamaño.

[16] Interfaz Gráfica de Usuario: es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático.

Actualmente existe una amplia gama de SGBD con características propias, no obstante todos deben tener en cuenta los siguientes aspectos de manera general: abstracción de la información, la independencia de los datos, redundancia mínima, consistencia en los datos, seguridad, integridad, respaldo y recuperación, tiempo de respuesta y control de concurrencia.

Existen SGBD muy potentes tanto en la clasificación de software propietario como software libre. Como privativos se encuentran: Oracle y Microsoft SQL Server que lideran el mercado por sus altas prestaciones, mientras que como opción libre, MySQL, gestor muy utilizado en la web por su simplicidad de uso, y PostgreSQL que si bien no soporta alguno de los aspectos más avanzados si representa una opción muy confiable y comprometedora.

PostgreSQL

Para el desarrollo de este sistema se decidió utilizar PostgreSQL entre otras cosas porque permite la aproximación de los datos a un modelo objetual-relacional, siendo capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas y herencia. Es altamente extensible: PostgreSQL soporta operadores, funciones, métodos de acceso y tipos de datos definidos por los usuarios, además soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92. Tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, Java, o TCL como lenguaje procedural embebido.

Posee un sistema denominado Control de Concurrencia Multi-Versión (MVCC- por sus siglas en inglés) utilizado para evitar bloqueos innecesarios entre acciones de lecturas para acceder a información de la Base de Datos y acciones de escrituras para actualizar los registros.(DOMINICANO, 2009)

1.6.3. Framework

Actualmente la reutilización de software en el proceso de desarrollo de aplicaciones juega un papel clave en temas como son: la productividad, la capacidad de mantenimiento, la portabilidad y la calidad. Por ello, la reutilización debe aplicarse a cada etapa del ciclo de vida (análisis, diseño, codificación, pruebas, y

mantenimiento) [BASS87]. De lo contrario, no se recogerán todos los beneficios potenciales que brinda. Por tal motivo, para el proceso de desarrollo del futuro sistema se hará uso de varios frameworks, debido a que la mayoría de estos sistemas han sido confeccionados para satisfacer dicho aspecto.

Recientemente, el interés en reutilizar software ha sido cambiado de la reutilización de componentes[17] simples por la reutilización de sistemas enteros o estructuras de aplicaciones. Un sistema software que pudiera ser reutilizado en este nivel para la creación de aplicaciones completas es llamado framework. Los framework son basados en la idea que deberían permitir la producción fácil de un conjunto de sistemas específicos pero similares, dentro de un cierto dominio, comenzando desde una estructura genérica. Son arquitecturas genéricas integradas por un extensible conjunto de componentes. Además, pueden contener subframeworks los cuales representen subconjuntos de componentes de un sistema más grande. (KAISER 2005)

Estos enfatizan en la reutilización del diseño sobre el código, aunque el código resultante puede ser reutilizado. Usualmente definen la estructura total de todas las aplicaciones derivadas de él, una organización entre clases y objetos, principales responsabilidades de clases individuales, cómo ellos colaboran y el hilo de control de éstas. El desarrollador es el responsable para personalizar el framework para una aplicación en particular. (KAISER 2005)

A continuación se analizarán los selectos:

Hibernate

Se empleará Hibernate por ser un potente framework de mapeo objeto/relacional y servicio de consultas para Java. Es la solución ORM (Object-Relational Mapping) más popular en el mundo Java. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada podrán generarse Bases de Datos (BBDD) en cualquiera de los entornos soportados: PostgreSQL, MySQL, etcétera.

[17] Un componente es un bloque de construcción reusable de software: una pieza de código encapsulada de una aplicación que puede ser combinada con otros componentes y con códigos adicionales para producir una aplicación específica.

Dentro de sus principales características se pueden mencionar las siguientes:

- Permite expresar consultas utilizando SQL nativo o consultas basadas en criterios.
- Soporta todos los sistemas gestores de bases de datos SQL y se integra de manera elegante y sin restricciones con los más populares servidores de aplicaciones J2EE y contenedores web, y por supuesto también puede utilizarse en aplicaciones de escritorio.
- Puede operar proporcionando persistencia de una manera transparente para el desarrollador.
- Soporta el paradigma de orientación a objetos de una manera natural: herencia, polimorfismo, composición y el framework de colecciones de Java.
- Soporte para modelos de objetos con una granularidad muy fina. Permite una gran variedad de mapeos para colecciones y objetos dependientes.
- Presenta un potente mecanismo de transacciones de aplicación llegando incluso a permitir las interacciones largas (aquellas que requieren la interacción con el usuario durante su ejecución).

Spring Framework

Spring Framework porque es una aplicación de código abierto que ayuda a hacer el desarrollo en J2EE mucho más fácil. Ayuda a estructurar aplicaciones completas en una manera consistente y productiva para crear arquitecturas coherentes.(ROD JOHNNSON, 2005)

Es el más popular y el más ambicioso de todos los framework de peso ligero. Es el único framework que interviene en todas las capas arquitectónicas de una aplicación J2EE. Además está diseñado para facilitar una flexibilidad arquitectónica. (ROD JOHNNSON, 2005)

Posee además disimiles módulos que facilitan el desarrollo de una aplicación en Java/J2EE, de los cuales como principales se pueden mencionar (ROD JOHNNSON, 2005):

- *Contenedor de Inversión de Control*: permite una potente gestión de configuración para POJOs[18] y trabaja con otras partes de Spring para proveer servicios como la administración de la configuración.

- *Un framework para la Programación Orientada a Aspectos (AOP)*: permite comportamientos que deberían ser esparcidos de otra manera a través de diferentes métodos para ser modularizados en un simple lugar. Spring usa la AOP para implementar importantes servicios tales como administración de transacciones declarativas, y además es usado para implementar códigos estándar que debería de otra manera ser esparcido entre las clases de la aplicación.

- *Una abstracción de acceso a datos:* Spring anima a un consistente acercamiento arquitectónico para acceso a datos, y posee una abstracción única y poderosa para implementarlo. Presenta una rica jerarquía de excepciones de acceso a datos, independiente de cualquier framework de persistencia en particular.

- *Simplificación de JDBC*[19] : presenta una capa de abstracción sobre JDBC que es significantemente mucho más simple y menos propensa a errores para usar que JDBC puro cuando se necesita usar acceso a través de SQL a bases de datos relacionales.

- *Administración de transacciones*: presenta una abstracción de transacciones que puede mover transacciones globales JTA (manipuladas por un servidor de aplicaciones) o transacciones locales usando JDBC, Hibernate, JDO u otro API de acceso a datos. Estas abstracciones presentan un consistente modelo de programación en una variedad de ambientes y es la base de la administración de transacciones programáticas y declarativas usadas por Spring.

- *Framework Web MVC*: Spring presenta un framework web MVC basado en peticiones. Éste tiene un acercamiento al framework de Struts pero el framework web de Spring es más flexible, y se

[18] POJO: Es el acrónimo de Plain Old Java Object. Clase del lenguaje de programación Java. Este nombre se le da a las clases que no son de algún tipo especial (como JavaBeans y EJB) y no cumplen ningún otro rol, ni implementan alguna interfaz especial.

[19] JDBC: Es el acrónimo de Java Database Connectivity, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema de operación donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

integra discretamente al contenedor de Inversión de control de Spring. Todos los demás rasgos de Spring se pueden usar con otros frameworks tales como Struts, JSF, WebWork, Tapestry, etcétera.

- *Soporte para Servicio de Mensajería de Java (Java Message Service, JMS)*: Spring presenta soporte para enviar y recibir mensajes de una forma mucho más simple que la que provee la especificación J2EE.

- *Soporte para Java Management Extension (JMX)*: Spring presenta soporte para la administración JMX de objetos de una aplicación para ser configurados.

- *Soporte para comprensivas estrategias de pruebas para desarrolladores de aplicación*: Spring no solamente ayuda a realizar un buen diseño, permitiendo efectivas pruebas de unidad, sino que presenta una comprensiva solución para pruebas de integración fuera de un servidor de aplicaciones.

Además es poseedor de un cúmulo de valores tecnológicos, los cuales facilitan su elección como herramienta para el desarrollo de otras aplicaciones y por consiguiente se hacen mención a las elementales:

- *Provee un consistente modelo de programación, útil en cualquier ambiente*: Muchas aplicaciones web simplemente no necesitan ser ejecutadas sobre un pesado servidor de aplicaciones, sino que es mejor ejecutarlas sobre un servidor web, como, Tomcat o Jetty. También es importante recordar que no todas las aplicaciones se ejecutan del lado del servidor. Spring provee un modelo de programación que aísla el código de la aplicación de los detalles del ambiente, logrando que el código sea menos dependiente del contexto de ejecución.(ROD JOHNNSON, 2005)

- *Facilita el diseño Orientado a Objetos (OO) en aplicaciones J2EE*: Se debería preguntar: ¿Cómo una aplicación J2EE, escrita en Java – un lenguaje OO – no es OO? En realidad muchas aplicaciones J2EE no merecen el nombre OO. Con Spring es más fácil eliminar los impedimentos del diseño OO en aplicaciones J2EE tradicionales haciendo el código más coherente, con bajo acoplamiento y más reusable.(ROD JOHNNSON, 2005)

- *No es un framework agresivo*: Este es el principal problema de los framework anteriores. Mientras que los framework tradicionales tales como EJB[20] o Apache Avalon que forzaban al código de las aplicaciones a ser dependientes del framework, implementando interfaces específicas de estos framework o heredando clases específicas de estos; ayuda a reducir las dependencias del código de la aplicación sobre el framework.

- *Facilita buenas prácticas de programación, tales como la programación a interfaces*: El uso de un contenedor de Inversión de Control reduce grandemente la complejidad del código a interfaces, más que a clases. El uso de los objetos a través de estas interfaces protege los requerimientos, los cuales pudieran cambiar en el desarrollo de la aplicación.

- *Promueve la selección arquitectónica*: Mientras Spring provee una columna vertebral arquitectónica, Spring apunta para facilitar el reemplazo de cada capa. Por ejemplo, con una capa media de Spring, se pudiera ser capaz de cambiar de un framework de mapeo objeto relacional (ORM) a otro con un impacto mínimo sobre el código de la lógica de negocio, o cambiar de Struts a Spring MVC o WebWork sin algún impacto en la capa media.

1.6.4. Herramientas para el modelado

El Lenguaje Unificado de Modelado (UML) constituye el lenguaje de modelado de sistemas de software más utilizado y conocido en la actualidad, además de ser compatible con RUP. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Por ser un *lenguaje* necesita servirse de una herramienta de modelado, la cual puede ser usada para capturar, guardar, integrar e incluso rechazar automáticamente información, y diseño de documentación de un sistema.

[20] EJB: Es el acrónimo de Enterprise Java Beans, un API que proporciona un modelo de componentes distribuido estándar del lado del servidor. Su objetivo es dotar al programador de un modelo que le permita centrarse en el desarrollo de la lógica de negocio en sí. El hecho de estar basado en componentes permite que éstos sean flexibles y sobre todo reutilizables.

Una de las ventajas que ofrece este lenguaje es que hace más fácil escoger dicha herramienta, antes el modelador primero tenía que seleccionar una notación de metodología, y después estaba limitado a una herramienta que la soportara. Ahora con UML como estándar, la elección de notación ya se ha hecho para el modelador. Y con todas las herramientas de modelado soportando UML, el modelador puede seleccionar la herramienta basada en las áreas claves de funcionalidad soportadas, que permiten resolver los problemas y documentar las soluciones. (LARMAN, 2002)

Como una buena caja de herramientas, una para el modelado de sistema debe ofrecer los instrumentos necesarios para hacer eficientemente varios trabajos, sin privarte nunca de la correcta. Dentro de la estructura de diseño de sistemas, esto incluye en sentido general lo siguiente:

- Soporte para toda la notación y semántica de UML.
- Soporte para una cantidad considerable de técnicas de modelado y diagramas para complementar UML - incluyendo modelado de datos, diagramas de flujo, y diseño de pantallas de usuario. Posibilidad de reutilizar información obtenida por otras técnicas todavía usadas, como modelado tradicional de procesos.
- Permitir a varios equipos de analistas trabajar en los mismos datos a la vez.
- Posibilidad de capturar los requisitos, asociarlos con elementos de modelado que los satisfagan y localizar cómo han sido satisfechos los requisitos en cada uno de los pasos del desarrollo.
- Posibilitar la creación de informes y documentación personalizados en sus diseños, y la salida de estos informes en varios formatos, incluyendo HTML para la distribución en la Internet o Intranet local.
- Posibilidad para generar y ’reverse’ código (por ejemplo C++, Java, etc.) para facilitar el análisis y diseño ’iterativo’, para volver a usar código o librerías de clase existentes, y para documentar el código.

Las Herramientas CASE[21] (Computer Aided Software Engineering) es una de las tantas herramientas de modelado UML, las cuales tienen como objetivo aumentar la productividad del proceso de desarrollo reduciendo los costes del mismo en términos de tiempo y dinero.

Herramientas CASE para modelado UML.

Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

La herramienta CASE que se utiliza durante el desarrollo del sistema es Visual Paradigm UML Enterprise Edition, pues la misma funciona sobre múltiples plataformas, es un producto de muy buena calidad, al igual que las imágenes y reportes generados, soporta aplicaciones web, es fácil de instalar y actualizar y es compatible con otras ediciones. Además ofrece un entorno de creación de diagramas para UML 2.0, que es el lenguaje de modelado que se va a utilizar, 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 y generación de código para Java principalmente y exportación de los proyectos como HTML.

1.7. Patrones de Diseño.

Los patrones de diseño describen un problema que ocurre en repetidas ocasiones en algún contexto determinado del desarrollo de software, y a su vez define una buena solución, que puede ser reusable, es decir, constituye una solución efectiva para diferentes problemas de diseño en distintas circunstancias. Esto ayuda a diseñar correctamente en menos tiempo, evitando los errores que tan comúnmente se atribuyen a dicha etapa.

[21] CASE: sus siglas en español significan Ingeniería de Software Asistida por Ordenador.

Para construir aplicaciones J2EE, de un modo profesional, se deben conocer técnicas avanzadas de análisis, diseño y desarrollo, ante tal necesidad la Sun MicroSystems[22] propuso una serie de patrones con el objetivo de facilitar la construcción de dichas aplicaciones.

1.7.1. Patrones J2EE.

Como se puede apreciar con la aparición del J2EE, todo un nuevo catálogo de patrones de diseño apareció. Desde que J2EE es una arquitectura por si misma que involucra otras arquitecturas, incluyendo Servlets, Java Server Pages (JSP), Enterprise JavaBeans, y más, merece su propio conjunto de patrones específicos para diferentes aplicaciones empresariales.

Catálogo de Patrones de Diseño J2EE para Aplicaciones Web.(DEEPAK ALUR, 2003)

- **Intercepting Filter**

  Son objetos que se crean entre el cliente y los componentes Web para procesar servicios comunes de una forma estándar sin requerir cambios en el código principal del procesamiento de la petición. Los filtros interceptan las peticiones entrantes y las respuestas salientes, permitiendo un pre y post-procesamiento. Se pueden añadir y eliminar estos filtros a discreción, sin necesitar cambios en nuestro código existente.

- **Front Controller**

  El mecanismo de manejo de peticiones de la capa de presentación debe controlar y coordinar el procesamiento de todos los usuarios a través de varias peticiones. Dichos mecanismos de control se pueden manejar de una forma centralizada o descentralizada. El sistema requiere entonces un punto de acceso centralizado para que el manejo de las peticiones soporte la integración de los servicios del sistema, recuperación de contenidos, control de vistas, y navegación, lo anterior se logra con la implementación de los controladores. El controlador además de manejar el control de peticiones, incluye la invocación de los servicios de seguridad como la autentificación y

[22] Sun Microsystems es una empresa informática de Silicon Valley, dedicada a la fabricación de semiconductores y software, constituida en 1982 por el alemán Andreas von Bechtolsheim y los norteamericanos Vinod Khosla, Bill Joy, Scott McNealy y Marcel Newman.

autorización, delega el procesamiento de negocio, controla la elección de una vista apropiada, el manejo de errores, y el control de la selección de estrategias de creación de contenido.

- **View Helper**

  Un objeto helper que encapsula la lógica de acceso a datos en beneficio de los componentes de la presentación. Por ejemplo, los JavaBeans pueden ser usados como patrón View Helper para las páginas JSP.

- **Composite View**

  Utilizar vistas compuestas que se componen de varias subvistas atómicas. Cada componente de la plantilla se podría incluir dinámicamente dentro del total y la distribución de la página se maneja independientemente del contenido. Esta solución promueve la creación de una vista compuesta basada en la inclusión y sustitución de fragmentos de plantilla modulares tanto estáticos como dinámicos. También promueve la reutilización de porciones atómicas de la vista asegurando un diseño modular. Es apropiado utilizar este patrón para generar páginas que muestran componentes que podrían combinarse en una gran variedad de formas. Este escenario ocurre, por ejemplo, con sites de portal que incluyen numerosas subvistas independientes, como noticias, información del tiempo, y valores de stocks en una sola página. La distribución de la página se maneja y modifica de forma independiente al contenido de las subvistas. Otro beneficio de este patrón es que los diseñadores Web pueden hacer un prototipo de la distribución de la site, conectando contenido estático en todas las regiones de la plantilla. Según va progresando el desarrollo de la site, ese contenido estático se puede sustituir por el contenido dinámico. Por ejemplo, una página JSP que incluye otras páginas JSP y HTML usando la directiva include o el action include es un patrón Composite View.

- **Service To Worker**

  Es como el patrón de diseño MVC con el Controlador actuando como Front Controller pero con una cosa importante: aquí el Dispatcher (el cual es parte del Front Controller) usa View Helpers a gran escala y ayuda en el manejo de la vista.

- **Dispatcher View**

  Es como el patrón de diseño MVC con el controlador actuando como Front Controller pero con un asunto importante: aquí el Dispatcher (el cual es parte del Front Controller) no usa View Helpers y realiza muy poco trabajo en el manejo de la vista. El manejo de la vista es manejado por los mismos componentes de la Vista.

- **Busisness Delegate**

  El Business Delegate actúa como una abstracción de negocio del lado del cliente; proporciona una abstracción para la implementación de los servicios del negocio. Utilizando Business Delegate se reduce el acoplamiento entre los clientes de la capa de presentación y los servicios de negocio del sistema. Dependiendo de la estrategia de implementación, Business Delegate podría aislar a los clientes de la posible volatilidad en la implementación del API de los servicios de negocio. Potencialmente, esto reduce el número de cambios que se deben hacer en el código de la capa de presentación cuando cambie el API del servicio de negocio o su implementación subyacente.

- **Session Facade**

  Session Facade abstrae las interacciones de los objetos de negocio y proporciona una capa de servicio que expone sólo las interfaces requeridas. Así, oculta a la vista de los clientes las interacciones complejas entre los participantes. El Session Facade controla las interacciones entre los datos de negocio y los objetos de servicio de negocio que participan en el flujo de trabajo, y encapsula la lógica de negocio asociada con los requerimientos, así el bean de sesión (representando al Session Facade) maneja las relaciones entre los objetos de negocio. El bean de sesión también maneja el ciclo de vida de los participantes creándolos, localizándolos, modificándolos y borrándolos según lo requiera el flujo de trabajo.

  Es importante examinar las relaciones entre los objetos de negocio. Algunas de estas relaciones son temporales, lo que significa que la relación es aplicable sólo a esa interacción o escenario. Otras relaciones podrían ser más permanentes. Las relaciones temporales se modelan mejor como flujos de trabajo en una fachada, donde la fachada maneja las relaciones entre los objetos de

negocio. Las relaciones permanentes entre dos objetos de negocio deberían estudiarse para determinar qué objeto de negocio mantiene la relación.

- **Transfer Object**

  Los clientes normalmente en las peticiones solicitan más de un valor a un Objeto de Negocio. Con el objetivo de reducir el número de llamadas remotas y evitar la sobrecarga asociada, se implementan los Transfer Objects para transportar los datos a través de las distintas capas lógicas de la aplicación. De modo que el cliente para obtener más de un valor hace una sola llamada a un método remoto del Objeto de Negocio en lugar de varias, pues el Objeto de Negocio construye un nuevo ejemplar Transfer Object, copia dentro los valores de los atributos individuales del objeto y lo devuelve al cliente. El cliente recibe el Transfer Object y puede entonces invocar los métodos para conocer los estados o propiedades (métodos Get) del mismo.

  Por lo tanto una clase Transfer Object debe proporcionar un constructor que acepte todos los atributos requeridos para crear objetos correspondientes a dicha clase. El constructor podría aceptar todos los valores de atributos del bean de entidad (entidad del dominio) para el que se ha diseñado el Transfer Object. Normalmente, los miembros del Transfer Object se definen como públicos, así eliminan la necesidad de los métodos get y set. Si se necesita alguna protección, los miembros podrían definirse como protected o private, y se definirían métodos get para sus valores. Si no ofrece métodos set para los valores, un Transfer Object está protegido frente a modificaciones después de su creación. Si sólo se permite la modificación de unos pocos miembros para facilitar las actualizaciones, entonces si que se deben proporcionar métodos set. Por lo tanto, la creación del Transfer Object varía dependiendo de los requerimientos de la aplicación.

- **Data Access Object (DAO)**

  La mayoría de las aplicaciones de la plataforma J2EE necesitan utilizar datos persistentes para su correcto funcionamiento, actualmente existen disímiles fuentes de datos con las que se puede contar para almacenar la información persistente de una aplicación, entiéndase bases de datos, archivos, servicios externos, etc. Esta diversidad trae consigo inconvenientes en cuanto a la forma

de acceder a dichos dispositivos de almacenamiento, este proceso varía dependiendo de la fuente de datos a la que se quiera llegar. Como solución a esta problemática surge el patrón DAO, el cual es aplicado para abstraer y encapsular la forma de acceso, manejando las conexiones a la fuente de datos para obtener y almacenar los mismos, de modo que el software cliente se centre en los datos que necesita y se olvide de cómo se realiza su acceso ó cuál es la fuente de almacenamiento a la que necesita dirigirse.

Específicamente, el patrón DAO es una solución al problema del diferencial de impedancia entre un programa de aplicación orientado a objetos y una base de datos relacional, empleando únicamente la interfaz de programación (API) nativa del manejador de base de datos.

La principal ventaja de su empleo durante el desarrollo de la aplicación, está determinada por el bajo nivel de acoplamiento obtenido entre clases, reduciendo la complejidad de realizar cambios, este planteamiento queda sustentado con la siguiente afirmación: un cambio del gestor de base de datos de la aplicación, sólo afectará al DAO y no a las clases encargadas de la lógica de negocio o de presentación.

Un principio básico de un buen diseño es identificar los aspectos de la aplicación que cambian o pueden cambiar y separarlos de los que van a permanecer fijos o prácticamente invariables. Muchos patrones de diseño se basan en encapsular de alguna forma la parte que cambia para hacer más fácil la extensión del sistema. En este caso, el DAO encapsula la parte que puede variar (la interacción con la fuente de datos).

1.8. Conclusiones.

A partir del análisis realizado durante este capítulo se puede arribar a la siguiente conclusión: la necesidad evidente de confeccionar una aplicación web que brinde la posibilidad de hacer ameno y menos complicado el trabajo de los especialistas del Grupo de Liberación y Pruebas de Software a través de la automatización del Proceso de Liberación de Productos de Software, de lo cual se desprende que para su desarrollo eficiente se precisa aplicar una y cada una de las tecnologías que fueron definidas anteriormente.

CAPÍTULO 2 CARACTERÍSTICAS DEL SISTEMA

2.1. Introducción.

Hasta el momento se han explicado las tecnologías a ser tenidas en cuenta para el desarrollo del sistema. En el presente capítulo se hace énfasis en las características del mismo, para lo cual se describen las principales funciones y el flujo actual del proceso que se llevan a cabo en el Grupo de Liberación y Pruebas de Software, de dicho análisis se desprenden aquellos subprocesos que son objeto de modelación para posterior automatización y por ende los requisitos tanto funcionales, como no funcionales que el sistema debe cumplir y debe tener respectivamente.

2.2. Funcionalidades del Grupo de Liberación y Pruebas de Software.

Actualmente la Dirección Central de Calidad se encuentra estructurada por grupos, donde Liberación y Pruebas de Software, es el grupo encargado de comprobar la calidad de los productos antes de ser entregados al cliente, a través de estándares y normas de calidad establecidos, siendo esta evaluación confiable tanto para los equipos de desarrollo, como para los clientes de la UCI. Para ello se rigen por un proceso que define el conjunto de tareas lógicas que deben ser llevadas a cabo para cumplimentar el objetivo principal del grupo. De modo que su análisis se hace vital para comprender el funcionamiento en detalles del mismo, siendo entonces el centro de atención de próximos acápites.

2.2.1. Flujo Actual del Proceso

El modo de trabajo del grupo Liberación y Pruebas de Software queda definido por la realización de las siguientes fases:

Confección de la Solicitud de Prueba

El Proceso de Liberación de Productos de Software comienza cuando el Jefe de un Proyecto determina que sus productos están aptos para ser sometidos a pruebas de software, para ello precisa de: confeccionar un documento a través del cual solicita oficialmente al Jefe del LIPS el inicio del proceso de comprobación de la calidad a los mismos, en este documento debe quedar registrado entre otros datos, los recursos tecnológicos necesarios que debe disponer el LIPS para que el proceso se lleve a cabo eficientemente.

Una vez confeccionada la solicitud, se procede al análisis de la misma, quedando definida así la segunda fase del proceso:

Análisis de la Solicitud de Prueba

Es responsabilidad del Jefe del LIPS analizar y tramitar la solicitud de prueba, con el propósito de determinar si la misma será aprobada, rechazada o cancelada, esta determinación depende fundamentalmente de la disponibilidad de recursos tanto humanos, como tecnológicos que dispone, así como de la calidad de la solicitud. En caso de ser aprobada, es responsabilidad de cada grupo de especialistas definir y garantizar uno y cada uno de los aspectos vitales para la ejecución de las pruebas de software, entiéndase creación de casos de pruebas, listas de chequeo, la definición de los tipos de pruebas y herramientas a aplicar, etc.

Elaboración del Pre-Plan de Pruebas

Garantizado el análisis de la solicitud de prueba se procede entonces a la confección del Pre-Plan de Pruebas, documento en el cual queda registrado cada uno de los aspectos abordados en la fase anterior, quedando así estructurada la planificación previa del proceso de prueba a desarrollar al producto.

Planificación de las Reuniones de Inicio

Confeccionado el Pre-Plan, el Jefe del LIPS concibe una reunión con el Líder del Proyecto para discutir dicho documento, de modo que se logre un acuerdo mutuo entre ambas partes en cuanto a la planificación definida.

Montaje del Entorno de Pruebas

Es en esta fase donde se adecua el LIPS en dependencia de los recursos tecnológicos definidos en la solicitud de pruebas necesarios para comprobar la calidad de los productos, por ejemplo: se planifican los dispositivos hardware y se instalan las aplicaciones informáticas a utilizar, etc.

Pruebas Exploratorias

Una vez definido el entorno de pruebas, se procede a realizar pruebas exploratorias a cada producto entregado, las mismas tienen como objetivo fundamental determinar si un producto posee un mínimo de calidad, empleando los criterios de criticidad, los cuales definen los aspectos básicos a garantizar.

Refinar - Elaborar las Listas de Chequeo y/o Casos de Prueba

Como su nombre lo indica es en esta fase donde se actualizan o se crean tanto Listas de Chequeo, como Casos de Prueba, que serán utilizadas durante la próxima fase: Pruebas y/o Revisiones Técnicas.

Pruebas y/o Revisiones Técnicas.

Una vez sentadas las bases, se procede a realizar una iteración de pruebas funcionales (de sobrecarga o stress, de seguridad) si el producto es una aplicación, o una revisión técnica en caso de que sea un documento o ambas dependiendo de la cantidad y el tipo de producto a comprobar, con el objetivo de encontrar No Conformidades.

Reunión con el Proyecto

Desde el punto de vista funcional esta fase tiene un carácter opcional: si las Pruebas Exploratorias o las iteraciones de Pruebas y/o Revisiones Técnicas se declaran fallidas, entonces se concibe una reunión con el equipo de desarrollo, en la cual el Jefe del LIPS expone las razones del por qué se detuvo el proceso de prueba al producto software especificando cada uno de los errores encontrados.

Actualización del Artefacto

De la fase anterior se desprende la necesidad de: disponer recursos por parte del equipo de desarrollo, con el propósito de corregir los errores detectados, una vez logrado lo anterior, se procede a someter al producto actualizado a una nueva iteración de Pruebas y/o Revisiones Técnicas.

Liberado

Es en esta fase, después de haber realizado satisfactoriamente varias iteraciones de Pruebas y/o Revisiones técnicas, se concluye que el producto está apto para ser entregado al cliente y por tanto se decide la liberación del mismo.

Analizadas las fases que comprende el PLPS, se procede a exponer (Ver Figura 5), la secuencia lógica de la realización de las mismas, por la cual se rigen los especialistas del Grupo de Liberación y Pruebas de Software para comprobar la calidad de los productos.

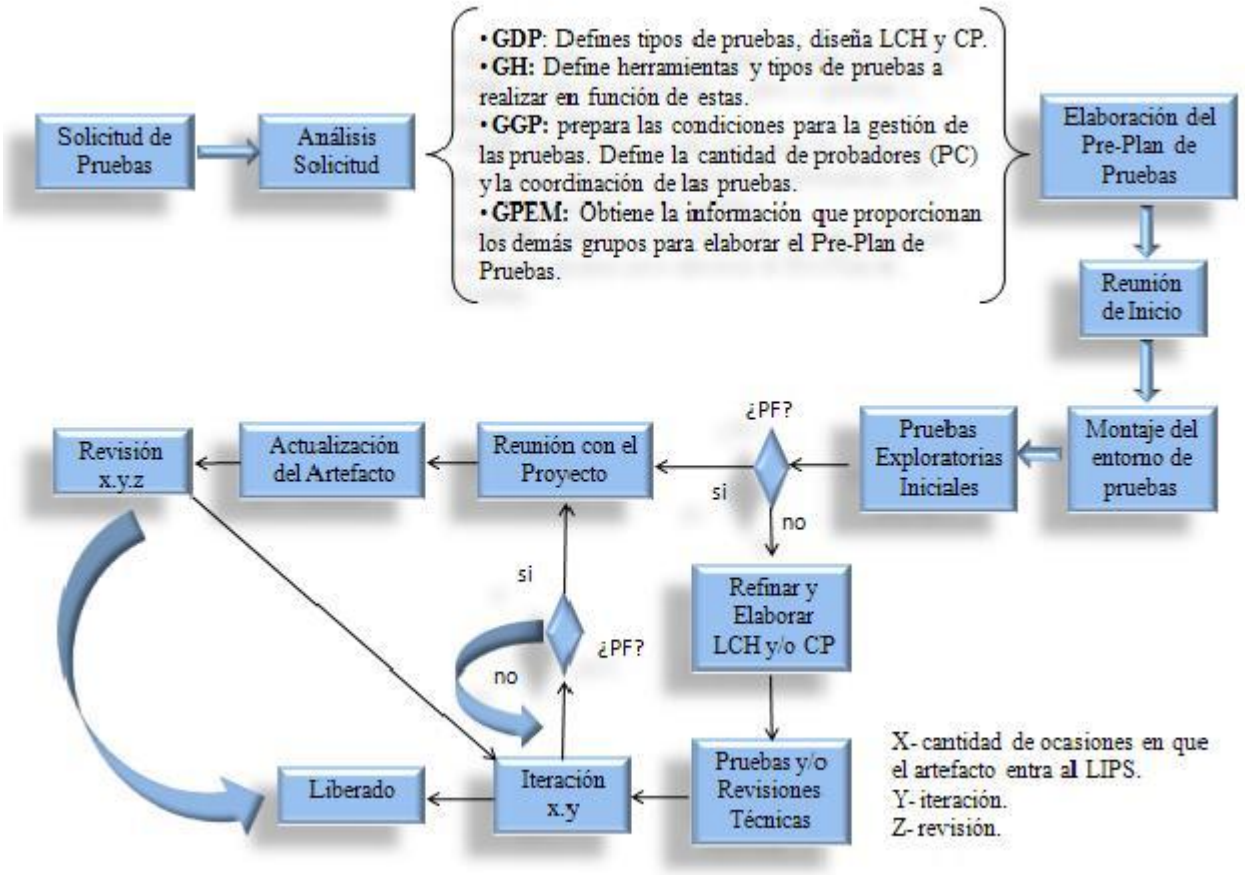


Figura 5: Proceso de Liberación de Productos Software.

2.3. Modelo del Dominio.

Después de un profundo análisis enfocado en la descripción de los procesos del negocio, realizada anteriormente se puede llegar a la conclusión de que el negocio que se esta analizando tiene un bajo nivel de estructuración, y cuenta con soluciones muy diversas y dispersas. Por lo que sería adecuado realizar

un modelo de dominio, el cual permitirá capturar los tipos de objetos conceptuales más importantes que existen o los eventos que suceden en el entorno donde estará el sistema. El modelo de dominio brindará a clientes, usuarios, desarrolladores e interesados, un vocabulario común para comprender el contexto del negocio en que se enmarcará el sistema.

Los objetos del dominio representan los entes o eventos que suceden en el entorno en el que se trabaja, el modelo de dominio captura los tipos más importantes de objetos en el contexto de este entorno. La modelación del dominio tiene como objetivo fundamental la comprensión y descripción de estos objetos, conceptos o eventos.

2.3.1. Principales Conceptos

- **Solicitud:** Gestiona todo un proceso, que comprende la solicitud o pedido por parte del líder de proyecto, de la realización de un conjunto de pruebas a su producto o a un artefacto específico que forme parte del mismo.
- **Pre-Plan de Pruebas:** Gestiona la planificación previa del proceso de prueba a desarrollar, definiendo los tipos de Listas de Chequeo, lo tipos de Pruebas que se van a aplicar a cada uno de los artefactos que conforman el producto durante el proceso.
- **Artefacto:** Término general para cualquier tipo de información creada, producida, modificada por cada uno de los trabajadores que interviene durante el desarrollo del sistema. Algunos ejemplos de artefactos pueden ser diagramas UML y su texto asociado, pueden ser modelos o elementos de los mismos, un documento, código fuente, ejecutables, etc.
- **Lista de Chequeo:** Utiliza un conjunto de herramientas predeterminadas y orientadas a identificar problemas por tipos de artefactos y sirven para motivar posibles soluciones o la detección de oportunidades de mejora.
- **Casos de Pruebas:** Especifica una forma de probar el sistema incluyendo la entrada o resultado con lo que se ha de comprobar y las condiciones bajo las que ha de probarse.
- **Pruebas Exploratorias:** Pruebas que se realizan inicialmente para verificar un mínimo de calidad del producto, no se planifican, ni se documentan.

- **No Conformidad:** Elemento con el cual el probador no se encuentra conforme durante el proceso de prueba.
- **Módulo de Trabajo:** Planificación previa de un conjunto de actividades que realizará un probador en su puesto de trabajo.

2.3.2. Diagrama de Clases del Dominio

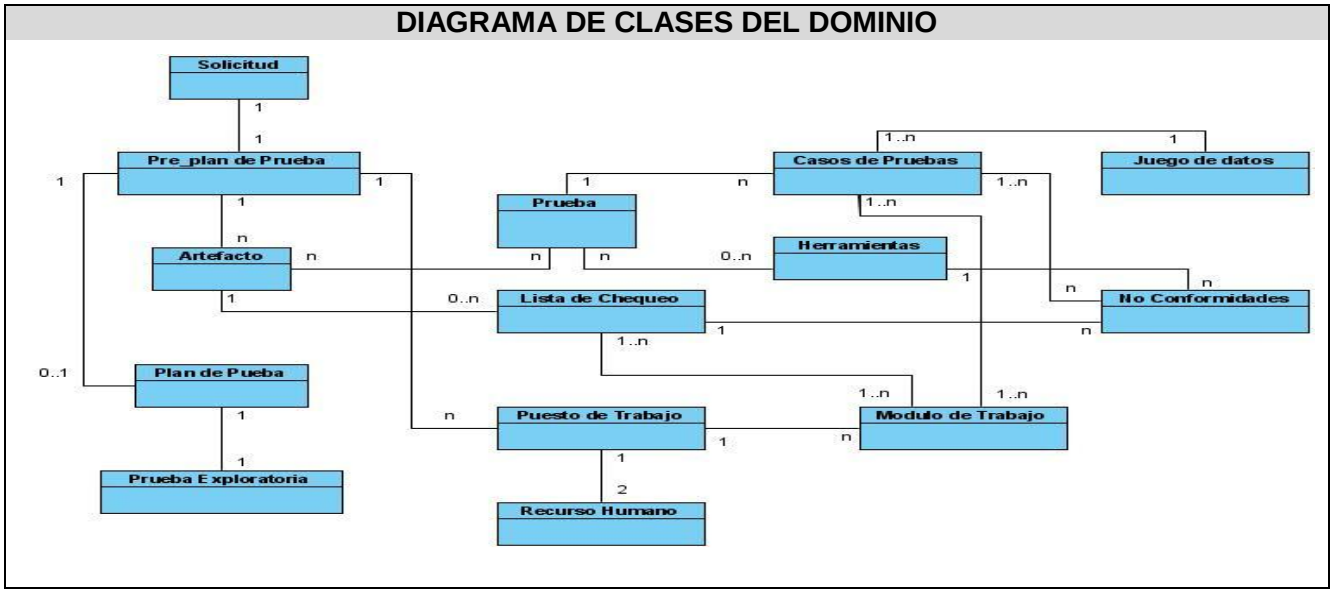


Figura 6: Diagrama de Clases del Dominio

2.4. Especificación de los Requisitos de Software.

La "IEEE (Institute of Electrical and Electronics Engineers) Standard Glossary of Software Engineering Terminology (1990)", define los requerimientos de software como condiciones o capacidades que tienen que ser alcanzadas o poseídas por un sistema para satisfacer un contrato, estándar u otro documento impuesto formalmente.

2.4.1. Requerimientos Funcionales.

Los requerimientos del sistema definen las funciones que el sistema será capaz de realizar. Son el conjunto de capacidades o condiciones que debe cumplir el software para ser exitoso en el entorno en el

cual se usará. Estos deben ser comprensibles por clientes, usuarios y desarrolladores, deben tener una sola interpretación y estar definidos en forma medible y verificable. El análisis realizado al flujo actual del Proceso de Liberación de Productos Software arrojó los siguientes requisitos:

R1. El sistema debe permitir gestionar los perfiles de usuarios.
- R1.1. Crear el perfil de usuario.
- R1.2. Modificar el perfil de usuario.
- R1.3. Mostrar datos del perfil de usuario.
- R1.4. Habilitar/Inhabilitar un perfil de usuario.

R2. El sistema debe permitir gestionar Solicitud de Inclusión de Usuario al Sistema.
- R2.1. Crear Solicitud de Inclusión de Usuario al Sistema.
- R2.2. Mostrar datos de Solicitud de Inclusión de Usuario al Sistema.
- R2.3. Eliminar Solicitud de Inclusión de Usuario al Sistema.

R3. El sistema debe permitir notificar la existencia de una nueva solicitud.

R4. Debe permitir tramitar el estado de Solicitud de Inclusión de Usuario al Sistema.
- R4.1. Aprobar Solicitud de Inclusión de Usuario al Sistema.
- R4.2. Rechazar Solicitud de Inclusión de Usuario al Sistema.
- R4.3. Cancelar Solicitud de Inclusión de Usuario al Sistema.

R5. El sistema debe permitir notificar estado de solicitud de inclusión de usuario a los interesados.

R6. El sistema debe permitir gestionar usuario.
- R6.1. Crear Usuario.
- R6.2. Mostrar datos Usuario.
- R6.3. Modificar Usuario.
- R6.4. Eliminar Usuario.

R7. El sistema debe permitir buscar usuario.

R8. El sistema debe permitir administrar los tipos de artefactos del sistema.
- R8.1. Crear tipos de artefactos.
- R8.2. Modificar tipos de artefactos.
- R8.3. Eliminar tipos de artefactos.

R9. El sistema debe permitir administrar nuevos tipos de pruebas.

R9.1. Crear tipos de pruebas.

R9.2. Modificar tipos de pruebas.

R9.3. Eliminar tipos de pruebas.

R10. El sistema debe permitir administrar herramientas de pruebas.

R10.1. Crear herramientas de pruebas.

R10.2. Modificar herramientas de pruebas.

R10.3. Eliminar herramientas de pruebas.

R11. El sistema debe permitir administrar nuevas Listas de Chequeo.

R11.1. Modificar Listas de Chequeo.

R11.2. Eliminar Listas de Chequeo.

R11.3. Crear Listas de Chequeo.

R12. El sistema debe permitir administrar horarios de trabajo.

R12.1. Crear horario de trabajo.

R12.2. Modificar horario de trabajo.

R13. El sistema debe permitir administrar puestos de trabajo.

R13.1. Crear puesto de trabajo.

R13.2. Inhabilitar puesto de trabajo.

R13.3. Habilitar puesto de trabajo.

R13.4. Modificar puesto de trabajo.

R14. El sistema debe permitir administrar Criterios de Criticidad.

R14.1. Crear nuevo criterio.

R14.2. Modificar criterio.

R14.3. Eliminar criterio.

R15. El sistema debe permitir al usuario autenticarse en el sistema para iniciar sesión de usuario.

R16. El sistema debe permitir gestionar Solicitud de Pruebas a un producto.

R16.1. Crear solicitud de prueba.

R16.2. Modificar datos de la solicitud.

R17. El sistema debe permitir notificar la existencia de una nueva solicitud.

R18. El sistema debe permitir tramitar estado de una solicitud de pruebas.

R18.1. Aprobar solicitud.

R18.2. Rechazar solicitud.

R18.3. Cancelar solicitud.

R19. El sistema debe permitir notificar estado de la solicitud de pruebas de un producto al personal involucrado en la solicitud.

R20. El sistema debe permitir gestionar No Conformidades.

R20.1. Crear No Conformidades.

R20.2. Modificar No Conformidad.

R20.3. Eliminar No Conformidad.

R21. El sistema debe permitir cambiar estado y la argumentación de la NC.

R22. El sistema debe permitir la emisión de decisión de fin de pruebas.

R23. El sistema debe permitir la emisión de reportes que resuma las NC pendientes.

R24. El sistema debe permitir notificar al Grupo de Gestión el inicio de una nueva iteración de pruebas (pertenece al R26).

R25. El sistema debe permitir asignación de los puestos de trabajo.

R26. El sistema debe permitir notificación de liberación del artefacto.

R27. El sistema debe permitir crear Pre-Plan de prueba.

R28. El sistema debe permitir crear tamaño de la muestra y momentos de muestreo.

R29. El sistema debe permitir gestión de la Evaluación de las Pruebas Exploratorias.

R29.1. Crear Evaluación de las Pruebas Exploratorias.

R29.2. Modificar Evaluación de las Pruebas Exploratorias.

R30. El sistema debe permitir notificación de los Resultados de las Pruebas Exploratorias.

R31. Modificar Pre-plan de Pruebas.

R32. El sistema debe permitir seleccionar tipos de pruebas a aplicar por artefacto.

R33. El sistema debe permitir seleccionar tipos de Listas de Chequeo a aplicar por artefacto.

R34. El sistema debe permitir gestionar planificación previa de recursos.

R34.1. Mostrar la cantidad de puestos de trabajo disponibles.

R34.2. Seleccionar la cantidad de puestos de trabajo a utilizar.

R34.3. Modificar la cantidad de puestos de trabajo.

R34.4. Mostrar la disponibilidad de estudiantes por día.

R34.5. Modificar la disponibilidad de estudiantes por día.

R34.6. Seleccionar la cantidad de turnos de trabajo a utilizar.

R34.7. Modificar la cantidad de turnos de trabajo.

R35. El sistema debe permitir notificar al Grupo de Herramientas que se definieron los tipos de pruebas.

R36. El sistema debe permitir seleccionar tipos de herramientas a usar para cada tipo de prueba.

R37. El sistema debe permitir crear citación de reuniones.

R38. El sistema debe permitir gestión de Reuniones de inicio.

R38.1. Crear Acta de Reuniones de Inicio.

R38.2. Mostrar plantilla de Acta de Reuniones de Inicio.

R38.3. Crear Plan de Pruebas.

R38.4. Mostrar el Pre-plan de Pruebas.

R38.5. Actualizar el Pre-plan de Pruebas.

R39. El sistema debe permitir registrar entrada de un probador al laboratorio.

R40. El sistema debe permitir ubicar al probador en un puesto de trabajo.

R41. El sistema debe permitir mostrar indicaciones del trabajo a realizar.

R42. El sistema debe permitir gestionar Módulos de Trabajo.

R42.1. Crear Módulo de Trabajo.

R42.2. Modificar Módulo de Trabajo.

R43. El sistema debe permitir mostrar alerta cuando que queden 5 minutos del turno de trabajo.

R44. Gestionar resultados de las pruebas automatizadas.

R44.1. Crear resultados de las pruebas automatizadas.

R44.2. Modificar resultados de las pruebas automatizadas.

R45. El sistema debe permitir buscar no conformidades.

R46. El sistema debe permitir cargar los artefactos de cada proyecto.

2.4.2. Requisitos No Funcionales

Los requisitos no funcionales especifican propiedades, que de una forma u otra restringen el entorno del sistema o de la implementación como por ejemplo rendimiento, interfaz de usuario, facilidad de mantenimiento, dependencias de la plataforma, entre otros.

A continuación se exponen aquellos que el sistema en cuestión debe tener:

Software

Cliente

Los clientes tendrán acceso al sistema a través de cualquier navegador Web. Se recomienda utilizar el Mozilla 3.x o Internet Explorer 6.x. Además debe estar instalado el Adobe Reader 5.x o superior.

Servidor

Sistema Gestor de Base de Datos: PostgreSQL 8.3.

Servidor Web: Apache Tomcat 6.0.

Hardware

Los requerimientos mínimos del sistema son: procesador Pentium III o superior, memoria RAM 256 MB o superior, capacidad de almacenamiento 10 Gb o superior.

Apariencia o interfaz externa

La interfaz de usuario debe ser apropiada para un control de operaciones efectivo. Esta debe ser sencilla, amigable, intuitiva y de fácil navegación por el usuario, permitiendo la utilización del sistema sin mucho entrenamiento, con el objetivo de evitar la resistencia humana al uso del nuevo sistema, ya que el factor humano determina en gran medida el éxito o el fracaso del mismo.

Seguridad y privacidad

El sistema tendrá acceso diferenciado con contraseña para cada usuario, expone la necesidad de autenticación doble ante acciones muy delicadas. El sistema contará con un módulo de seguridad que permite a los usuarios acceder solamente a los lugares que se les ha dado acceso. Permite definir perfiles que establecen las acciones permisibles a realizar por los distintos usuarios.

La información almacenada en el sistema estará protegida de acceso no autorizado. Por otro lado a los usuarios autorizados se les garantizará el acceso a la información sin que los mecanismos utilizados para lograr la seguridad retrasen la obtención de los datos deseados en un momento dado.

Portabilidad

El sistema podrá ejecutarse sobre la plataforma Unix y Windows, siendo posible el acceso a través de los navegadores web Firefox o Internet Explorer.

Rendimiento

Para un funcionamiento óptimo la aplicación debe soportar al menos 100 PC físicas conectadas de forma on-line. Debe ser capaz de dar respuesta en un corto plazo de tiempo, es recomendable no más de 0 ó 5 segundos; además debe haber consistencia en los datos que se obtienen en las solicitudes. El sistema debe estar disponible al usuario en todo momento.

Confiabilidad

Garantía de un tratamiento adecuado de las excepciones y validación de las entradas de los usuarios.

Eficiencia

Garantizar velocidad de respuesta.

Tener base de datos normalizada, para garantizar la integridad de la información y reducir los tiempos de respuesta.

Permitir numerosas conexiones simultáneas.

2.5. Definición de los Casos de Uso del Sistema.

Un caso de uso define la secuencia de transacciones desarrolladas por un sistema en respuesta a un evento que inicia un actor al interactuar con él, este actor representa el rol que juega una o varias personas, un equipo u otro sistema automatizado en el propio sistema. Por la marcada importancia de estos en aras de favorecer el funcionamiento de un sistema software, se hace imprescindible definir y describir específicamente cuáles interactúan con el sistema en cuestión.

2.5.1. Actores del Sistema.

Actores	Justificación
Administrador	Es el responsable de gestionar los diferentes perfiles con los que va a contar el sistema. Incorporará al sistema y da baja del mismo a usuarios definidos para cada uno de los perfiles establecidos. Es el responsable además de definir cuáles van a ser los momentos laborables y horarios de trabajo. Es responsable también

	de administrar recursos, herramientas que se utilizarán durante las pruebas de calidad de software a cada uno de los productos creados como parte fundamental del PLPS.
Solicitante de Usuario	Responsable de gestionar las Solicitudes de Inclusión de Usuarios al Sistema.
Especialistas del Grupo de Muestreo (EGM)	Son los responsables de determinar si un producto está apto para entrar al LIPS con el fin de realizarles el PLPS, dependiendo de los resultados de las Pruebas Exploratorias.
Especialistas del Grupo de Diseño de Pruebas (EGDP)	Encargados de seleccionar los diferentes tipos de pruebas, así como Listas de Chequeos a aplicar por artefactos. Gestionan además todos los Casos de Pruebas, así como sus respectivos Juegos de Datos para los artefactos de tipo Aplicación.
Especialistas del Grupo de Gestión de Pruebas (EGGP)	Planifican tanto los Puestos como los módulos de Trabajo necesarios para la realización de las pruebas.
Jefe del LIPS	Sus responsabilidades van a estar determinadas entre otras cosas por: la aprobación o no las Solicitudes de Pruebas que se envíen. Es la persona que decide cuando un producto está apto para ser entregado al cliente. Es responsable además de crear y enviar las citaciones de las reuniones que se realizaran durante el proceso, de tramitar decisión de fin de prueba y del Monitoreo del sistema.
Probadores	Son aquellas personas que se encargaran de realizar pruebas a productos software, con el fin de crear No Conformidades en caso de que existan.
Coordinador de Pruebas	Es un actor abstracto (EGM, EGDP, EGGP, Jefe del LIPS, EGH) realiza todas las actividades referentes a la gestión de las Reuniones de Inicio con el Jefe de Proyecto, además que es responsable de emitir decisión de fin de pruebas.
Especialistas del Grupo de Herramientas de Pruebas (EGH)	Son los responsables de definir los tipos de herramientas a aplicar por tipo de pruebas.
Jefe de Proyecto	Es el responsable de Gestionar las Solicitudes de Pruebas, además de Actualizar las No Conformidades.

Vicedecano de Producción	Es responsable de Gestionar las Solicitudes de Pruebas.

Tabla 3: Actores del Sistema

2.5.2. Estructura del Sistema a Automatizar

Para organizar de forma eficiente el desarrollo del sistema, el mismo se dividió en subsistemas, los cuales se representan a través de paquetes, siendo ésta, la única forma que expone UML para agrupar elementos en este tipo de estructuras organizativas (XAVIER FERRÉ GRAU, 2004). A continuación se exponen aquellos que son objetos de análisis y diseño en este trabajo: Administración, Gestión Operativa, Gestión del (Cliente y Especialista). Los mismos se describen a continuación:

- **Administración:** Este paquete recoge todas las funcionalidades referentes a la parte administrativa del sistema.
- **Gestión del (Cliente y Especialista):** Este paquete recoge todo lo relacionado con las acciones que tanto el cliente como el especialista deben realizar para solicitar la entrada de un producto software al LIPS y comenzar el proceso de pruebas respectivamente.
- **Gestión Operativa:** Este paquete recoge todas las funcionalidades relacionadas con las acciones propias que se realizan en el laboratorio durante el PLPS.

REPRESENTACIÓN DE LOS SUBSISTEMAS DE LA APLICACIÓN

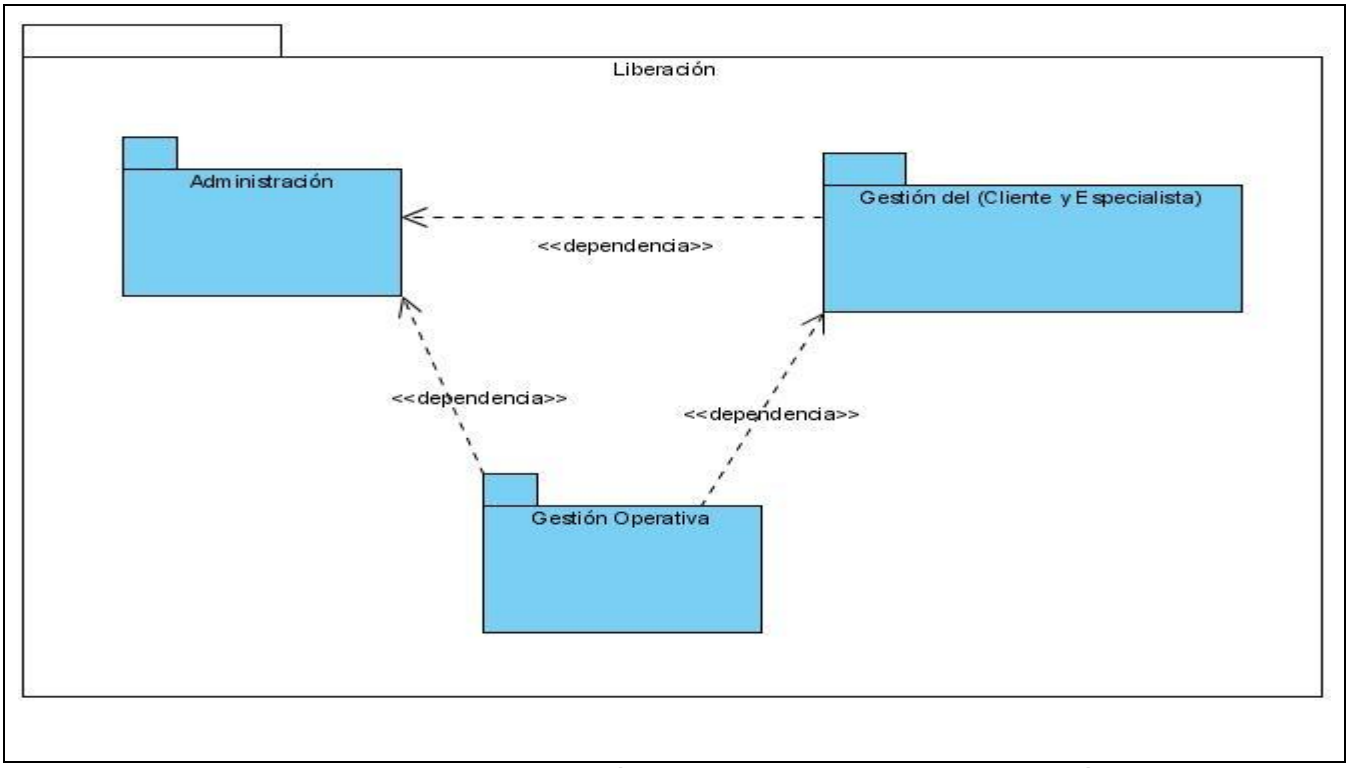


Figura 7: Representación de los subsistemas de la aplicación.

Una vez definida la estructura organizativa del sistema y los actores que interactúan con el mismo, se procede entonces a definir los casos de uso.

2.5.3. Listado de los Casos de Uso del Sistema.

Cód	Nombre del caso de uso	Paquete	Justificación de la selección
CU-1	Gestionar Perfil.	Administración	Permite crear, modificar o habilitar /deshabilitar un perfil de usuario.
CU-2	Gestionar Puestos de Trabajo.	Administración	Permite crear un puesto de trabajo o modificar uno existente.
CU-3	Conocer Puestos de Trabajo.	Gestión Operativa	Permite conocer el puesto de trabajo asignado para realizar las pruebas.
CU-4	Gestionar Solicitud de Inclusión de Usuario.	Administración	Permite crear o eliminar una Solicitud de Inclusión de Usuario.
CU-5	Tramitar Estado de Solicitud de Inclusión de Usuario.	Administración	Permite aprobar, rechazar o cancelar una Solicitud de Inclusión de Usuario.

CU-6	Gestionar Usuario.	Administración	Permite modificar o eliminar usuarios en el sistema.
CU-7	Buscar Usuario.	Administración	Permite buscar por tres criterios usuarios en el sistema.
CU-8	Autenticar.	Gestión Operativa	Permite autenticarse en el sistema.
CU-9	Probar Artefacto Asignado.	Gestión Operativa	Permite probar un artefacto asignado.
CU-10	Gestionar No Conformidades.	Gestión Operativa	Permite modificar o eliminar no conformidades.
CU-11	Actualizar No Conformidades.	Gestión Operativa	Permite actualizar el estado de una no conformidad.
CU-12	Gestionar Módulos de Trabajo.	Gestión Operativa	Permite crear, modificar o eliminar un módulo de trabajo.
CU-13	Gestionar Horario de Trabajo.	Administración	Permite crear el horario de trabajo o modificarlo.
CU-14	Gestionar Herramientas de Prueba.	Administración	Permite crear, modificar o eliminar herramientas de prueba.
CU-15	Gestionar Tipos de Pruebas	Administración	Permite crear, modificar o eliminar tipos de pruebas.
CU-16	Gestionar Tipo de Artefactos.	Administración	Permite crear, modificar o eliminar tipos de artefactos.
CU-17	Asignar Puestos de Trabajo a Proyectos	Gestión Operativa	Permite asignar un puesto de trabajo a un proyecto determinado.
CU-18	Gestionar Resultados Pruebas Automatizadas	Gestión Operativa	Permite crear o modificar los resultados de las pruebas automatizadas.
CU-19	Emitir Decisión de fin de pruebas	Gestión Operativa	Permite emitir la decisión de fin de pruebas de un proyecto.
CU-20	Tramitar decisión de fin de pruebas	Gestión Operativa	Permite aprobar, rechazar o cancelar decisión de fin de pruebas.
CU-21	Gestionar Criterios de Criticidad	Administración	Permite crear, modificar o eliminar criterios de criticidad.
CU-22	Definir Tipos de Prueba X Artefacto	Gestión del (Cliente y Especialista)	Permite definir los tipos de prueba aplicar por artefacto.
CU-23	Gestionar Listas de Chequeo	Gestión del (Cliente y Especialista)	Permite crear, modificar o eliminar listas de chequeo.
CU-24	Definir Tipo de Herramienta X Tipo de Prueba	Gestión del (Cliente y Especialista)	Permite definir las herramientas aplicar por cada tipo de prueba.
CU-25	Gestionar Evaluación PE	Gestión del (Cliente y Especialista)	Permite crear o modificar evaluación de las pruebas exploratorias.
CU-26	Cargar Artefactos X Proyecto	Gestión del (Cliente y Especialista)	Permite cargar al sistema los artefactos de cada proyecto.

CU-27	Gestionar Solicitud de Prueba	Gestión del (Cliente y Especialista)	Permite crear o modificar una Solicitud de Pruebas.
CU-28	Tramitar Estado de Solicitud de Pruebas	Gestión del (Cliente y Especialista)	Permite aprobar, rechazar o cancelar una Solicitud de Pruebas.
CU-29	Crear Citación de Reuniones	Gestión del (Cliente y Especialista)	Permite crear citación de reuniones.
CU-30	Confeccionar Acta de Reunión	Gestión del (Cliente y Especialista)	Permite confeccionar acta de reunión.
CU-31	Modificar Pre-Plan de Prueba	Gestión del (Cliente y Especialista)	Permite modificar el Pre-Plan de Prueba.
CU-32	Confeccionar Plan de Prueba	Gestión del (Cliente y Especialista)	Permite confeccionar el Plan de Prueba.

Tabla 4: Casos de Uso del primer ciclo de desarrollo del sistema.

A continuación, las figuras representan la relación que existe entre los Actores y los Casos de Uso del sistema para la gestión del Proceso de Liberación de Productos Software previamente definidos, a través del diagrama de Casos de Uso, haciendo énfasis en aquellos que integran cada uno de los subsistemas a diseñar.

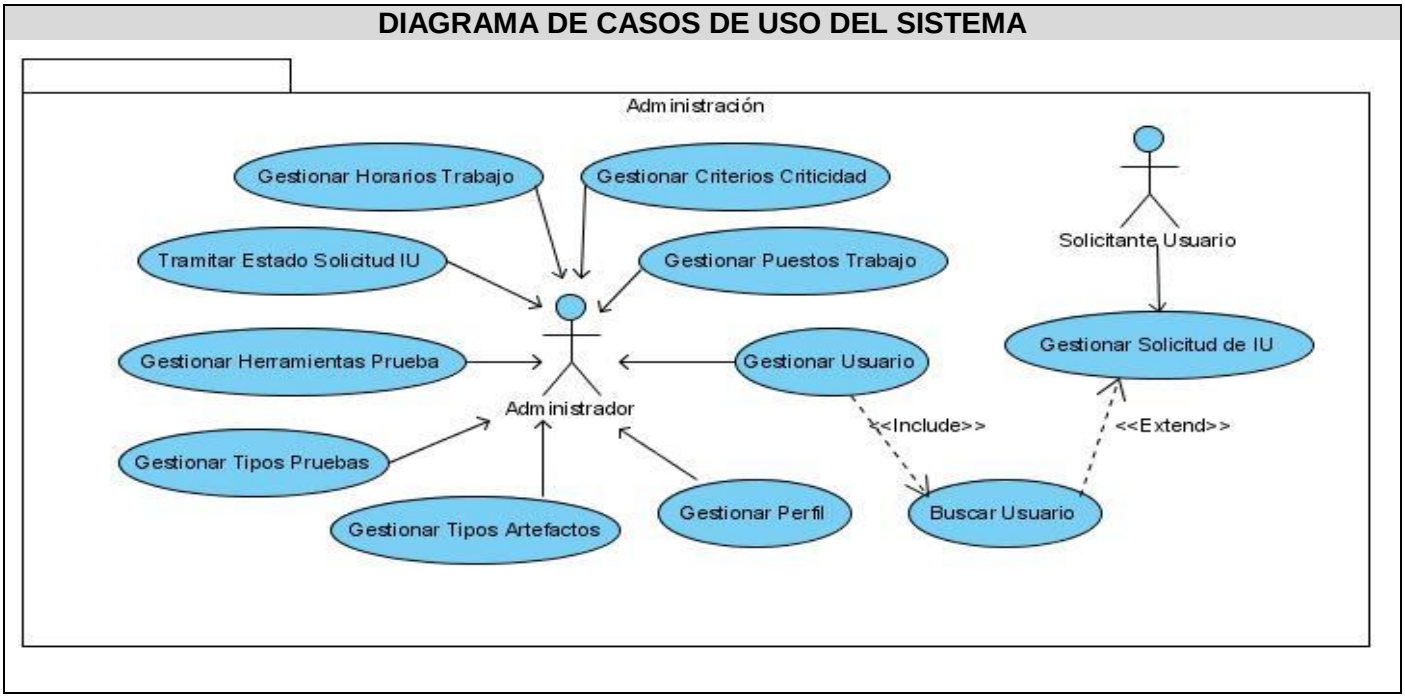


Figura 8: Diagrama de Casos de uso del Sistema (Subsistema Administración)

DIAGRAMA DE CASOS DE USO DEL SISTEMA

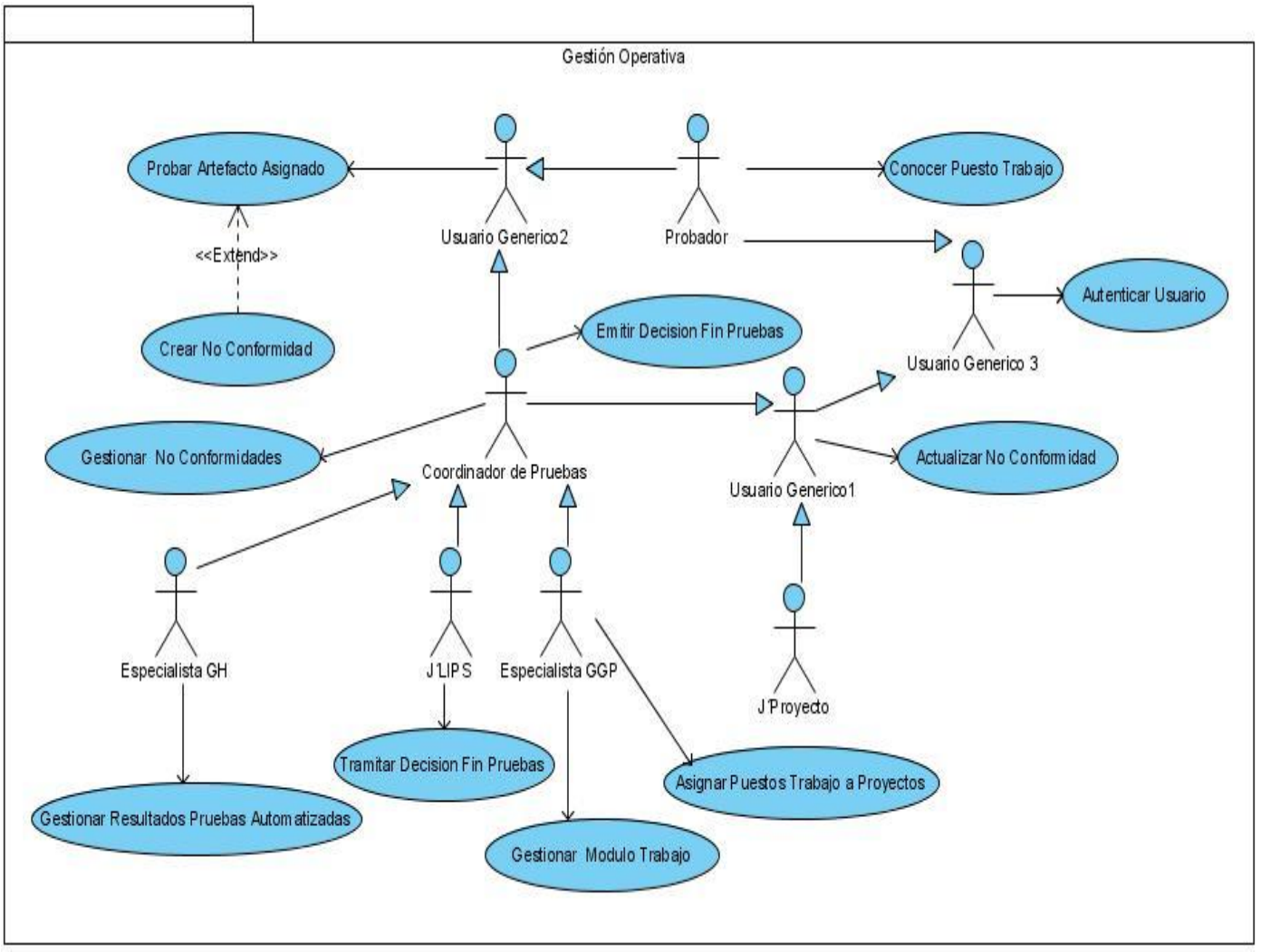


Figura 9: Diagrama de Casos de uso del Sistema (Subsistema Gestión Operativa)

DIAGRAMA DE CASOS DE USO DEL SISTEMA

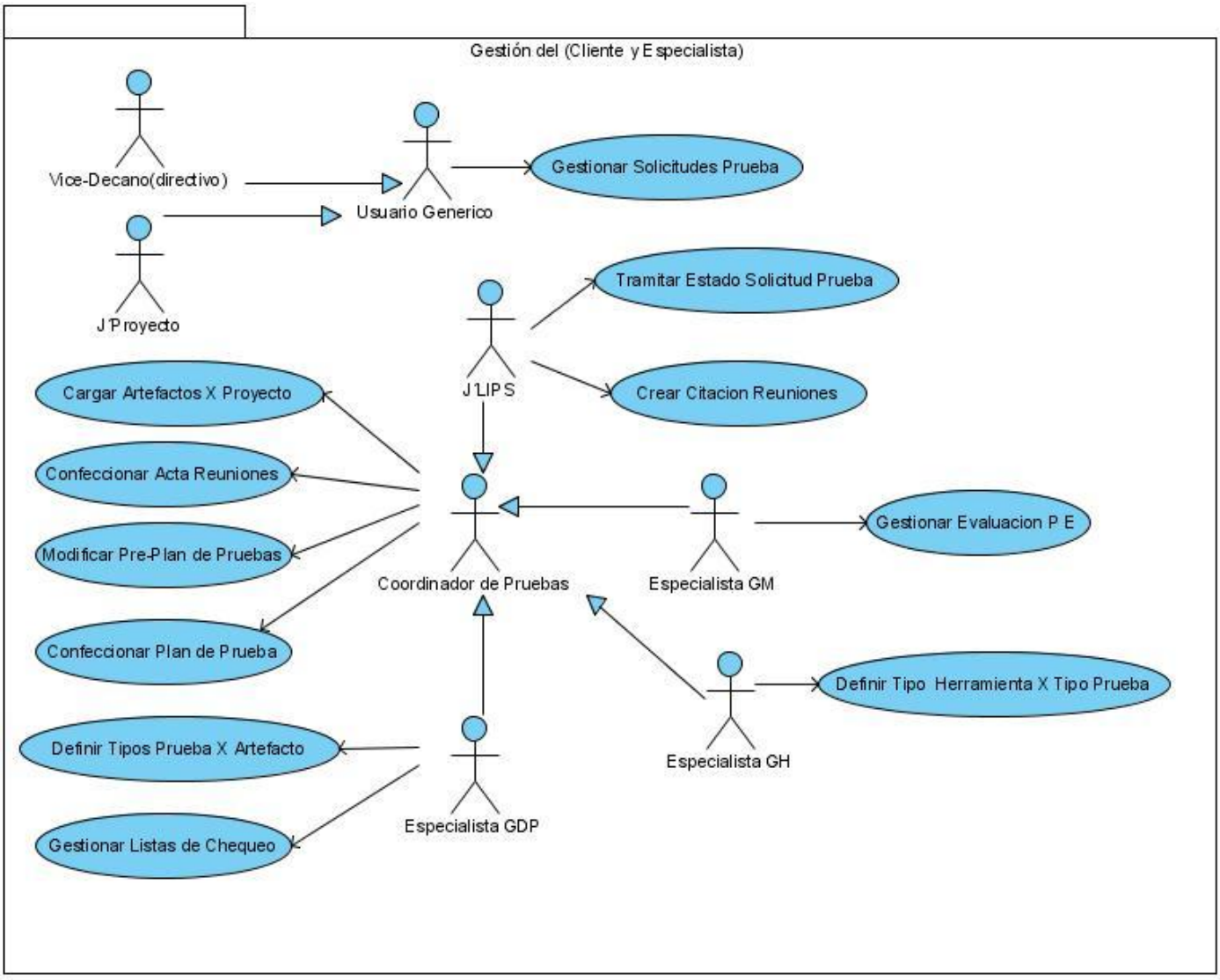


Figura 10: Diagrama de Casos de uso del Sistema (Subsistema Gestión del Cliente y Especialistas)

2.5.4. Descripción de los Casos de Uso del Sistema.

Las descripciones están recogidas en el Anexo 1.

2.6. Conclusiones

En este capítulo se realizó la modelación del dominio, donde se expusieron los objetos conceptuales del dominio que representan los eventos o sucesos que tienen lugar en el entorno en que se trabaja, brindándole a clientes, usuarios, desarrolladores e interesados, un vocabulario común para comprender el contexto del negocio en que se enmarcará el sistema, se realizó además el levantamiento de los requisitos funcionales y no funcionales que el sistema debe cumplir y poseer respectivamente. Además se representó el diagrama de casos de uso del sistema y de los subsistemas: Administración, Gestión Operativa y Gestión del Cliente y Especialistas. Se describieron los casos de uso, así como los actores que se van a relacionar con ellos, dando de esta forma una visión global de cómo va a funcionar el sistema.

CAPÍTULO 3 Análisis y Diseño del Sistema.

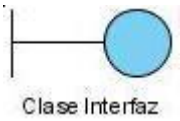
3.1. Introducción.

Durante el proceso de desarrollo de un sistema la fase de elaboración juega un papel primordial, centrándose fundamentalmente en los flujos de trabajo Análisis y Diseño del Sistema, de ahí que el presente capítulo aborde ambos flujos, representando a través de sus principales artefactos las clases que interactúan en la realización los casos de uso capturados previamente, dándole cumplimiento a los requisitos funcionales.

3.2. Análisis del Sistema.

Durante este flujo de trabajo se analizan los requisitos previamente capturados, refinándolos y estructurándolos, a través de clases y paquetes del análisis, con el objetivo de alcanzar una comprensión precisa de los mismos, de facilitar su preparación, su modificación, en general, su mantenimiento. Cabe destacar que para dar cumplimiento a lo anterior no se tiene en cuenta el lenguaje de programación que se va a utilizar en la construcción de la aplicación, debido a que el objetivo del análisis es comprender perfectamente los requisitos del software y no precisar como se implementará la solución. Es uno de los flujos de trabajo que se realizan en el proceso de desarrollo de software durante la fase de elaboración.

A continuación, se describen las clases que serán utilizadas en la realización de los diagramas de clases de análisis de cada caso de uso, de modo que se logre una mejor visión del sistema. Los prototipos para identificar las diferentes clases que participan son los siguientes:



Las clases interfaz se utilizan para modelar la interacción entre el sistema y sus actores (usuarios y sistemas externos). Esta interacción a menudo implica recibir (y presentar) información y peticiones de (y hacia) los usuarios. Representan a menudo abstracciones de ventanas, formularios, paneles, interfaces de impresoras, etcétera.

Las clases de control encapsulan el comportamiento de cada caso de uso y coordinan las secuencias, transacciones entre las clases interfaz y las clases entidad.

Las clases de entidad se utilizan para modelar información que posee una vida larga y que es a menudo persistente.

3.2.1. Diagramas de Clases del Análisis.

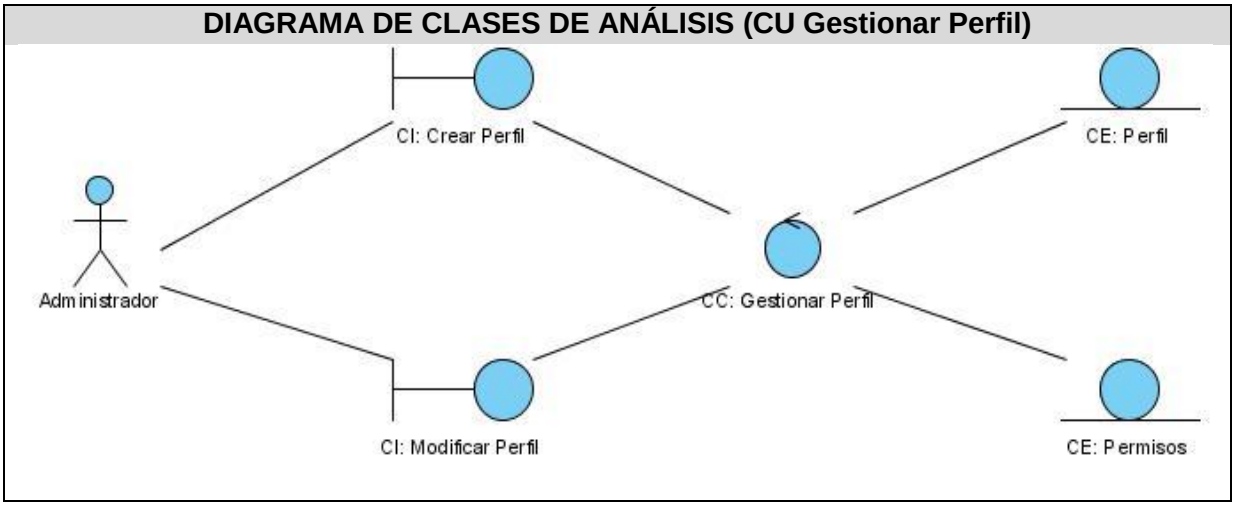


Figura 11: Diagrama de Clases de Análisis (CU Gestionar Perfil)

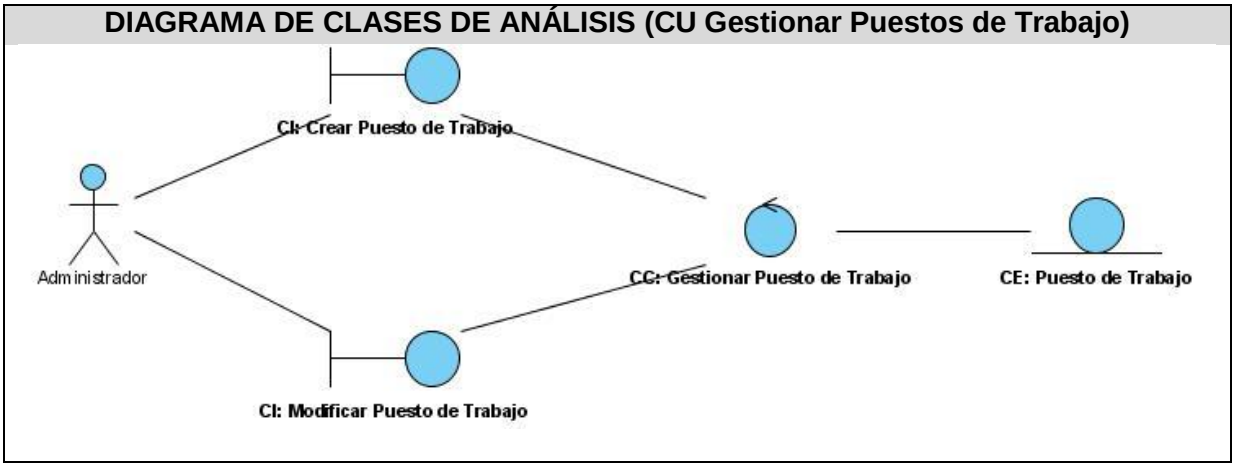


Figura 12: Diagrama de Clases de Análisis (CU Gestionar Puestos de Trabajo)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Conocer Puesto de Trabajo)

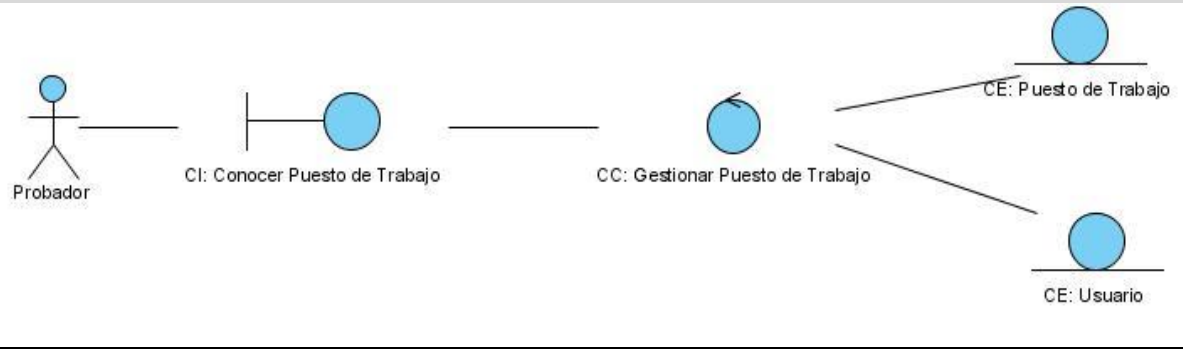


Figura 13: Diagrama de Clases de Análisis (CU Conocer Puesto de Trabajo)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Gestionar Solicitud de Inclusión de Usuario)

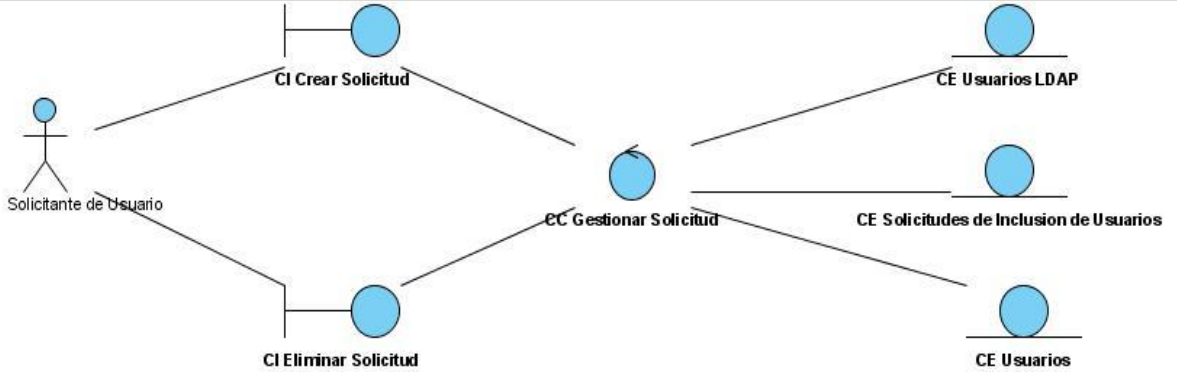


Figura 14: Diagrama de Clases de Análisis (CU Gestionar Solicitud de Inclusión de Usuario)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Tramitar estado de Solicitud de Inclusión de Usuario)

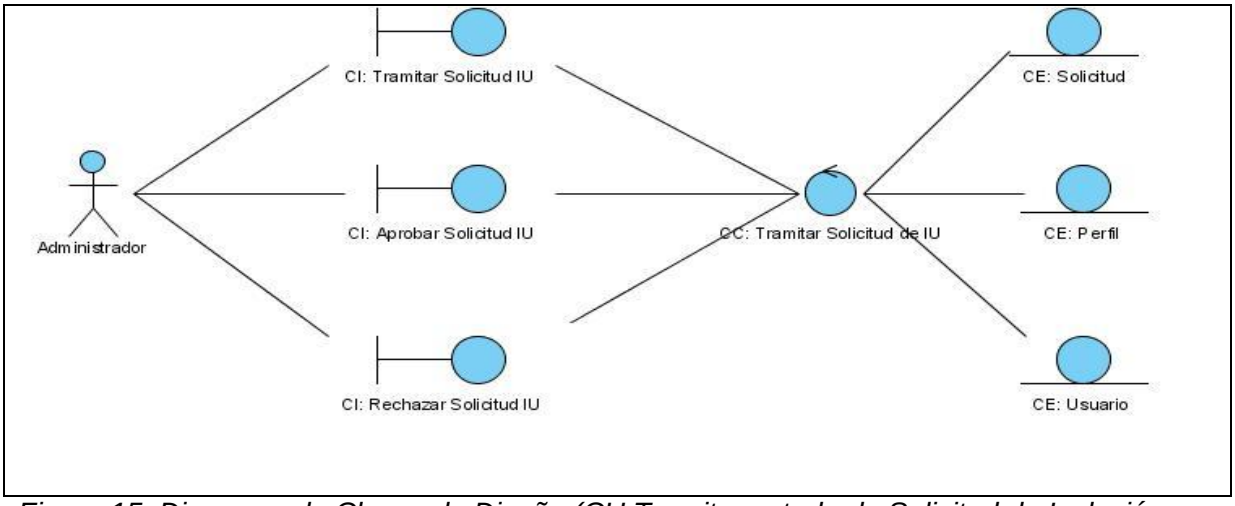


Figura 15: Diagrama de Clases de Diseño (CU Tramitar estado de Solicitud de Inclusión de Usuario)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Gestionar Usuario**)**

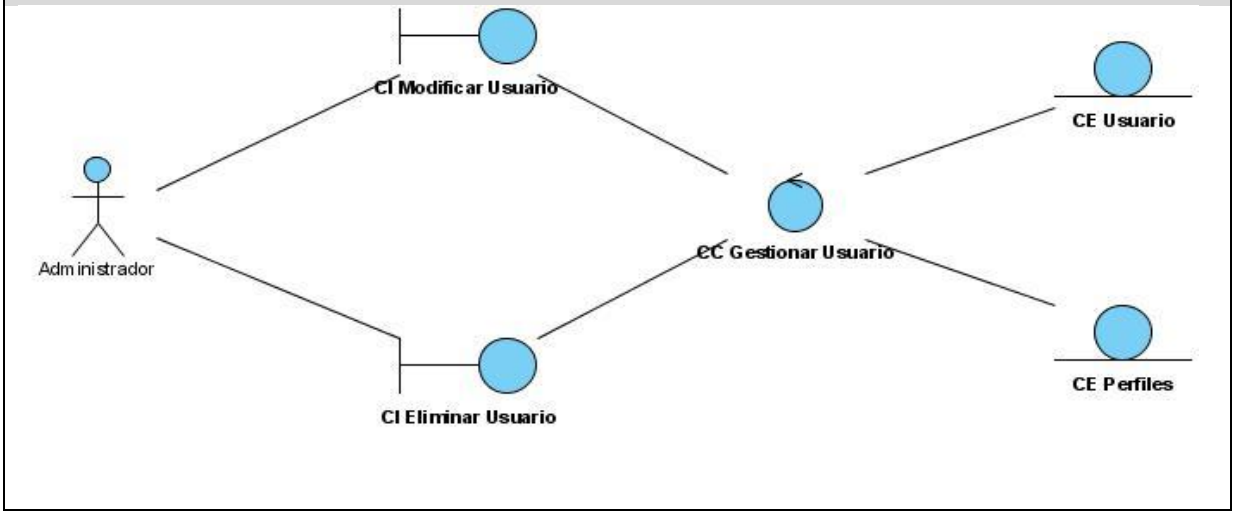


Figura 15: Diagrama de Clases de Diseño (CU Gestionar Usuario)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Autenticar Usuario)

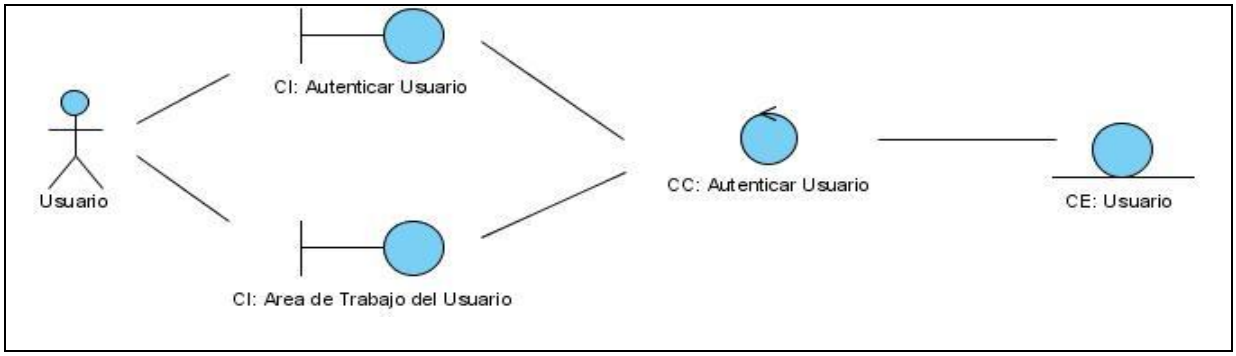


Figura 16: Diagrama de Clases de Diseño (CU Autenticar Usuario)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Probar Artefacto Asignado)

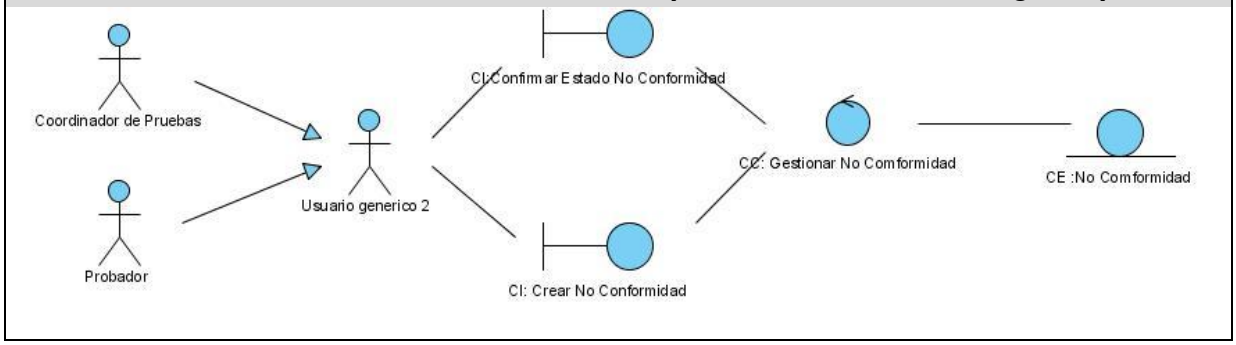


Figura 17: Diagrama de Clases de Diseño (CU Probar Artefacto Asignado)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Gestionar No Conformidades)

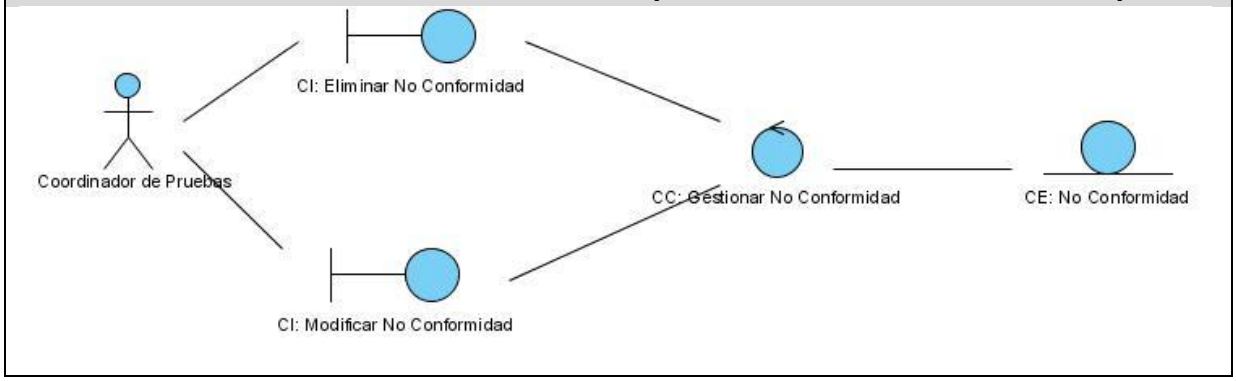


Figura 18: Diagrama de Clases de Diseño (CU Gestionar No Conformidades)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Actualizar No Conformidades)

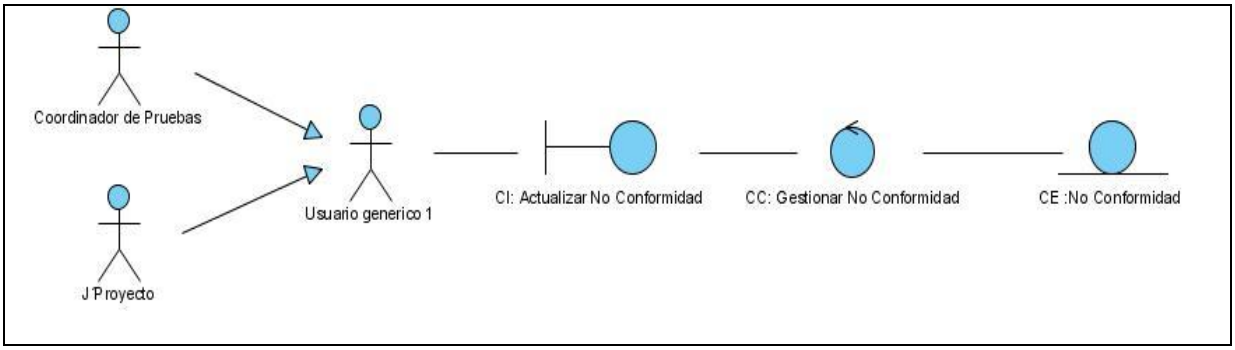


Figura 19: Diagrama de Clases de Diseño (CU Actualizar No Conformidades)

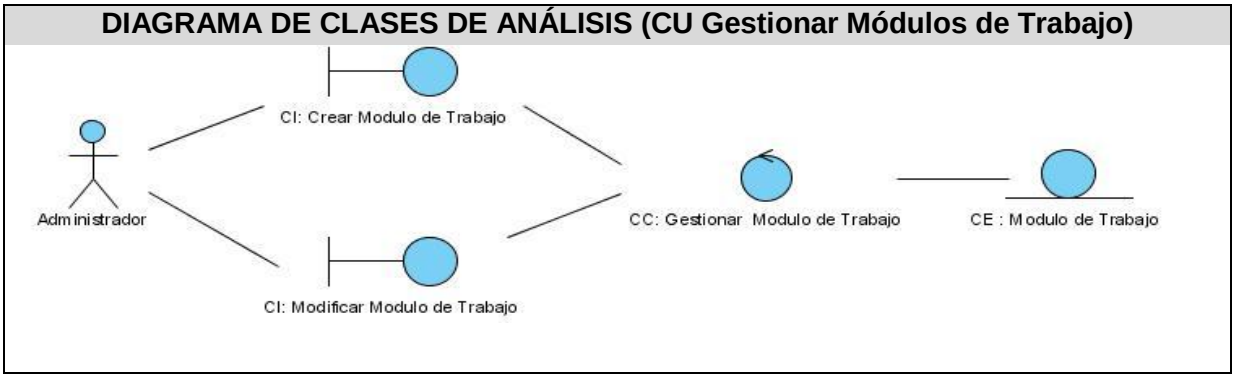


Figura 20: Diagrama de Clases de Diseño (CU Gestionar Módulos de Trabajo)

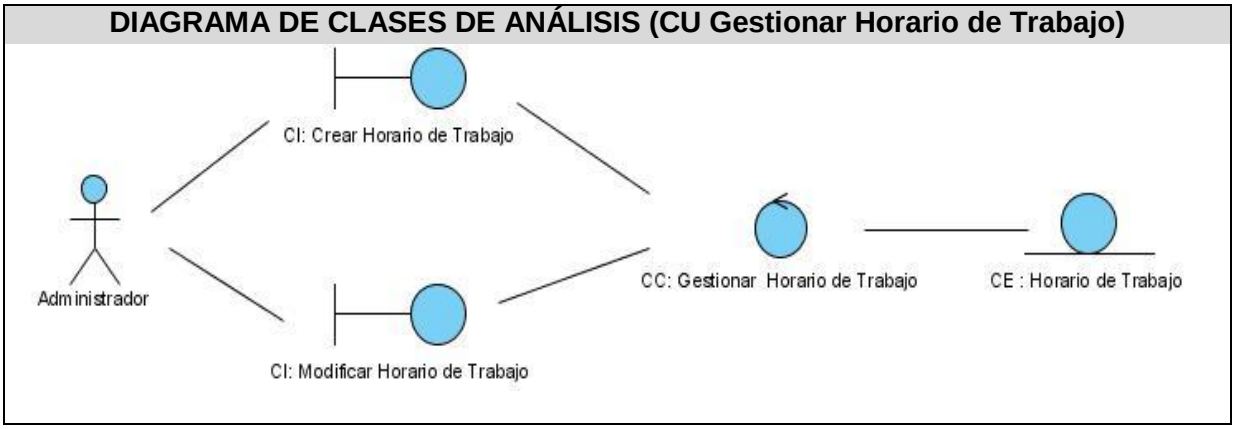


Figura 21: Diagrama de Clases de Diseño (CU Gestionar Horario de Trabajo)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Gestionar Herramientas de Prueba)

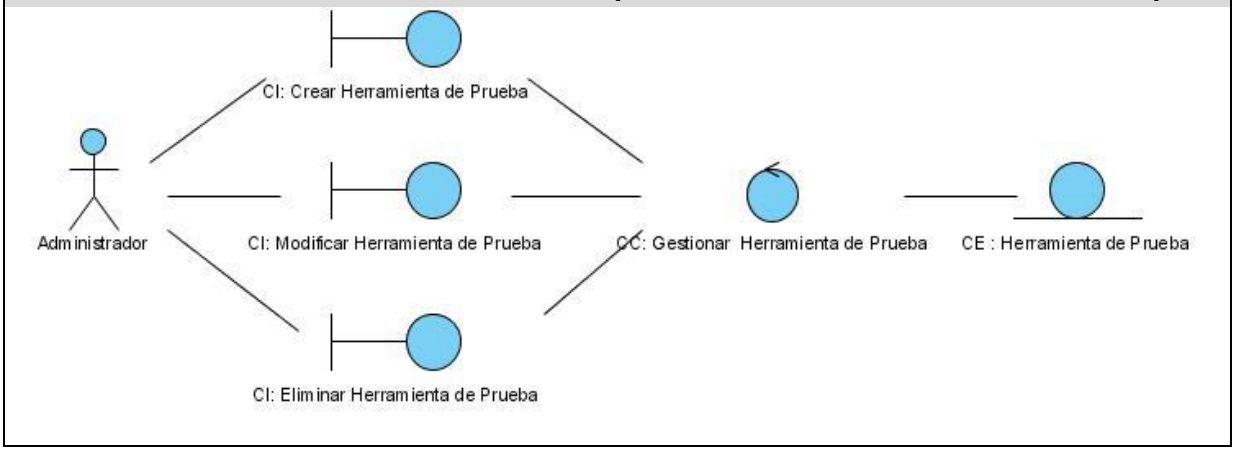


Figura 22: Diagrama de Clases de Diseño (CU Gestionar Herramientas de Prueba)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Gestionar Tipos de Pruebas)

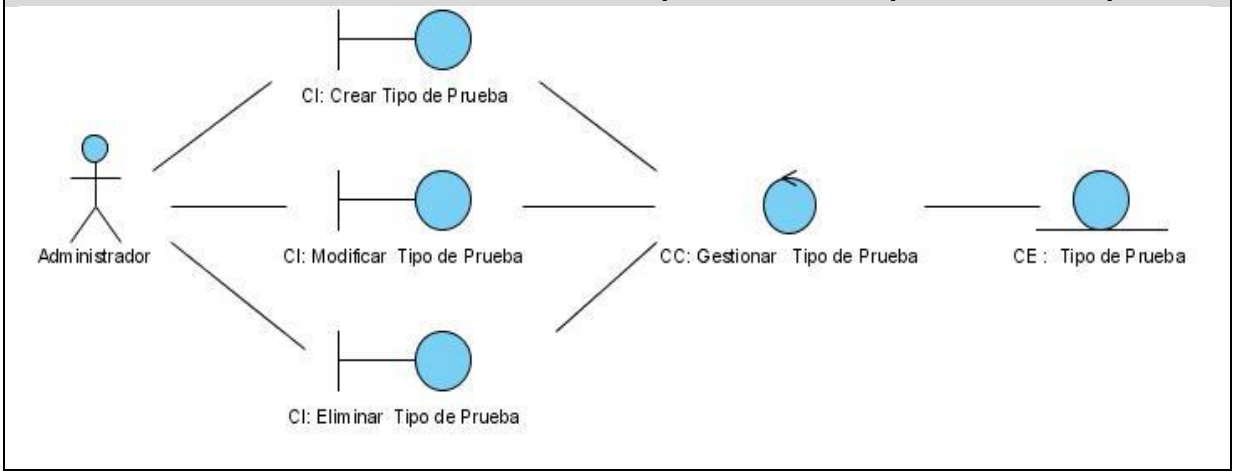


Figura 23: Diagrama de Clases de Diseño (CU Gestionar Tipos de Pruebas)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Gestionar Tipos de Artefactos)

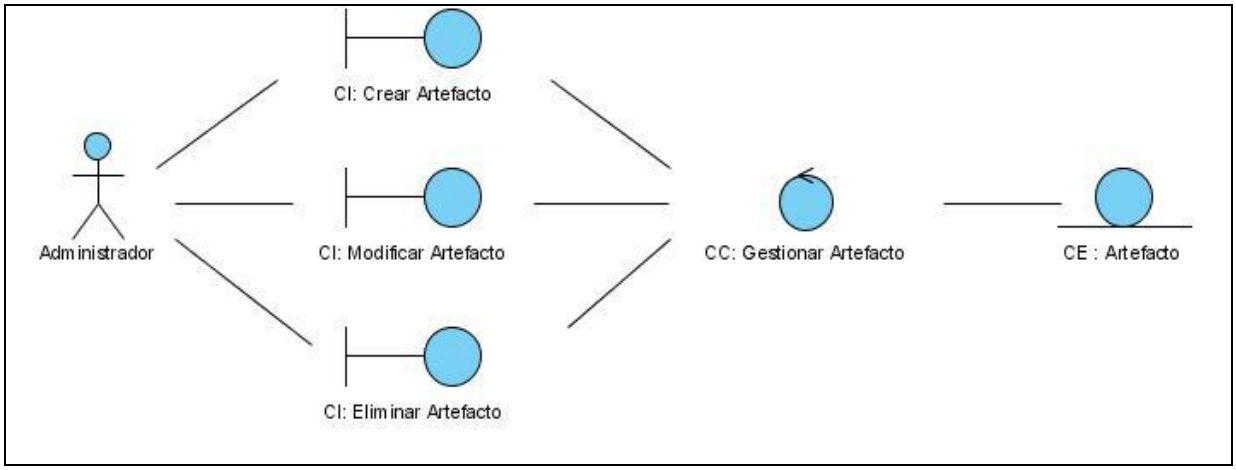


Figura 24: Diagrama de Clases de Diseño (CU Gestionar Tipos de Artefactos)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Asignar Puestos de Trabajo a Proyectos)

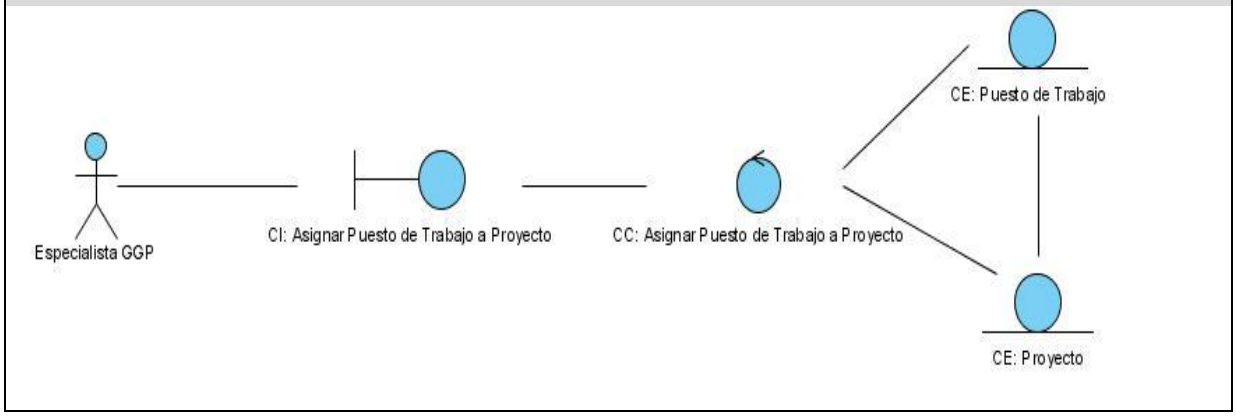


Figura 25: Diagrama de Clases de Diseño (CU Asignar Puestos de Trabajo a Proyectos)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Gestionar Resultados Pruebas Automatizadas)

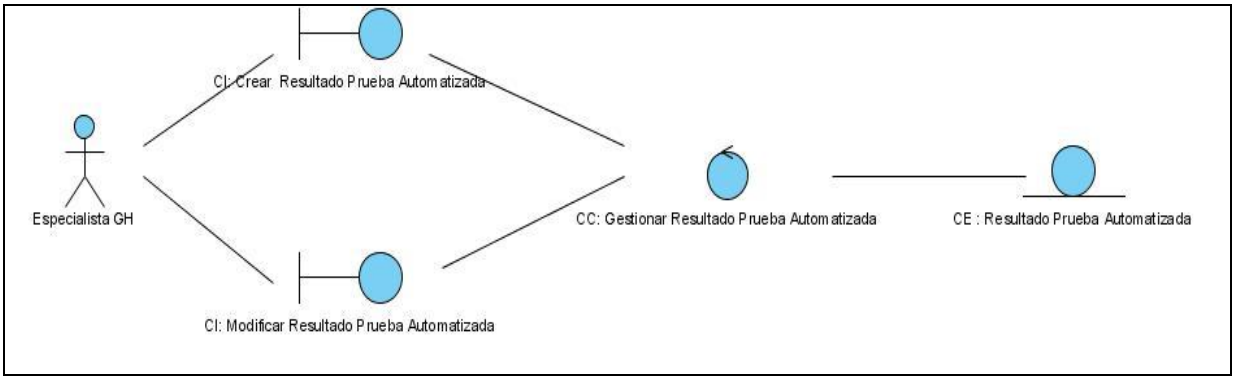


Figura 26: Diagrama de Clases de Diseño (CU Gestionar Resultados Pruebas Automatizadas)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Emitir decisión de fin de pruebas)

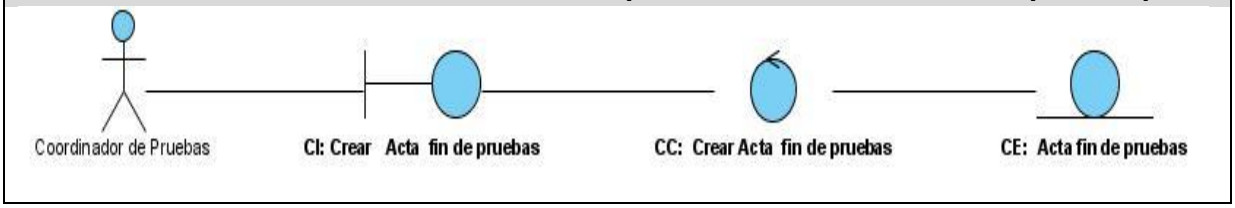


Figura 27: Diagrama de Clases de Diseño (CU Emitir decisión de fin de pruebas)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Tramitar decisión de fin de pruebas)

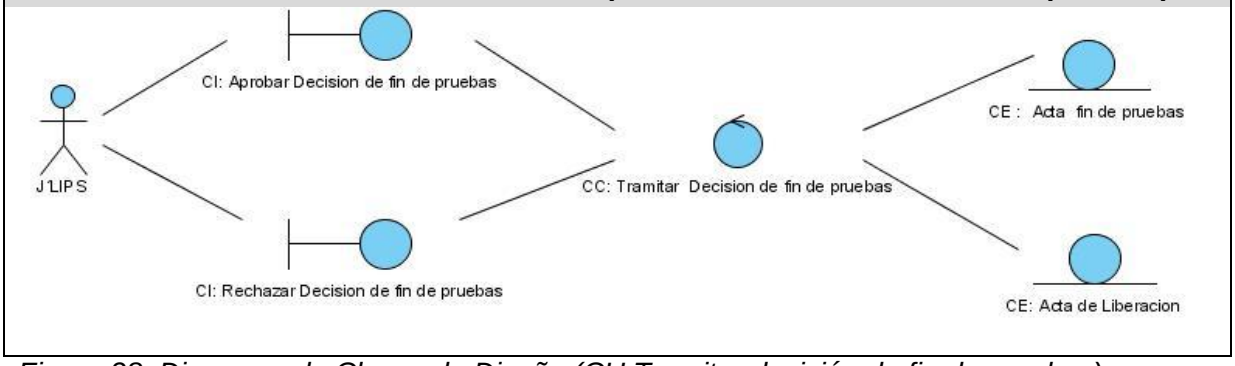


Figura 28: Diagrama de Clases de Diseño (CU Tramitar decisión de fin de pruebas)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Gestionar Criterios de Criticidad)

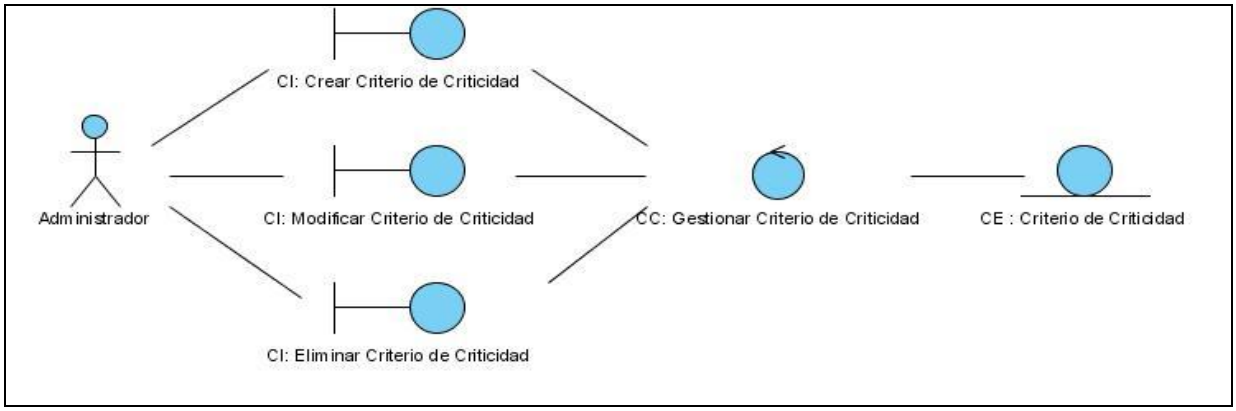


Figura 29: Diagrama de Clases de Diseño (CU Gestionar Criterios de Criticidad)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Definir Tipos de Prueba X Artefacto)

Figura 30: Diagrama de Clases de Diseño (CU Definir Tipos de Prueba X Artefacto)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Gestionar Listas de Chequeo)

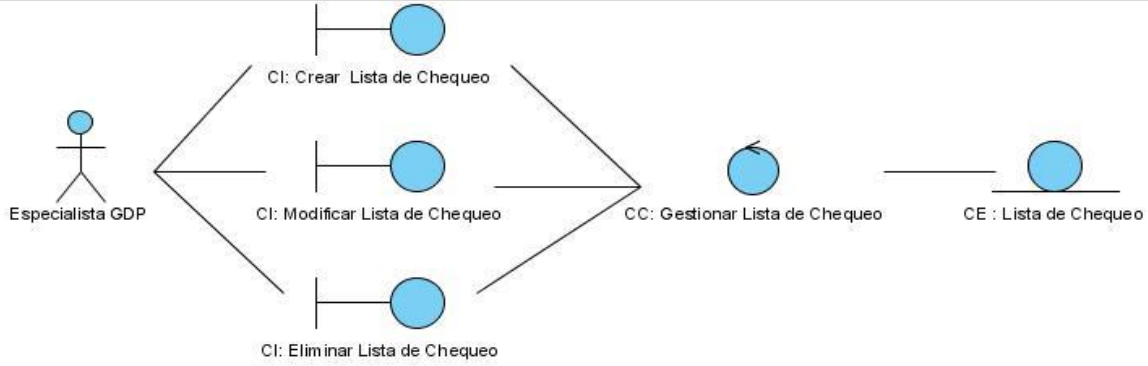


Figura 31: Diagrama de Clases de Diseño (CU Gestionar Listas de Chequeo)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Definir Tipo de Herramienta X Tipo de Prueba)

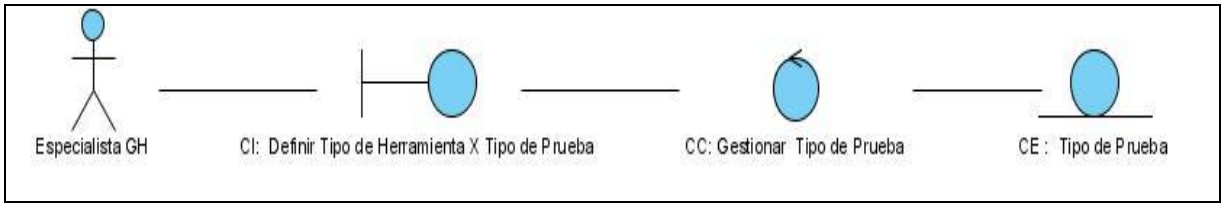


Figura 32: Diagrama de Clases de Diseño (CU Definir Tipo de Herramienta X Tipo de Prueba)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Gestionar Evaluación PE)

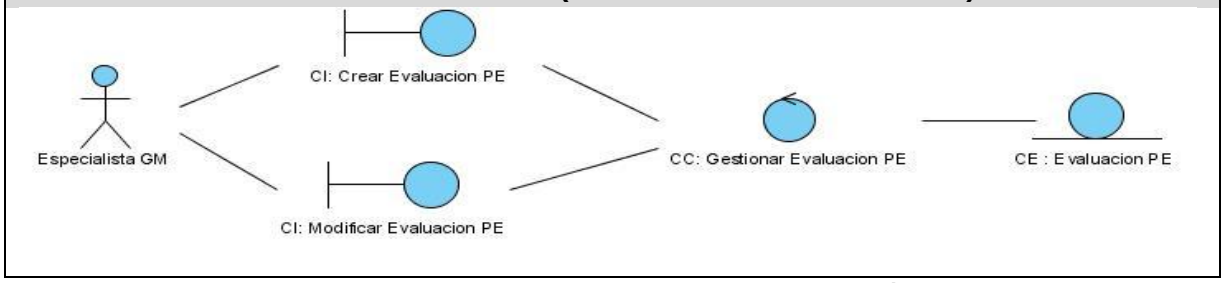


Figura 33: Diagrama de Clases de Diseño (CU Gestionar Evaluación PE)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Cargar Artefacto X Proyecto)

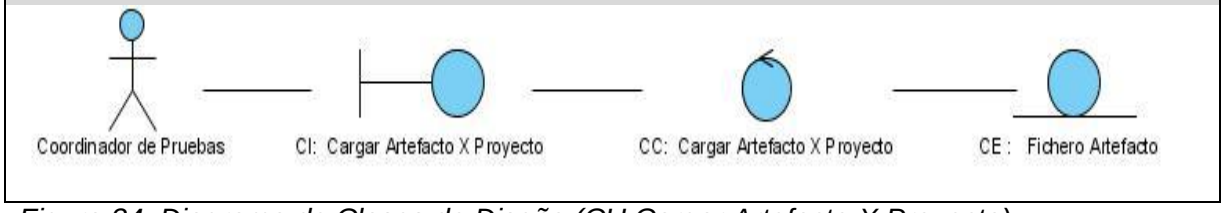


Figura 34: Diagrama de Clases de Diseño (CU Cargar Artefacto X Proyecto)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Gestionar Solicitudes de Prueba)

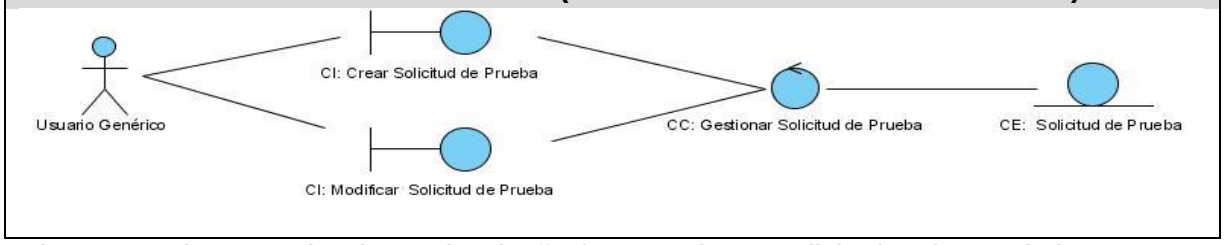


Figura 35: Diagrama de Clases de Diseño (CU Gestionar Solicitudes de Prueba)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Tramitar Estado de Solicitud de Pruebas)

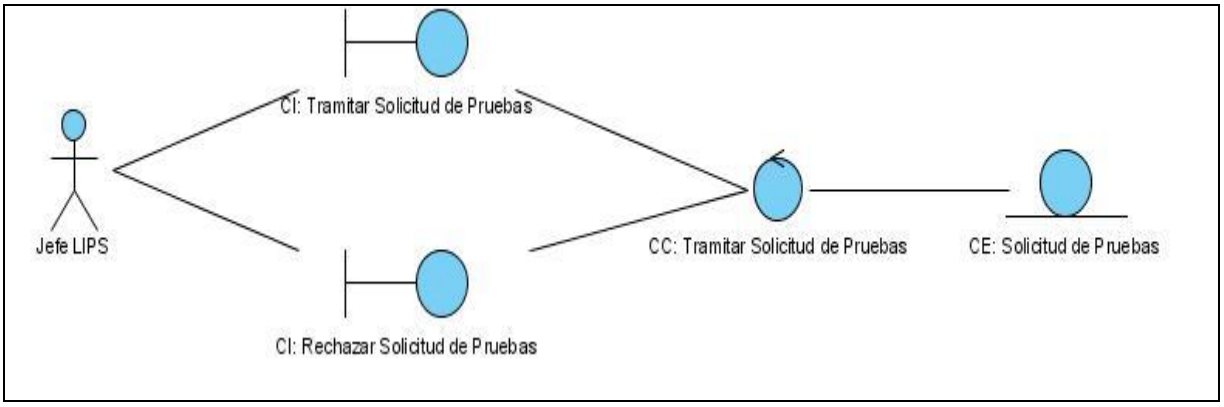


Figura 36: Diagrama de Clases de Diseño (CU Tramitar Estado de Solicitud de Pruebas)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Crear Citación de Reuniones)

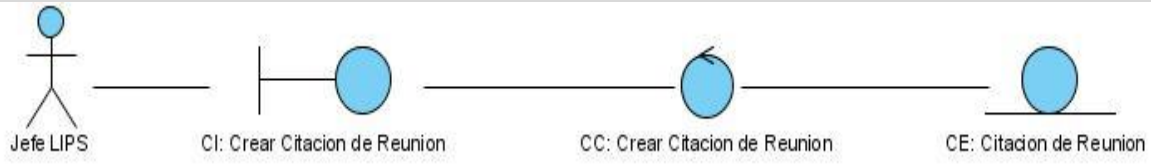


Figura 37: Diagrama de Clases de Diseño (CU Crear Citación de Reuniones)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Confeccionar Acta de Reunión)

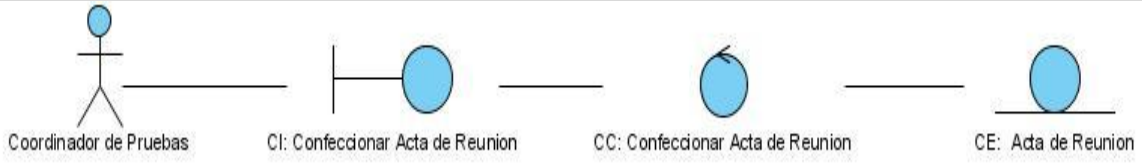


Figura 38: Diagrama de Clases de Diseño (CU Confeccionar Acta de Reunión)

DIAGRAMA DE CLASES DE ANÁLISIS (CU Modificar Pre-Plan de Prueba)

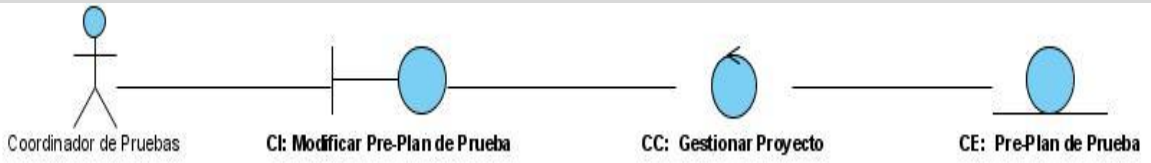


Figura 39: Diagrama de Clases de Diseño (CU Modificar Pre-Plan de Prueba)

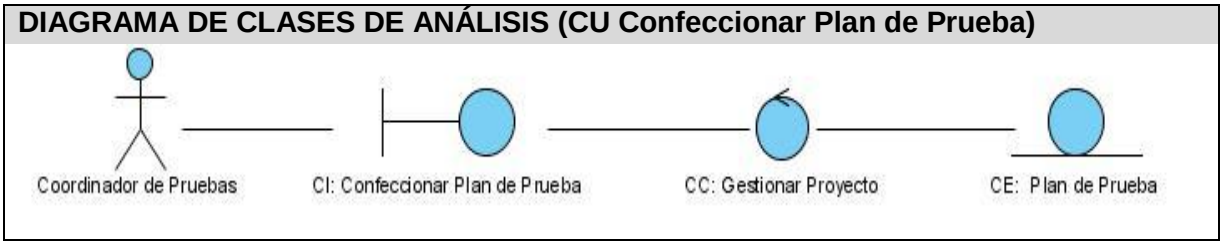


Figura 40: Diagrama de Clases de Diseño (CU Confeccionar Plan de Prueba)

3.3. Diseño del Sistema.

El diseño del sistema es otro flujo de trabajo que se realiza al final de la fase de elaboración, el cual permite, mediante el modelo de diseño describir la realización física de los casos de uso por las clases de diseño y sus objetos, centrándose en el impacto que tienen en el sistema los requisitos funcionales, no funcionales y otras restricciones relacionadas con el entorno de implementación. A continuación se describen los tipos principales de clases que serán utilizadas durante la implementación del sistema, para ello se precisa analizar la arquitectura definida para el mismo, la cual indica la estructura, el funcionamiento y la interacción entre las partes del software.(PRESSMAN, 1997)

3.3.1. Arquitectura de Software.

Al ser el sistema una aplicación web, la arquitectura del mismo está basada fundamentalmente en el estilo arquitectónico en capas, permitiendo orientar el diseño de la aplicación a través de capas lógicas, de forma que se prevean problemas de escalabilidad, seguridad, disponibilidad, integración, etcétera, obteniendo aplicaciones más robustas, flexibles, fáciles de mantener y dar soporte, siendo mucho más sencillo cambiar un componente que una aplicación monolítica. Este estilo permite además desacoplar dichas capas lógicas, lo anterior significa que cada una puede ser modificada tanto como sea posible sin provocar un impacto no deseado en las demás. Una capa no es consciente de lo que le ocurre a la capa superior, su responsabilidad radica solo en proporcionarle servicios; su dependencia es puramente con la capa inmediatamente inferior, sirviéndose de las prestaciones que le brinda la misma. Esta dependencia entre capas es normalmente entre interfaces, asegurando que el acople sea el más bajo posible.

Seguidamente se analiza en más detalle cada una de las capas lógicas en las que se encuentra dividido el sistema, comenzando por las capas inferiores hacia las superiores.

Capa de Acceso a Datos

Manteniendo la filosofía de que es mejor programar orientado a interfaces que a clases, pues su adecuada aplicación puede hacer que el diseño de las aplicaciones sea más elegante, escalable y sostenible, posibilitando la reutilización, la adaptabilidad y la facilidad de cambio en los mismos; se desprende la necesidad de que la arquitectura definida soporte el uso de interfaces de acceso a datos entre la capa de servicios de negocio y el API de persistencia utilizado, en este caso Hibernate.

El término Objeto de Acceso a Datos o Data Access Object (DAO), en inglés, es ampliamente difundido y utilizado en el desarrollo de aplicaciones web.

Los DAOs encapsulan la persistencia de los objetos de dominios, proveen la persistencia de los objetos transitorios y las actualizaciones de los objetos existentes en la base de datos. Las implementaciones de los DAOs estarán disponibles para los objetos (típicamente para los objetos de negocio: aquellos que contienen detalles específicos de operación o aplicación) haciendo uso de la inyección de dependencias con los objetos de negocio y las instancias de los DAOs.

Las interfaces de los DAOs exponen a la *capa de servicios de negocio* los métodos del DAO para persistir en la base de datos una determinada entidad de dominio, básicamente los tipos de métodos son los siguientes (PÉREZ, 2007):

- **Métodos para descubrir**: Estos localizan los objetos almacenados para ser usados por la capa de servicios de negocio.
- **Métodos para persistir o salvar**: Estos hacen persistentes a los objetos transitorios.
- **Métodos para eliminar**: Estos eliminan a los objetos guardados en el medio de almacenamiento (generalmente una base de datos).
- **Métodos para conteos y otras funciones agregadas**: Estos devuelven los resultados de operaciones que son más eficientes implementarlas usando funcionalidades de la base de datos (procedimientos almacenados, etcétera.) que iterar sobre los mismos objetos.

Clases de la Capa Acceso a Datos

- En esta capa sólo van a encontrarse las definiciones y las implementaciones reales de las Interfaces DAOs, para estas últimas se definió el siguiente estándar de codificación: comienzan con el nombre de la interfaz correspondiente y terminan con abreviatura “Impl”. Ejemplo: EstudianteDAOImpl. Estas clases son las que realmente se conectan a la base de datos usando Hibernate.

Capa de Servicios de Negocio

Esta capa contiene toda la lógica que modela los procesos de negocio, es donde se realiza todo el procesamiento necesario para solventar las peticiones realizadas por los usuarios.

Es en esta capa donde radican los objetos de negocio o Business Objects (DEEPAK ALUR, 2003). Los objetos de negocio separan los datos y la lógica de negocio usando un modelo de objetos. Al igual que en la capa de acceso a datos, se definen interfaces, las cuales exponen a la *capa de presentación* los métodos necesarios para las operaciones del negocio sobre una determinada entidad del dominio. A modo de aclaración: puede ocurrir que un objeto de negocio exponga métodos de negocio de varias entidades de dominio o de procesos específicos de negocio.

Funcionalidades específicas de esta capa (PÉREZ, 2007):

- **Lógica de Negocios específica de Procesos de Negocio**: En determinadas ocasiones lo óptimo es que las entidades de dominio contengan la lógica de negocio aplicable a un conjunto de casos de uso de la aplicación**.** Sin embargo, existen casos de uso específicos que precisan ser realizados en la capa de servicios de negocio, de las situaciones anteriormente expuestas, se llegó a un consenso, plasmado en la arquitectura definida donde se propuso que: las entidades de dominio no presentan ningún tipo de lógica de negocio, esta responsabilidad recae sobre los objetos de negocio, permitiendo usar a los objetos de dominio como Transfer Objects que se mueven entre las capas arquitectónicas de la aplicación.

- **Puntos de entrada muy bien definidos para las operaciones de negocio implementadas:** Los objetos de negocio brindan las interfaces usadas por la capa de presentación.

- **Ejecución de restricciones de seguridad**: Las restricciones de seguridad en esta capa están en los puntos de entradas a la capa media, es decir en los objetos de negocio.
- **Control de Transacciones**: Las políticas transaccionales de la aplicación son planteadas al nivel de los objetos de negocio.

Clase de la Capa de Servicios de Negocio

- Es en esta capa donde se encuentran las definiciones y las implementaciones reales (Objetos de Negocio) de las interfaces de negocio, para estas últimas se aplica el mismo estándar de codificación definido para la capa de acceso a datos: comienzan con el nombre de la interfaz correspondiente y terminan con abreviatura “Impl”. Ejemplo: GestionarEstudianteImpl. Esta clase es la que desarrolla los métodos que contendrá la lógica de negocio correspondiente a la entidad del dominio. Ejemplo: Estudiante.
- Entidad del Dominio: clase que contienen todos los datos, no tienen lógica de negocio. Pueden ser objetos (individuales o colecciones).

Capa de Presentación

Es la capa encargada de generar la interfaz de usuario en función de las acciones llevadas a cabo por él mismo. Esta capa descansa sobre una capa de servicios de negocio. Esto significa que la misma será fina y no contendrá lógica de negocio, sino simplemente lo concerniente a aspectos de presentación, por ejemplo, el código para manipular las interacciones web. Por tanto, se pueden elegir más de una capa de presentación en la misma aplicación, sin impactar en las capas arquitectónicas inferiores, las cuales no tienen conocimiento sobre la capa de presentación.

Capa Web

Esta capa web es la responsable de tratar con las interacciones del usuario y obtener los datos que pueden ser mostrados en un formato determinado. Normalmente está compuesta por tres tipos de objetos:

- **Controladores**: Estos objetos son responsables de procesar las peticiones del usuario, invocando las funcionalidades necesarias, expuestas por la capa de servicios de negocio y devolviendo un modelo requerido para ser mostrado.
- **Modelo**: Estos objetos contienen los datos resultantes de la ejecución de la lógica de negocio, los cuales deberían ser mostrados en la respuesta.
- **Vistas**: Estos objetos son responsables de mostrar el modelo en la respuesta de la petición. La forma de mostrar el modelo podrá ser de diferentes tipos de vistas, por ejemplo, archivos JSP, HTML, PDF, documentos de Excel, etcétera. Las vistas no son responsables de modificar los datos o incluso de obtener los datos; estas simplemente sirven para mostrar los datos del modelo que han sido suministrados por un controlador.

La combinación de los objetos anteriormente expuestos, no es más que la aplicación del patrón arquitectónico MVC (Model View Controller) (MARTIN FOWLER, 2002).

Clases de la Capa Presentación

Como se está modelando una aplicación web se hace uso de las extensiones brindadas por UML para el modelado de este tipo de soluciones informáticas, presentando como elementos más significativos a 3 clases con los siguientes estereotipos “Server Page”, “Client Page”, “Form” empleados para el código servidor, código cliente y formularios respectivamente.

<<Client Page>> Son las páginas que van a funcionar como interfaz a los usuarios (*Vistas*). Se construirán dinámicamente para ser visualizadas en el explorador de los usuarios.

<<Server Page>> Son las páginas servidoras que construyen a las páginas clientes y tienen toda la lógica de presentación (*Controladoras*). Invocan todos los métodos necesarios de la capa de servicio de negocio a través de objetos de negocio.

<<Form>> Colección de elementos de entrada que son parte de un página cliente. Se relaciona directamente con la etiqueta de igual nombre del HTML. Sus atributos son los elementos de entrada del formulario (input boxes, text areas, radio buttons, check boxes y hidden fields).

Una vez definida los tipos de clases que contienen cada una de las capas lógicas de la aplicación, se procede a mostrar cómo interactúan sus correspondientes objetos del diseño a través de diagramas de clases y diagramas de secuencia para realizar y llevar a cabo los casos de uso previamente capturados.

3.3.2. Diagramas de Clases.

A través de este tipo de diagramas se identifican las clases, así como sus relaciones necesarias para realizar los casos de uso.

DIAGRAMA DE CLASES DEL DISEÑO (CU Gestionar Perfil: Sección Crear Perfil)

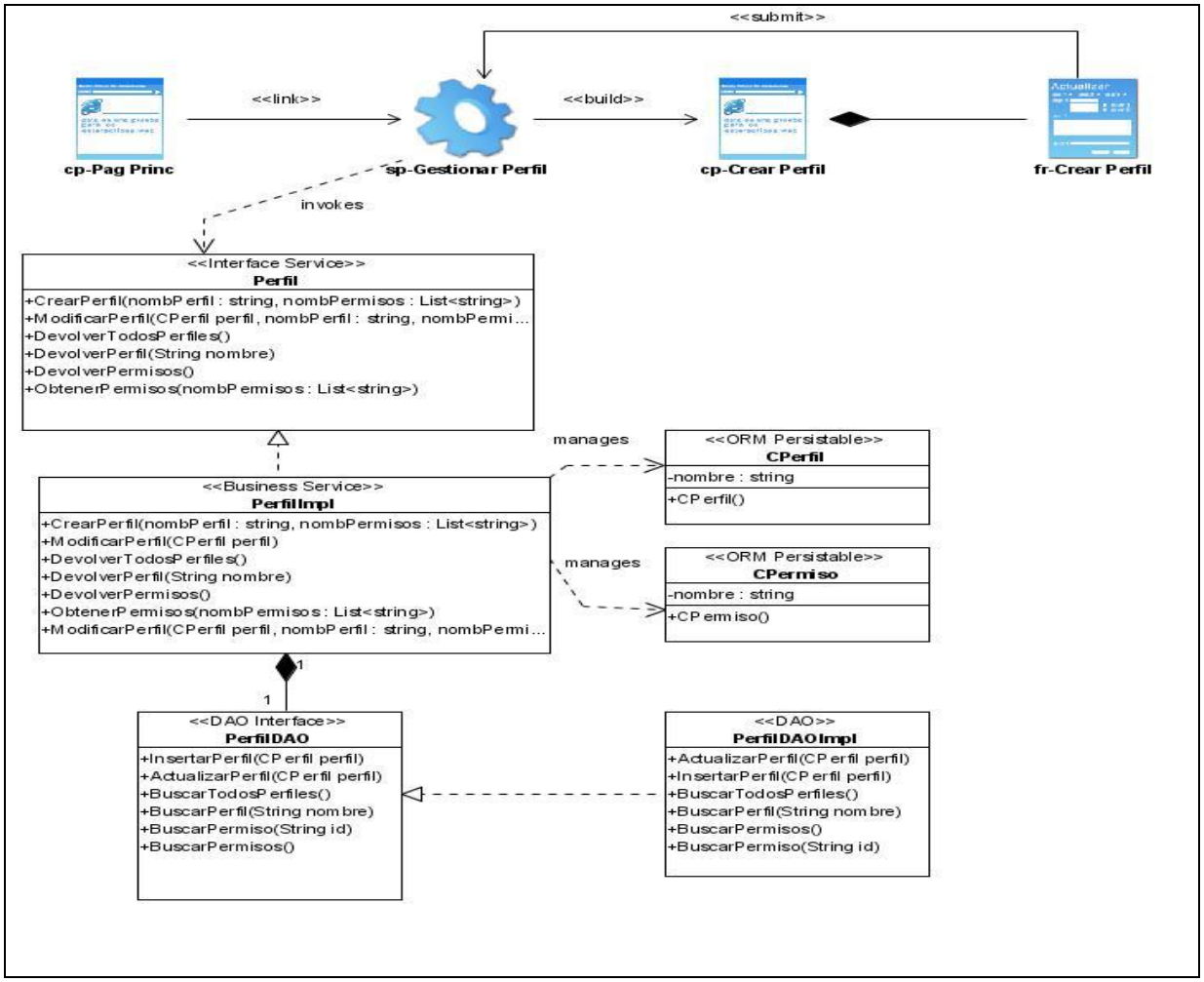


Figura:41 Diagrama de Clases del Diseño (CU Gestionar Perfil)

DIAGRAMA DE CLASES DEL DISEÑO (CU Gestionar Perfil: Sección Modificar Perfil)

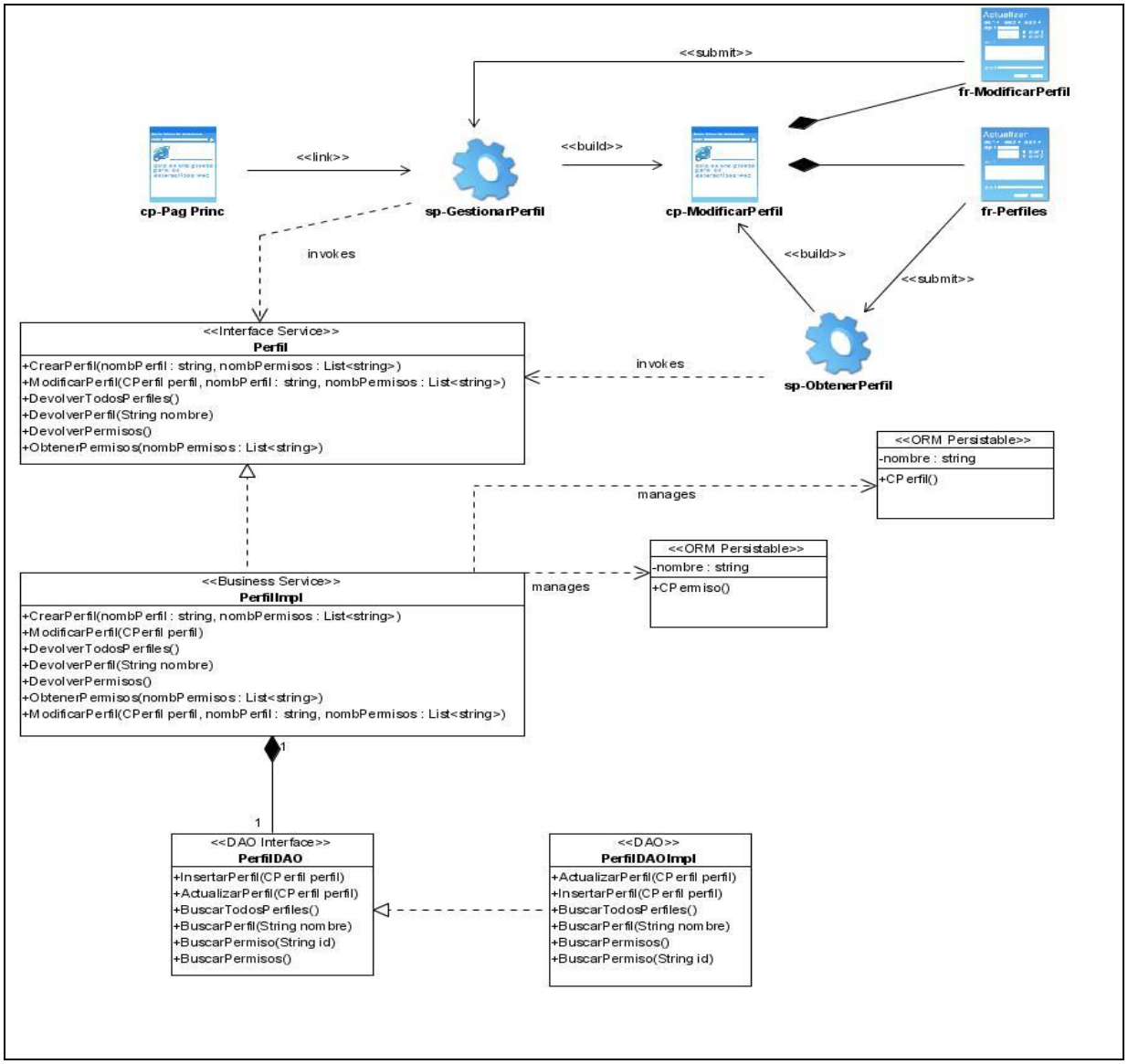


Figura 42: Diagrama de Clases del Diseño (CU Gestionar Perfil)

DIAGRAMA DE CLASES DEL DISEÑO (CU Gestionar Puestos de Trabajo: Sección Crear Puesto de Trabajo)

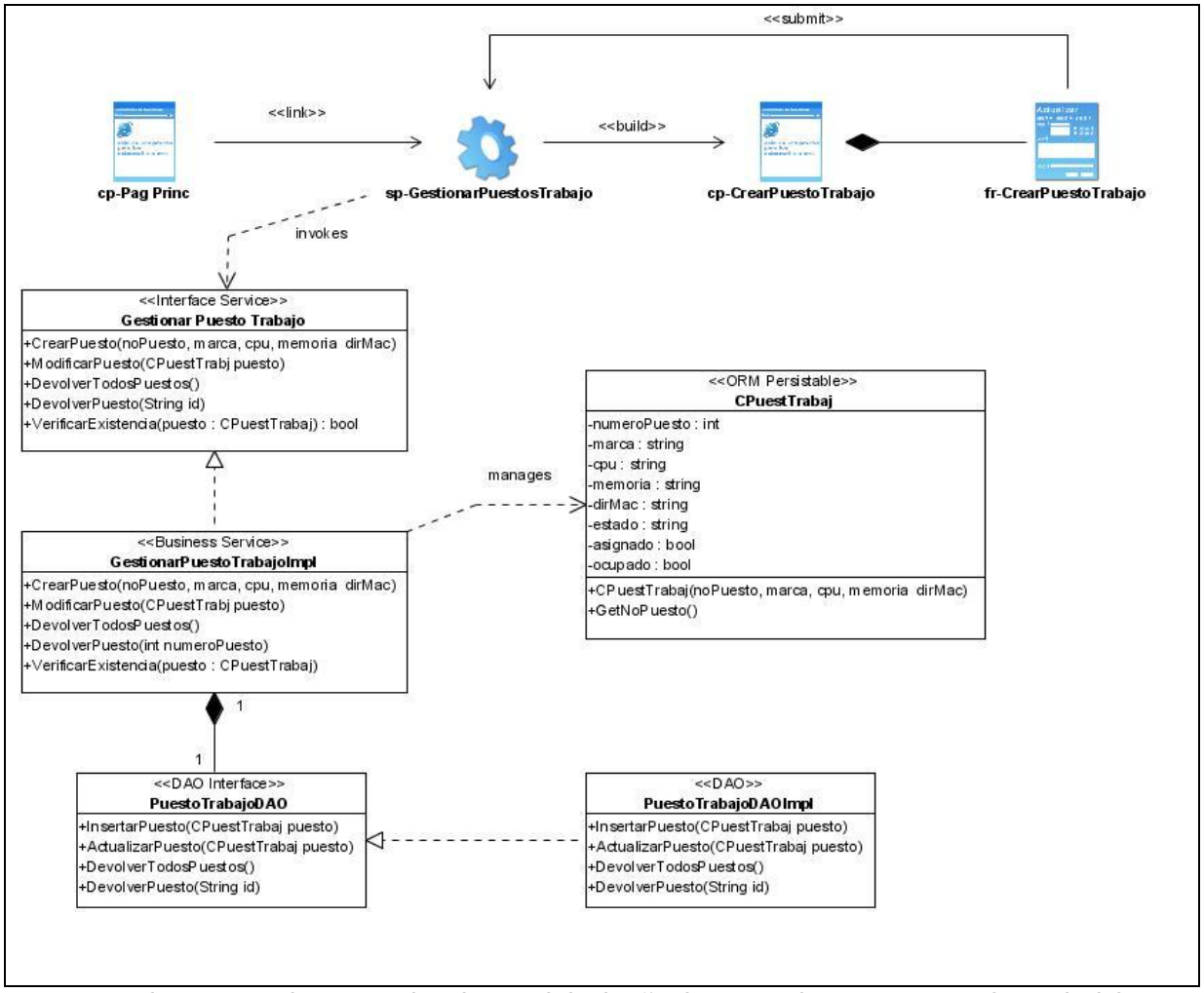


Figura 43: Diagrama de Clases del Diseño (CU Gestionar Puestos de Trabajo)

DIAGRAMA DE CLASES DEL DISEÑO (CU Gestionar Puestos de Trabajo: Sección Modificar Puesto de Trabajo)

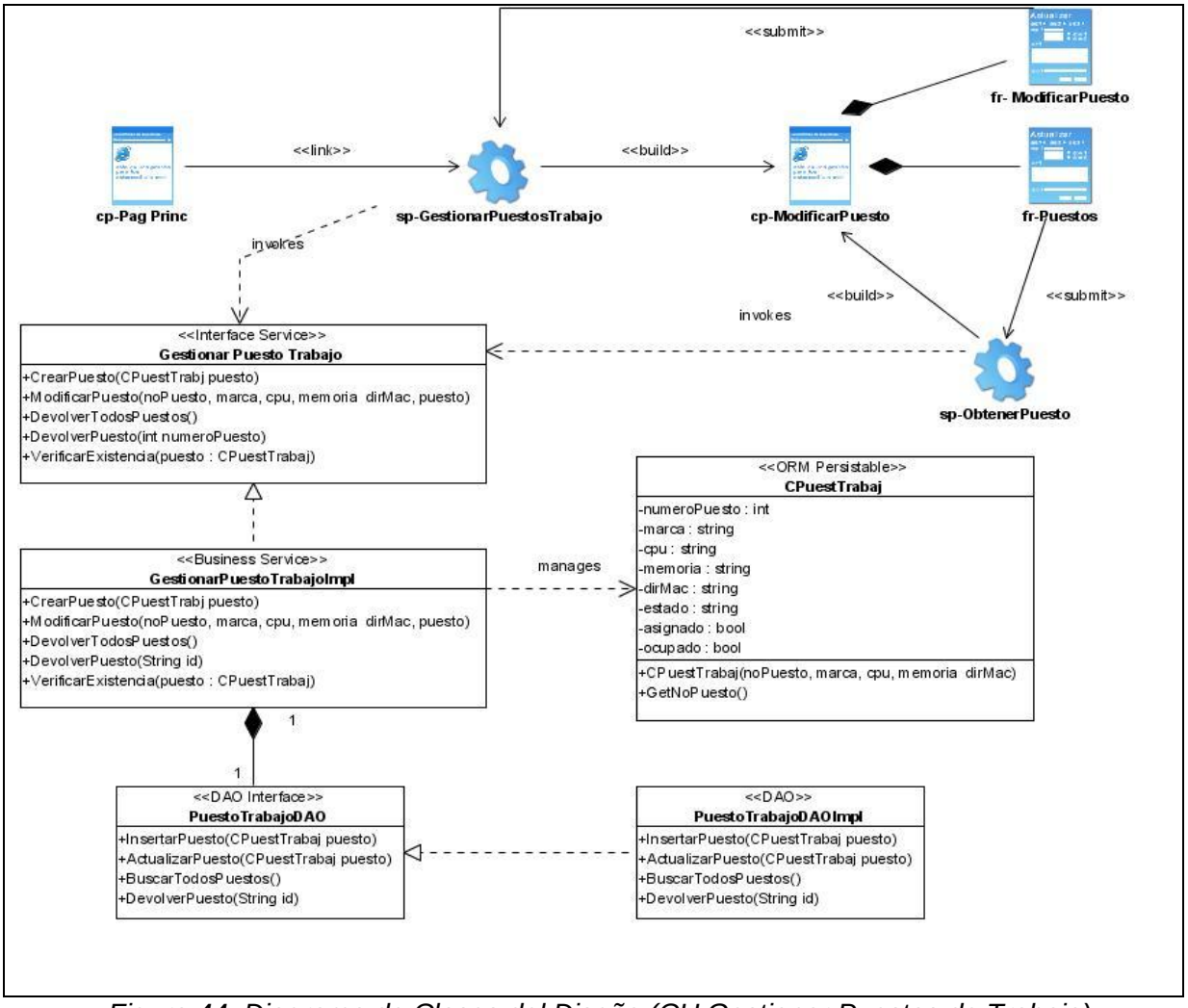


Figura 44: Diagrama de Clases del Diseño (CU Gestionar Puestos de Trabajo)

DIAGRAMA DE CLASES DEL DISEÑO CU Conocer Puesto de Trabajo

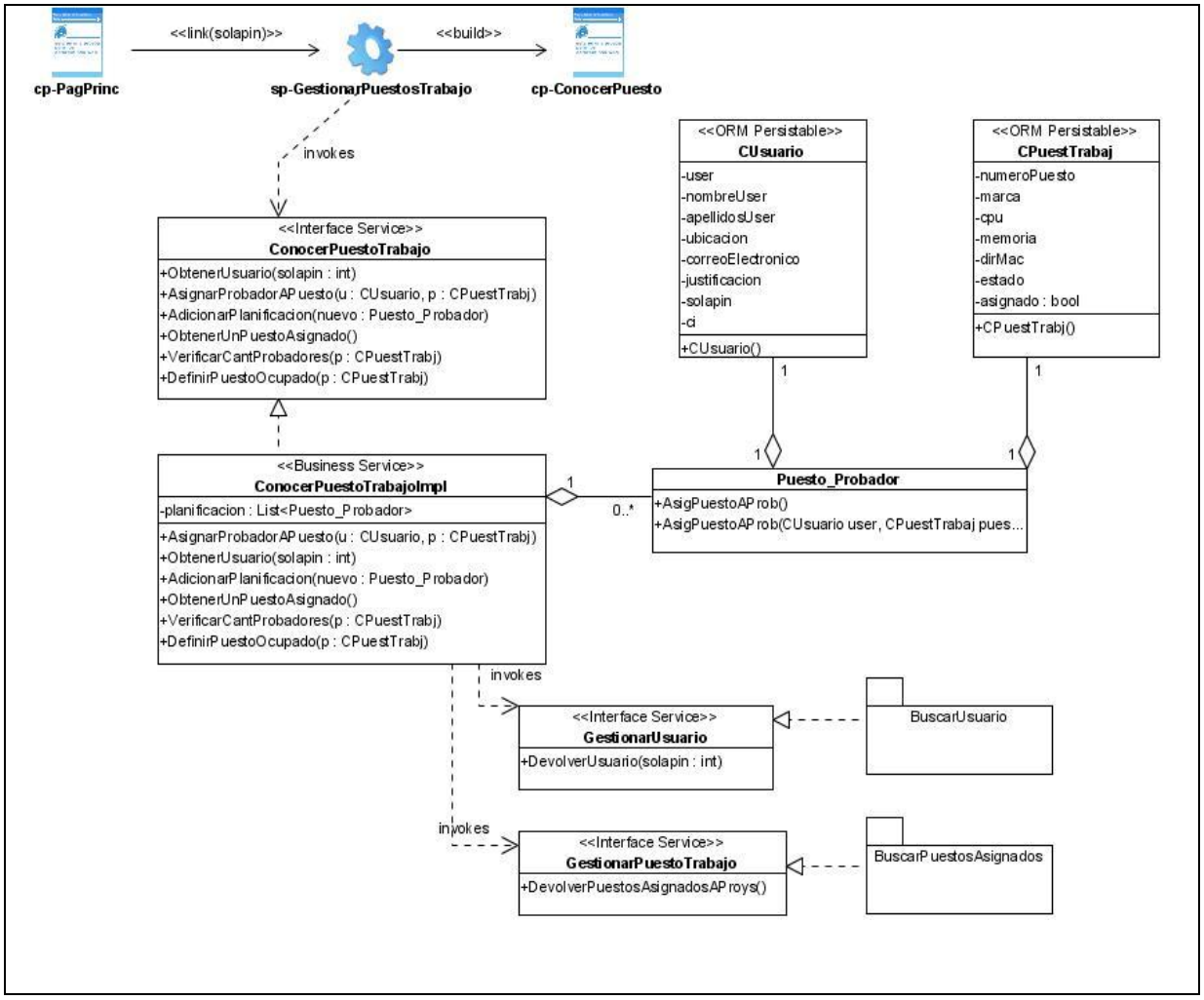


Figura 45: Diagrama de Clases del Diseño (CU Conocer Puesto de Trabajo)

DIAGRAMA DE CLASES DEL DISEÑO (CU Gestionar Solicitud de Inclusión de Usuario: Sección Crear Solicitud de Inclusión de Usuario)

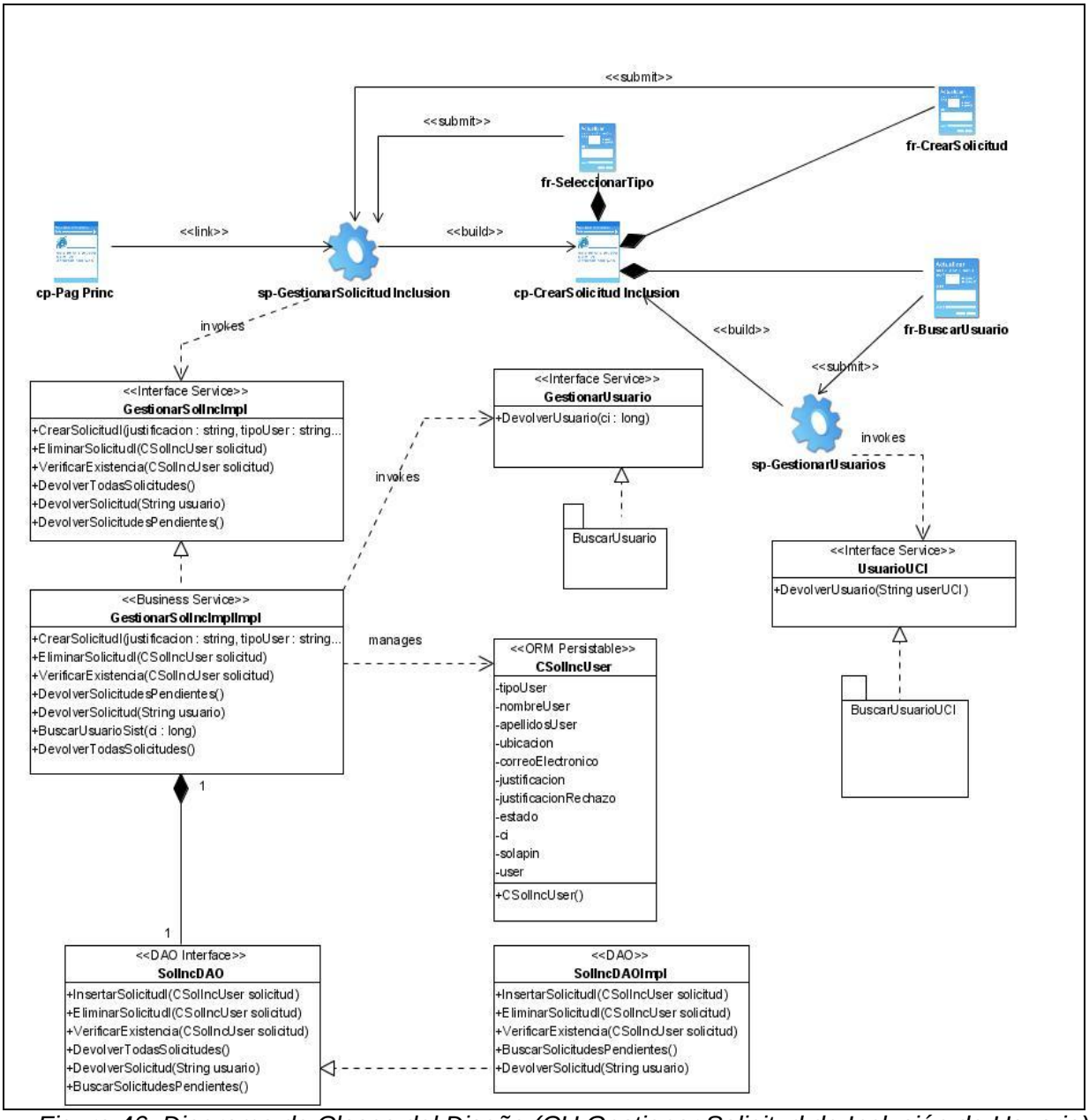


Figura 46: Diagrama de Clases del Diseño (CU Gestionar Solicitud de Inclusión de Usuario)

DIAGRAMA DE CLASES DEL DISEÑO (CU Gestionar Solicitud de Inclusión de Usuario: Sección Eliminar Solicitud de Inclusión de Usuario)

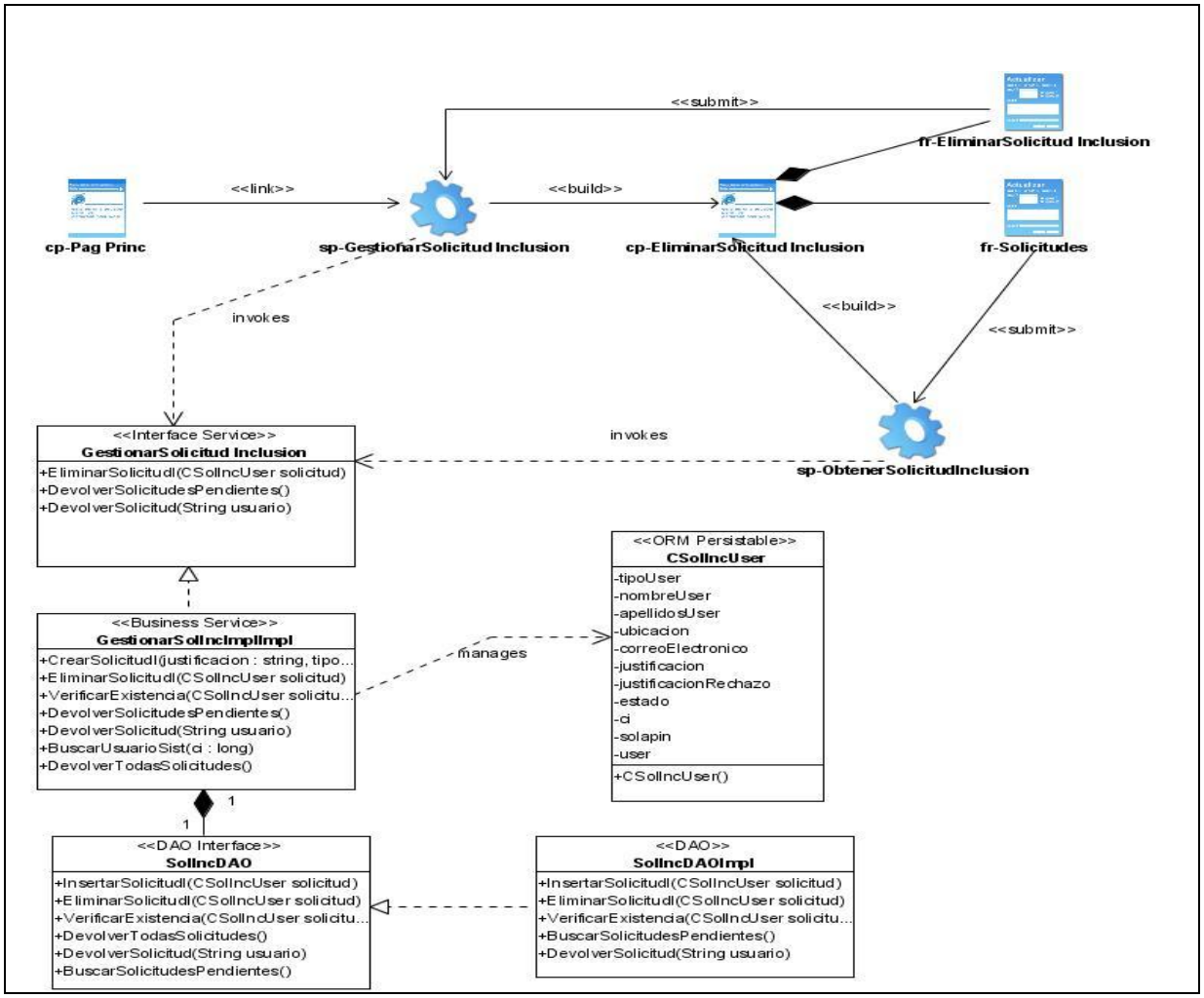


Figura 47: Diagrama de Clases del Diseño (CU Gestionar Solicitud de Inclusión de Usuario)

DIAGRAMA DE CLASES DEL DISEÑO (CU Tramitar estado de Solicitud de Inclusión de Usuario: Sección Aprobar Solicitud de Inclusión de Usuario)

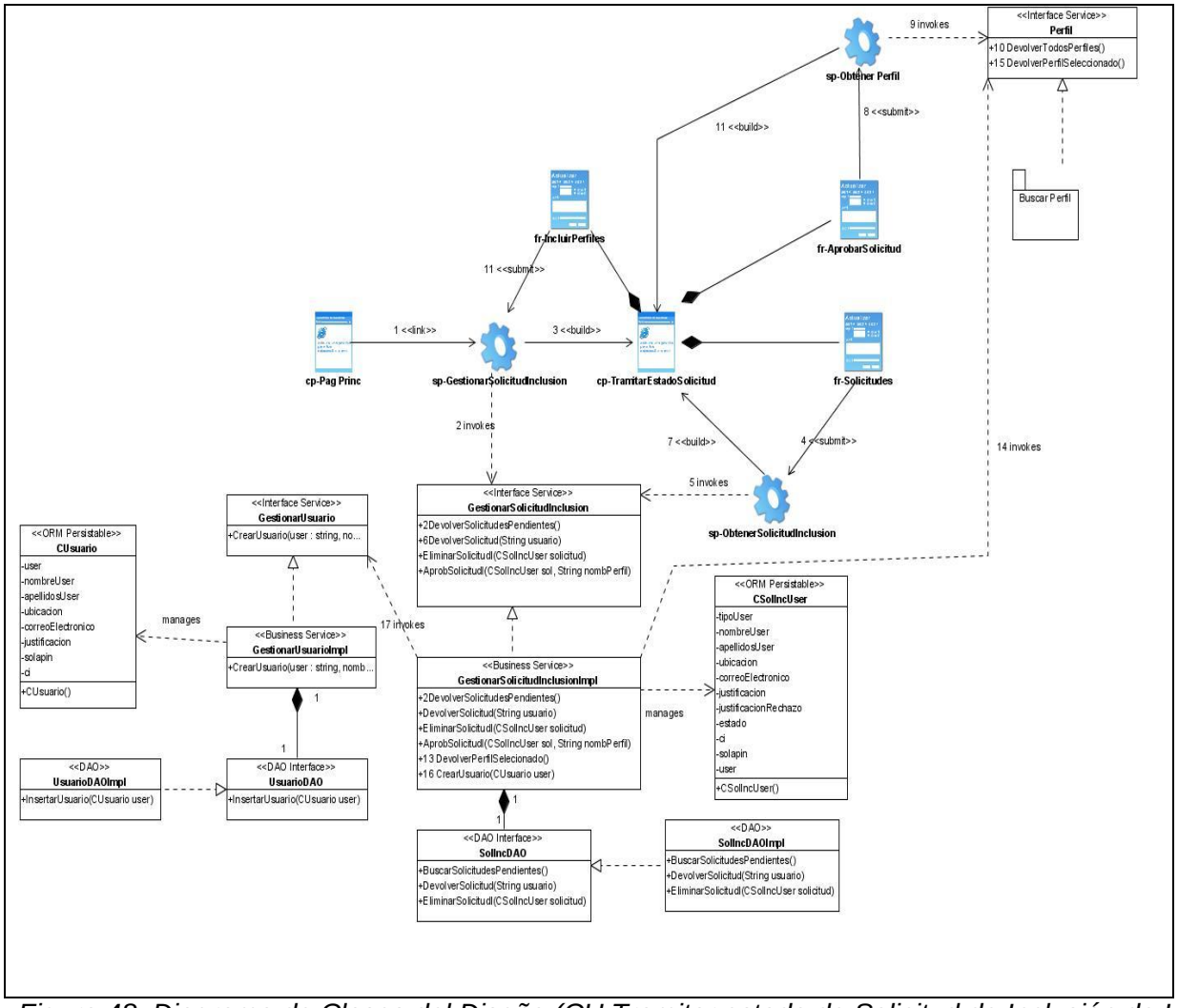


Figura 48: Diagrama de Clases del Diseño (CU Tramitar estado de Solicitud de Inclusión de Usuario)

DIAGRAMA DE CLASES DEL DISEÑO (CU Tramitar estado de Solicitud de Inclusión de Usuario: Sección Rechazar Solicitud de Inclusión de Usuario)

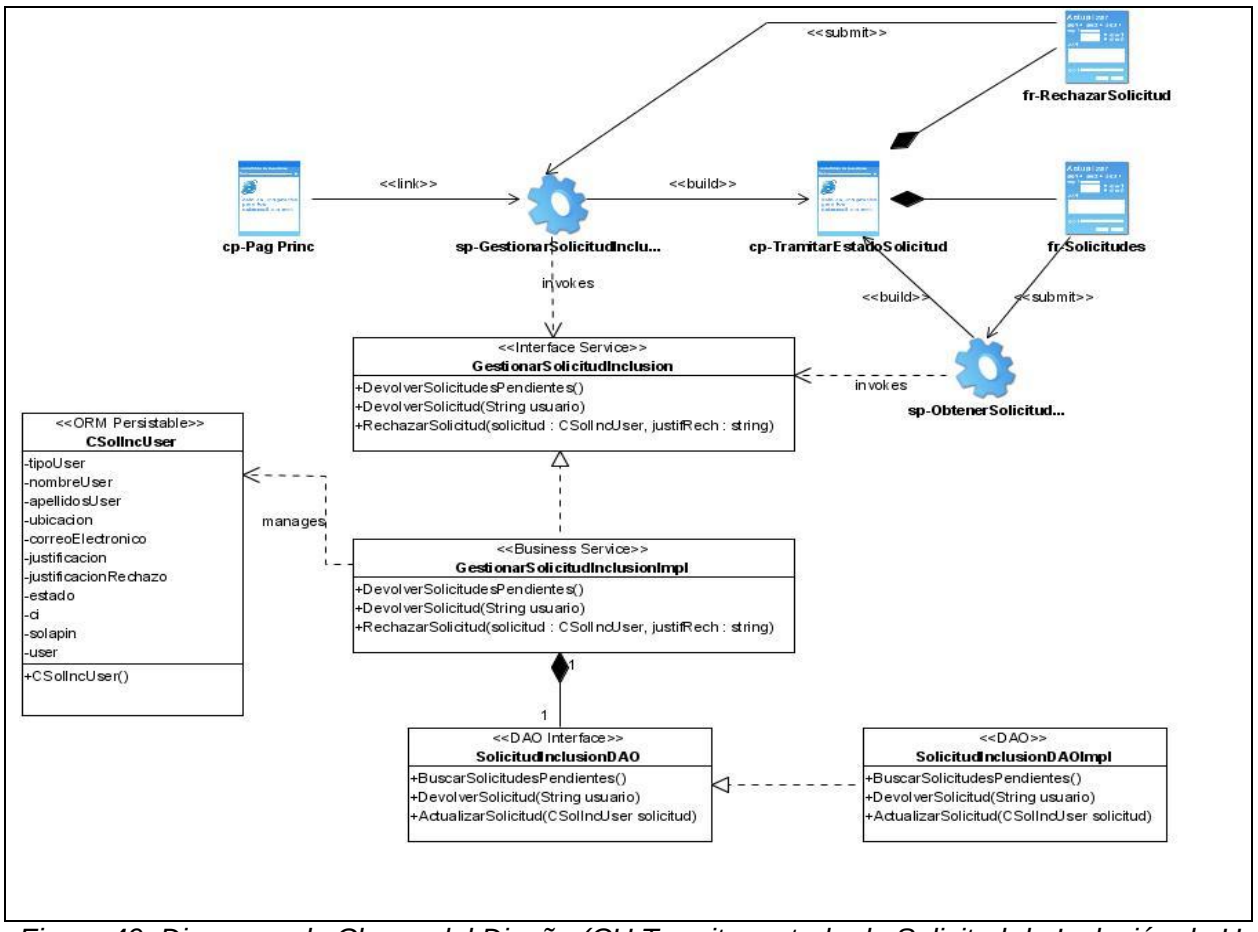


Figura 49: Diagrama de Clases del Diseño (CU Tramitar estado de Solicitud de Inclusión de Usuario)

DIAGRAMA DE CLASES DEL DISEÑO (CU Gestionar Usuario: Modificar Usuario)

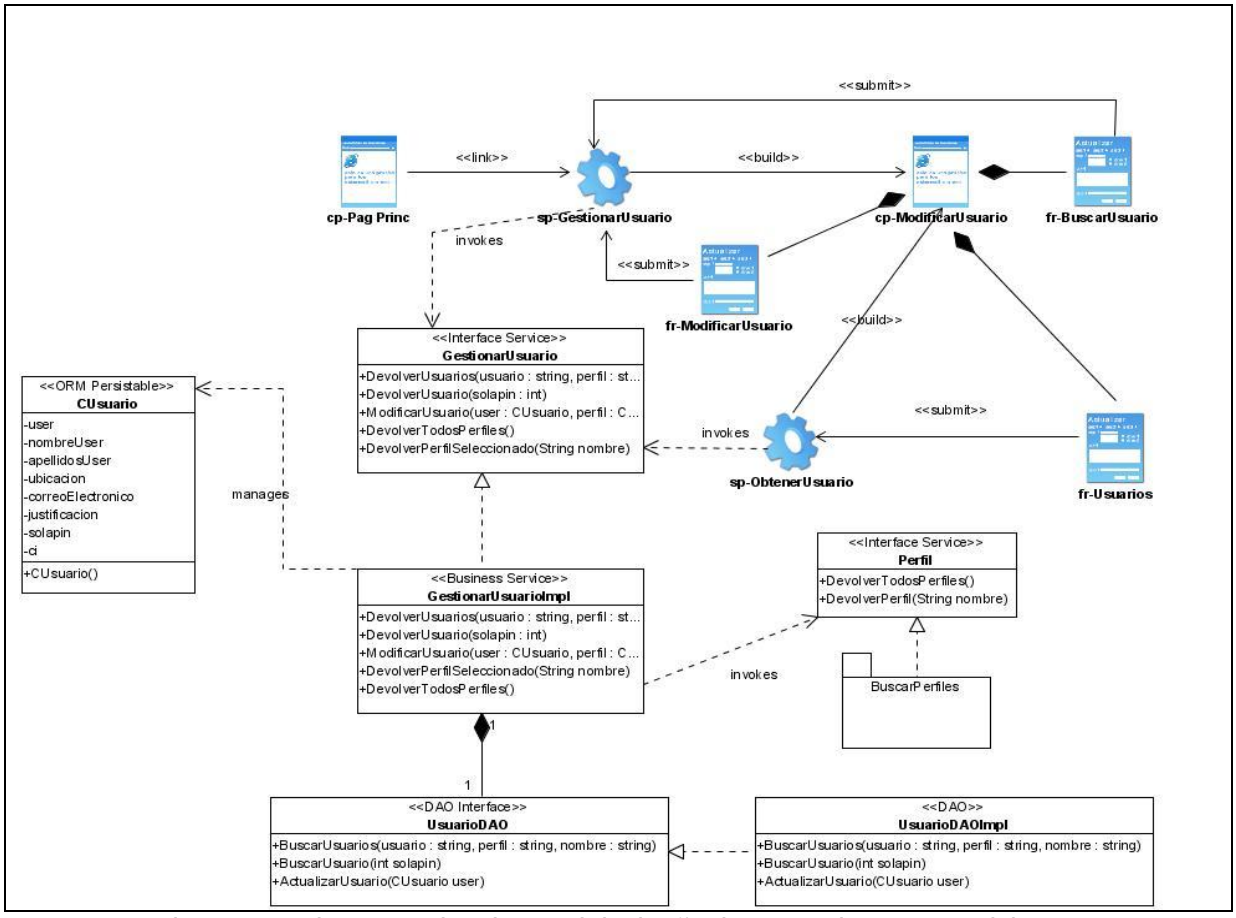


Figura 50: Diagrama de Clases del Diseño (CU Gestionar Usuario)

DIAGRAMA DE CLASES DEL DISEÑO (CU Gestionar Usuario: Eliminar Usuario)

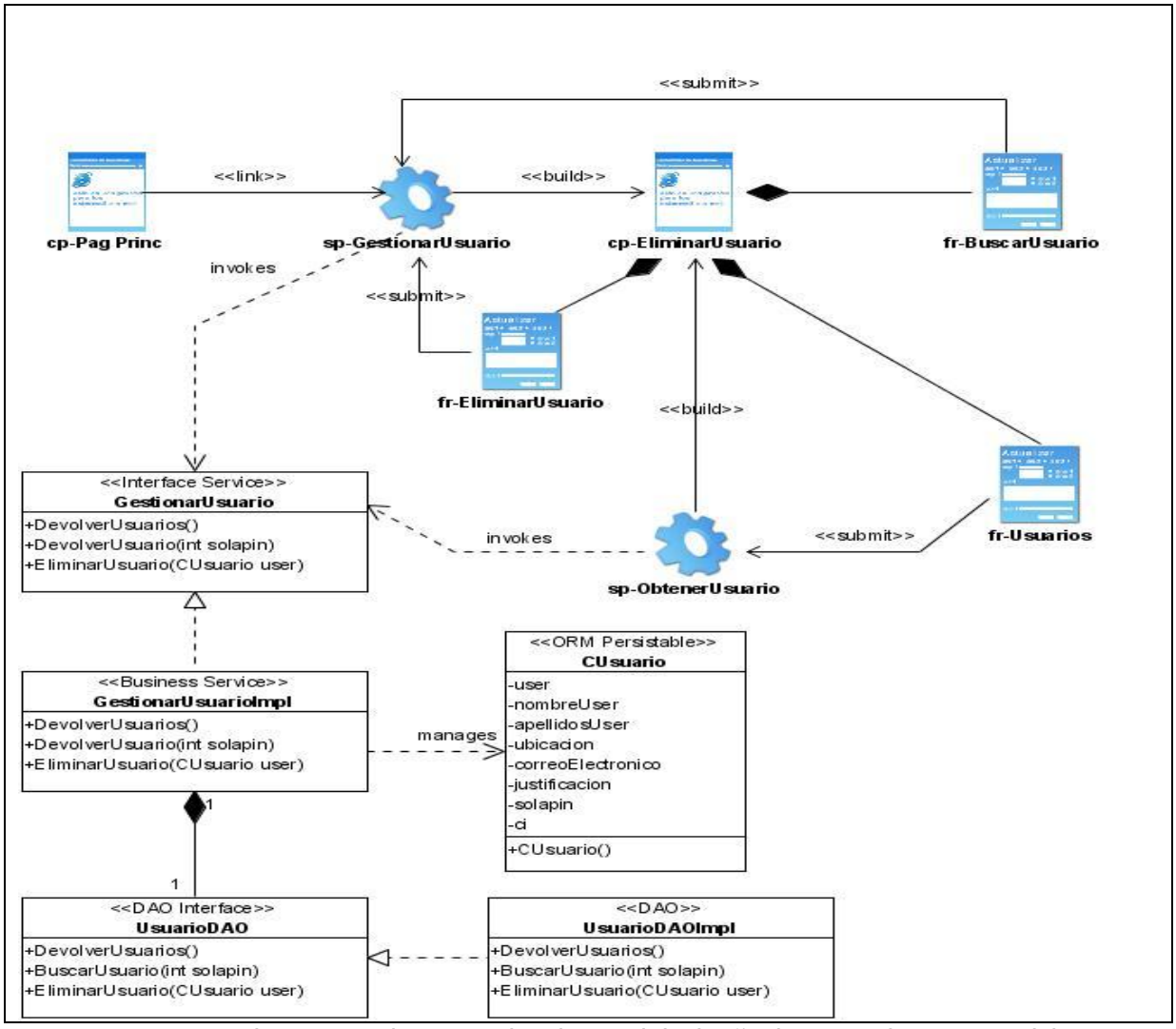


Figura 51: Diagrama de Clases del Diseño (CU Gestionar Usuario)

DIAGRAMA DE CLASES DEL DISEÑO (CU Autenticar Usuario)

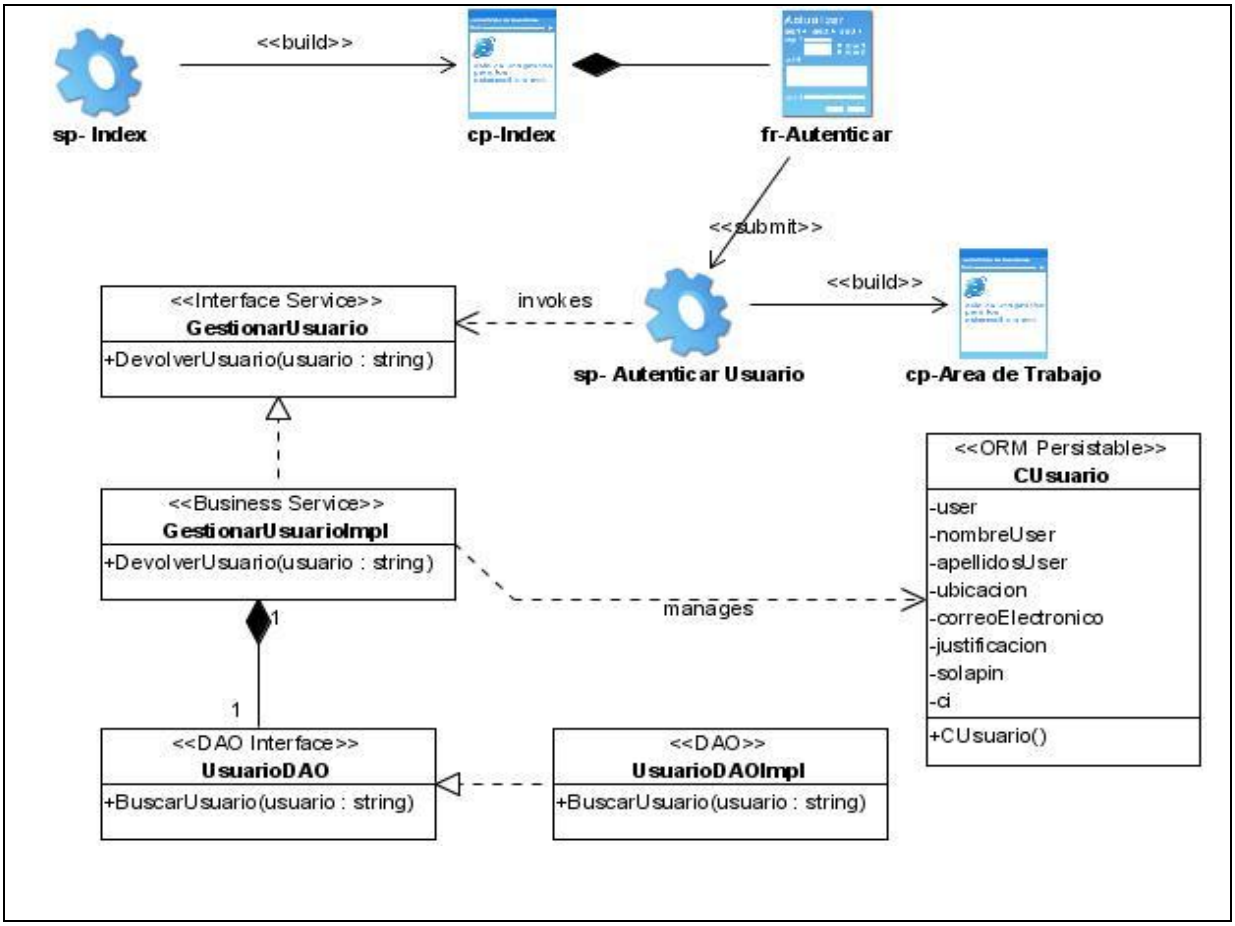


Figura 52: Diagrama de Clases del Diseño (CU Autenticar Usuario)

DIAGRAMA DE CLASES DEL DISEÑO (CU Probar Artefacto Asignado: Crear No Conformidad)

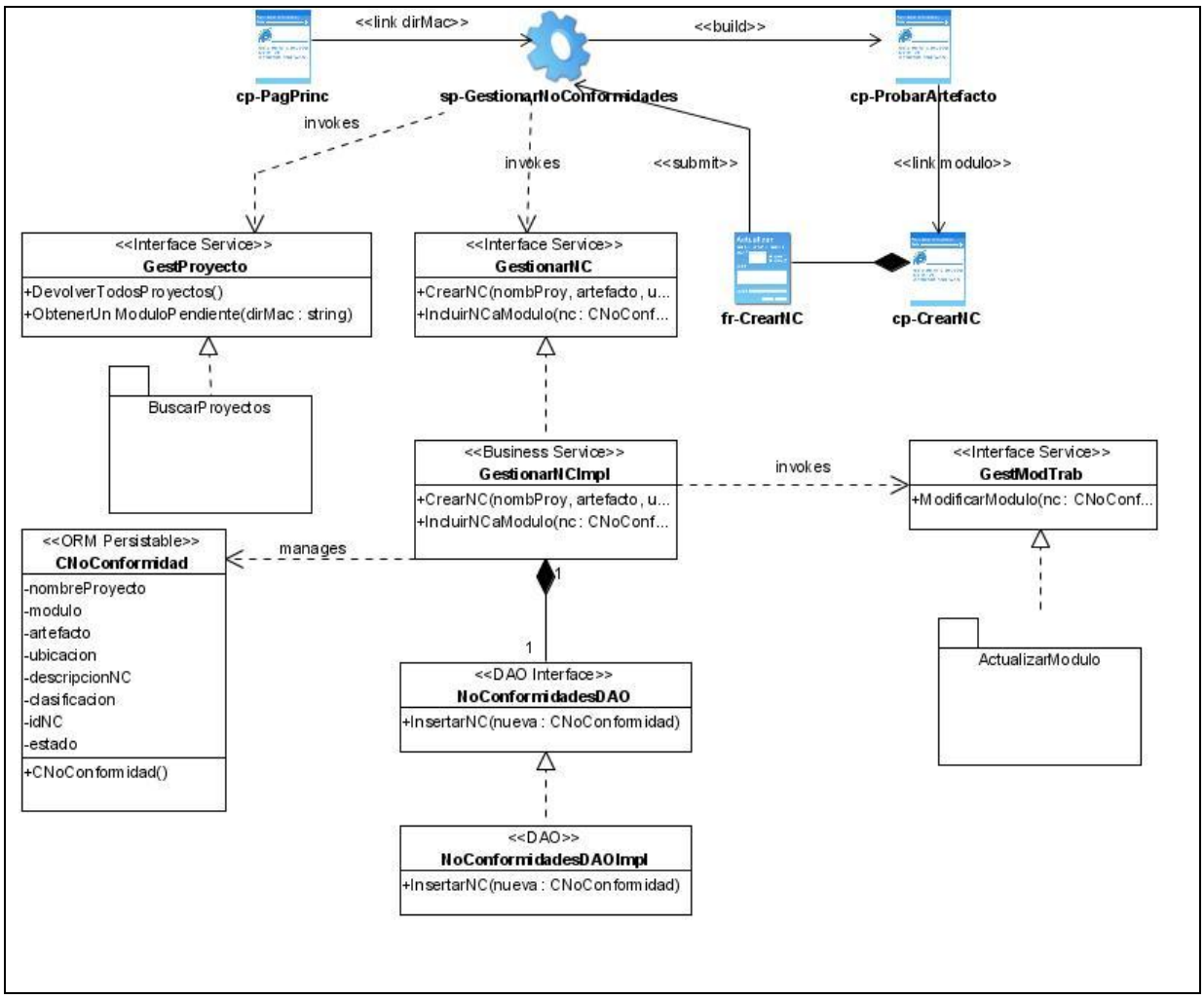


Figura 53: Diagrama de Clases del Diseño (CU Probar Artefacto Asignado)

DIAGRAMA DE CLASES DEL DISEÑO (CU Gestionar No Conformidades: Modificar No Conformidad)

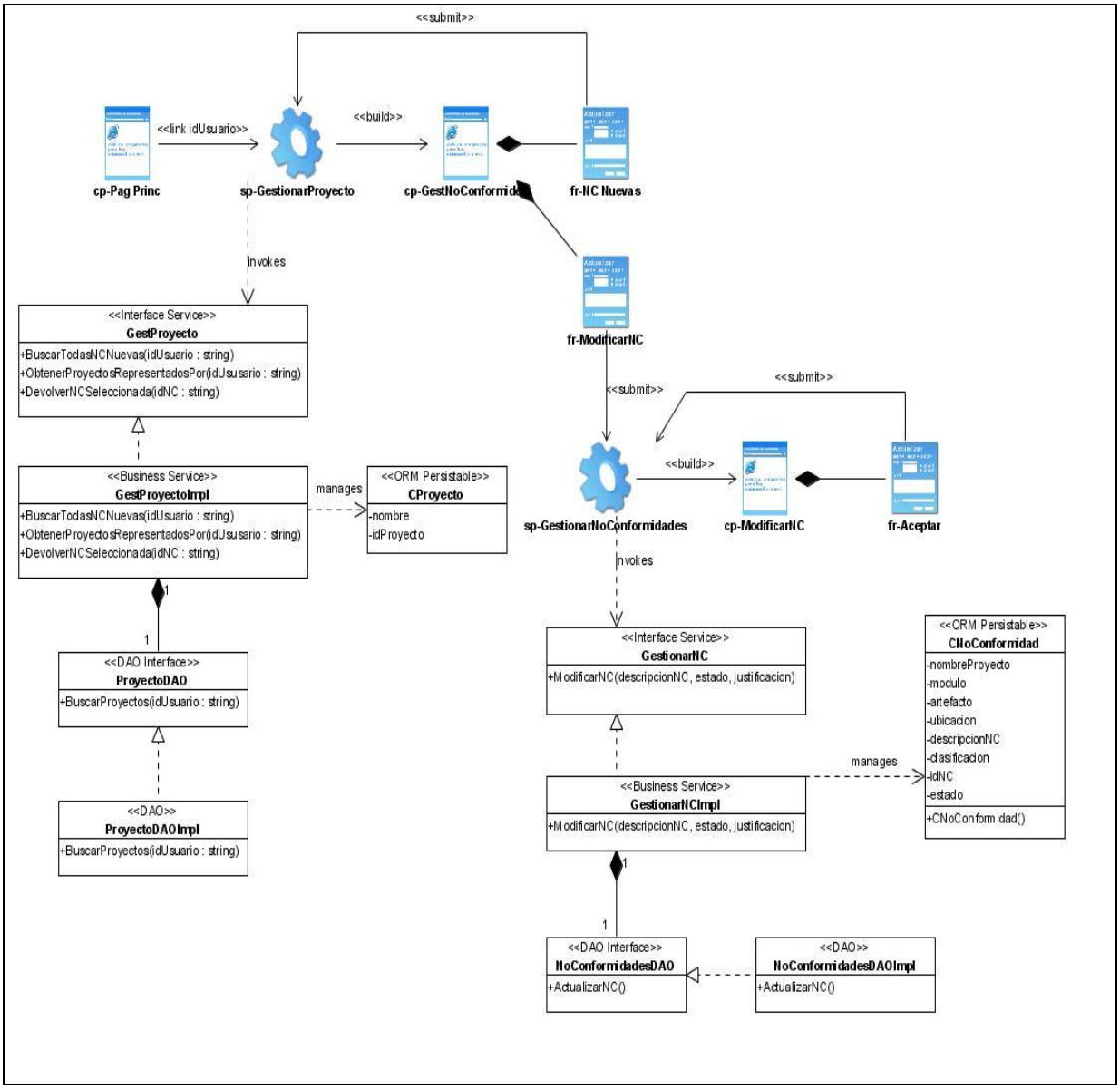


Figura 54: Diagrama de Clases del Diseño (CU Gestionar No Conformidades)

DIAGRAMA DE CLASES DEL DISEÑO (CU Gestionar No Conformidades: Eliminar No Conformidad)

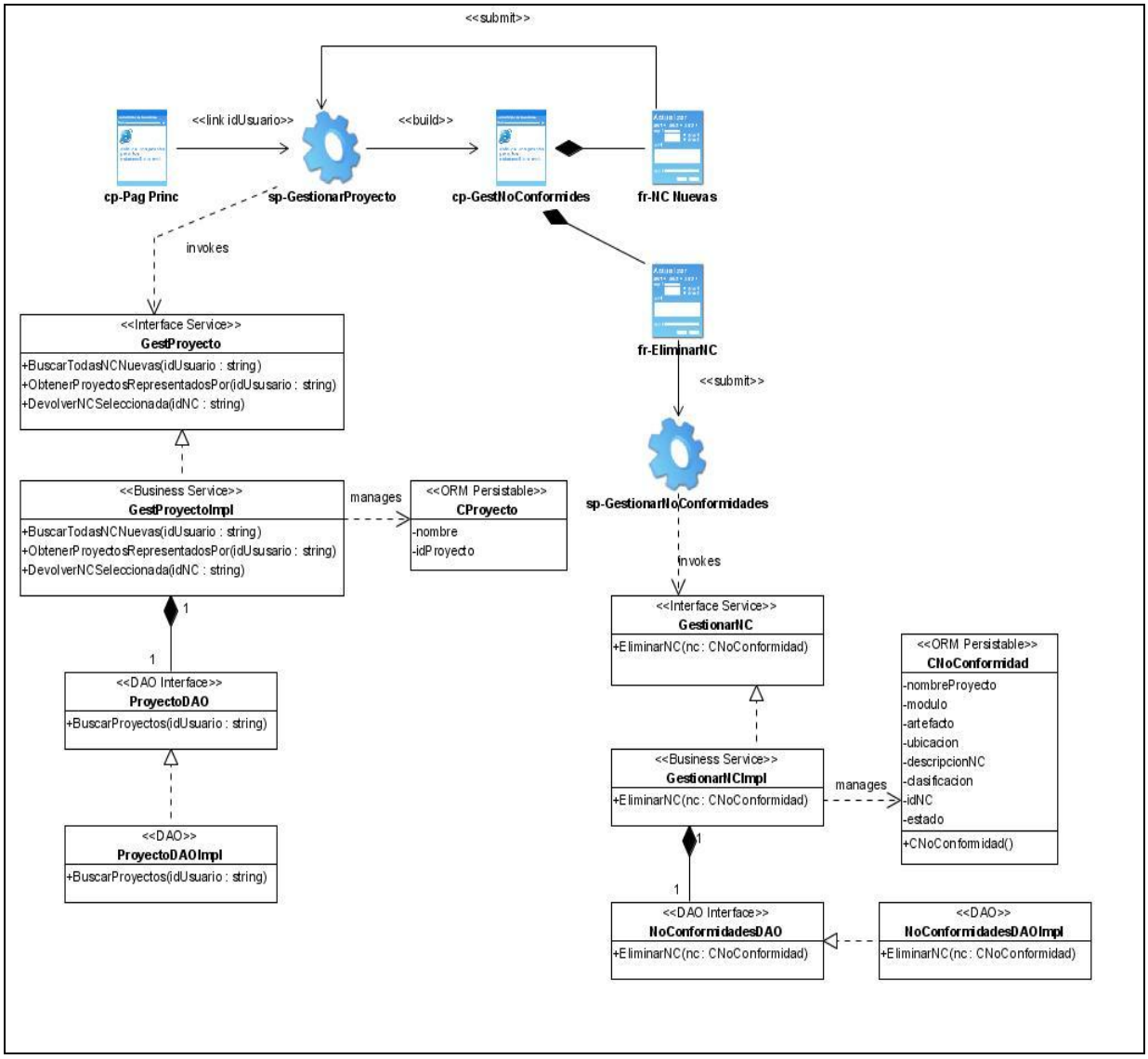


Figura 55: Diagrama de Clases del Diseño (CU Gestionar No Conformidades)

3.3.3. Diagramas de Interacción.

Los diagramas de secuencia muestran las interacciones entre los objetos que estarán presentes en la aplicación, ordenadas en secuencia temporal durante un escenario concreto. A los casos de uso que tienen flujos distintos se le realizó un diagrama de secuencia por cada flujo para un mejor entendimiento de los mismos.

DIAGRAMA DE INTERACCIÓN CU Gestionar Perfil (Crear Perfil)

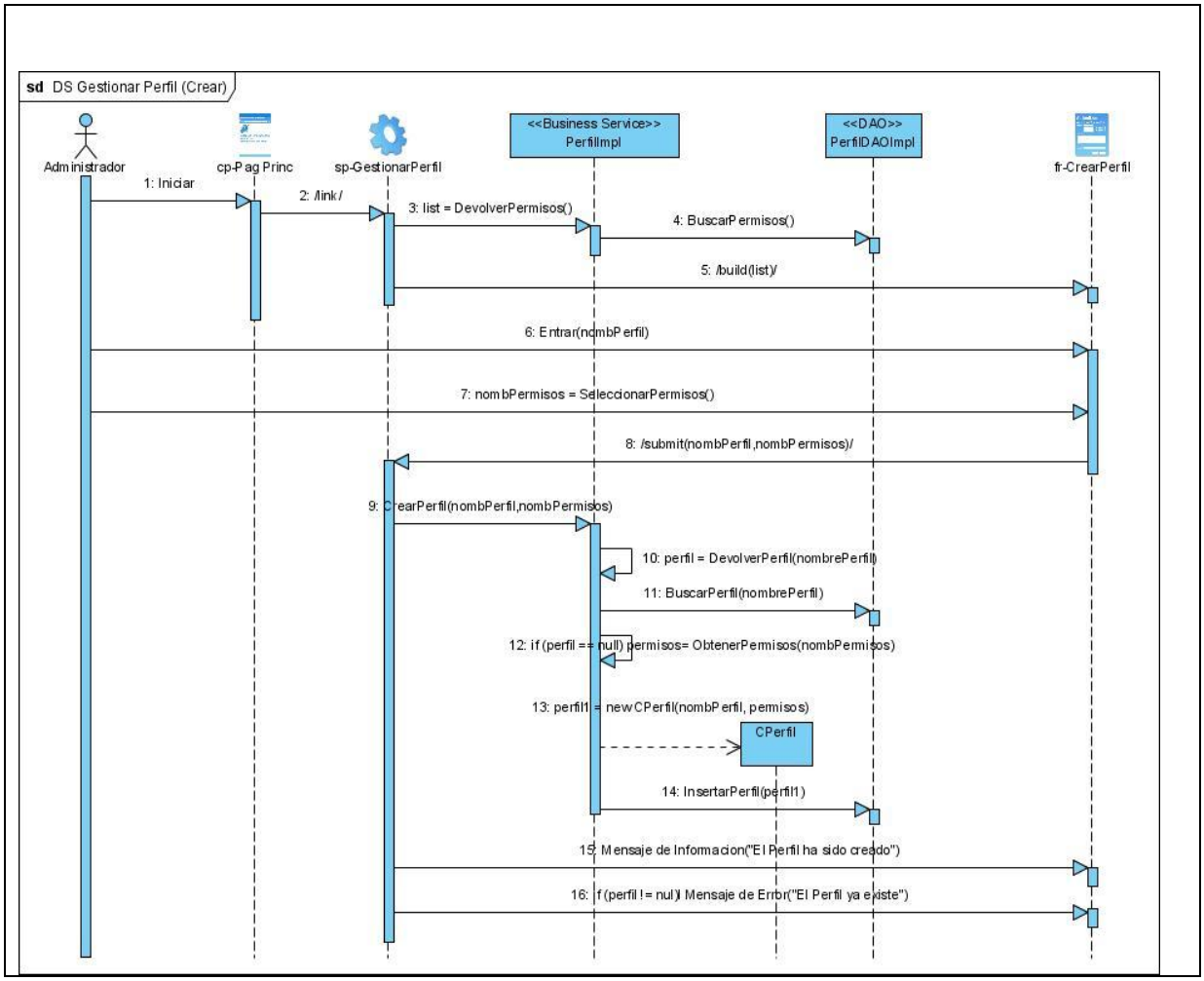


Figura 56: Diagrama de Interacción (CU Gestionar Perfil)

DIAGRAMA DE INTERACCIÓN CU Gestionar Perfil (Modificar Perfil)

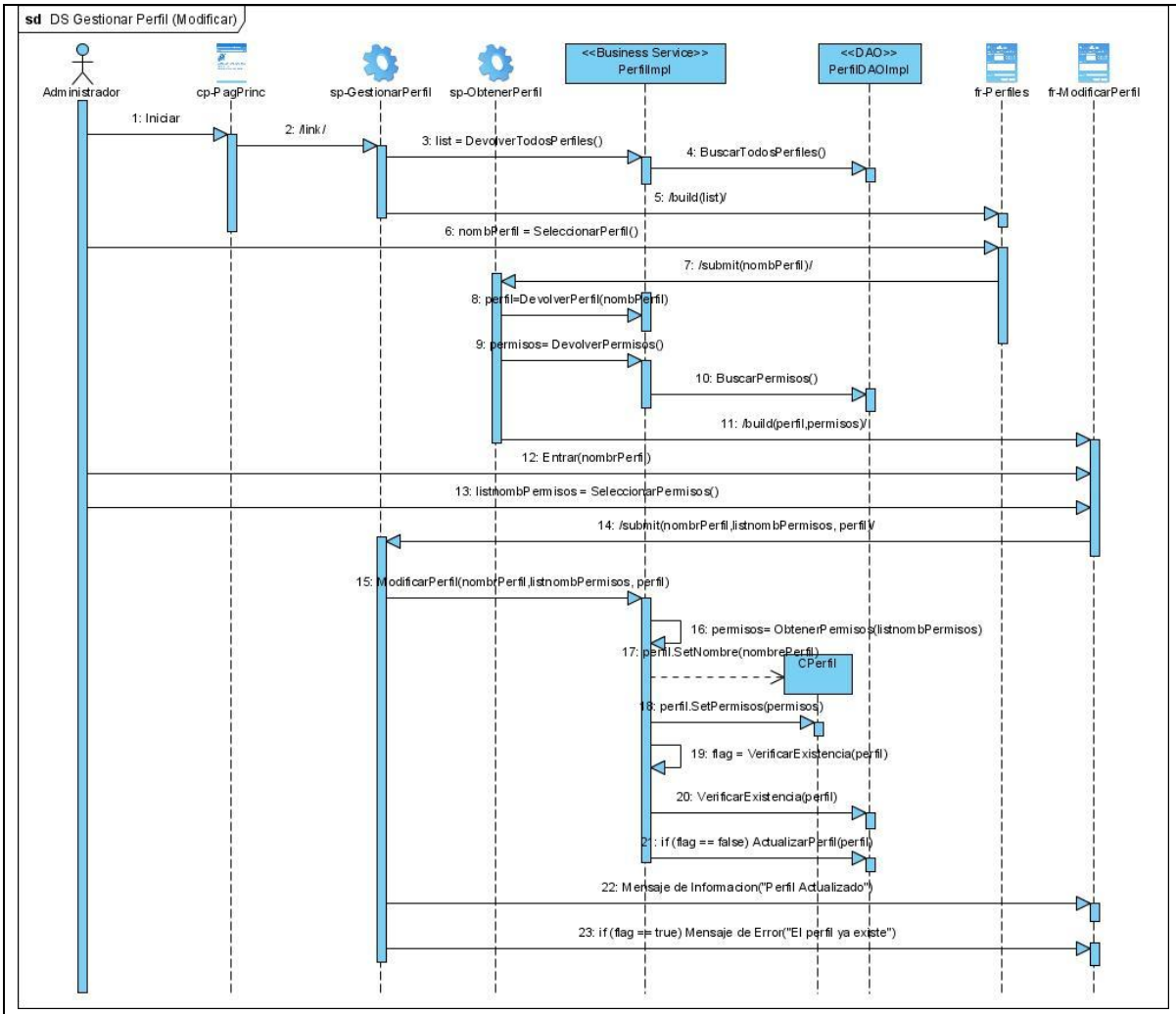


Figura 57: Diagrama de Interacción (CU Gestionar Perfil)

DIAGRAMA DE INTERACCIÓN CU Gestionar Puestos de Trabajo (Crear Puesto de Trabajo)

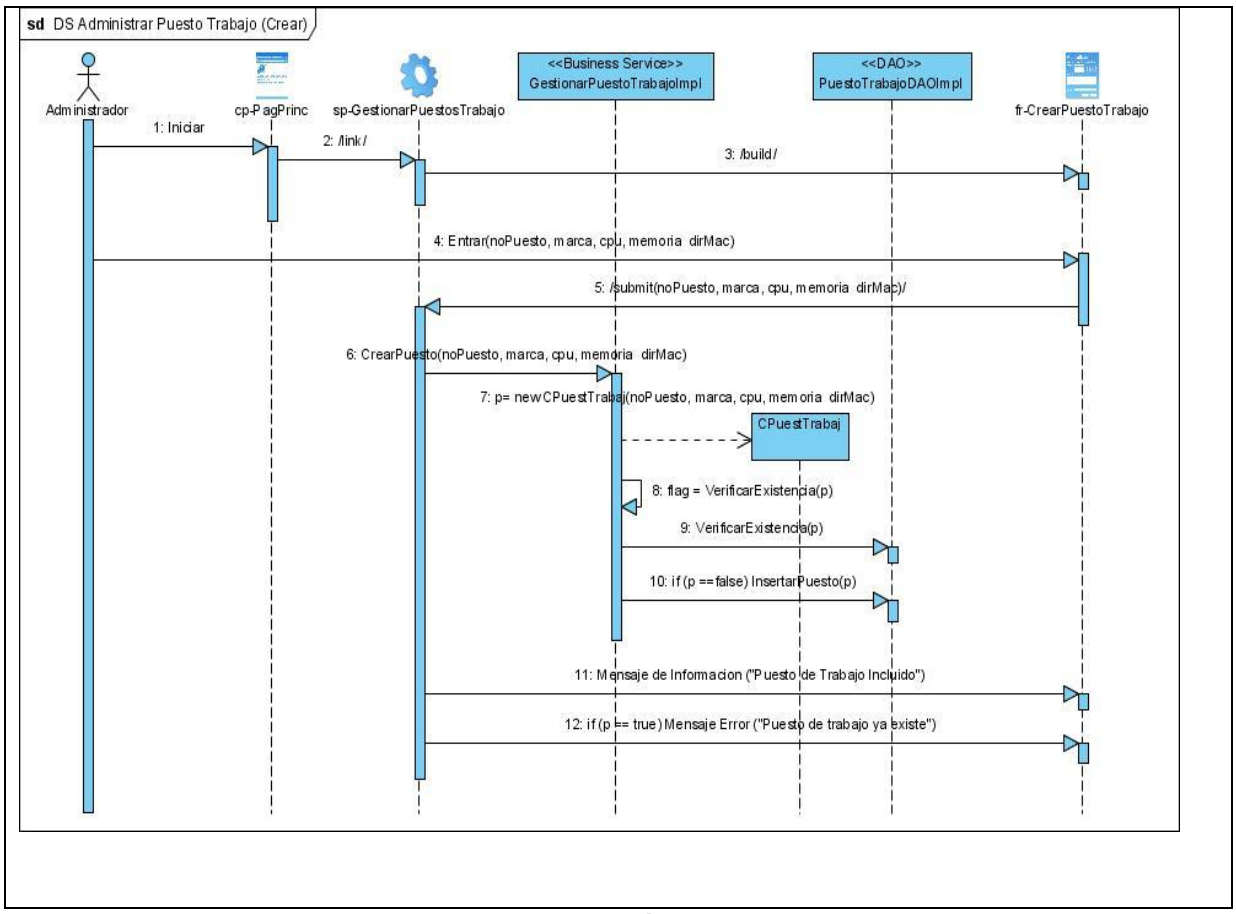


Figura 58: Diagrama de Interacción (CU Puestos de Trabajo)

DIAGRAMA DE INTERACCIÓN CU Gestionar Puestos de Trabajo (Modificar Puesto de Trabajo)

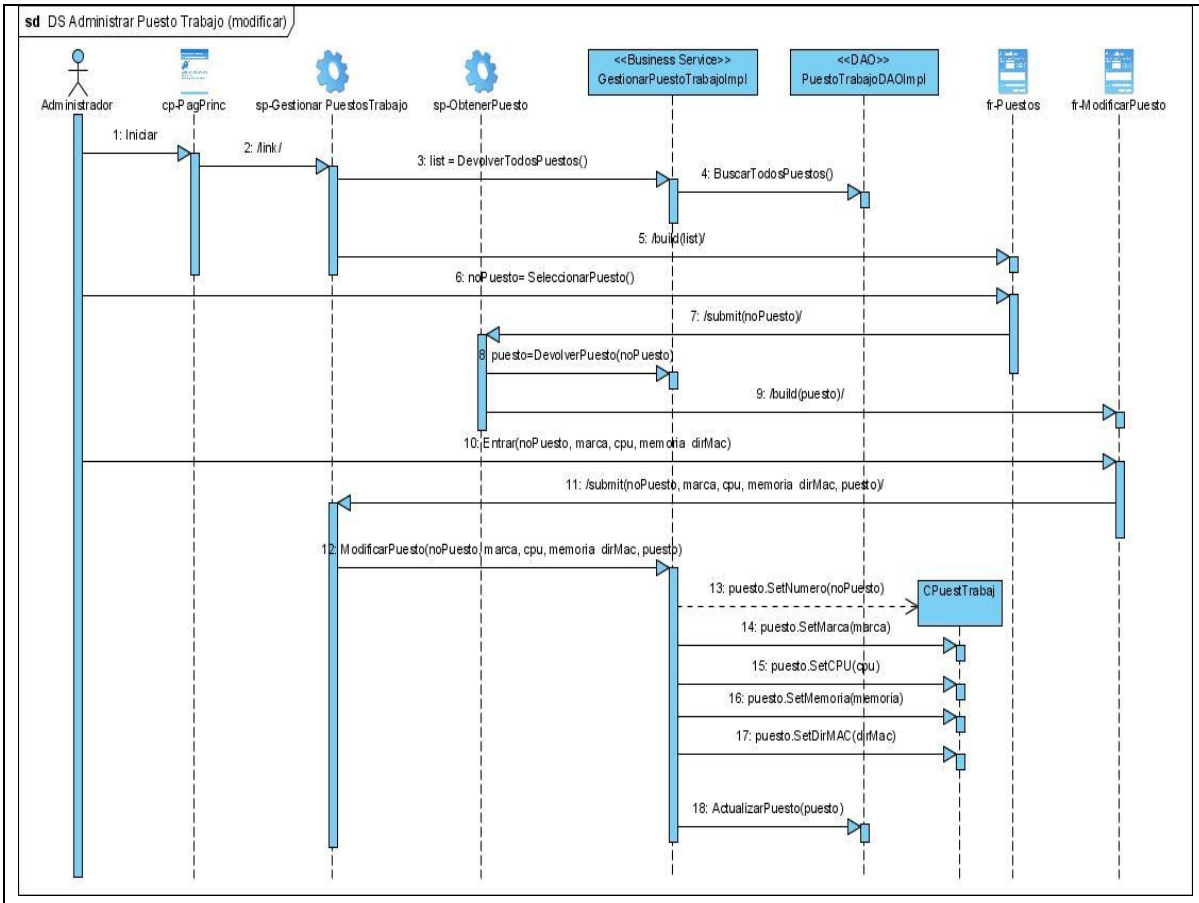


Figura 59: Diagrama de Interacción (CU Puestos de Trabajo)

DIAGRAMA DE INTERACCIÓN CU Conocer Puesto de Trabajo

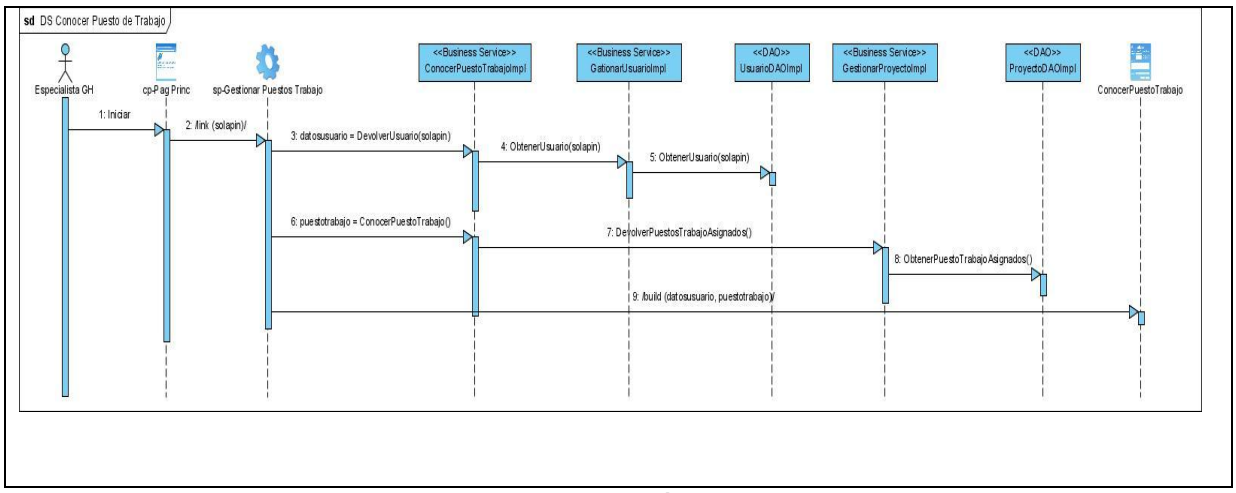


Figura 60: Diagrama de Interacción (CU Conocer Puesto de Trabajo)

DIAGRAMA DE INTERACCIÓN CU Gestionar Solicitud de Inclusión de Usuario (Crear Solicitud de Inclusión de Usuario UCI)

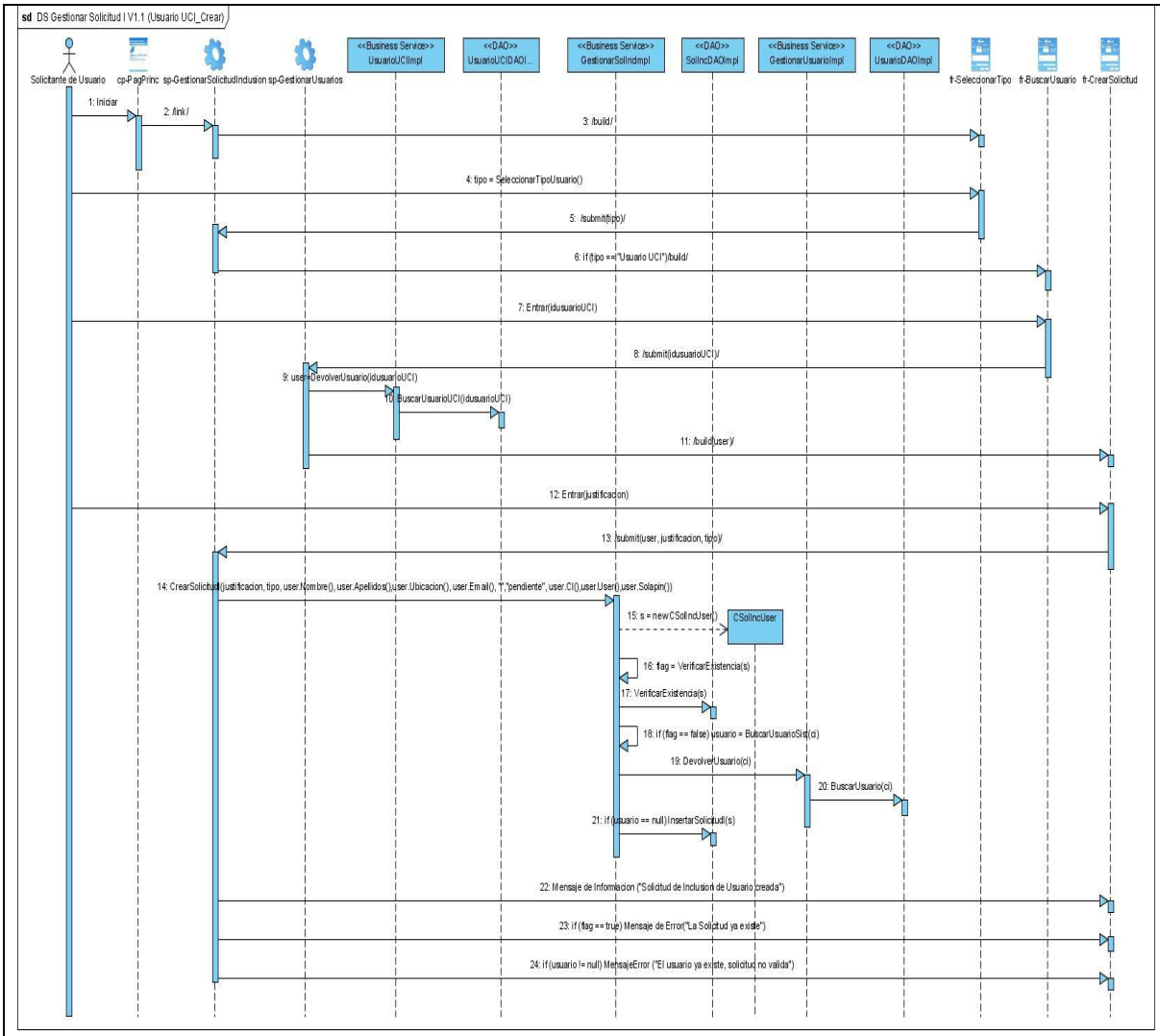


Figura 61: Diagrama de Interacción (CU Gestionar Solicitud de Inclusión de Usuario)

DIAGRAMA DE INTERACCIÓN CU Gestionar Solicitud de Inclusión de Usuario (Crear Solicitud de Inclusión de Usuario Externo)

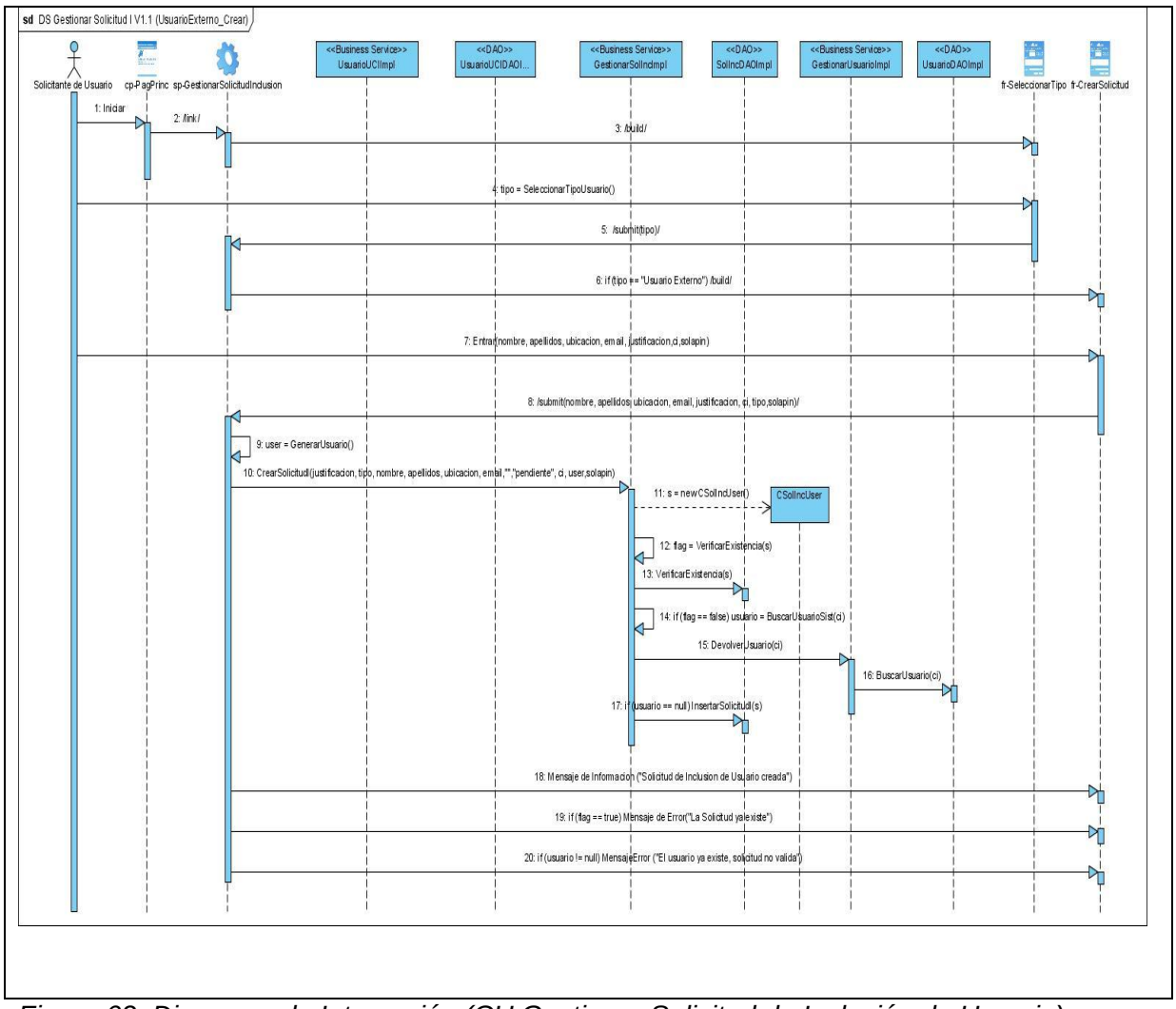


Figura 62: Diagrama de Interacción (CU Gestionar Solicitud de Inclusión de Usuario)

DIAGRAMA DE INTERACCIÓN CU Gestionar Solicitud de Inclusión de Usuario (Eliminar Solicitud de Inclusión de Usuario)

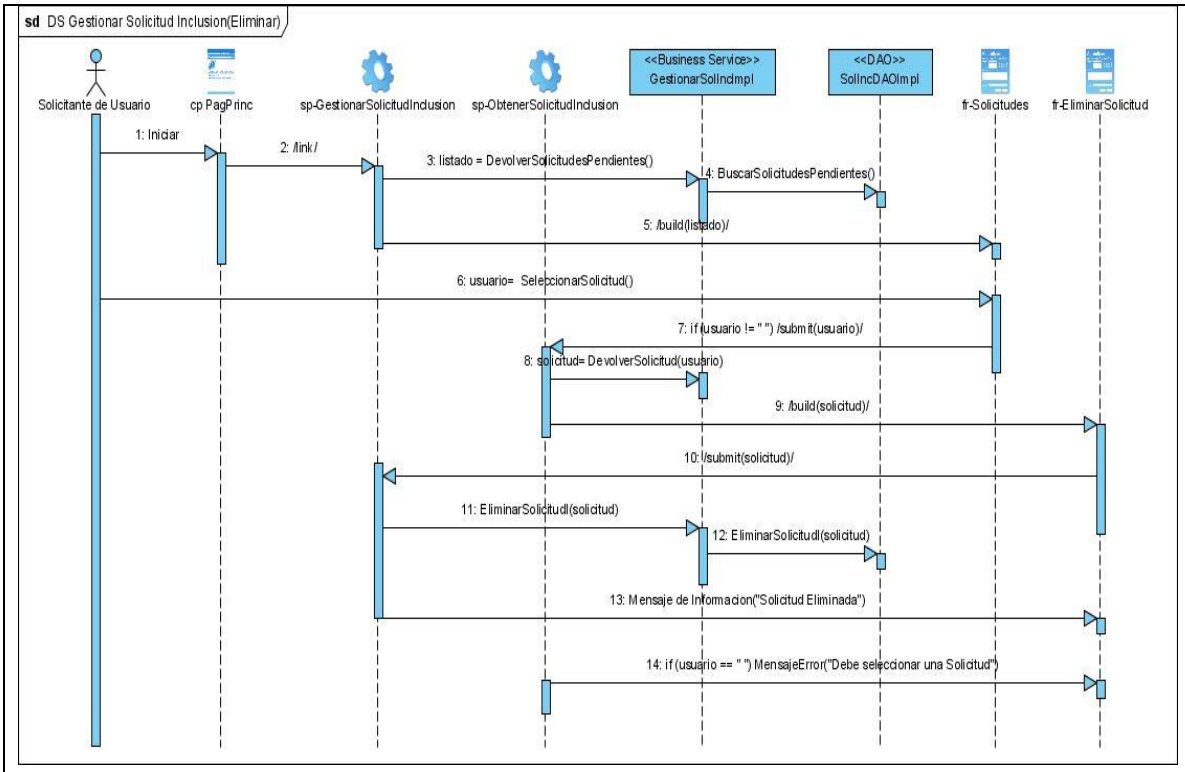


Figura 63: Diagrama de Interacción (CU Gestionar Solicitud de Inclusión de Usuario)

DIAGRAMA DE INTERACCIÓN CU Tramitar estado de Solicitud de Inclusión de Usuario (Aprobar Solicitud de Inclusión de Usuario)

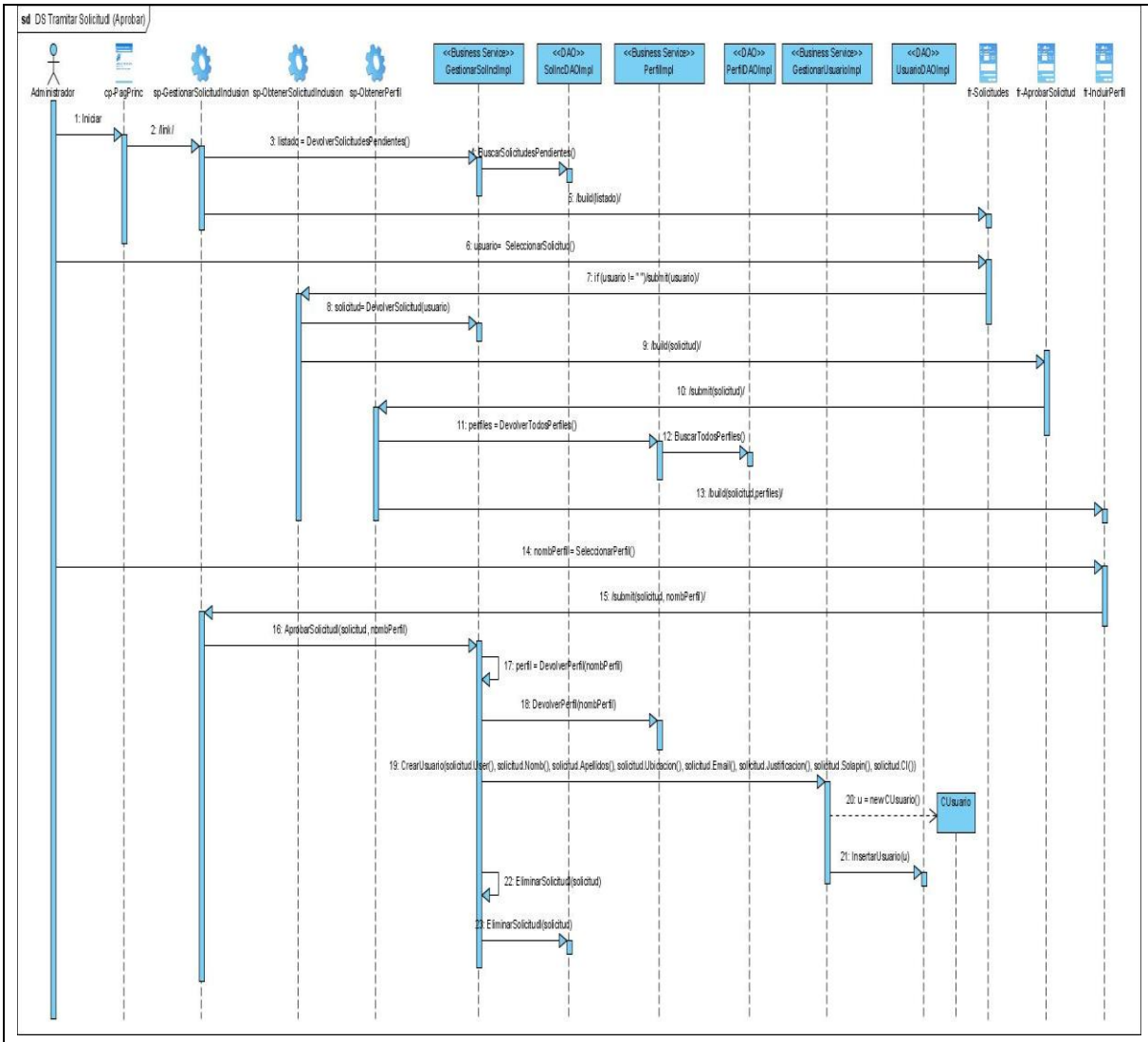


Figura 64: Diagrama de Interacción (CU Tramitar estado de Solicitud de Inclusión de Usuario)

DIAGRAMA DE INTERACCIÓN CU Tramitar estado de Solicitud de Inclusión de Usuario (Rechazar Solicitud de Inclusión de Usuario)

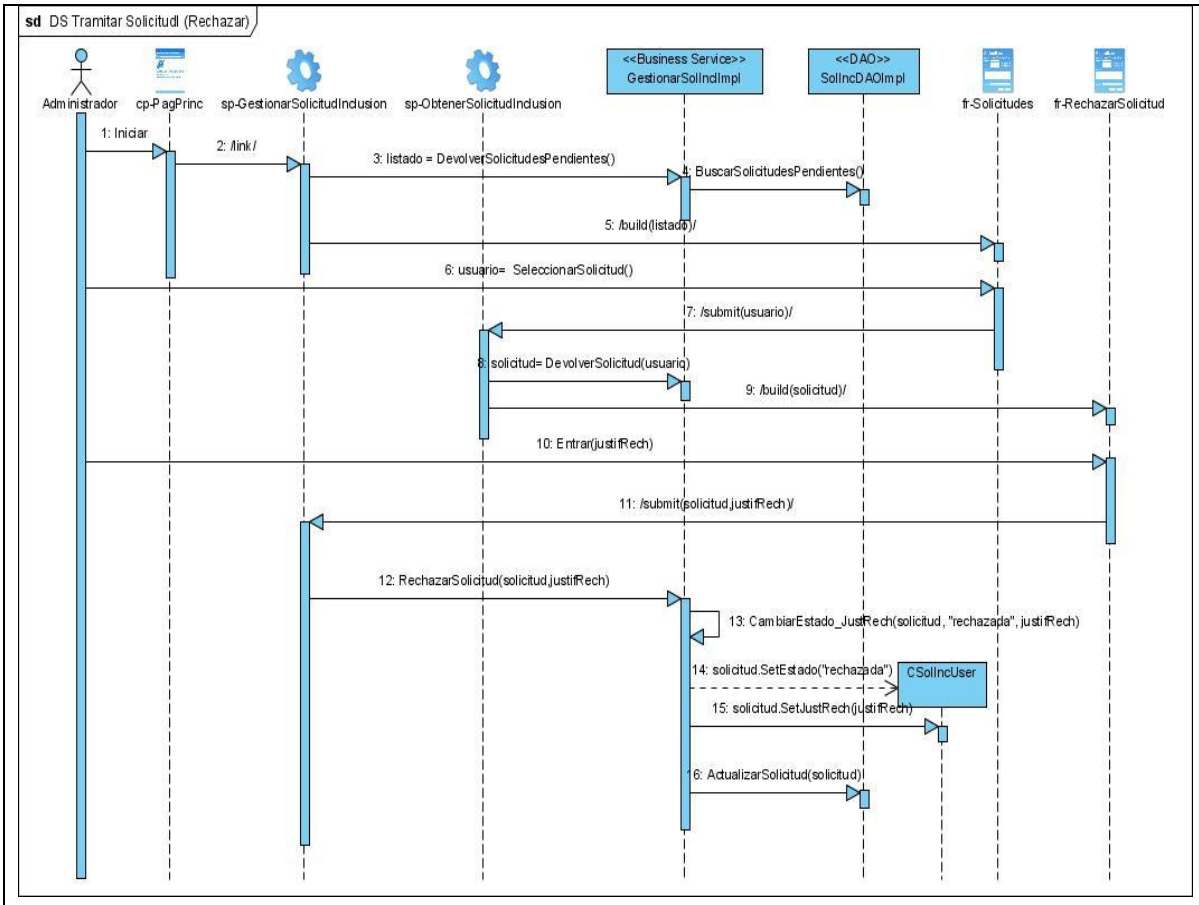


Figura 65: Diagrama de Interacción (CU Tramitar estado de Solicitud de Inclusión de Usuario)

DIAGRAMA DE INTERACCIÓN CU Gestionar Usuario (Modificar Usuario)

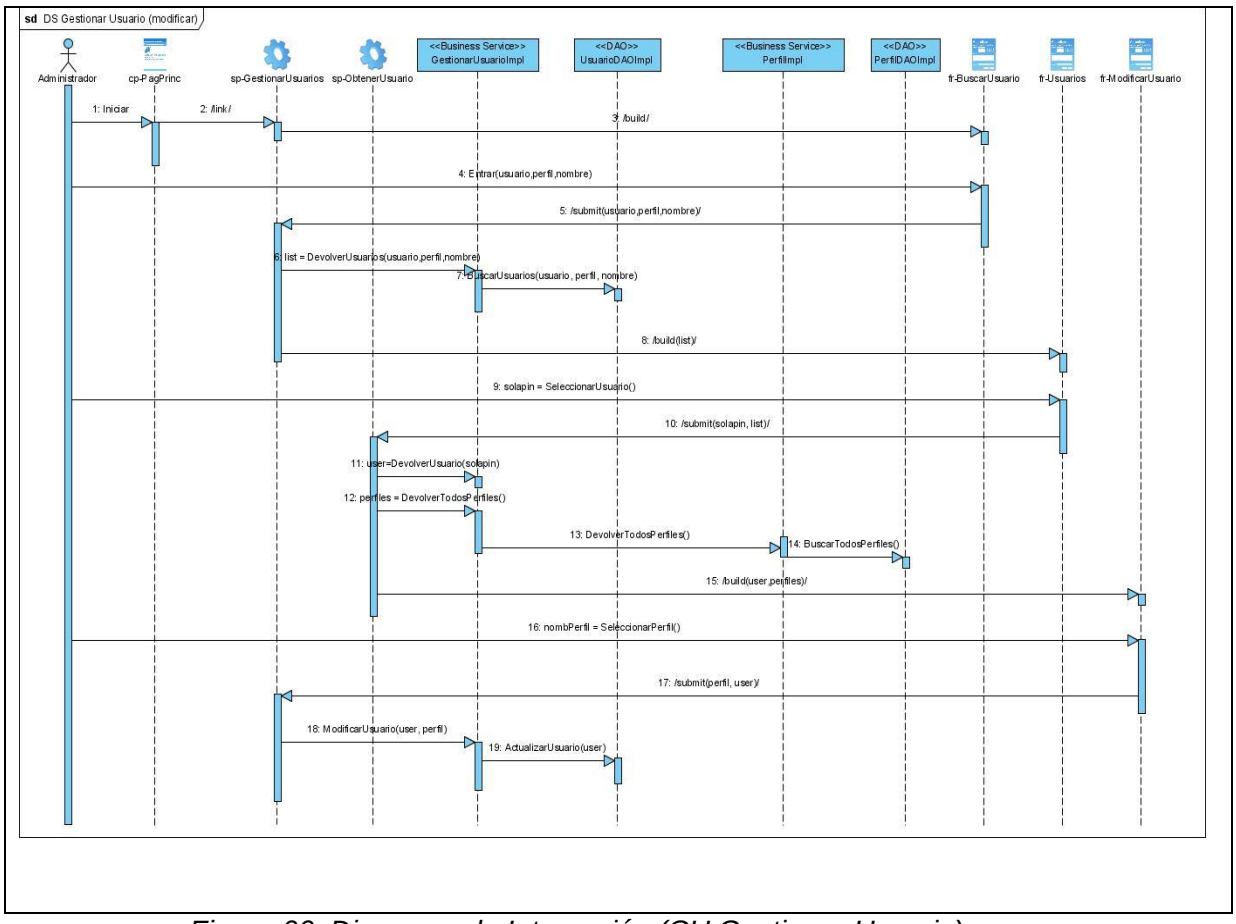


Figura 66: Diagrama de Interacción (CU Gestionar Usuario)

DIAGRAMA DE INTERACCIÓN CU Gestionar Usuario (Eliminar Usuario)

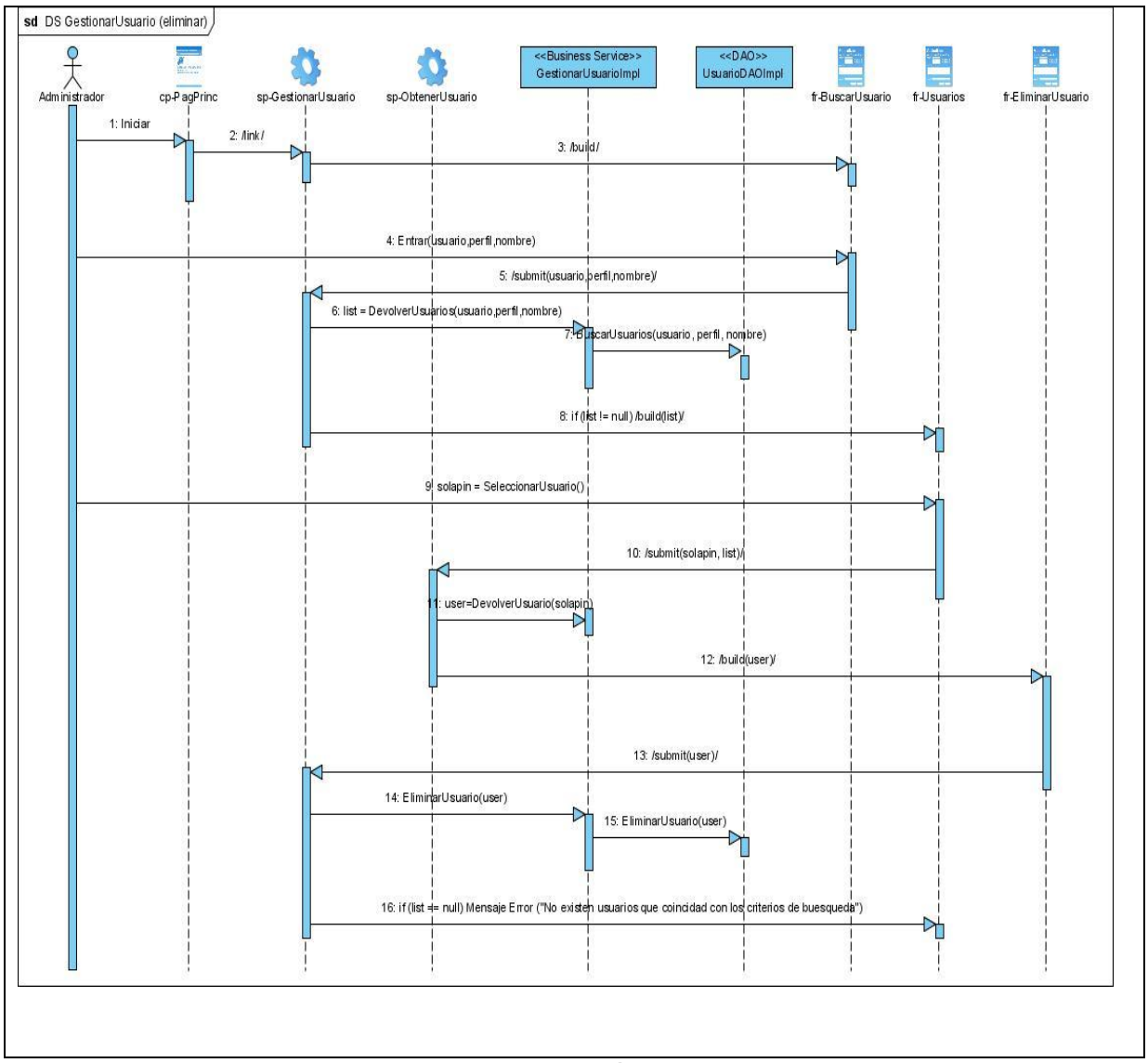


Figura 67: Diagrama de Interacción (CU Gestionar Usuario)

DIAGRAMA DE INTERACCIÓN CU Autenticar Usuario

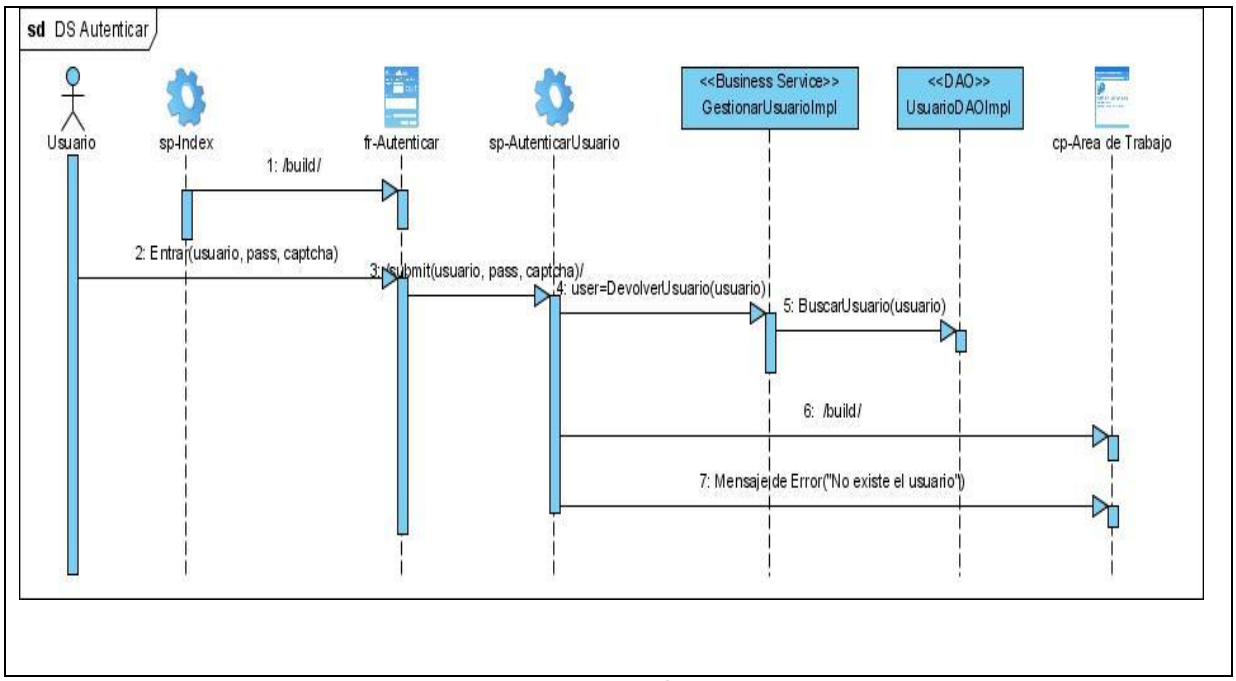


Figura 68: Diagrama de Interacción (CU Autenticar Usuario)

DIAGRAMA DE INTERACCIÓN CU Probar Artefacto Asignado (Crear No Conformidad)

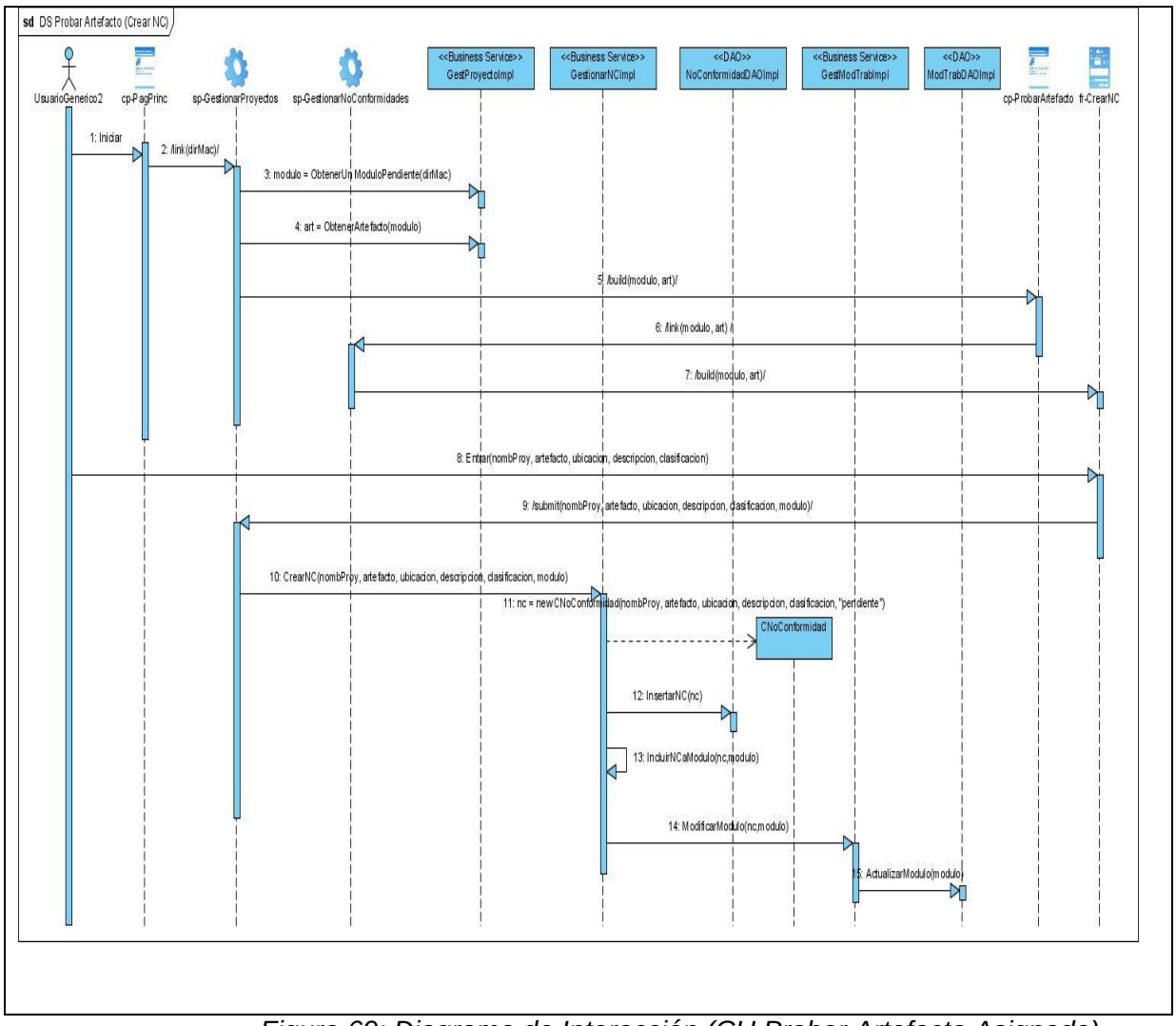


Figura 69: Diagrama de Interacción (CU Probar Artefacto Asignado)

DIAGRAMA DE INTERACCIÓN CU Gestionar No Conformidades (Modificar No Conformidad)

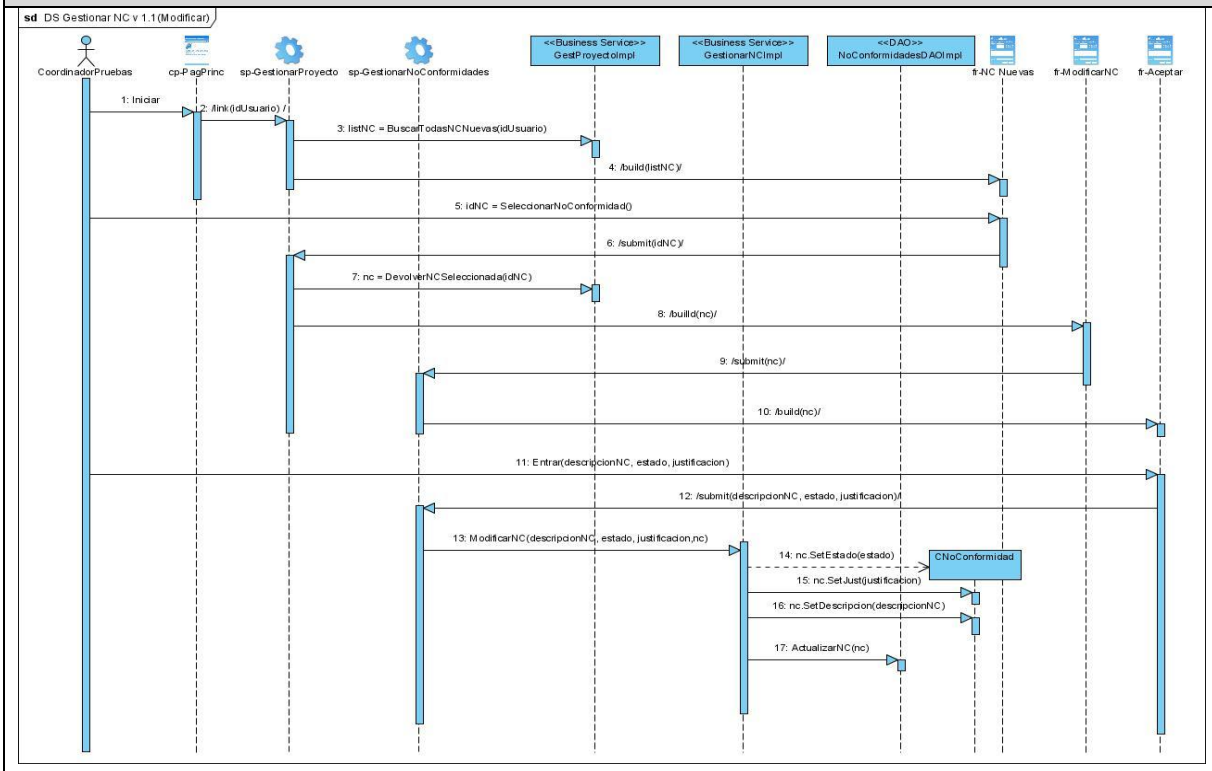


Figura 70: Diagrama de Interacción (CU Gestionar No Conformidades)

DIAGRAMA DE INTERACCIÓN CU Gestionar No Conformidades (Eliminar No Conformidad)

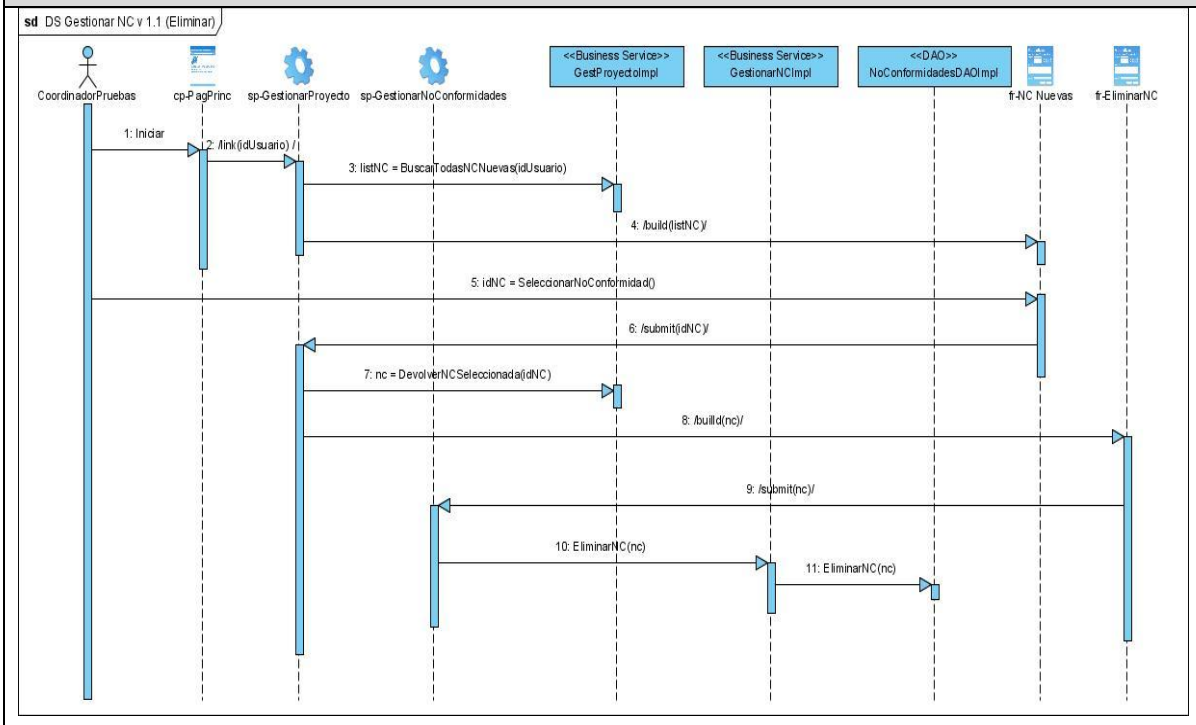


Figura 71: Diagrama de Interacción (CU Gestionar No Conformidades)

3.3.4. Descripción de las clases del diseño.

Nombre: PerfilImpl	
Tipo de Clase: Objeto del negocio (capa de lógica del negocio).	
Para cada responsabilidad:	
Nombre:	**CrearPerfil (perfil)**
Descripción:	Permite instanciar al constructor de la clase PerfilImpl, para que se cree un nuevo perfil.
Nombre:	**DevolverTodosPerfiles ()**
Descripción:	Permite devolver un listado con todos los perfiles existentes en el sistema.
Nombre:	**DevolverPermisos ()**
Descripción:	Permite devolver un listado con todos los permisos existentes en el sistema.

Tabla 5: Descripción de Clases (PerfilImpl)

Nombre: PerfilDAOImpl	
Tipo de Clase: Objeto de acceso a datos (capa de dato).	
Para cada responsabilidad:	
Nombre:	**CrearPerfil (perfil)**
Descripción:	Permite conectarse a la Base Datos y hacer persistir el objeto.
Nombre:	**DevolverTodosPerfiles ()**
Descripción:	Permite conectarse a la Base Datos y devolver un listado con todos los perfiles existentes en el sistema.
Nombre:	**DevolverPermisos ()**
Descripción:	Permite conectarse a la Base Datos y devolver un listado con todos los permisos existentes en el sistema.

Tabla 6: Descripción de Clases (PerfilDAOImpl)

Nombre: CPermiso	
Tipo de Clase: Entidad del dominio (capa de lógica del negocio).	
Atributo	**Tipo**
Nombre	string
Nombre:	**CPermiso ()**
Descripción:	Es el constructor de la clase, permite crear un nuevo permiso (objeto) con sus atributos.

Tabla 7: Descripción de Clases (CPermiso)

Nombre: GestionarPuestoTrabajoImpl	
Tipo de Clase: Objeto del negocio (capa de lógica del negocio).	
Para cada responsabilidad:	
Nombre:	**CrearPuesto (puesto)**
Descripción:	Permite instanciar al constructor de la clase GestionarPuestoTrabajoImpl, para que se cree un nuevo puesto de trabajo.
Nombre:	**DevolverTodosPuestos ()**
Descripción:	Permite devolver un listado con todos los puestos de trabajo existentes en el sistema.
Nombre:	**ModificarPuesto (puesto)**
Descripción:	Permite modificar los datos de un puesto determinado.

Tabla 8: Descripción de Clases (GestionarPuestoTrabajoImpl)

Nombre: PuestoTrabajoDAOImpl	
Tipo de Clase: Objeto de acceso a datos (capa de dato).	
Para cada responsabilidad:	
Nombre:	**CrearPuesto (puesto)**
Descripción:	Permite conectarse a la Base Datos y hacer persistir el objeto.

Nombre:	**DevolverTodosPuestos ()**
Descripción:	Permite conectarse a la Base Datos y devolver un listado con todos los puestos de trabajo existentes en el sistema.
Nombre:	**ModificarPuesto (puesto)**
Descripción:	Permite conectarse a la Base Datos sobrescribiendo el objeto modificado.

Tabla 9: Descripción de Clases (PuestoTrabajoDAOImpl)

Nombre: CPuestoTrabajo	
Tipo de Clase: Entidad del dominio (capa de lógica del negocio).	
Atributo	**Tipo**
Número Puesto	int
Marca	string
CPU	string
Memoria	int
Dir Mac	string
Nombre:	**CPuestoTrabajo ()**
Descripción:	Es el constructor de la clase, permite crear un nuevo puesto de trabajo (objeto) con sus atributos.

Tabla 10: Descripción de Clases (CPuestoTrabajo)

Nombre: GestionarSolicitudInclusiónImpl	
Tipo de Clase: Objeto del negocio (capa de lógica del negocio).	
Para cada responsabilidad:	
Nombre:	**CrearSolicitudI (solicitud)**
Descripción:	Permite instanciar al constructor de la clase GestionarSolicitudInclusiónImpl, para que se cree una nueva solicitud de inclusión de usuario.
Nombre:	**DevolverTodosSolicitudesPendientes ()**
Descripción:	Permite devolver un listado con todas las solicitudes pendientes existentes en el sistema.
Nombre:	**EliminarSolicitudI (solicitud)**
Descripción:	Permite eliminar una solicitud de inclusión de usuario.

Tabla 11: Descripción de Clases (GestionarSolicitudInclusiónImpl)

Nombre: SolicitudInclusiónDAOImpl	
Tipo de Clase: Objeto de acceso a datos (capa de dato).	
Para cada responsabilidad:	
Nombre:	**InsertarSolicitudPuesto (solicitud)**
Descripción:	Permite conectarse a la Base Datos y hacer persistir el objeto.
Nombre:	**DevolverTodosSolicitudesPendientes ()**

Descripción:	Permite conectarse a la Base Datos y devolver un listado con todas las solicitudes de inclusión de usuario pendientes, existentes en el sistema.
Nombre:	**EliminarSolicitudl (solicitud)**
Descripción:	Permite conectarse a la Base Datos eliminando el objeto especificado.

Tabla 12: Descripción de Clases (SolicitudInclusiónDAOImpl)

Nombre: CSolIncUser	
Tipo de Clase: Entidad del dominio (capa de lógica del negocio).	
Atributo	**Tipo**
Tipo User	string
User	string
Nombre User	string
Apellidos User	string
Ubicación	string
Correo Electrónico	string
Justificación	string
Nombre:	**CSolIncUser ()**
Descripción:	Es el constructor de la clase, permite crear una nueva solicitud de inclusión de usuario (objeto) con sus atributos.

Tabla 13: Descripción de Clases (CSolIncUser)

Nombre: GestionarUsaurioImpl	
Tipo de Clase: Objeto del negocio (capa de lógica del negocio).	
Para cada responsabilidad:	
Nombre:	**ModificarUsuario (usuario)**
Descripción:	Permite modificar los datos del usuario.
Nombre:	**EliminarUsuario (usuario)**
Descripción:	Permite eliminar un usuario del sistema.

Tabla 14: Descripción de Clases (GestionarUsaurioImpl)

Nombre: UsuarioDAOImpl	
Tipo de Clase: Objeto de acceso a datos (capa de dato).	
Para cada responsabilidad:	
Nombre:	**ModificarUsuario (usuario)**
Descripción:	Permite conectarse a la Base Datos sobrescribiendo el objeto modificado.
Nombre:	**EliminarUsuario (usuario)**
Descripción:	Permite conectarse a la Base Datos eliminando el objeto especificado.

Tabla 15: Descripción de Clases (SolicitudInclusiónDAOImpl)

Nombre: CUsuario	
Tipo de Clase: Entidad del dominio (capa de lógica del negocio).	
Atributo	**Tipo**
User	string
Nombre User	string
Apellidos User	string
Ubicación	string
Correo Electrónico	string
Justificación	string
Nombre:	**CUsuario ()**
Descripción:	Es el constructor de la clase, permite crear un nuevo usuario (objeto) con sus atributos.

Tabla 16: Descripción de Clases (CUsuario)

Nombre: GestionarNCImpl	
Tipo de Clase: Objeto del negocio (capa de lógica del negocio).	
Para cada responsabilidad:	
Nombre:	**ModificarNC (nc)**
Descripción:	Permite modificar los datos de una no conformidad.
Nombre:	**DevolverTodasNCNuevas ()**
Descripción:	Permite devolver un listado con todas las no conformidades nuevas existentes en el sistema.

Tabla 17: Descripción de Clases (GestionarNCImpl)

Nombre: NoConformidadDAOImpl	
Tipo de Clase: Objeto de acceso a datos (capa de dato).	
Para cada responsabilidad:	
Nombre:	**ModificarNC (nc)**
Descripción:	Permite conectarse a la Base Datos sobrescribiendo el objeto modificado.
Nombre:	**DevolverTodasNCNuevas ()**
Descripción:	Permite conectarse a la Base Datos y devolver un listado con todas las no conformidades nuevas, existentes en el sistema.

Tabla 18: Descripción de Clases (NoConformidadDAOImpl)

Nombre: CNoConformidad	
Tipo de Clase: Entidad del dominio (capa de lógica del negocio).	
Atributo	**Tipo**
Nombre Proyecto	string
Módulo	string
Artefacto	string
Ubicación	string

Descripción de la NC	string
Clasificación	string
Estado	string
Nombre:	**CNoConformidad ()**
Descripción:	Es el constructor de la clase, permite crear una nueva no conformidad (objeto) con sus atributos.

Tabla 19: Descripción de Clases (CNoConformidad)

Nombre: GestionarHorarioImpl	
Tipo de Clase: Objeto del negocio (capa de lógica del negocio).	
Para cada responsabilidad:	
Nombre:	**CrearHorario (horario)**
Descripción:	Permite instanciar al constructor de la clase GestionarHorarioImpl, para que se cree un nuevo horario de trabajo.

Tabla 20: Descripción de Clases (GestionarHorarioImpl)

Nombre: HorarioDAOImpl	
Tipo de Clase: Objeto de acceso a datos (capa de dato).	
Para cada responsabilidad:	
Nombre:	**InsertarHorario (horario)**
Descripción:	Permite conectarse a la Base Datos y hacer persistir el objeto.

Tabla 21: Descripción de Clases (HorarioDAOImpl)

Nombre: CHorario	
Tipo de Clase: Entidad del dominio (capa de lógica del negocio).	
Atributo	**Tipo**
Cantidad turnos	int
Nombre:	**CHorario ()**
Descripción:	Es el constructor de la clase, permite crear un nuevo horario de trabajo (objeto) con sus atributos.

Tabla 22: Descripción de Clases (CHorario)

Nombre: CTurno	
Tipo de Clase: Entidad del dominio (capa de lógica del negocio).	
Atributo	**Tipo**
Hora inicio	string
Hora fin	string
Nombre:	**CTurno ()**
Descripción:	Es el constructor de la clase, permite crear un turno de trabajo (objeto) con sus atributos.

Tabla 23: Descripción de Clases (CNoConformidad)

Nombre: GestionarTipoPruebaImpl	
Tipo de Clase: Objeto del negocio (capa de lógica del negocio).	
Para cada responsabilidad:	
Nombre:	**CrearTipoPrueba (prueba)**
Descripción:	Permite instanciar al constructor de la clase GestionarTipoPruebaImpl, para que se cree un nuevo tipo de prueba.
Nombre:	**VerificarExistencia (prueba)**
Descripción:	Permite verificar si el objeto existe.

Tabla 24: Descripción de Clases (GestionarTipoPruebaImpl)

Nombre: TipoPruebaDAOImpl	
Tipo de Clase: Objeto de acceso a datos (capa de dato).	
Para cada responsabilidad:	
Nombre:	**InsertarTipoPrueba (prueba)**
Descripción:	Permite conectarse a la Base Datos y hacer persistir el objeto.
Nombre:	**VerificarExistencia (prueba)**
Descripción:	Permite conectarse a la Base Datos y verificar la existencia del objeto.

Tabla 25: Descripción de Clases (TipoPruebaDAOImpl)

Nombre: CTipoPrueba	
Tipo de Clase: Entidad del dominio (capa de lógica del negocio).	
Atributo	**Tipo**
Nombre Tipo Prueba	string
Descripción	string
Nombre:	**CTipoPrueba ()**
Descripción:	Es el constructor de la clase, permite crear un tipo de prueba (objeto) con sus atributos.

Tabla 26: Descripción de Clases (CTipoPrueba)

3.4. Diseño de la Base de Datos.

La base de datos es la solución utilizada para garantizar el almacenamiento de la información persistente de la aplicación. En este epígrafe se muestra el diseño de la base de datos del sistema propuesto, a través del modelo de datos generado a partir del diagrama de clases persistentes.

3.4.1. Diagrama de clases persistentes.

Las clases persistentes son aquellas que se corresponden con las entidades del dominio del software, necesitan ser capaces de guardar su estado en un medio permanente, dicha necesidad está dada por el almacenamiento físico de la información de la clase, para la copia de seguridad en caso del fracaso del sistema, o para el intercambio de información. A continuación se muestra el diagrama de clases persistentes.

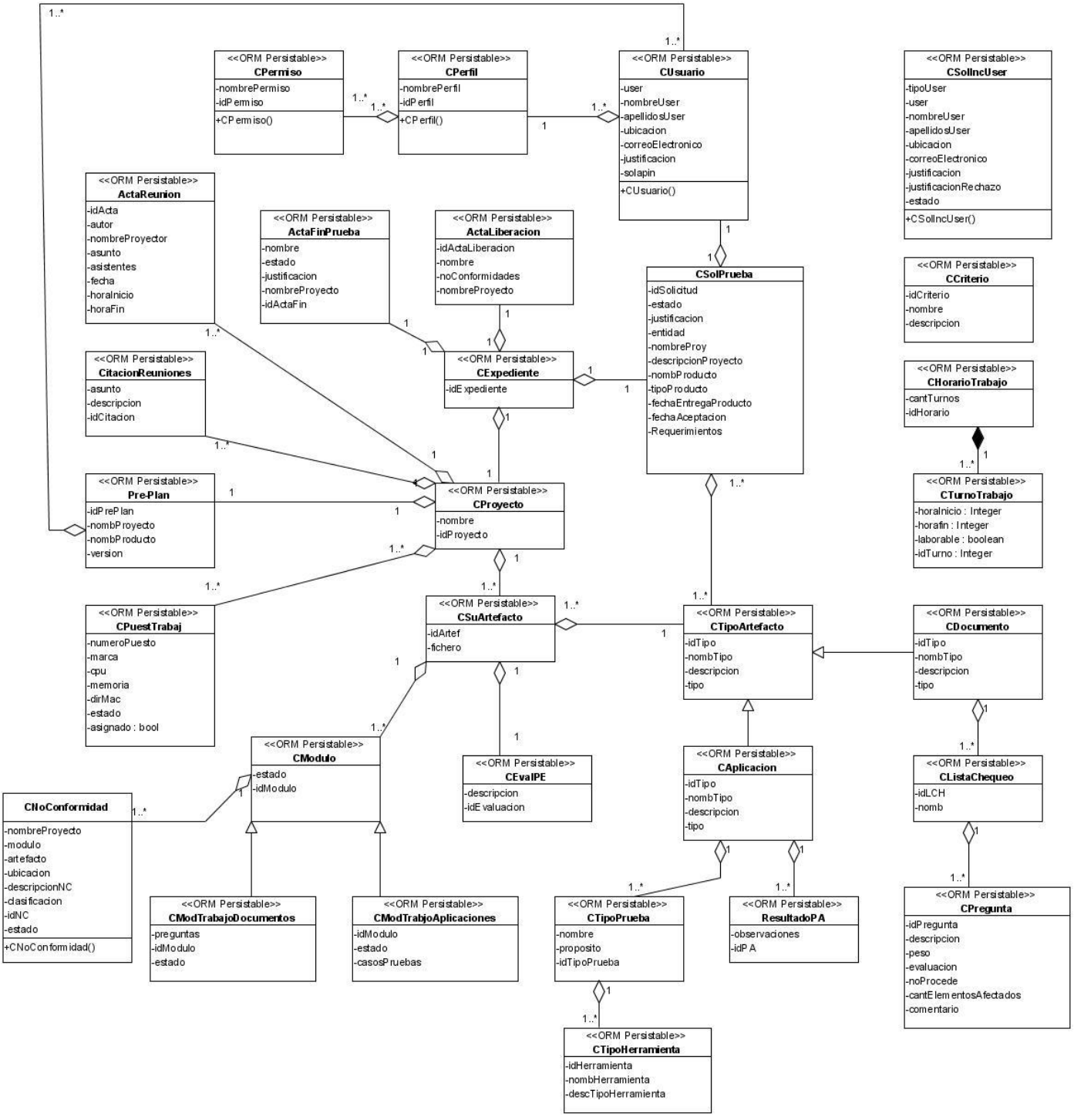


Figura 72: Diagrama de Clases Persistentes.

3.5. Modelo de Datos.

Un modelo de datos es un conjunto de conceptos que se utilizan para describir la estructura de una base de datos: los datos, las relaciones y las restricciones que deben cumplirse entre ellos.

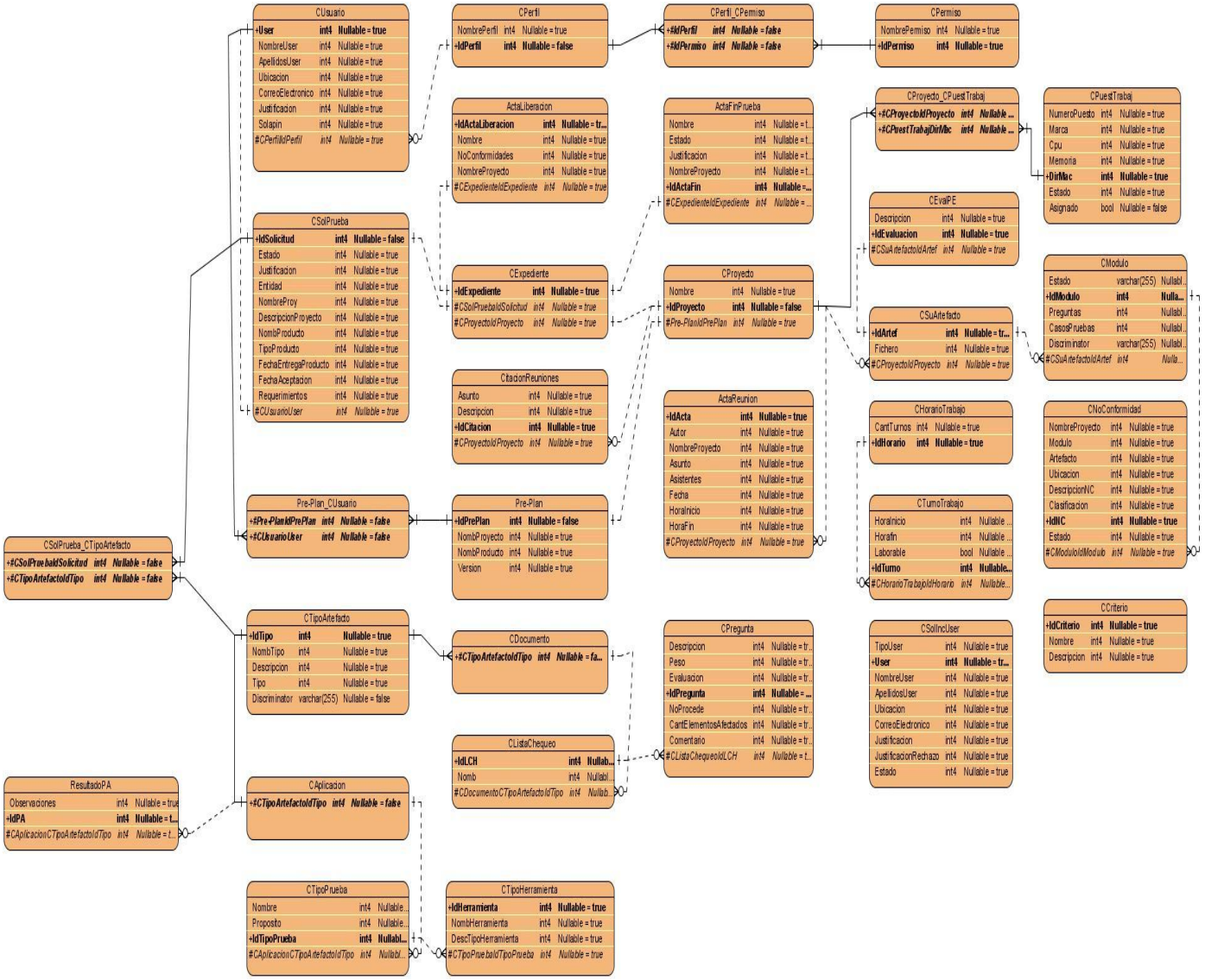


Figura 73: Modelo de Datos

3.6. Principios de Diseño.

Los principios de diseño que se necesitan para la construcción exitosa de la aplicación pueden codificarse, explicarse y aplicarse de forma sistemática. Este enfoque se basa en patrones-guías y principios estructurados. Por tanto, después de introducir la presentación de los diagramas de interacción, se hace necesario explicar aquellos patrones que se aplicaron en dichos diagramas.

3.6.1. Aplicación de los Patrones de Diseño.

MVC (Model View Controller)

Uno de los patrones que ha demostrado ser fundamental a la hora de diseñar aplicaciones web es el Modelo-Vista-Control (MVC). Este patrón propone la separación en distintos componentes, la interfaz de usuario (vistas), el modelo de negocio y la lógica de control. Una vista es una “fotografía” del modelo (o una parte del mismo) en un determinado momento. Un control recibe un evento disparado por el usuario a través de la interfaz, accede al modelo de manera adecuada a la acción realizada, y presenta en una nueva vista el resultado de dicha acción. Por su parte, el modelo consiste en el conjunto de objetos que modelan los procesos de negocio que se realizan a través del sistema.

En esta aplicación web, las vistas serían Java Server Pages (JSP) que el usuario visualiza en el navegador. A través de estas páginas el usuario interactúa con la aplicación, enviando eventos al servidor. En el servidor se encuentra el código de control para estos eventos, que serían los servlets, estereotipados como “server page” según UML, que en función del evento invocan las funcionalidades necesarias, expuestas por la capa de servicios de negocio, como resultado preparan o actualizan los datos representados por el modelo, los mismos se devuelven al usuario a través de paginas JSP.

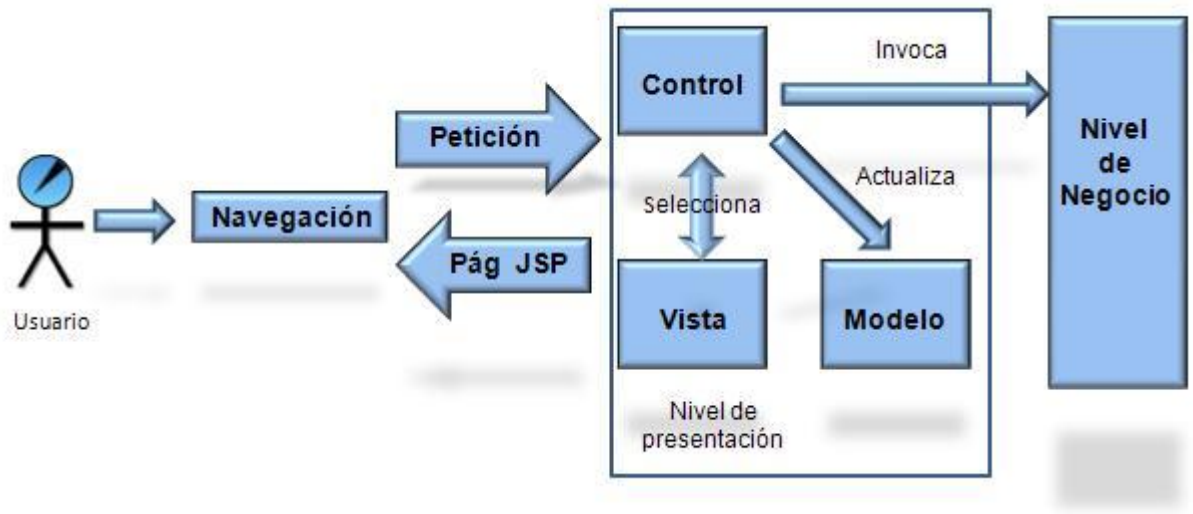


Figura 74: Modelo Vista Controlador

La clave está en la separación entre vista y modelo, mejorando la reusabilidad. El modelo suele ser más estable a lo largo del tiempo y sujeto a menos variaciones, mientras que las vistas pueden cambiar con frecuencia, ya sea por cambio del medio de presentación o por necesidades de usabilidad de la interfaz o simple renovación de la estética de la aplicación. Con esta clara separación las vistas pueden cambiar sin afectar al modelo y viceversa. Los controladores son los encargados de funcionar como puentes entre ambos, determinando el flujo de salida de la aplicación.

DAO (Data Access Object- Patrón J2EE, de la capa de acceso a datos)

Como decisión de diseño de la aplicación, se creó para cada objeto de negocio un DAO distinto, el cual hace persistir en la base de datos las entidades del dominio que maneja dicho objeto de negocio, de modo que se cumpla un principio básico de desarrollo de software, definido originalmente por Dijkstra en los años 70, denominado “separación de incumbencias”: Distintas responsabilidades no deben ser delegadas a una misma clase.

Funcionamiento del patrón DAO en la Aplicación.

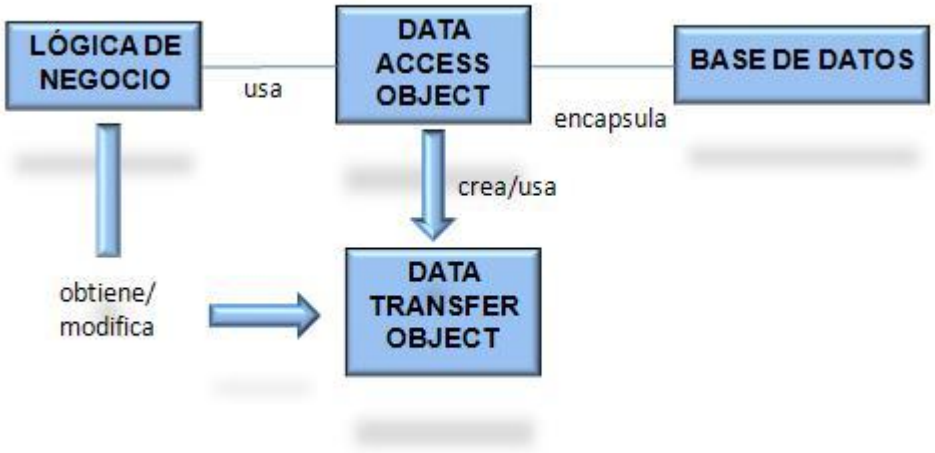


Figura 75: Funcionamiento del Patrón DAO

El gráfico anterior representa el funcionamiento del patrón DAO en la aplicación. De modo que cuando la capa de lógica de negocio necesite interactuar con la base de datos, va a hacerlo a través del DAO, quien se encarga de conectarse directamente a la misma utilizando Hibernate, con el fin de ejecutar las transacciones que precisa la capa lógica de negocio, para ello el DAO hace uso de los Transfer Object (TO) para transportar los datos desde la base de datos hasta la capa de lógica de negocio y viceversa, este proceso se hace transparente para dicha capa.

Transfer Object (Patrón J2EE de la capa lógica de negocio)

Al definir una arquitectura multicapa para el desarrollo del sistema, provocó el surgimiento de un problema típico en este tipo de situaciones ¿Cómo pasar los datos de una capa lógica a otra? Solicitar o Enviar datos uno a uno no es una solución adecuada, pues incrementa innecesariamente el número de llamadas a la base de datos. Se prescindió entonces de la aplicación del patrón Transfer Object, el cual provee una forma compacta y organizada de pasar dichos datos de una capa a otra. Como se vio en el patrón DAO analizado previamente, la comunicación entre la capa de negocio y la de acceso a datos se hace a través de este tipo de objetos.

Específicamente para esta aplicación se definió que las entidades de dominio, no van a poseer ningún tipo de lógica de negocios, esta responsabilidad recae sobre los objetos de negocio, permitiendo usar a dichos objetos como Transfer Objects que se mueven entre las capas arquitectónicas de la aplicación.

3.7. Validación de la Solución Propuesta.

Este trabajo abarca las primeras fases del proceso de desarrollo de la herramienta que utilizarán los especialistas del Grupo de Liberación y Pruebas de Software en el LIPS para llevar a cabo el Proceso de Liberación de todos los productos que se confeccionan tanto dentro como fuera de la UCI, es decir, solo comprende la modelación de la misma, con el objetivo de garantizar una codificación más sencilla y eficiente, por tal motivo se precisa que todos los artefactos creados por los analistas del sistema tengan la calidad requerida, como respuesta los especialistas del grupo antes mencionado, sometieron dichos entregables a rigurosas pruebas y revisiones técnicas en el LIPS, aplicando así el propio Proceso de Liberación Productos Software que se pretende automatizar, a continuación se muestran los resultados obtenidos:

Artefacto Especificación de Requisitos de Software

Según los lineamientos definidos por el LIPS al artefacto Especificación de Requisitos de Software se le aplica la Lista de Chequeo correspondiente *(Ver Anexo 3)*.

Específicamente este artefacto consta de 46 requisitos funcionales, 7 no funcionales escritos en un total de 6 páginas.

La primera iteración de pruebas por parte de los especialistas del Grupo de Liberación y Pruebas de Software arrojó un total de 2 no conformidades:

- No Conformidad relacionada con 1 error de redacción.
- No Conformidad relacionada con 1 error de formato.

Artefacto Especificación de Casos de Uso del Sistema

Según los lineamientos definidos por el LIPS al artefacto Especificación de Casos de Uso del Sistema se le aplica la Lista de Chequeo Correspondiente *(Ver Anexo 3)* se obtuvo los siguientes resultados:

El artefacto que se analiza consta de 32 casos de uso, escritos en un total de 53 páginas.

La primera iteración de pruebas arrojó un total de 4 no conformidades:

- No Conformidad relacionada con 1 error de redacción.
- No Conformidades relacionadas con 2 errores de formato.
- No Conformidad relacionada con 1 error de correspondencia con otra documentación.

Cada una de las No Conformidades detectadas en la primera iteración de pruebas fueron resueltas, de modo que en la segunda iteración a la que fueron sometidos todos los entregables no se detectaron No Conformidades, garantizando un 100 % de efectividad en los mismos.

3.8. Conclusiones

Los flujos de trabajo análisis y diseño son importantes en el proceso de desarrollo del sistema proporcionando un dominio total del problema así como una visión amplia del tamaño y la complejidad del mismo. Después de haber modelado todos los procesos que han sido objeto de estudio durante el presente trabajo, mediante la construcción de los artefactos propuestos por la metodología RUP correspondientes a los flujos de trabajo antes mencionados como son: diagrama de clases de análisis, diagrama de clases del diseño, diagrama de interacción y descripción de las clases, se concluye que estos sirven como guía de fácil comprensión para los desarrolladores que implementarán el sistema.

CONCLUSIONES GENERALES

En un mundo donde la presencia de soluciones informáticas (software) se hace inevitable, por encontrarse en el centro de todas las grandes transformaciones que se suceden en la sociedad del siglo XXI; por ser la piedra angular en temas de primer orden, como la economía digital, la evolución de las empresas y la administración del conocimiento, por solo mencionar algunos, se hace imprescindible entonces garantizar la calidad de los mismos por parte de las Empresas Desarrolladoras de Software, de ahí surge la necesidad de este trabajo. A partir de los objetivos planteados, en aras de dar respuesta a la pregunta que comprende el problema principal a resolver durante la presente investigación, se obtuvieron los siguientes resultados:

- Al realizar un análisis detallado de los aspectos teóricos que rigen la investigación se hizo evidente la necesidad de desarrollar una herramienta web que permita a los Especialistas de Calidad llevar a cabo el Proceso de Liberación de Productos Software eficientemente.
- Ante la necesidad de desarrollar dicha herramienta se precisó entonces de la implementación de una metodología que guió el proceso de desarrollo de la aplicación en sus fases iniciales, teniendo en cuenta que el alcance de este trabajo abarca hasta el diseño de la misma.
- Se hizo necesario la realización de un estudio minucioso de los aspectos teórico-prácticos comprendidos en el proceso que rige la comprobación de la calidad de productos software en el Laboratorio Industrial de Pruebas de Software, el cual sentó las bases para capturar todos los requisitos funcionales que debe cumplir el sistema a desarrollar.
- Ante la marcada complejidad del Proceso de Liberación de Productos de Software se hizo necesario brindarle tanto a clientes, usuarios, desarrolladores e interesados, un vocabulario común para comprender el contexto del negocio en que se enmarcó el sistema, a través de la modelación del dominio.
- Se realizó el diseño de dicho sistema, confeccionando los artefactos más importantes en aras de crear las pautas iniciales para la codificación del mismo por parte de los programadores.

RECOMENDACIONES

A continuación se exponen un conjunto de recomendaciones en pos de cumplimentar el proceso de desarrollo del sistema para la gestión del Proceso de Liberación de Productos de Software, debido a que este trabajo solo abarca las fases iniciales que propone RUP para la confección del mismo; por lo tanto se propone para posteriores trabajos lo siguiente:

- La codificación completa de los subsistemas diseñados para el Grupo de Liberación y Pruebas de Software.
- Someter los mismos a un riguroso proceso de pruebas funcionales por parte de los especialistas de calidad, del mismo grupo.
- Hacer la instalación piloto en el Laboratorio Industrial de Pruebas de Software, entidad que utilizará la herramienta.
- Diseñar e implementar un nuevo subsistema: Monitoreo y Reportes, el cual no quedó comprendido en el alcance de este trabajo.

REFERENCIAS BIBLIOGRÁFICAS

BUSTELO RUESTA, C. A. G.-M. H., ELISA. *Tendencias en la gestión de la información, la documentación y el conocimiento en las organizaciones.* Diciembre 2001, vol. Volumen 10, 4-7 p. Disponible en: www.infoarea.es.

CANÓS, J. H., P. LETELIER, AND M.C. PENADÉS. *Metodologías Ágiles en el Desarrollo de Software.* Editado por: Valencia, D.-U. P. D. 2004, Disponible en: http://www.e-participa.com/www/eParDemos/eParCC/DataS/GestorSolicitudes-EAD.pdf.

DEEPAK ALUR, J. C., DAN MALKS, GRADY BOOCH, MARTIN FOWLER. *Core J2EE Patterns: Best Practices and Design Strategies.* 2 ed. 2003.

DOMINICANO, F. C. L. *Gestor de Base de Datos: MySQL, PostgreSQL, SQLite.* 2009, Disponible en: http://www.eaprende.com/gestor-de-basededatos-mysql-postresql-sqlite.html#post.

JACOBSON, I. *"Applying UML in The Unified Process" Presentación. Rational Software.* 1998, Disponible en: http://www.rational.com/uml.

LARMAN, C. *UML y Patrones.* 2 ed. Prentice Hall, 2002. 520 p.

LETELIER, P. Y. P., Mª CARMEN. *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Editado por: Valencia., D. D. S. I. Y. C. U. P. D. 2004, Disponible en: http://issi.dsic.upv.es/publications/archives/f-1067610619332/HistorialGrupo.pdf.

MARTIN FOWLER. *Patterns of Enterprice Application Architecture.* 2002.

PÉREZ, L. *ArBaWeb: Arquitectura base sobre la Web* UCI, 2007.

PRESSMAN, R. *Ingeniería del Software: Un enfoque práctico.* 1997.

ROD JOHNNSON, J. H., ALEF ARENDSEN, THOMAS RISBERG, COLIN SAMPALEANU. *Professional Java Development with the Spring Framework.* Wiley Publishing Inc, 2005. ISBN 13:978-0-7645-7483-2.

WESLEY, A. *Metodologia RUP.* 2004, Disponible en: http://www.forosdelweb.com/1967481-post34.html.

XAVIER FERRÉ GRAU, M. I. S. S. *Desarrollo Orientado a Objetos con UML.* 2004, Disponible en: http://www.clikear.com/manuales/uml/index.aspx.

BIBLIOGRAFÍA

Jorge Rodríguez. Spring y el principio de Hollywood. [En línea] [Citado: el 10 de enero del 2009.] http://blog.continuum.cl/wp-content/uploads/2008/08/spring_y_el_principio_de_hollywood.pdf

Dos Ideas. Inyección De Dependencia. [En línea] [Citado: el 20 de febrero del 2009.] http://www.dosideas.com/wiki/Inyeccion_De_Dependencia

Dos Ideas. Integrar Singleton En Spring. [En línea] [Citado: el 20 de febrero del 2009.] http://www.dosideas.com/wiki/Integrar_Singletons_En_Spring

***Urs Peter**. Configuration-hell remedy with Singleton injection. [En línea] [Citado: el 5 de enero del 2009.]* http://springtips.blogspot.com/2007/06/configuration-hell-remedy-with.html

Cristian. Ejemplo básico de programación con JDBC y MySQL. [En línea] [Citado: el 3 de marzo del 2009.] http://casidiablo.net/ejemplo-basico-de-programacion-con-jdbc-y-mysql/

María A. Mendoza Sánchez. Metodologías De Desarrollo De Software. [En línea] [Citado: el 15 de diciembre del 2008.] http://www.informatizate.net/articulos/pdfs/metodologias_de_desarrollo_de_software_07062004.pdf

Andrés Dorado. Patrones de diseño. [En línea] [Citado: el 26 de febrero del 2009.] http://www.slideshare.net/2008PA2Info3/tema-de-revision-2003

Ramiro Lago. Tecnología orientada a procesos de negocio. [En línea] [Citado: el 4 de abril del 2009.] http://www.proactiva-calidad.com/java/principal.html

Deepak Alur, John Crupi, Dan Malks. Core J2EE Patterns Best Practices and Design Strategies. [En línea] [Citado: el 19 de enero del 2009.] http://books.google.com.cu/books?id=1dx34EMVyi8C&dq=Core+J2EE%E2%84%A2+Patterns:+Best+Practices+and+Design+Strategies&printsec=frontcover&source=bn&hl=es&ei=zKAUSvKADuGrtgeFirSZBA&sa=X&oi=book_result&ct=result&resnum=4#PPP1,M1

Julio Tentor. Arquitectura de N-Capas y N-Niveles. [En línea] [Citado: el 17 de abril del 2009.] http://www.jtentor.com.ar/post/Arquitectura-de-N-Capas-y-N-Niveles.aspx

BIBLIOGRAFÍA

Dra. Anaisa Hernández González. Un Método para el Diseño de la Base de Datos a partir del Modelo Orientado a Objetos. [En línea] [Citado: el 5 de mayo del 2009.] *http://74.125.47.132/search?q=cache:zMDSHI7VoD4J:www.dict.uh.cu/Revistas/Educ_Sup/012002/Art080102.pdf+Un+M%C3%A9todo+para+el+Dise%C3%B1o+de+la+Base+de+Datos+a+partir+del+Modelo+Orientado+a+Objetos&cd=1&hl=es&ct=clnk&gl=cu.*

Luis Fernández Sanz. La Importancia de la calidad del software. [En línea] [Citado: el 29 de abril del 2009.] *http://www.baquia.com/articulos/software/noticia/14128/la-importancia-de-la-calidad-del-software.*

ANEXOS

Anexo 1: Descripción detallada de los Casos de Uso del Sistema.

1. Gestionar Perfil

Nombre del CU	Gestionar perfil.
Actores	Administrador. (inicia)
Propósito	Permitir a un administrador la gestión de un perfil de usuario determinado.
Resumen	El caso de uso se inicia cuando el administrador decide gestionar perfil de usuario. El sistema debe permitir crear, modificar o habilitar /deshabilitar un perfil de usuario, se realiza la operación deseada, finalizando así el caso de uso.
Referencias	**R1**
Precondiciones	El administrador debe estar autenticado.
Poscondiciones	

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El administrador selecciona opción Gestionar perfil.	1.1. El sistema muestra las opciones de crear o modificar perfil de usuario.
2. El administrador selecciona una de las opciones.	2.1. El sistema gestiona la opción seleccionada. a) Crear. b) Modificar.
Sección "Crear"	
Acciones del Actor	**Respuesta del Sistema**
3. El administrador selecciona la opción Crear.	3.1. El sistema muestra un listados con los permisos existentes, los campos a llenar y a seleccionar para crear el nuevo perfil; (nombre del perfil y permisos).
4. El administrador llena los campos y envía la información.	4.1. El sistema verifica completitud de los datos. 4.2. El sistema verifica la existencia del perfil (nombre iguales o nombre distintos con permisos iguales).

	4.3. El sistema crea el perfil. 4.4. El sistema muestra un mensaje de información “Perfil creado”.
Flujo Alternativo	
Acciones del Actor	**Respuesta del Sistema**
4. El administrador selecciona la opción Cancelar.	4.1. El sistema muestra un mensaje de confirmación “Está seguro que desea cancelar la acción”.
5. El administrador acepta mensaje de confirmación.	5.2. El sistema elimina los datos creados. 5.1. El sistema muestra la vista anterior.
6. El administrador no llena todos los campos.	6.1. El sistema muestra un mensaje de error “Debe llenar todos los campos”. 6.2. El sistema regresa al paso 4 del curso normal de eventos.
Sección “Modificar”	
Acciones del Actor	**Respuesta del Sistema**
3. El administrador selecciona la opción Modificar.	3.1. El sistema muestra un listado con los perfiles existentes.
4. El administrador selecciona el perfil a modificar.	4.1. El sistema muestra los datos del perfil y los campos a modificar.
5. El administrador modifica los campos que desea y selecciona la opción Modificar.	5.1. El sistema verifica completitud de los datos. 5.2. El sistema verifica la existencia del perfil modificado (nombre iguales o nombre distintos con permisos iguales). 5.3. El sistema muestra un mensaje “Está seguro que desea modificar estos campos”.
6. El administrador acepta mensaje de confirmación.	6.1. El sistema modifica los campos. 6.2. El sistema muestra un mensaje de información “Perfil modificado”.
Flujo Alternativo	
Acciones del Actor	**Respuesta del Sistema**
5. El administrador selecciona la opción Cancelar.	5.1. El sistema muestra un mensaje de confirmación “Está seguro que desea cancelar la acción”.
6. El administrador acepta mensaje de confirmación.	6.1. El sistema elimina los datos creados. 6.2. El sistema muestra la vista anterior.
7. El administrador no llena todos los	7.1. El sistema muestra un mensaje de error

campos.	"Debe llenar todos los campos". 7.2. El sistema regresa al paso 5 del curso normal de eventos.
8. El administrador selecciona la opción Modificar sin seleccionar un perfil.	8.1. El sistema muestra un mensaje de error "Debe de seleccionar un perfil".
Prioridad:	Crítico.

2. Gestionar puestos de trabajo.

Nombre del CU	Gestionar puestos de trabajo.
Actores	Administrador. (inicia)
Propósito	Permitir a un administrador gestionar puestos de trabajo.
Resumen	El caso de uso se inicia cuando el administrador decide gestionar puestos de trabajo. El sistema debe permitir crear o modificar un puesto de trabajo, se realiza la operación deseada, finalizando así el caso de uso.
Referencias	R13
Precondiciones	El administrador debe estar autenticado.
Poscondiciones	
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Gestionar puestos de trabajo.	1.1. El sistema muestra las opciones de crear o modificar puestos de trabajo.
2. El administrador selecciona una de las opciones.	2.1. El sistema gestiona la opción seleccionada. a) Crear. b) Modificar.
Sección "Crear"	
Acciones del Actor	**Respuesta del Sistema**
3. El administrador selecciona la opción Crear.	3.1. El sistema muestra los siguientes campos a llenar para crear un nuevo puesto de trabajo: No. Puesto, Marca, CPU, Memoria y Dirección Mac
4. El administrador llena los campos y envía la información.	4.1. El sistema verifica completitud de los datos. 4.2. El sistema verifica existencia del puesto de trabajo.

	4.3. El sistema crea el puesto de trabajo. 4.4. El sistema muestra un mensaje de información “Puesto de Trabajo creado”.
Flujo Alternativo	
Acciones del Actor	**Respuesta del Sistema**
4. El administrador selecciona la opción cancelar.	4.1. El sistema muestra un mensaje de confirmación “Está seguro que desea cancelar la acción”.
5. El administrador acepta mensaje de confirmación.	5.1. El sistema elimina los datos creados. 5.2. El sistema muestra la vista anterior.
6. El administrador no llena todos los campos.	6.1. El sistema muestra un mensaje de error “Debe llenar todos los campos.” 6.2. El sistema regresa al paso 4 del curso normal de eventos.
Sección “Modificar”	
Acciones del Actor	**Respuesta del Sistema**
3. El administrador selecciona la opción Modificar.	3.1. El sistema muestra un listado con los puestos de trabajos existentes.
4. El administrador selecciona el puesto de trabajo a modificar.	4.1. El sistema muestra los datos del puesto de trabajo y los campos a modificar.
5. El administrador modifica los campos que desea y selecciona la opción Modificar.	51. El sistema verifica completitud de los datos. 5.2. El sistema verifica existencia del puesto de trabajo. 5.3. El sistema muestra un mensaje “Está seguro que desea modificar estos campos”.
6. El administrador acepta mensaje de confirmación.	6.1. El sistema modifica los campos. 6.2. El sistema muestra un mensaje de información “Puesto de Trabajo modificado”.
Flujo Alternativo	
Acciones del Actor	**Respuesta del Sistema**
5. El administrador selecciona la opción Cancelar.	5.1. El sistema muestra un mensaje de confirmación “Está seguro que desea cancelar la acción”.
6. El administrador acepta mensaje de confirmación.	6.1. El sistema elimina los datos creados. 6.2. El sistema muestra la vista anterior.
7. El administrador no llena todos los campos.	7.1. El sistema muestra un mensaje de error “Debe llenar todos los campos”.

	7.2. El sistema regresa al paso 6 del curso normal de eventos.
8. El administrador selecciona la opción Modificar sin seleccionar un puesto de trabajo.	8.1. El sistema muestra un mensaje de error “Debe de seleccionar un puesto de trabajo”.
Prioridad:	Crítico.

3. Conocer puestos de trabajo.

Nombre del CU	Conocer puestos de trabajo.
Actores	Probador. (inicia)
Propósito	Permitir a un probador registrarse en el LIPS y conocer su puesto de trabajo.
Resumen	El caso de uso se inicia cuando el probador pasa su identificación por el lector de código de barra y termina cuando es ubicado en un puesto de trabajo.
Referencias	R39, R40
Precondiciones	El probador debe portar su identificación.
Poscondiciones	
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El probador pasa la identificación por el lector de código de barra.	1.1. El sistema muestra los siguientes datos: (nombre, apellidos, grupo, puesto de trabajo).
Prioridad:	No Crítico.

4. Gestionar Solicitud de Inclusión de Usuario.

Nombre del CU	Gestionar Solicitud de IU.
Actores	Solicitante de Usuario (inicia)
Propósito	Permitir a un solicitante de usuario gestionar solicitud de IU.
Resumen	El caso de uso se inicia cuando el solicitante de usuario decide gestionar solicitud de IU. El sistema debe permitir crear o eliminar dicha solicitud, se realiza la operación deseada, finalizando así el caso de uso.
Referencias	R2, R3

Precondiciones	El Solicitante de Usuario debe estar autenticado.
Poscondiciones	

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El solicitante de usuario selecciona opción Gestionar Solicitud de IU.	1.1. El sistema muestra las opciones de crear o eliminar Solicitud de IU.
2. El solicitante de usuario selecciona una de las opciones.	2.1. El sistema gestiona la opción seleccionada. a) Crear. b) Eliminar
Sección "Crear"	
Acciones del Actor	**Respuesta del Sistema**
3. El solicitante de usuario selecciona la opción Crear. 3.1. El solicitante de usuario selecciona el tipo de usuario a incluir (usuario uci).	3.1. El sistema le da la posibilidad de buscar el usuario (*ver caso de uso buscar usuario R7*). 3.2. El sistema muestra resultado de la búsqueda (nombre, apellidos, ubicación y correo electrónico) y un campo a llenar (justificación).
4. El solicitante de usuario llena el campo y envía la información.	4.1. El sistema verifica completitud de los datos. 4.2. El sistema verifica la existencia de la solicitud. 4.3. El sistema crea la Solicitud de inclusión de usuario. 4.4. El sistema notifica la existencia de nuevas solicitudes. 4.5. El sistema muestra un mensaje de información "Solicitud creada".
Flujo Alternativo	
Acciones del Actor	**Respuesta del Sistema**
3.1. El solicitante de usuario selecciona el tipo de usuario a incluir (usuario externo).	3.1. El sistema muestra los siguientes campos a llenar nombre, apellidos, ubicación, correo electrónico y justificación.
4. El solicitante de usuario llena los campos y envía la información.	4.1. El sistema verifica completitud de los datos. 4.2. El sistema verifica la existencia de la solicitud.

	4.3. El sistema crea la Solicitud de inclusión de usuario. 4.4. El sistema muestra un mensaje de información “Solicitud creada”.
5. El solicitante de usuario selecciona la opción Cancelar.	5.1. El sistema muestra un mensaje de confirmación “Está seguro que desea cancelar la acción”.
6. El solicitante de usuario acepta mensaje de confirmación.	6.1. El sistema elimina los datos creados. 6.2. El sistema muestra la vista anterior.
7. El solicitante de usuario no llena todos los campos.	7.1. El sistema muestra un mensaje de error “Debe llenar todos los campos”. 7.2. El sistema regresa al paso 4 del curso normal de eventos.
Sección “Eliminar”	
Acciones del Actor	**Respuesta del Sistema**
3. El solicitante de usuario selecciona la opción Eliminar.	3.1. El sistema muestra un listado con las solicitudes (pendientes).
4. El solicitante de usuario selecciona la solicitud que desea eliminar.	4.1. El sistema muestra los datos de la solicitud seleccionada.
5. El solicitante de usuario selecciona la opción Eliminar.	5.1. El sistema muestra un mensaje de confirmación “Está seguro que desea eliminar esta solicitud”.
6. El solicitante de usuario acepta mensaje de confirmación.	6.1. El sistema elimina la solicitud seleccionada. 6.2. El sistema muestra un mensaje de información “Solicitud eliminada”.
Flujo Alternativo	
Acciones del Actor	**Respuesta del Sistema**
5. El solicitante de usuario selecciona la opción Cancelar.	5.1. El sistema muestra un mensaje de confirmación “Está seguro que desea cancelar la acción”.
6. El solicitante de usuario acepta mensaje de confirmación.	6.1. El sistema cancela todas las acciones. 6.2. El sistema muestra la vista anterior.
7. El solicitante de usuario selecciona la opción Eliminar sin seleccionar una Solicitud de Inclusión de Usuario.	7.1. El sistema muestra un mensaje de error “Debe de seleccionar una Solicitud de Inclusión de Usuario”.
Prioridad:	Crítico.

5. Tramitar Estado de Solicitud de Inclusión de Usuario.

Nombre del CU	Tramitar Estado de Solicitud de IU.
Actores	Administrador. (inicia)
Propósito	Permitir a un administrador tramitar estado de la solicitud de IU.
Resumen	El caso de uso se inicia cuando el Administrador decide tramitar estado de solicitud de IU. El sistema debe permitir aprobar, rechazar o cancelar una solicitud de IU, se realiza la operación deseada, finalizando así el caso de uso.
Referencias	R4, R5
Precondiciones	La solicitud de IU debe estar creada.
Poscondiciones	

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Tramitar Estado.	1.1. El sistema muestra un listado con todas las solicitudes pendientes.
2. El administrador selecciona la solicitud que desea tramitar.	2.1. El sistema muestra los datos de esa solicitud (nombre, apellidos, ubicación, justificación) y las opciones de aprobar, rechazar o cancelar dicha solicitud IU.
3. El administrador selecciona una de las opciones.	3.1. El sistema gestiona la opción seleccionada. a) Aprobar. b) Rechazar. c) Cancelar.
Sección "Aprobar"	
Acciones del Actor	**Respuesta del Sistema**
4. El administrador selecciona la opción Aprobar.	4.1. El sistema muestra un listado con los perfiles a seleccionar.
5. El administrador selecciona el perfil que desea y envía la información.	5.1. El sistema crea el usuario. 5.2. El sistema elimina la solicitud de inclusión de usuario. 5.3. El sistema notifica el estado de la solicitud de IU al solicitante de usuario. 5.4. El sistema muestra un mensaje de

	información “Usuario creado”.
Flujo Alternativo	
Acciones del Actor	**Respuesta del Sistema**
5. El administrador selecciona la opción Cancelar.	5.1. El sistema muestra un mensaje de confirmación “Está seguro que desea cancelar la acción”.
6. El administrador acepta mensaje de confirmación.	6.1. El sistema cancela todas las acciones. 6.2. El sistema muestra la vista anterior.
Sección “Rechazar”	
Acciones del Actor	**Respuesta del Sistema**
4. El administrador selecciona la opción Rechazar.	4.1. El sistema muestra el siguiente campo a llenar (justificación del Rechazo).
5. El administrador llena el campo y envía la información.	5.1. El sistema verifica completitud de los datos. 5.2. El sistema muestra un mensaje de confirmación “Está seguro que desea rechazar la solicitud”.
6. El administrador acepta mensaje de confirmación.	6.1. El sistema rechaza la solicitud de inclusión de usuario. 6.2. El sistema guarda justificación de rechazo 6.3. El sistema notifica al solicitante de usuario el rechazo de la solicitud. 6.4. El sistema muestra un mensaje de información “Solicitud rechazada”.
Flujo Alternativo	
Acciones del Actor	**Respuesta del Sistema**
5. El administrador selecciona la opción Cancelar.	5.1. El sistema muestra un mensaje de confirmación “Está seguro que desea cancelar la acción”.
6. El administrador acepta mensaje de confirmación.	6.1. El sistema cancela todas las acciones. 6.2. El sistema muestra la vista anterior.
Sección “Cancelar”	
Acciones del Actor	**Respuesta del Sistema**
4. El administrador selecciona la opción Cancelar.	4.1. El sistema muestra un mensaje de confirmación “Está seguro que desea cancelar la acción”.
5. El administrador acepta mensaje de	5.1. El sistema cancela todas las acciones.

confirmación.	5.2. El sistema muestra la vista anterior.
Prioridad:	Crítico.

6. Gestionar Usuario.

Nombre del CU	Gestionar Usuario.
Actores	Administrador (inicia)
Propósito	Permitir a un administrador gestionar usuarios en el sistema.
Resumen	El caso de uso se inicia cuando el Administrador decide gestionar usuarios. El sistema debe permitir modificar o eliminar usuarios, se realiza la operación deseada, finalizando así el caso de uso.
Referencias	R6, R7
Precondiciones	El usuario debe estar creado.
Poscondiciones	

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción Gestionar usuario.	1.1. El sistema muestra las opciones de modificar o eliminar usuario.
2. El administrador selecciona una de las opciones.	2.1. El sistema gestiona la opción seleccionada. a) Modificar. b) Eliminar.
Sección "Modificar"	
Acciones del Actor	**Respuesta del Sistema**
3. El administrador selecciona la opción Modificar.	3.1. El sistema le da la posibilidad de buscar el usuario a modificar (*ver caso de uso buscar usuario R7*). 3.2. El sistema muestra resultado de la búsqueda.
4. El administrador selecciona el usuario a modificar.	4.1. El sistema muestra los datos del usuario encontrado y el campo a modificar (perfiles).
5. El administrador modifica el campo perfiles y selecciona la opción Modificar.	5.1. El sistema verifica completitud de los datos. 5.2. El sistema muestra un mensaje "Está seguro que desea modificar estos campos".
6. El administrador acepta mensaje de	6.1. El sistema modifica los campos.

confirmación.	6.2. El sistema muestra un mensaje de información “Usuario modificado”.
Flujo Alternativo	
Acciones del Actor	**Respuesta del Sistema**
5. El administrador selecciona la opción Cancelar.	5.1. El sistema muestra un mensaje de confirmación “Está seguro que desea cancelar la acción”.
6. El administrador acepta mensaje de confirmación.	6.1. El sistema elimina los datos creados. 6.2. El sistema muestra la vista anterior.
7. El administrador no llena todos los campos.	7.1. El sistema muestra un mensaje de error “Debe llenar todos los campos”. 7.2. El sistema regresa al paso 5 del curso normal de eventos.
8. El administrador selecciona la opción Modificar sin seleccionar un usuario.	8.1. El sistema muestra un mensaje de error “Debe de seleccionar un usuario”.
Sección “Eliminar”	
Acciones del Actor	**Respuesta del Sistema**
3. El administrador selecciona la opción Eliminar.	3.1. El sistema le da la posibilidad de buscar el usuario a eliminar (*ver caso de uso buscar usuario R9*). 3.2. El sistema muestra resultado de la búsqueda.
4. El administrador selecciona el usuario que desea eliminar.	4.1. El sistema muestra los datos del usuario seleccionado.
5. El administrador selecciona la opción Eliminar.	El sistema muestra un mensaje de confirmación “Está seguro que desea eliminar este usuario”.
5. El administrador acepta mensaje de confirmación.	5.1. El sistema elimina el usuario seleccionado. 5.2. El sistema muestra un mensaje de información “Usuario eliminado”.
Flujo Alternativo	
Acciones del Actor	**Respuesta del Sistema**
4. El administrador selecciona la opción Cancelar.	4.1. El sistema muestra un mensaje de confirmación “Está seguro que desea cancelar la acción”.
5. El administrador acepta mensaje de confirmación.	5.1. El sistema cancela todas las acciones. 5.2. El sistema muestra la vista anterior.

Prioridad:	Crítico.

7. Buscar Usuario.

Nombre del CU	Buscar Usuario (include de Gestionar Usuario).	
Actores		
Propósito	Permitir buscar un usuario determinado en el sistema.	
Resumen	El caso de uso se inicia cuando se decide buscar un usuario y finaliza cuando el perfil es encontrado.	
Referencias	R7	
Precondiciones	El administrador debe estar autenticado.	
Poscondiciones	El usuario es encontrado.	
Curso Normal de los Eventos		
Acciones del Actor		**Acciones del Actor**
1. El administrador selecciona la opción Buscar usuario.		1.1. El sistema muestra los siguientes campos a llenar: (usuario, perfil, nombre) para realizar la búsqueda.
2. El administrador llena los campos y envía la información.		2.1. El sistema verifica completitud de los datos. 2.2. El sistema muestra resultado de la búsqueda.
Flujo Alternativo		
Acciones del Actor		**Respuesta del Sistema**
2. El administrador no llena todos los campos.		2.2. El sistema muestra un mensaje “Debe de llenar los campos obligatorios”. 2.3. El sistema regresa al paso 2 del curso normal de eventos.
Prioridad:	Crítico.	

8. Autenticar.

Nombre del CU	Autenticar.
Actores	Usuario genérico 3 (Usuario Genérico 1 y Probador). (inicia)
Propósito	Permitir a un usuario genérico autenticarse en el sistema.
Resumen	El caso de uso se inicia cuando el usuario decide autenticarse en el sistema y finaliza cuando el sistema le otorga los permisos

	según el rol que desempeña.
Referencias	R15
Precondiciones	
Poscondiciones	El usuario ha sido autenticado.

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario genérico llena los campos, usuario, contraseña y captcha.	1.1. El sistema verifica completitud de los datos. 1.2. El sistema le otorga los permisos según el rol que desempeña dentro del sistema.
Flujo Alternativo	
Acciones del Actor	**Respuesta del Sistema**
	1.1. El sistema muestra un mensaje de error "El nombre de usuario o contraseña son incorrectos". 1.2. El sistema regresa al paso 1 del curso normal de eventos.
Prioridad:	Crítico.

9. Probar Artefacto Asignado.

Nombre del CU	Probar artefacto asignado.
Actores	Usuario genérico 2 (Probador y Coordinador de Pruebas). (inicia)
Propósito	Permitir a un usuario genérico probar un artefacto.
Resumen	El caso de uso se inicia cuando el sistema muestra el módulo de trabajo donde el usuario genérico 2 probará el artefacto asignado y termina cuando el artefacto es probado.
Referencias	R 20, R42, R43
Precondiciones	El usuario genérico 2 debe estar autenticado.
Poscondiciones	

Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**

	1.1. El sistema muestra el módulo de trabajo (la pantalla se divide en dos, en la parte derecha la lista de chequeo, en la parte izquierda el artefacto a probar y en cada aspecto de la lista de chequeo un campo analizada/no analizada.) y las opciones de crear no conformidad, confirmar estado de la no conformidad, guardar, guardar y cerrar, cancelar.
2.1. El usuario genérico selecciona una de las opciones.	2.1. El sistema gestiona la opción seleccionada. a) Crear No Conformidad. b) Guardar. c) Guardar y Cerrar. d) Cancelar.
Sección "Crear No Conformidad"	
Acciones del Actor	**Respuesta del Sistema**
3. El usuario genérico selecciona la opción Crear No Conformidades.	3.1. El sistema muestra los siguientes campos a llenar: (nombre del proyecto, módulo, artefacto, ubicación, texto de la no conformidad, clasificación) y las opciones crear, cancelar o nueva.
4. El usuario genérico llena los campos y selecciona la opción Crear.	4.1. El sistema verifica completitud de los datos. 4.2. El sistema crea la No Conformidad. 4.3. El sistema muestra un mensaje de información "No Conformidad creada".
Flujo Alternativo	
Acciones del Actor	**Respuesta del Sistema**
4. El usuario genérico selecciona la opción Cancelar.	4.1. El sistema muestra un mensaje de confirmación "Está seguro que desea cancelar la acción".
5. El usuario genérico acepta mensaje de confirmación.	5.1. El sistema elimina los datos creados. 5.2. El sistema muestra la vista anterior.
6. El usuario genérico no llena todos los campos.	6.1. El sistema muestra un mensaje de error "Debe llenar todos los campos". 6.2. El sistema regresa al paso 4 del curso normal de eventos.

Sección "Guardar"	
Acciones del Actor	**Respuesta del Sistema**
3. El usuario genérico selecciona la opción Guardar.	3.1. El sistema guarda todos los cambios.
Sección "Guardar y Cerrar"	
Acciones del Actor	**Respuesta del Sistema**
3. El usuario genérico selecciona la opción Guardar y Cerrar.	3.1. El sistema guarda los cambios y cierra el módulo de trabajo.
Sección "Cancelar"	
Acciones del Actor	**Respuesta del Sistema**
3. El usuario genérico selecciona la opción Cancelar.	3.1. El sistema muestra un mensaje de confirmación "Está seguro que desea cancelar la acción".
4. El usuario genérico acepta mensaje de confirmación.	4.1. El sistema elimina los datos creados. 4.2. El sistema muestra la vista anterior.
Prioridad:	Crítico.

10. Gestionar No Conformidades.

Nombre del CU	Gestionar No Conformidades.
Actores	Coordinador de Pruebas (inicia)
Propósito	Permitir a un coordinador de pruebas Gestionar una No Conformidad.
Resumen	El caso de uso se inicia cuando el coordinador decide gestionar no conformidades. El sistema debe permitir modificar o eliminar no conformidades, se realiza la operación deseada, finalizando así el caso de uso.
Referencias	R 20, R21
Precondiciones	La No Conformidad debe estar creada, se le debe enviar una notificación de la existencia de nuevas NC creadas.
Poscondiciones	
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El coordinador de pruebas selecciona la opción Gestionar No Conformidades.	1.1. El sistema muestra un listado de las nuevas NC.
2. El coordinador de pruebas selecciona una de las No Conformidades.	2.1. El sistema muestra los datos de la no conformidad seleccionada y las opciones de

	modificar, eliminar o cancelar.
3. El coordinador de pruebas selecciona una de las opciones.	3.1. El sistema gestiona la opción seleccionada. a) Modificar. b) Eliminar. c) Cancelar.
Sección "Modificar"	
Acciones del Actor	**Respuesta del Sistema**
4. El coordinador de pruebas selecciona la opción Modificar.	4.1. El sistema muestra una nueva pantalla con los campos a modificar.
5. El coordinador de pruebas modifica los campos que desea (descripción, estado de NC, justificación) y envía la información.	5.1. El sistema verifica completitud de los datos. 5.2. El sistema muestra un mensaje "Está seguro que desea modificar estos campos".
6. El coordinador de pruebas acepta mensaje de confirmación.	6.1. El sistema modifica los campos. 6.2. El sistema muestra un mensaje de información "No Conformidad modificada". 6.3. Notifica al equipo de desarrollo existencia de NC pendientes de solución.
Flujo Alternativo	
Acciones del Actor	**Respuesta del Sistema**
4. El coordinador de pruebas selecciona la opción Cancelar.	4.1. El sistema muestra un mensaje de confirmación "Está seguro que desea cancelar la acción".
5. El coordinador de pruebas acepta mensaje de confirmación.	5.1. El sistema elimina los datos creados. 5.2. El sistema muestra la vista anterior.
6. El coordinador de pruebas no llena todos los campos.	6.1. El sistema muestra un mensaje de error "Debe llenar todos los campos". 6.2. El sistema regresa al paso 5 del curso normal de eventos.
7. El coordinador de pruebas selecciona la opción Modificar sin seleccionar una No Conformidad.	7.1. El sistema muestra un mensaje de error "Debe de seleccionar una No Conformidad".
Sección "Eliminar"	
Acciones del Actor	**Respuesta del Sistema**
4. El coordinador de pruebas selecciona la opción Eliminar.	4.1. El sistema muestra un mensaje de confirmación "Está seguro que desea eliminar esta No Conformidad".

5. El coordinador de pruebas acepta mensaje de confirmación.	5.1. El sistema elimina la No Conformidad seleccionada. 5.2. El sistema muestra un mensaje de información “No Conformidad eliminada”.
Flujo Alternativo	
Acciones del Actor	**Respuesta del Sistema**
4. El coordinador de pruebas selecciona la opción Cancelar.	4.1. El sistema muestra un mensaje de confirmación “Está seguro que desea cancelar la acción”.
5. El coordinador de pruebas acepta mensaje de confirmación.	5.1. El sistema cancela todas las acciones. 5.2. El sistema muestra la vista anterior.
6. El coordinador de pruebas selecciona la opción Eliminar sin seleccionar una No Conformidad.	6.1. El sistema muestra un mensaje de error “Debe de seleccionar una No Conformidad”.
Sección “Cancelar”	
Acciones del Actor	**Respuesta del Sistema**
4. El coordinador de pruebas selecciona la opción Cancelar.	4.1. El sistema muestra un mensaje de confirmación “Está seguro que desea cancelar la acción”.
5. El coordinador de pruebas acepta mensaje de confirmación.	5.1. El sistema elimina los datos creados. 5.2. El sistema muestra la vista anterior.
Prioridad:	Crítico.

Anexo 2: Resumen de preguntas realizadas durante las entrevistas.

1. ¿Cuáles son las actividades que se desarrollan en el Grupo de Liberación y Pruebas de Software durante el Proceso de Liberación de Productos Software?
2. ¿En qué consiste cada una de esas actividades antes mencionada? Describir por pasos como se realizan.
3. ¿Qué otras actividades no relacionadas con el Proceso de Liberación de Productos Software se realizan en dicho Grupo?
4. ¿Qué tipo de artefactos o entregables se procesan?
5. ¿Cuántos proyectos productivos se pueden procesar simultáneamente?

Anexo 3: Listas de Chequeos aplicadas durante las revisiones técnicas a los entregables confeccionados.

Lista de Chequeo Especificación de Requisitos. Estructura.

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	1. ¿Se especifica el Nombre del Proyecto en el documento de especificación de requisitos?				
crítico	2. ¿Se especifica el Nombre del Producto en el documento de especificación de requisitos?				
crítico	3. ¿Se especifica la versión actual del documento de especificación de requisitos?				
crítico	4. ¿Se especifica el control de versiones en el documento de especificación de requisitos?				
crítico	5. ¿Se especifica en el control de versiones la fecha, la versión, la descripción y el autor de todas las iteraciones que ha pasado				

	en el documento de especificación de requisitos?				
crítico	6. ¿Coincide la versión de la primera página con la última versión que está en el control de versiones? Ver en el documento de especificación de requisitos.				
crítico	7. ¿Aparecen las Reglas de Confidencialidad en el documento de especificación de requisitos?				
crítico	8. ¿Se especifica la tabla de contenido en el documento de especificación de requisitos?				
crítico	9. ¿Se especifica en el documento de especificación de requisitos la introducción?				
crítico	10. ¿Se especifica en la introducción en el documento de especificación de requisitos el propósito de este artefacto?				
crítico	11. ¿Se especifica en la introducción en el				

	documento de especificación de requisitos el alcance de este artefacto?				
	12. ¿Si existen definiciones, acrónimos y abreviaturas, se han definido estos en el documento de especificación de requisitos?				
	13. ¿Si existen referencias, se han definido en el documento de especificación de requisitos?				
crítico	14. ¿Se especifica los Requisitos Funcionales en el documento de especificación de requisitos?				
crítico	15. ¿Se especifica los Requisitos No Funcionales en el documento de especificación de requisitos?				
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**

crítico	1. ¿Están todos los requisitos redactados de forma simple y clara para aquellos que vayan a consultarlo en un futuro? Ver documento de Especificación de Requisitos.				
	2. ¿Debería especificarse algún requisito con más detalle? Ver documento de Especificación de Requisitos.				
	3. ¿Debería especificarse algún requisito con menos detalles? Ver documento de Especificación de Requisitos.				
	4. ¿Todos los requisitos identificados se centran en lo que el sistema debe hacer y no como el sistema debe hacerlo? Ver documento de Especificación de Requisitos.				
crítico	5. ¿Han sido abordadas e identificadas los valores de				

	entradas y salidas? Ver documento de Especificación de Requisitos.				
	6. ¿Han sido incluidos las respuestas válidas y no válidas de los valores de entrada? Ver documento de Especificación de Requisitos.				
	7. ¿Han sido identificadas las restricciones de interfaz externa? Ver documento de Especificación de Requisitos.				
	8. ¿Los requerimientos de soporte y usabilidad se han identificados? Ver documento de Especificación de Requisitos.				
	9. ¿Se han identificado los requerimientos de seguridad (confidencialidad, integridad, disponibilidad)? Ver documento de Especificación de Requisitos.				
	10. ¿Se puede verificar cada requisito? (Un requisito se dice que es verificable si existe algún proceso no excesivamente costoso por el cual una persona				

	o una máquina pueda chequear que el software satisface dicho requerimiento, ejemplo la especificación del caso de uso). Ver documento de Especificación de Requisitos.				
	11. ¿Se han enumerado los requisitos incluso los que se derivan de otros requisitos? Ver documento de Especificación de Requisitos.				
	12. ¿Se puede trazar cada requisito al origen en el entorno del problema, (caso de uso del negocio)? Ver documento de Especificación de Requisitos.				
	13. ¿Se han especificado todos los posibles cambios en los requisitos, incluyendo la probabilidad de cambio? Ver documento de Especificación de Requisitos.				
	14. ¿No aparece un mismo requisito en más de un lugar del documento de especificación? Ver documento de Especificación de Requisitos.				

crítico	15. ¿No existe contradicción entre lo especificado por un requisito y lo especificado por otro? Ver documento de Especificación de Requisitos.				
	16. ¿Existe correspondencia entre el modelo de caso de uso, las Especificaciones Suplementarias y las especificaciones de requerimientos? Ver documento de Especificación de Requisitos.				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	1. ¿Ha identificado errores ortográficos?				
Crítico	2. ¿Se entiende claramente lo que se ha especificado en el documento?				
	3. ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?				
	4. ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene				

	el documento?				

Lista de Chequeo: Especificación de Casos de Uso. Estructura.

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	1. ¿Se especifica el nombre del Proyecto en el Modelo de Caso de Uso del Sistema?				
crítico	2. ¿Se especifica el nombre del Producto en el Modelo de Caso de Uso del Sistema?				
crítico	3. ¿Se especifica en el control de versiones la fecha, la versión, la descripción y el autor de todas las iteraciones que ha pasado el modelo del Caso de Uso del Sistema?				
	4. ¿Coincide la última versión que está en el control de las versiones con la versión que está en la primera página del Modelo				

	de Caso de Uso del Sistema?				
crítico	5. ¿Se especifica en el Modelo de Caso de Uso del Sistema la tabla de contenido?				
crítico	6. ¿Se especifica en el Modelo de Caso de Uso del Sistema la introducción?				
crítico	7. ¿Se especifica en la introducción del Modelo de Caso de Uso del Sistema el propósito de este artefacto?				
crítico	8. ¿Se especifica en la introducción del Modelo de Caso de Uso del Sistema el alcance de este artefacto?				
	9. ¿Si existen definiciones, acrónimos y abreviaturas, se ha definido en la introducción del Modelo de Caso de Uso del Sistema?				
	10. ¿Si existen referencias, se ha definido en la introducción del Modelo de Caso de Uso del Sistema?				
crítico	11. ¿Se especifica en el Modelo de Caso de Uso del				

	Sistema los actores del sistema?				
crítico	12. ¿Se especifica en el Modelo de Caso de Uso del Sistema el diagrama de Caso de Uso del mismo?				
crítico	13. ¿Se especifica en el Modelo de Caso de Uso del Sistema la especificación de cada Caso de Uso?				
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Aspectos Generales					
	1. ¿Cada caso de uso registra claramente lo que el sistema debe hacer?				
crítico	2. ¿Están clasificado los casos de uso que definen la arquitectura básica del sistema?(críticos)				
	3. ¿Se ha identificado los casos de uso que darán soporte y mantenimiento al sistema?				
crítico	4. ¿Se ha descrito con precisión todas las alternativas o excepciones?				

	5. ¿Están clasificados los casos de uso que sirven de apoyo a los caso de uso que cubren las principales funciones que el sistema debe realizar? (secundarios)				
	6. ¿Están clasificados los casos de uso que no son claves para la arquitectura? (secundarios, apoyo)				
Nombre del Caso de Uso					
crítico	1. ¿Está en infinitivo y refleja de manera clara el objetivo del usuario sobre el sistema?				
crítico	2. ¿El nombre del caso de uso es único?				
	3. ¿El nombre del caso de uso es intuitivo?				
Descripción					
	1. ¿El resumen dice como se inicia, como termina y las operaciones principales que realiza el caso de uso?				
Precondición					
	1. ¿Se escribe una precondición si y solo si a partir de la ocurrencia de un suceso determinado comienza el caso de uso?				

	2. ¿La precondición es válida tanto para flujos básicos como flujos alternativos?				
Poscondición					
	1. ¿La pos condición plasma cambios que suceden en el sistema al terminarse de ejecutar el caso de uso?				
Complejidad del CU					
	1. ¿Se especifica la complejidad del caso de uso?				
Forma de presentar la información					
crítico	1. ¿Está descrito el caso de uso en presente?				
crítico	2. ¿Se describe de manera comprensible y detallada las acciones del actor frente al sistema? ¿Está lo más parecido a un manual de ayuda?				
Actores del CU					
crítico	1. ¿El Caso de Uso está relacionado con al menos un actor?				
crítico	2. ¿Si hay dos actores interactuando con el caso de uso está generalizado en uno solo?				
crítico	3. ¿Si el caso de uso es abstracto (include, extend,				

	generalización-especialización), no lo inicializa ningún actor?				
Flujo Básico					
	1. ¿Comienza diciendo "El caso de uso se inicia cuando el actor…"?				
	2. ¿Termina diciendo en un evento independiente "El caso de uso termina"?				
	3. ¿No existen abreviaturas?				
crítico	4. ¿Las partes del flujo de eventos que se repiten en otro caso de uso se especifican como un Caso de Uso incluido?				
crítico	5. ¿Si las alternativas que se describen casi nunca ocurren o son alternativas comunes a otros casos de uso se especifican como un Caso de Uso extendido?				
	6. ¿Si existe un proceso general y a partir de él se especializan otros se especifican como una generalización/especialización?				
Flujo Alterno					
crítico	1. ¿Las alternativas o excepciones se reflejan como flujos alternos?				

	2. ¿En todos los CU que se introducen datos tienen un flujo alterno donde el sistema valida la integridad de los datos que se introducen y muestra un mensaje en caso de que los datos estén incompletos?				
	3. ¿Los flujos alternativos se nombran con el número del paso que lo generó en el flujo básico, una letra, ordenados alfabéticamente que lo produjo?				
crítico	4. ¿En la sección flujos alternativos se describen todas las excepciones que existan por muy evidentes que parezcan?				
Casos de uso incluidos y extendidos					
crítico	1. ¿Al describir el caso de uso base se mencionan todos los casos de Uso que Extienden, se incluyen o se generalizan del Caso de Uso?				
crítico	2. ¿La descripción de los Casos de Uso incluidos, extendidos y especializados se realiza aparte?				
Navegabilidad					

crítico	1. ¿La navegabilidad en los caso de uso de inclusión se inicia desde el caso uso base hasta el caso de uso incluido?				
crítico	2. ¿La navegabilidad en los caso de uso de extensión se inicia desde el caso uso extendido hasta el caso de uso base?				
	3. ¿La navegabilidad en la generalización/especialización se inicia desde el caso de uso especializado a al generalizado y se representa con una relación de herencia?				
Relaciones					
crítico	1. ¿Las relaciones de inclusión y extensión entre los caso de uso se han representado con línea discontinua?				
Información General					
crítico	1. ¿El diagrama de casos de uso expresa en detalles y claramente lo que debe hacer el sistema?				
	2. Si la modelación de las interacciones con el sistema es muy extensa ¿ha empleado los paquetes de caso de uso?				

Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	5. ¿Ha identificado errores ortográficos?				
Crítico	6. ¿Se entiende claramente lo que se ha especificado en el documento?				
	7. ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?				
	8. ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?				

Lista de Chequeo Documento de Análisis. Estructura.

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	1. ¿Está el documento acorde con a la plantilla				

	estándar del proyecto o del expediente de proyecto?				
crítico	2. ¿Contiene las secciones obligatorias definidas en el expediente? (Ver Expediente de Proyecto)				
Elementos definidos por la metodología					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Diagrama de clases de Análisis.					
Crítico	1. ¿El nombre de la clase es único?				
Crítico	2. ¿Se han colocado las responsabilidades (atributos y métodos) de las clases?				
Crítico	3. ¿Se ha verificado que no exista más de una clase con la misma responsabilidad?				

Crítico	4. ¿Existen relaciones entre las clases (asociación, agregación, composición, herencia)?				
	5. ¿Se ha representado la multiplicidad?				
	6. ¿Se ha establecido la navegabilidad en la relación de asociación entre las clases?				
Crítico	7. ¿Se han detallado y especificado claramente el tipo de clase (interfaz, controladora, entidad)?				
Crítico	8. ¿De existir complejidad en el sistema que se está modelando están las clases organizadas por paquetes?				

	9. ¿Cada caso de uso tiene asociado un diagrama de clases del análisis?				
	10. ¿Se han aplicado patrones (GRASP o GoF) para el mejor entendimiento de las clases?				
Interacción entre las clases					
	1. ¿Cada diagrama de interacción es inicializado por un actor que opera directamente con el sistema?				
	2. ¿Se han identificado la sucesión de los eventos a partir de la descripción textual y detallada de los casos de uso?				

	3. ¿Comienza el nombre de los eventos con un verbo?				
	4. ¿En el caso de entrada de datos existe un evento que valide dichos datos?				
	5. En dependencia del número de escenarios de un caso de uso ¿Se ha realizado el diagrama de interacción de cada uno de estos escenarios?				
	6. ¿No existe relación directa entre una clase interfaz y una entidad?				
Crítico	7. De haber empleado el				

	diagrama de colaboración ¿todos los enlaces entre los objetos están acompañados de eventos (numeración)?				
Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	9. ¿Ha identificado errores ortográficos?				
Crítico	10. ¿Se entiende claramente lo que se ha especificado en el documento?				
	11. ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?				
	12. ¿El total de páginas que aparecen en las reglas de				

	confidencialidad coincide con el total de páginas que tiene el documento?				

Nota: En todas las Listas de Chequeos se destina un punto para registrar los defectos y las dificultades detectadas, a continuación se muestra, la información que debe contener.

3.8.1. Registro de defectos y dificultades detectados

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo Desarrollo
<Nombre del Elemento>	*< 1>*	*<Descripción de la No Conformidad>*	*<Descripción del Aspecto correspondiente>*	*<Etapa de detección del error>*	*<Si la NC es significativa se debe seleccionar que tipo de significativa es.* *Si las NC es de Aplicación se clasifican en:* • *Ortografía.* • *Funcionalidad.* • *Validación.* • *Excepciones.*	*<X>*	*<X>*	*[Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó.]*	*[Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué.]*

					• *Correspondencia con documentación.* • *Opciones que no funcionan.* *Si las NC es de Documentación se clasifican en:* • *Ortografía.* • *Redacción.* • *Correspondencia con otra documentación.* • *Formato.* • *Error técnico.* *>*			*RA: Resuelta* *PD:Pendiente* *NP:No Proceede*	

3.8.2. Evaluación del Artefacto

Se aborta la revisión del artefacto revisado si:

- ✓ *El promedio de las No Conformidades críticas por casos de uso es superior a uno.*

Este criterio es empleado únicamente para los casos de uso. En caso de otro artefacto la revisión se aborta si:

- ✓ *Existen al menos dos indicadores críticos evaluados de mal.*
- ✓ *Existe más de una falta de ortografía por página o pantalla en caso de ser una interfaz.*

- ✓ *Incumple con más del 50 % de los indicadores a evaluar de la sección Estructura del Documento que posee la lista de chequeo.*
- ✓ *Se mantienen las No Conformidades de una revisión a otra.*

***Se evalúa de regular la calidad del artefacto revisado** si el artefacto no cumple los criterios para ser abortado y:*

- ✓ *Existe una No Conformidad crítica.*
- ✓ *La cantidad de elementos afectados de un indicador evaluado de mal es superior a tres.*

Estos criterios se cumplen para todas las secciones que tiene la lista de chequeo.

***El artefacto es evaluado de bien** si no cumple ninguno de los criterios anteriores y:*

- ✓ *No existe ninguna No Conformidad relacionada con indicadores con peso crítico.*
- ✓ *Si la cantidad de elementos afectados de un indicador que no sea crítico no es mayor que dos.*

3.8.3. Evaluación: _____

3.8.4. Nombre y Apellido del Evaluador: _____