UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD # 2

Análisis y diseño de un sistema para la gestión de descarga desde la producción en la UCI para los proyectos productivos.

Trabajo de Diploma para optar por el título de

INGENIERO EN CIENCIAS INFORMÁTICAS

Autores: Yanel Machado Machado

Yoandris Pupo Rodríguez

Tutor: Ing. Reynier Pérez Mira

Ciudad de La Habana, 2008

"Año 50 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Dirección de la Infraestructura Productiva de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.
Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Autores: Yanel Machado Machado
Yoandris Pupo Rodríguez

Tutor: Reynier Pérez Mira

_____

_____ _____

Firma de Autores

Firma del Tutor

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

El Trabajo de Diploma, titulado Análisis y diseño de un sistema para la gestión de descarga desde la producción en la UCI para los proyectos productivos, fue realizado en la Universidad de las Ciencias Informáticas. Esta entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface

☐ Totalmente

☐ Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes (cuantificar):

_____

_____

_____

_____

_____

_____

_____

Como resultado de la implantación de este trabajo se reportará un efecto económico que asciende a <valor en MN o USD del efecto económico>

Y para que así conste, se firma la presente a los ____ días del mes de _____ del año _____.

_____ _____

Representante de la entidad Cargo

_____ _____

Firma Cuño

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Análisis y diseño de un sistema para la gestión de descarga desde la producción en la UCI para los proyectos productivos.

Autor: Yanel Machado Machado y Yoandris Pupo Rodríguez

El tutor del presente Trabajo de Diploma considera que durante su ejecución los estudiantes mostraron las cualidades que a continuación se detallan.

<Aquí el tutor debe expresar cualitativamente su opinión y medir (usando la escala: muy alta, alta, adecuada) entre otras las cualidades siguientes:

- Independencia
- Originalidad
- Creatividad
- Laboriosidad
- Responsabilidad>

<Además, debe evaluar la calidad científico-técnica del trabajo realizado (resultados y documento) y expresar su opinión sobre el valor de los resultados obtenidos (aplicación y beneficios) >

Por todo lo anteriormente expresado considero que los estudiantes están aptos para ejercer como Ingenieros Informáticos; y propongo que se le otorgue al Trabajo de Diploma la calificación de <nota>.

<Además, si considera que los resultados poseen valor para ser publicados, debe expresarlo también>

Reynier Pérez Mira

_____ _____

Firma Fecha

"La ciencia humana consiste más en destruir errores que en descubrir verdades."

Sócrates

Dedicatoria:

Al Comandante en Jefe Fidel Castro Ruz y a la Revolución.

A la memoria de mi padre Ganicio Machado Velázquez por apoyarme en todo momento y darme el infinito cariño de padre ejemplar.

A mi madre Diosdada Machado Machado por su amor, cariño y apoyo.

A mi hermanita querida Carmen Luisa Machado Machado por su ayuda y dedicación.

A mi cuñado Norge Noa Frómeta por su ayuda.

A mis sobrinos Davier Noa Machado y Dairiet Noa Machado.

A mi querida novia Eneysi Osorio Castro por su amor hacia mí, por su ayuda y por compartir conmigo todas las alegrías y tristezas.

A toda mi familia, amistades, vecinos y amigos.

A todos muchas gracias de corazón, los quiero mucho…

Yanel Machado Machado.

A mi querida madre Rosa M. Rodríguez Pérez por su amor, dedicación y apoyo durante todos estos años y ser el pilar de mi formación como hombre y profesional.

A mis queridos hermanos Alejandro y Alexander por su ayuda y cariño incondicional y por estar siempre unidos en los buenos y malos momentos.

A mi querida hermana Arianna por su amor y respeto.

A mi padrastro Arístides Romero González por sus consejos y ayuda.

A mi padre Bárbaro Pupo Nuez por su cariño.

A mis abuelos, mis vecinos y toda mi familia que de una forma u otra me han apoyado y ayudado todo este tiempo.

A mi otra familia que durante estos cinco años hemos compartido y vivido buenos momentos.

A mis amigos y compañeros de lucha……..

Yoandris Pupo Rodríguez.

Agradecimientos:

La lista de personas a quien debo agradecer es interminable, ruego disculpas no mencionar el nombre de muchas personas que siempre hacen posible la realidad de los sueños, mi primer agradecimiento a mi padre Ganicio Machado Velázquez aunque ya no esta físicamente no se aparta ni un instante de mi memoria, siempre me apoyo y contribuyó a mi educación desde la cuna, a mi madre por su apoyo incondicional y su amor infinito hacia mí. A mi hermana Carmen Luisa Machado Machado por su apoyo y ayuda, a mi cuñado Norge Noa Frómeta, a mis sobrinos, a mis tías y tíos a todos los primos mi agradecimiento. Agradezco a mi novia Eneysi Osorio Castro por su amor hacia mí y su comprensión. Mi agradecimiento especial a mi tutor Ing. Reynier Pérez Mira por su apoyo y dedicación.

Agradezco a mi amigo y profesor Lic. Julio Antonio Tejera Castillo por sus consejos y ayuda.

Yanel Machado Machado.

A mi madre por respetar y defender siempre mis decisiones.
A mi padre por desearme siempre lo mejor.
A mis colegas y amigos que siempre me han ayudado.
Al Ing. Yordan Nuez por ser un buen amigo y ayudarme en momentos difíciles.
A nuestra profesora y siempre compañera Yadilka por soportarnos y ayudarnos en todo lo que pudo durante estos cinco años.
A mis hermanos, por quererme siempre.
A mi primer sobrinito, Dariel, por entretenernos y ser tan cariñoso.
A mi padrastro, Arístides, por ser un buen ejemplo.
A mandy y al yayi, por su ayuda y ser buenos amigos.
A mi tutor, Ing. Reynier Pérez Mira, por preocuparse tanto por la calidad de este trabajo.

Yoandris Pupo Rodríguez

Resumen

En la actualidad la Infraestructura Productiva (IP) de la Universidad de las Ciencias Informáticas (UCI) no cuenta con un sistema para la gestión de descargas que les permita llevar el control y seguimiento de forma efectiva y ágil de los softwares que son descargados.

Por lo que este trabajo tiene como objetivo principal realizar el análisis y diseño de un sistema para la gestión de descarga en la UCI que facilite el control de los pedidos de descarga de software en la Infraestructura Productiva, para cumplir con dicho objetivo se realizó un estudio de algunas de las tecnologías, herramientas y lenguajes de programación más utilizados en la actualidad para el desarrollo de aplicaciones web.

Se hace un estudio del negocio, para poder comprender todo el proceso de gestión de planillas y documentos elaborados durante una solicitud de descarga de software, se realiza el levantamiento de requisitos funcionales y no funcionales que deberá tener el sistema utilizando técnicas de recopilación de información, así como los diagramas correspondientes al análisis y diseño del sistema.

Finalmente después de realizar el análisis costo beneficio dejamos algunas recomendaciones para la implementación del software, así como para futuras versiones que sean necesarias construir.

Palabras claves: sistema, gestión de descargas, control, seguimiento, descarga de software (productos y/o información), planillas.

Índice

Capítulo 1 Fundamentación Teórica.

Figuras:

Capítulo 2 Características del sistema.

Tablas:

Figuras:

Capítulo 3 Análisis y diseño del sistema.

Figuras:

Capitulo 4 Estudio de factibilidad.

Figuras:

Tablas:

INTRODUCCIÓN.

Nuestro país sufre el más cruel bloqueo económico y financiero impuesto por el imperialismo norteamericano desde el triunfo de la Revolución. Cuba tiene que pagar a precio de oro las distintas tecnologías que se utilizan en la rama de la informática ya que el bloqueo prohíbe la venta a la Isla de tecnología norteamericana, que como se sabe domina la industria del hardware y el software a nivel mundial.

Hoy para Cuba es un sueño conectarse a la malla mundial de fibra óptica submarina que posee ocho puntos en territorios muy próximos a la Isla, en el Caribe y que optimizaría extraordinariamente la comunicación. El sistema Arcos (Americas Region Caribbean Optical-ring System) (ver Anexo 1) conecta con fibra óptica a EE.UU., México, Centroamérica, Sudamérica y el Caribe, y brinda un servicio de ancho de banda de alta velocidad**,** pero Arcos está liderada por New World Network, accionista norteamericano mayoritario que tiene una participación del orden de 88.2%, por eso, negociar una conexión sería según el bloqueo una violación. [1]

Cuba cuenta solamente con una conexión satelital a Internet que sólo dispone de 124 megabytes/seg en bajada de satélite y de 65 megabytes/seg en subida. Este ancho de banda es impuesto por los Estados Unidos que es quien imputa lo que podemos contratar. Cada megabytes nos cuesta alrededor de cuatro veces más caro que a todo el mundo y tenemos que pelearlo con uñas y dientes.

La Universidad de las Ciencias Informáticas (UCI) institución educacional superior surgida a raíz de la Batalla de Ideas que libra el pueblo cubano actualmente se encuentra en proceso de informatización de los servicios que se prestan en la misma a estudiantes, trabajadores y proyectos productivos. Una de sus características fundamentales es que se fundamenta en el principio martiano estudio-trabajo, donde la mayoría de sus estudiantes a partir del segundo año de la carrera se incorporan a proyectos productivos, con el objetivo de producir software y al mismo tiempo esto contribuye a su formación como ingenieros informáticos. Los estudiantes, profesores y el personal que labora diariamente en la institución necesitan y utilizan diariamente gran cantidad de información para desarrollar sus proyectos y adquirir conocimientos.

La mayoría de esta información debe ser obtenida a través de Internet destacándose en estas categorías distintas aplicaciones de desarrollo, herramientas, tutoriales, videos, multimedias, libros,

manuales, etc. Actualmente la Infraestructura Productiva de la Universidad no cuenta con personal, servidores, y otros medios dedicados a garantizar un banco de información actualizada que garantice suplir las necesidades de los desarrolladores. La mayoría de los proyectos productivos y personal autorizado precisan descargar cualquier tipo de software y sus respectivas documentaciones para su posterior uso, lo que conlleva a que en reiteradas ocasiones se conste con gran flujo de información a manejar debido a que se hace el pedido de la descarga de forma tradicional mediante correos a los administradores de redes autorizados a descargar el software de Internet, así en ocasiones se descargan los mismos programas una y otra ves sin que se tenga registrado de forma adecuada cuales softwares fueron tratados con anterioridad, esto conlleva a que exista una sobrecarga de información repetida en los servidores. Debido a esta problemática en la Universidad se descargan alrededor de 150 gigabytes de información mensualmente haciéndose un uso indebido del servicio de Internet en la UCI.

Teniendo en cuenta la problemática planteada anteriormente se ha definido como **problema científico,** a tratar: ¿Cómo mejorar los procesos para la gestión de descarga desde la producción en la Universidad de las Ciencias Informáticas?

A partir de lo antes expuesto se plantea como **objeto de estudio** el proceso de análisis y diseño de un sistema para la gestión de descargas.

Por tanto, se considera como **campo de acción** la gestión de descarga desde la producción en la UCI para los proyectos productivos.

Se trazó como **objetivo general**: Realizar el análisis y diseño de un sistema para mejorar la gestión de descargas en la UCI.

Se utilizarán métodos teóricos y empíricos que aportarán gran cantidad de información para guiar por el camino correcto la investigación a realizar. Se investigarán las teorías y documentos relacionados con los sistemas gestores de descarga existentes.

Preguntas científicas

- ¿Cuáles son los principales aspectos teóricos propios de la gestión de la información en los sistemas de descargas?

- ¿Cuáles son los principales flujos de trabajo en los sistemas para la gestión de descargas?
- ¿Se podrá analizar y diseñar un sistema para la gestión de descargas en la UCI?

Para dar cumplimiento a las preguntas científicas planteadas se trazaron las siguientes **tareas de investigación:**

- Realizar un estudio de las herramientas necesarias para desarrollar el sistema, desde la fase inicial.
- Elaboración del modelo de datos.
- Realización de los diagramas correspondientes al análisis del sistema.
- Realización de los diagramas correspondientes al diseño del sistema.

En el Capítulo 1 Fundamentación Teórica se describen algunos conceptos que se van a tratar a lo largo del trabajo. Se realiza un estudio del gestor de descarga FlashGet. También se estudian algunas tecnologías, lenguajes de programación y la metodología a utilizar en el desarrollo del sistema gestor de descarga.

En el Capítulo 2 Características del sistema se describe el problema y se hace una descripción de la propuesta del sistema, se realiza el modelado del negocio. Se hace el levantamiento de requisitos funcionales y no funcionales, se crea el diagrama de paquetes de los casos de uso del sistema.

En el Capítulo 3 Análisis y diseño del sistema se trata la construcción de la solución seleccionándose el framework de desarrollo a utilizar. Se abordan algunos patrones utilizados en el diseño, se modelan los diagramas de clases del análisis y diseño.

En el Capítulo 4 Estudio de la factibilidad se analiza si es factible desarrollar el sistema utilizando la técnica de estimación de esfuerzo Análisis por Puntos de Caso de Uso.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Introducción.

En este capítulo se tratan los principales aspectos relacionados con los sistemas de descargas existentes, sus principales características y/o funcionalidades para la elaboración de este trabajo. Se realiza un estudio de algunas herramientas y programas prácticos para la elaboración de estos sistemas así como la fundamentación de las mismas.

1.1 Sistemas gestores de descarga.

Un sistema gestor de descargas es el programa encargado de manejar toda la información necesaria para realizar una descarga de Internet, ftp o de otro recurso compartido en la red. Estos sistemas reúnen un conjunto de propiedades y características fundamentales entre ellas podemos encontrar la organización de todos los objetos que van a ser descargados de acuerdo a su categoría, posible descripción de estos para facilitar el tratamiento de los mismos de forma correcta al finalizar la descarga, asignación de prioridades, deben ser programas portables capaces de trabajar con gran cantidad de información y de rápido acceso a sus bases de datos para obtener mayor respuesta de trabajo con el objetivo general de que el usuario quede satisfecho.

1.2 FlashGet.

FlashGet es un gestor de descargas reconocido a nivel mundial tanto por su eficacia, manejo de las descargas y rapidez como por su popularidad. Este programa está pensado para acelerar las descargas desde internet, ftp o cualquier otro recurso compartido en la red. Optimiza al máximo el ancho de banda que se disponga. Esto se logra abriendo varias conexiones de forma simultánea hacia los servidores de donde se ejecuta la descarga.

Incluye gran cantidad de detalles. Permite ordenar las listas de archivos descargados además de poder hacerle comentarios a los mismos [2].

FlashGet permite cosas como las siguientes:

- Detener descargas a la mitad y reanudarlas posteriormente.
- Archivar las descargas por temas.
- Adjuntar descripciones a los archivos que nos hemos bajado.
- Mantener un completo historial de descargas.
- Programar horarios para descargas.
- Gestionar las velocidades de descarga, etc.

En caso de disponer de varios sitios alternativos de descarga, FlashGet los memoriza y se beneficia de ello. Lanza varios procesos de descarga simultáneos sobre el mismo archivo, y cada uno va descargando una parte, para luego ponerlas todas en común. [3]

1.3 Servidor web Apache.

El servidor de páginas web Apache es uno de los servidores más populares en el mundo, muchas personas diariamente se suman a utilizar este servidor por la amplia documentación que existe sobre el mismo y por las características que posee, como son:

- Es gratuito de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras.
- Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor web Apache. Cualquiera que posea experiencia en la programación de C o Perl puede programar módulos para realizar determinadas funciones.
- Soporte para scripts PHP y Servlets de Java.
- Utiliza el modelo multiproceso (MPM) con distintas implementaciones lo que lo hace más flexible, transportable y escalable.
- Es seguro.

1.4 Sistemas gestores de bases de datos.

1.4.1 PostgreSQL.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto PostGres, de la Universidad de Berkeley. PostgreSQL es una versión libre (OpenSource) de este proyecto y está basado en SQL92/SQL99.

Este ORDBMS fue uno de los primeros sistemas basados en el modelo objeto-relación de la actualidad ya que incluye características de la orientación a objetos como puede ser la herencia, tipos de datos, restricciones, funciones, disparadores, reglas e integridad transaccional. A pesar de todo esto no es un sistema gestor de bases de datos del todo orientado a objetos. [4]

Principales características:

- Incluye la herencia entre tablas (aunque no entre objetos, ya que no existen).
- **DBMS Objeto-Relacional:** PostgreSQL aproxima los datos a un modelo objeto-relacional, y es 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, herencia, y arreglos.
- **Altamente Extensible:** PostgreSQL soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.
- **Soporte SQL Comprensivo:** PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.
- **Integridad Referencial:** PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.
- **API Flexible:** La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.
- **Lenguajes Procedurales:** PostgreSQL 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, o TCL como lenguaje procedural embebido.
- **MVCC:** MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios. Cuando se usa algún DBMS con capacidades SQL, tal como MySQL o Access hay ocasiones en las que una lectura tiene que esperar para acceder a información de la base de datos. La espera está provocada por usuarios que están escribiendo en la base de datos. Resumiendo, el lector está bloqueado por los escritores que están actualizando registros. Mediante el uso de MVCC, PostgreSQL evita este problema por completo. MVCC está

considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor. En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. PostgreSQL es capaz de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles.

- **Cliente/Servidor:** PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectarse a PostgreSQL.
- **Write Ahead Logging (WAL):** La característica de PostgreSQL conocida como Write Ahead Logging incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en el hipotético caso de que la base de datos se caiga, existirá un registro de las transacciones a partir del cual podremos restaurar la base de datos.

1.4.2 Manejador de bases de datos. PgAdmin.

Todas las bases de datos construidas necesitan ser administradas y que mejor para cumplir con dicho objetivo que un programa con interfaz gráfica. Para PostgreSQL existen muchas herramientas de propósito general que administran sus bases de datos entre las que se encuentra PgAdmin una de las más utilizadas por la comunidad de desarrollo del software libre.

PgAdmin diseña, mantiene y administra fácilmente las bases de datos construidas en PostgreSQL, funciona en distintos sistemas operativos como Windows 95/98, NT, Xp, así como en plataformas libres. Está liberada bajo la licencia Open Source.

Es la herramienta más popular y completa, diseñada para responder a las necesidades de los usuarios permitiéndole escribir desde simples consultas y sentencias SQL hasta diseñar complejas bases de datos. [5]

Algunas de las características de PgAdmin son:

- Entradas SQL aleatorias.

- Pantallas de información y ayudas para bases de datos, tablas, índices, secuencias, vistas, programas de arranque, funciones y lenguajes.
- Preguntas y respuestas para configurar usuarios, grupos y privilegios.
- Control de revisión con mejora de la generación de script.
- Ayudas para importar y exportar datos.
- Ayuda para migrar bases de datos.

1.4.3 MySQL.

MySQL es un sistema gestor de bases de datos relacional, licenciado bajo GPL de la GNU y fue creado por la empresa sueca MySQL AB. MySQL es uno de los ORDBMS más utilizados en el mundo del software libre debido a su gran rapidez y capacidad de manejar grandes cantidades de datos de forma eficiente gracias a su diseño multihilo.

Principales características de MySQL:

- Está escrito en C y C++.
- Probado en un amplio rango de compiladores diferentes.
- Funciona en diferentes plataformas.
- APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.
- Uso completo de multi-hilos mediante hilos del kernel. Pueden usarse fácilmente múltiples CPUs si están disponibles.
- Proporciona sistemas de almacenamientos transaccionales y no transaccionales.
- Un sistema de reserva de memoria muy rápido basado en hilos.
- Joins muy rápidos usando un multi-join de un paso optimizado.
- Tablas hash en memoria, que son usadas como tablas temporales.
- Las funciones SQL están implementadas usando una librería altamente optimizada y deben ser tan rápidas como sea posible. Normalmente no hay reserva de memoria tras toda la inicialización para consultas.

Tipos de columnas:

- Diversos tipos de columnas: enteros con/sin signo de 1, 2, 3, 4, y 8 bytes de longitud, **FLOAT**, **DOUBLE**, **CHAR**, **VARCHAR**, **TEXT**, **BLOB**, **DATE**, **TIME**, **DATETIME**, **TIMESTAMP**, **YEAR**, **SET**, **ENUM**, y tipos espaciales OpenGIS.
- Registros de longitud fija y longitud variable.

Sentencias y funciones:

- Soporte completo para operadores y funciones en las cláusulas de consultas **SELECT** y **WHERE**, por ejemplo:

```
Mysql-> SELECT CONCAT (first_name, ' ', last name)
-> FROM citizen
-> WHERE income/dependents > 10000 AND age > 30;
```

- Soporte completo para las cláusulas SQL **GROUP BY** y **ORDER BY**. Soporte de funciones de agrupación (**COUNT ()**, **COUNT (DISTINCT...)**, **AVG ()**, **STD ()**, **SUM ()**, **MAX ()**, **MIN ()**, y **GROUP_CONCAT ()**)
- Soporte para **LEFT OUTER JOIN** y **RIGHT OUTER JOIN** cumpliendo estándares de sintaxis SQL y ODBC.
- Soporte para alias en tablas y columnas como lo requiere el estándar SQL.
- **DELETE**, **INSERT**, **REPLACE**, y **UPDATE** devuelven el número de filas que han cambiado (han sido afectadas). Es posible devolver el número de filas que serían afectadas usando una bandera al conectarse con el servidor.
- El comando específico de MySQL **SHOW** puede usarse para obtener información acerca de la base de datos, el motor de base de datos, tablas e índices. El comando **EXPLAIN** puede usarse para determinar cómo el optimizador resuelve una consulta.
- Puede mezclar tablas de distintas bases de datos en la misma consulta (como en MySQL 3.22).

Seguridad:

- Un sistema de privilegios y contraseñas que es muy flexible y seguro y que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor.

Escalabilidad y límites:

- Soporte a grandes bases de datos. Se ha usado MySQL Server con bases de datos que contienen 50 millones de registros. También se conoce a usuarios que usan MySQL Server con 60.000 tablas y cerca de 5.000.000.000.000 de registros.
- Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).Un índice puede usar prefijos de una columna para los tipos de columna **CHAR**, **VARCHAR**, **BLOB**, o **TEXT**.

Conectividad:

- Los clientes pueden conectarse con el servidor MySQL usando sockets TCP/IP en cualquier plataforma. En sistemas Windows de la familia NT (NT, 2000, XP, o 2003), los clientes pueden usar named pipes para la conexión. En sistemas Unix, los clientes pueden conectarse usando ficheros socket Unix.
- En MySQL 5.0, los servidores Windows soportan conexiones con memoria compartida si se inicializan con la opción **--shared-memory**. Los clientes pueden conectarse a través de memoria compartida usando la opción **–protocol = memory**.
- La interfaz para el conector MyODBC proporciona a MySQL soporte para programas clientes que usen conexiones ODBC (Open Database Connectivity). Por ejemplo, puede usar MS Access para conectarse al servidor MySQL. Los clientes pueden ejecutarse en Windows o Unix. El código fuente de MyODBC está disponible. Todas las funciones para ODBC 2.5 están soportadas, así como muchas otras.
- La interfaz para el conector J MySQL proporciona soporte para clientes Java que usen conexiones JDBC. Estos clientes pueden ejecutarse en Windows o Unix. El código fuente para el conector J está disponible.

Localización:

- El servidor puede proporcionar mensajes de error a los clientes en muchos idiomas.
- Soporte completo para distintos conjuntos de caracteres, incluyendo **latin1** (ISO-8859-1), **german**, **big5**, **ujis**, y más. Por ejemplo, los caracteres escandinavos '**â**', '**ä**' y '**ö**' están permitidos en nombres de tablas y columnas. El soporte para Unicode está disponible.
- Todos los datos se guardan en el conjunto de caracteres elegido. Todas las comparaciones para columnas normales de cadenas de caracteres son case-insensitive.

Clientes y herramientas:

- MySQL server tiene soporte para comandos SQL para chequear, optimizar, y reparar tablas. Estos comandos están disponibles a través de la línea de comandos y el cliente mysqlcheck**.** MySQL también incluye **myisamchk**, una utilidad de línea de comandos muy rápida para efectuar estas operaciones en tablas **MyISAM**.
- Todos los programas MySQL pueden invocarse con las opciones --**help** o **-?** para obtener asistencia en línea.[6]

1.5 Metodología.

1.5.1 RUP.

El proceso racional unificado o RUP como indican sus siglas en inglés es una metodología para el desarrollo de software que define quién hace qué, cómo y cuándo debe hacerse en un proyecto. RUP define 3 características esenciales [7]:

- Está dirigido por casos de uso. Los cuales orientan el proyecto hacia las necesidades del usuario.
- Centrado en la arquitectura. Aquí se relacionan las tomas de decisiones que indican cómo tiene que ser construido el sistema y en qué orden.
- Es iterativo e incremental. Aquí se divide el proyecto total en pequeños mini proyectos donde los casos de uso y la arquitectura juegan sus objetivos de forma más depurada.

RUP define 4 fases fundamentales en las que se realizan varias iteraciones según la amplitud del proyecto a realizar y en las que tienen mayor o menor peso dentro de las actividades del proyecto:

1. Intercepción (inicio): Se crea un plan de fases, se identifican los principales casos de uso y los riesgos. Se define el alcance del proyecto.
2. Elaboración (definición, análisis, diseño): Se crea el plan del proyecto, se completan los casos de uso y se eliminan los riesgos.
3. Construcción (implementación): Se concentra en la elaboración de un producto totalmente operativo y eficiente. Se crea el manual de usuario.
4. Transición (fin del proyecto y puesta en producción): Se instala el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto surgen nuevos requisitos que se deben analizar previo a la instalación.

Principales actividades de RUP según las fases:

En la **fase de Inicio** podemos encontrar 2 actividades fundamentales ya que esta basa su existencia en el modelado del negocio y la captura de requisitos.

Modelado del negocio:

- Entender la estructura y la dinámica del cliente para el cual el sistema va a ser desarrollado.
- Entender el objetivo fundamental del problema a resolver e identificar posibles mejoras.
- Asegurar qué clientes, usuarios finales y desarrolladores tengan un entendimiento común del problema a resolver.

Requisitos: No son más que las funcionalidades que va a tener nuestro producto final de acuerdo con las especificaciones del cliente, de modo que los usuarios finales los comprendan y los acepten.

- Se establece y mantiene un acuerdo entre clientes y otros stakeholders sobre lo que el sistema podrá hacer.
- Facilitar a los desarrolladores un mejor entendimiento de los requisitos del sistema.
- Definir el ámbito del sistema.
- Generar una base para estimar costo y tiempo de desarrollo del sistema.
- Definir una interfaz de usuario para el sistema, enfocada a las necesidades y metas del usuario.

En la **fase de elaboración**, las iteraciones se orientan al desarrollo de la base para la arquitectura, abarcan más los flujos de trabajo de requerimientos, refinamiento del modelo de negocios, análisis, diseño y una parte de la implementación del código orientado a la base de la arquitectura.

Análisis y diseño:

En esta actividad se especifican los requerimientos y su descripción de cómo se van a implementar en el sistema.

- Desarrollar una arquitectura para el sistema.
- Transformar los requisitos al diseño del sistema.
- Adaptar el diseño para que sea consistente con el entorno de implementación.

Fase de Construcción:

Implementación: en esta actividad se implementan las clases y objetos en ficheros fuentes, binarios y ejecutables. El resultado final es un sistema ejecutable.

- Si se encuentran errores en el diseño, se notifican.
- Planificar los subsistemas a implementar y en el orden de integración de los mismos, formando el plan de integración.
- Cada desarrollador decide en que orden implementa los elementos del subsistema.
- Se integra el sistema siguiendo el plan de integración.

Prueba: este flujo de trabajo es el encargado de evaluar la calidad del producto que se esta desarrollando.

- Verificar las funciones del producto de software según lo diseñado.
- Encontrar y documentar defectos en la calidad del software.
- Verificar que los requisitos tengan su apropiada implementación.

Fase de transición:

Despliegue: Esta actividad tiene como objetivo la creación de versiones del software para su distribución a los usuarios finales.

- Probar el producto en su entorno final.
- Empaquetar el software para su distribución.
- Distribuir el software.
- Instalar el software.
- Proveer asistencia y ayuda a los usuarios.

En la siguiente figura se pueden observar los hitos de cada fase explicada anteriormente.

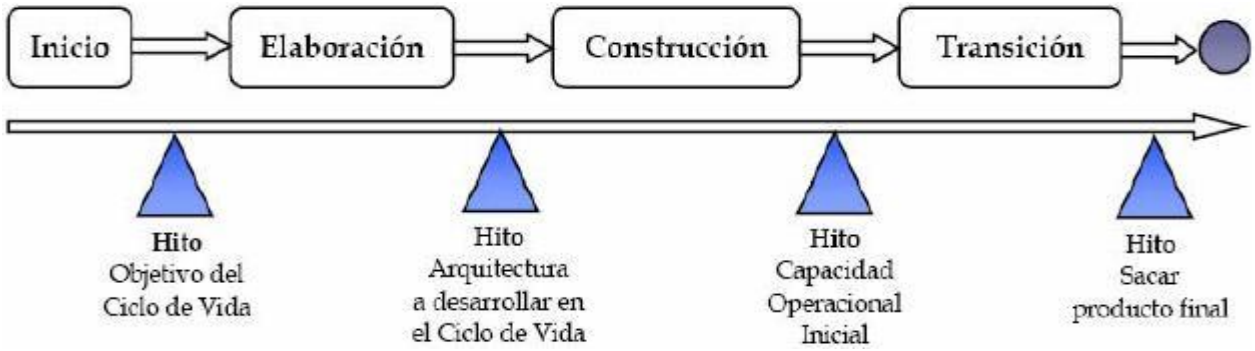

Figura 1. 1 Hitos.

Para cada una de las cuatro fases se debe trabajar en seis disciplinas o flujos de trabajo, por lo que el proceso es iterativo, y a medida que se avanza en el tiempo de desarrollo se debe lograr pasar a una etapa superior. El esfuerzo que se dedica a cada una de las disciplinas de ingeniería depende del objetivo de la fase en que se encuentre el proyecto. Los tres últimos flujos, son llamados flujos de gestión o soporte y tienen actividades en todas las fases del proyecto como se muestra en la figura 1.2.

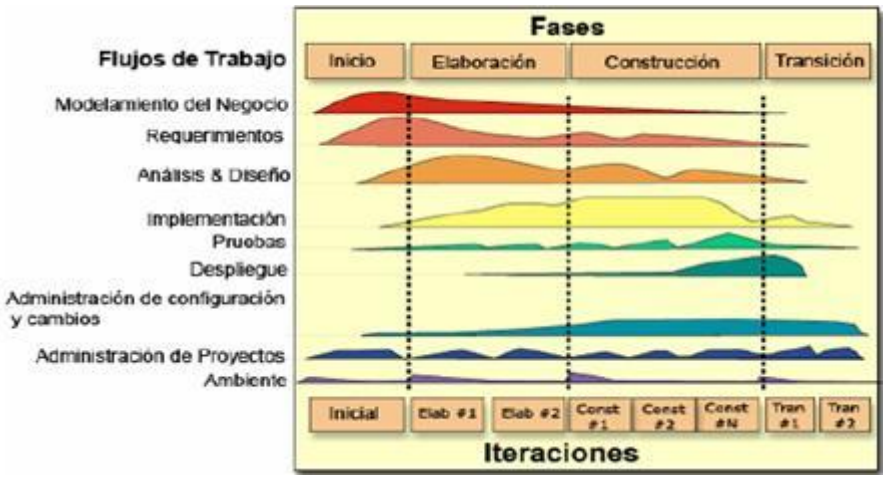

Figura 1. 2 Flujos de trabajo de RUP.

1.6 Lenguajes y herramientas para el desarrollo.

1.6.1 ¿Por qué UML como lenguaje de modelado gráfico?

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido concebido por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa. [8]

Bloques de construcción.

Existen tres tipos de bloques de construcción:

- **Elementos:** Son los modelos UML (clases, casos de uso, estados, anotaciones).
- **Relaciones:** Ligan elementos entre sí, establecen la forma en que interactúan.
- **Diagramas:** Representación gráfica de un grupo de elementos y sus relaciones.

Diagramas.

- **Los diagramas de casos de uso:** Describen las relaciones y las dependencias entre un grupo de casos de uso y los actores participantes en el proceso.

- **Los diagramas de clases:** Muestran las diferentes clases que componen un sistema y cómo se relacionan unas con otras.
- **Los diagramas de secuencia:** Muestran el intercambio de mensajes en un momento dado.
- **Los diagramas de colaboración:** Muestran las interacciones que ocurren entre los objetos que participan en una situación determinada.
- **Los diagramas de estado:** Muestran los diferentes estados de un objeto durante su vida, y los estímulos que provocan los cambios de estado en un objeto.
- **Los diagramas de actividad:** Describen la secuencia de las actividades en un sistema. Los diagramas de actividad son una forma especial de los diagramas de estado que únicamente (o mayormente) contienen actividades.
- **Los diagramas de componentes:** Muestran los componentes del software y los artefactos por los que está compuesto como son los archivos de código fuente, las librerías o las tablas de una base de datos.
- **Los diagramas de implementación:** Muestran las instancias existentes al ejecutarse así como sus relaciones. También se representan los nodos que identifican recursos físicos, típicamente un ordenador así como interfaces y objetos (instancias de las clases).

1.7 ¿Qué es CSS?

Hojas de Estilo en Cascada (CSS) (Cascading Style Sheets), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. [9]

Las ventajas que obtenemos cuando trabajamos con CSS son:

- **Separación del contenido y presentación:** Las hojas de estilo generalmente se encuentran en archivos separados del código principal. Esto permitirá que en un equipo de trabajo, programador y diseñador puedan realizar sus tareas de forma independiente aunque paralela, sin correr el riesgo de que haya interferencias entre ambos, y ello no alterará el resultado final.

- **Flexibilidad:** Utilizando las hojas de estilos se puede cambiar en cualquier momento alguna parte o la totalidad del diseño de las páginas web con sólo modificar la hoja de estilo, sin que ello suponga modificar el contenido.

- **Unificación del diseño de las páginas del sitio:** Mantener una misma apariencia de las páginas de un sitio web se puede volver una tarea pesada y tediosa si tenemos que copiar y pegar código cada vez que cree una página nueva, o que se desee modificar una misma cosa en todas sin embargo enlazando las páginas web con las hojas de estilo, se agiliza este proceso y se minimiza el trabajo.

- **Optimización de los tiempos de carga y de tráfico en el servidor:** Al dividir contenido y apariencia se obtienen archivos más ligeros, y esto nos reporta dos beneficios: Por un lado, se reducen notablemente los tiempos de carga del sitio en el navegador. A esto se le une la capacidad de éste para mantener la hoja de estilo en caché. Por otro lado, se reduce el volumen de tráfico del servidor.

- **Precisión o elasticidad:** Desde el momento en que se use CSS, el tamaño y posicionamiento de los elementos que formen las páginas web podrá ser exacto. Se le indicará al navegador donde debe colocarse dichos elementos o aquellas imágenes que se empleen así como el ancho y alto de los mismos. También se emplean medidas variables o relativas que permitan expandir el contenido hasta ocupar la totalidad de la ventana de navegación como se desee, o contraerla a sólo una parte de la misma, con independencia de la resolución de pantalla del usuario.

- **Limpieza del código fuente:** Si se escribe una hoja de estilos independiente, el código fuente de la web va a resultar menos pesado y se agilizarán las tareas de localización de las líneas que se buscan.
- **Compatibilidad y continuidad:** Las reglas establecidas por la especificación CSS-1 fijaron los estándares del diseño, y se mantienen y respetan en la CSS-2.
- **Estandarización frente a especificaciones propietarias:** La adopción de estándares por la W3C ofrece la ventaja de la compatibilidad del código entre los diferentes navegadores web. El uso del estándar CSS de la W3C evitará visualizaciones incorrectas de las páginas web en distintos navegadores.

1.8 JavaScript.

JavaScript es un lenguaje de programación interpretado. Sencillo de aprender, su código es embebido en el HTML o llamado desde otro fichero haciendo fácil la creación de páginas web con contenido dinámico. Su mayor potencial está dado porque permite realizar validaciones de datos o elementos del lado del cliente reduciendo de esta manera la carga del servidor y las transacciones que se efectúan a través de http. Actualmente es soportado por la mayoría de los navegadores.

1.9 ¿Por qué PHP?

PHP es un lenguaje de secuencia de comandos del lado del servidor diseñado para la web. Las siglas PHP equivalían inicialmente a Personal Home Page (Página de Inicio Personal), pero su equivalente actual es Hipertext Preprocessor (Procesador de Hipertexto). Se puede incrustar el código PHP en páginas web, estas se verán en el lado cliente una vez interpretadas por el servidor como una página HTML. En el año 2007 más de 40 millones de sitios han empezado a utilizar PHP. [10][11]

Entre las ventajas que nos ofrece PHP podemos citar:

- Muy sencillo de aprender.
- Soporta la programación orientada a objeto (POO), clases y herencia.
- Es un lenguaje multiplataforma (Soportado aproximadamente en 25 plataformas).
- Biblioteca nativa de funciones.
- Tratamiento de errores.
- Puede interactuar con muchos motores de bases de datos tales como MySQL, Access, Oracle, PostgreSQL, y muchos otros.
- Existe mucha y variada documentación entre las que se pueden encontrar (libros, tutoriales, videos tutoriales, etc.).

1.10 Visual Paradigm.

Es una herramienta CASE que da soporte al modelado visual de los sistemas software, permitiendo representar los sistemas desde diferentes perspectivas y resaltando las características más importantes de estos.

Visual Paradigm nos ofrece:

- Entorno de creación de diagramas para UML.
- 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 múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs.
- Disponibilidad en múltiples plataformas.

1.11 Dreamweaver.

Es una herramienta de desarrollo profesional de sitios web muy avanzada. Tiene características muy novedosas con amplías opciones de diseño que permiten ir desarrollando y visualizando el resultado. Se puede trabajar codificando o diseñando visualmente o en ambas opciones a la misma vez. El desarrollador puede utilizar CSS, la cual mejorará la apariencia del contenido. Es fácil de aprender.

Abarca actualmente más del 80% del mercado de edición de páginas web. Soporta las principales tecnologías de servidor como por ejemplo PHP, JSP, ASP.net entre otras.

1.12 Zend Studio.

El Zend Studio es uno de los mejores IDEs del momento. Son muchos los desarrolladores que trabajan con Zend Studio ya que es un excelente editor de texto para páginas web en PHP, permite la depuración de código.

Zend Studio consta de dos partes en las que se dividen las funcionalidades del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene la interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración. [12]

La interfaz está compuesta por varias partes, en las que se encuentra un explorador de archivos, una ventana de depuración, los menús y otra para mostrar el código de las páginas.

Contiene una ayuda con todas las librerías de funciones del lenguaje, ofrece los nombres de las funciones y parámetros que deben recibir. Esta ayuda no solo se queda en las funciones definidas en el lenguaje, sino que también reporta ayudas de las funciones que se creen.

Zend Studio facilita encontrar errores y corregirlos ya que dispone de una herramienta de debug o depuración. Esta permite ejecutar páginas y conocer en todo momento el contenido de las variables de la aplicación y las variables del entorno como las cookies, las recibidas por formulario o en la sesión. Se pueden colocar puntos de parada de los scripts y realizar las acciones típicas de depuración.

1.13 Propuesta.

Con lo explicado y analizado en este capítulo se ha decidido que es necesario realizar el análisis y diseño de una aplicación web que permita gestionar el proceso de descarga de software e información desde la producción en la en la UCI para los proyectos productivos desde Internet, la aplicación debe brindar algunas estadísticas relacionadas con la cantidad de información que se maneja en la misma. Debe ser una aplicación con una interfaz amigable y que responda a las necesidades de los clientes.

Para lograr este objetivo se propone como metodología a emplear RUP, Zend Studio, Visual Paradigm, Dreamweaver, PostgreSQL, PgAdmin como herramientas. UML y PHP para el modelado e implementación respectivamente, también se empleará como material de apoyo CSS para los estilos y JavaScript para las validaciones de datos del lado del cliente. Además de Apache como servidor web.

Conclusiones.

En el presente capítulo se ha definido el concepto de sistema gestor de descargas, se han estudiado las principales herramientas, metodologías y lenguajes a utilizar para la construcción del sistema así como las justificaciones de sus elecciones, además se ha elaborado la propuesta del sistema a desarrollar.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA.

2.1 Introducción.

En el presente capítulo se utiliza como metodología para el desarrollo y modelado del negocio RUP. Se identifican los actores del negocio, los casos de uso del negocio y los trabajadores del negocio, brindando una visión general del sistema a desarrollar, además se brindan los actores del sistema, los casos de uso del sistema, se capturan los requisitos funcionales y no funcionales que debe cumplir el sistema a desarrollar.

2.2 Modelo del negocio o modelo del dominio.

En RUP existen al menos dos aproximaciones para expresar el contexto de un sistema en una forma reutilizable para desarrolladores de software: **modelo del dominio y modelo del negocio**.

Un modelo del dominio captura los tipos más importantes de objetos en el contexto del sistema, y enlaza estos objetos unos con otros. La identificación y asignación de un nombre para estos objetos nos ayuda a desarrollar un glosario de términos que permitirá comunicarse mejor a todos los que están trabajando en el sistema, también nos ayudará a identificar algunas de las clases a medida que analicemos y diseñemos nuestro sistema.

Un modelo del negocio es una técnica para comprender los procesos de negocio de la organización, describiendo los procesos existentes u observados con el objetivo de comprenderlos. **El modelo del negocio** especifica los procesos del negocio que soportará el sistema.

Después de un estudio del caso que nos ocupa decidimos desarrollar **un modelo del negocio** ya que coincidimos en que:

Los procesos están bien definidos, las fronteras están bien establecidas, se identifican las personas que empiezan, se benefician y quienes desarrollan las actividades en cada uno de los procesos del negocio [13].

2.3 Estado actual del negocio.

Actualmente en la UCI no se utiliza ningún sistema para la gestión de descargas, por lo que todo el proceso se realiza de forma manual y con ayuda del correo electrónico por lo que es un proceso lento, tardío y no ofrece estadísticas que nos permitan realizar un análisis de los softwares que se descargan ni de la cantidad de mb invertidos por concepto de descarga desde Internet. El flujo empieza cuando algún asesor de arquitectura y tecnologías (AAT) de una facultad o especialistas de la dirección de la IP (EDIP) llena una planilla de solicitud de descarga y la envía por correo electrónico al especialista del grupo de soporte al desarrollo (EGSD), este la revisa, sí la aprueba entonces realiza una búsqueda manual a ver si lo solicitado se había descargado con anterioridad, de ser positivo se les informa a través de un correo a los implicados en el proceso donde puede encontrar lo solicitado, si en la búsqueda no se encuentra entonces envía esta planilla a la Dirección de Servicios Legales(DSL) para la aprobación de la licencia del producto, si éste la aprueba notifica al EGSD. Cuando el EGSD comprueba que el producto fue aprobado por el DSL se conecta a Internet y utilizando un gestor de descarga realiza la misma y cuando finaliza la descarga informa al asesor o al especialista donde puede encontrar lo solicitado a través de un correo electrónico. Si el EGSD no aprueba la solicitud informa por correo electrónico al AAT o al EDIP de que la solicitud fue denegada y el por qué se les negó.

2.4 Reglas del Negocio.

- Las solicitudes serán revisadas por la Dirección de Servicios Legales de la IP con el fin de validar el/los software/s a usar en la producción. No serán revisados aquellos que generen archivos de tipo audio, video, librerías entre otros.
- Cada 15 días en los Consejos Técnicos se debe dar un parte al Director de la Dirección Técnica y a los Asesores de Arquitectura y Tecnologías de la planilla generada a nivel UCI en la quincena así como del estado de aprobación y nivel de respuesta o estado de las solicitudes.

2.5 Actores del negocio.

Según la metodología RUP seleccionada un actor del negocio es cualquier individuo, grupo, organización, entidad, máquina o sistema de información externo, con los que el negocio interactúa. Lo

que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados.

- **Actores del negocio. Justificación.**

Actores del negocio	Justificación
Asesor de arquitectura y tecnología.	El Asesor de arquitectura y tecnología es quien inicia el proceso de solitud de descarga de software para los proyectos productivos de su facultad en la Universidad de las Ciencias Informáticas, proceso que se realiza por el grupo de especialistas y soporte al desarrollo de la propia Universidad.
Especialista de la Dirección de la IP (EDIP).	El EDIP es quien solicita la descarga de información para la dirección de la infraestructura productiva de la Universidad de las Ciencias Informáticas, proceso que se realiza por el grupo de especialistas y soporte al desarrollo de la propia Universidad.

Tabla 2. 1 Actores del negocio.

2.6 Trabajadores del negocio.

Trabajadores del negocio. Justificación.

Trabajadores del negocio	Justificación
Asesor de arquitectura y tecnología.	El asesor es el encargado de enviar las solicitudes de descargas de los proyectos productivos al EGSD.
Especialista del grupo de soporte y apoyo al desarrollo.	El Especialista es el responsable de aprobar una descarga, enviar las solicitudes a la Dirección de Servicios Legales para su aprobación, realiza la descarga con algún gestor de descarga

	e informa al AAT o al EDIP.
Especialista de la dirección de la IP.	Es EDIP es el encargado de crear la planilla de solicitud de descarga de software de la infraestructura productiva.
Dirección de Servicios Legales.	El DSL es el encargado de aprobar y chequear todas las licencias de los pedidos de descarga de software que se hacen.

Tabla 2. 2 Trabajadores del negocio.

2.7 Diagrama de caso de uso del negocio.

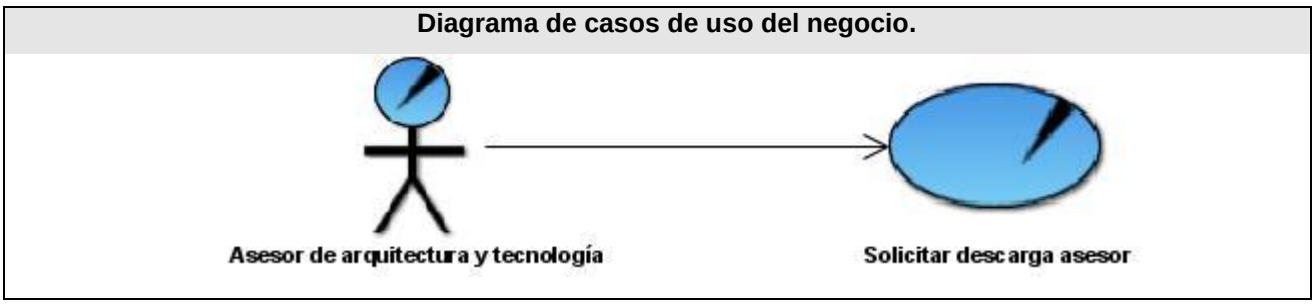

Figura 2. 1 Solicitar descarga AAT

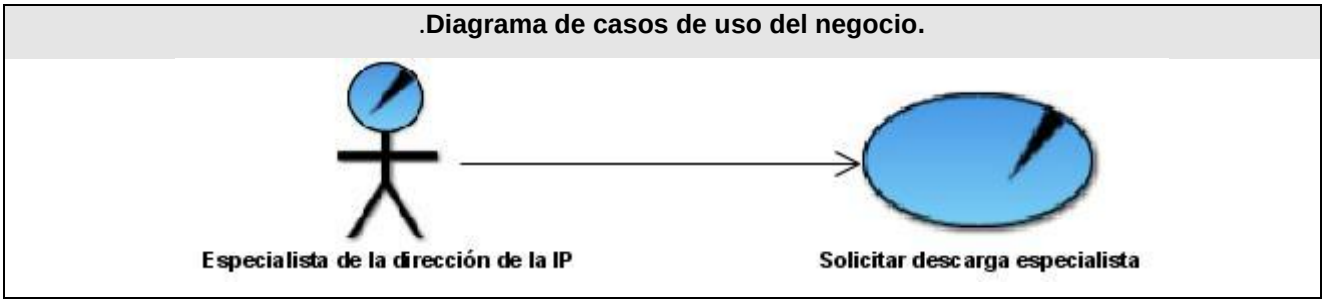

Figura 2. 2 Solicitar descarga EDIP.

2.8 Descripción de los casos de uso del negocio.

En el Anexo 2 se muestran las descripciones textuales de los casos de usos del negocio descritos en el diagrama.

2.9 Diagrama de Actividades.

Un diagrama de actividad describe un proceso que explora el orden de las tareas o actividades que logran los objetivos del negocio. Es similar a un diagrama de estados en el cual todos o la mayoría de los estados son estados de actividad y en la cual todas o la mayoría de las transiciones se disparan al completarse las acciones en los estados fuentes precedentes.

En el Anexo 3 se muestran los diagramas de actividades.

2.10 Diagrama de clases del modelo de objetos.

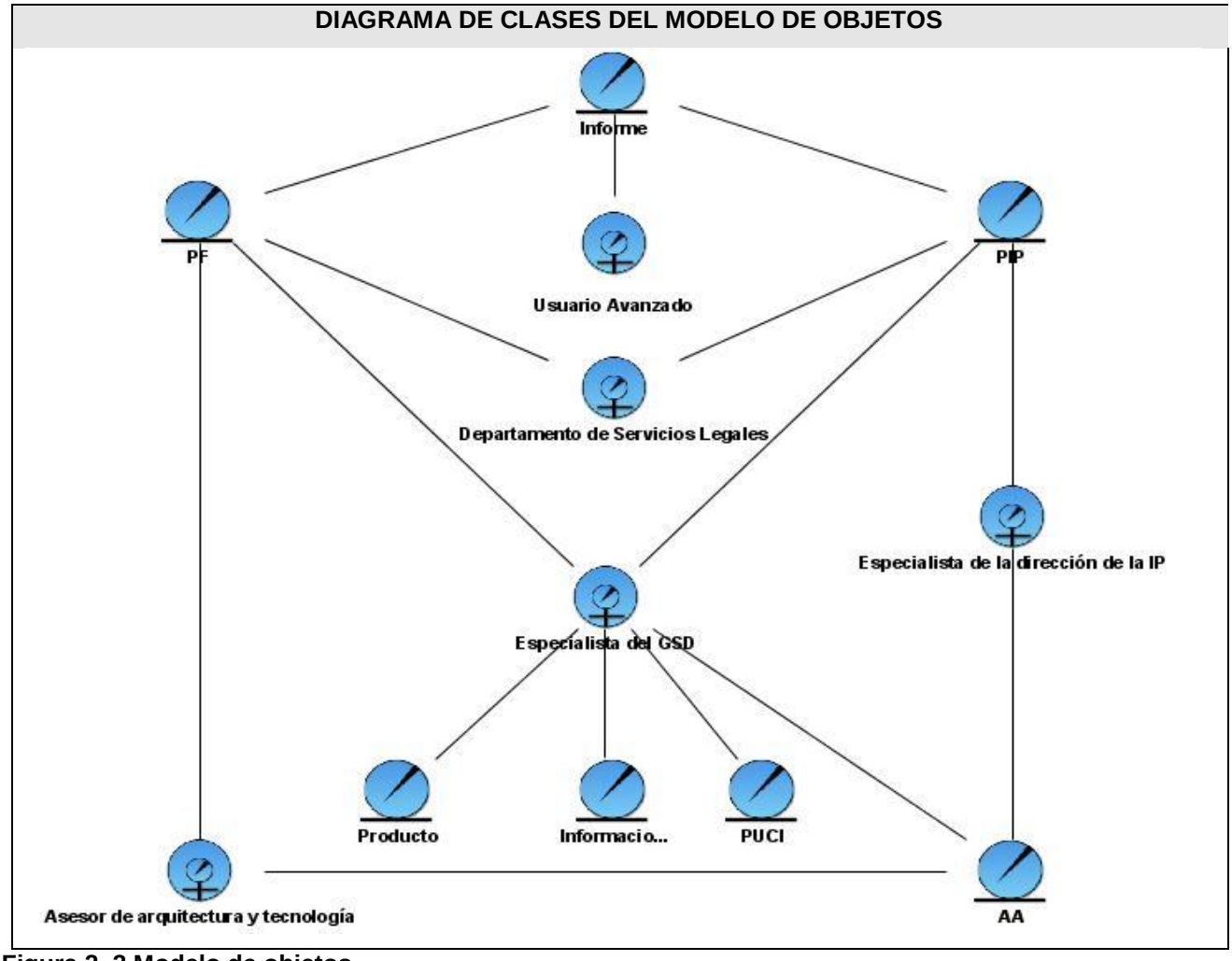

Figura 2. 3 Modelo de objetos.

2.11 Requerimientos.

Los desarrolladores de software tras el cursar de los años han elaborado y probado muchos estándares, metodologías de desarrollo y herramientas que sin dudas son muy útiles y ofrecen ventajas para elaborar softwares. Sin embargo hoy en día muchos proyectos fracasan o no cumplen con las exigencias de sus usuarios, entre otros factores, esto es debido a que en ocasiones se realiza una mala captura de requisitos por parte de los desarrolladores.

Por lo planteado anteriormente es que la captura de requisitos es uno de los flujos de trabajo que tiene más importancia en el proceso de desarrollo y se le debe prestar mucha atención ya que:

Los usuarios no saben cómo puede hacerse más eficiente la operación en su conjunto. La mayoría de los usuarios no saben qué parte de su trabajo puede transformarse en software. Francamente, con frecuencia los usuarios no saben cuáles son los requisitos ni tampoco cómo especificarlos de una forma precisa. [14]

2.12 Definición de requisitos y clasificación.

El término requisito de software puede definirse como una condición que el sistema debe cumplir o capacidad que debe tener.

Los requisitos pueden clasificarse en dos grandes categorías que son:

- Requisitos funcionales.
- Requisitos no funcionales.

Los requisitos funcionales son los que especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable, por ejemplo, pudiera desearse que el sistema responda dentro de un intervalo de tiempo especificado o que obtenga los resultados de los cálculos con un nivel de precisión dado.

2.13 Técnicas de obtención de Información.

Los analistas y desarrolladores utilizan variados métodos o técnicas para recopilar información o capturar datos de una determinada situación. Esta última la mayoría de las veces determina que técnica emplear aunque es muy recomendado utilizar varias de ellas para obtener claridad y veracidad en los datos.

Las técnicas más mencionadas y utilizadas son:

- Entrevista.
- Encuesta.
- Observación.
- Muestreo.
- Introspección.

Para lograr veracidad, claridad y eficiencia en la captura de los requisitos se utilizan las técnicas de la entrevista y la introspección. [15] [16] [17]

¿Qué es la entrevista?

- La entrevista: es la práctica que permite al investigador obtener información de primera mano.
- La entrevista: se utiliza para recabar información en forma verbal, a través de preguntas que propone el analista.
- La entrevista: Es una técnica para obtener datos o información que consisten en un diálogo entre dos o más personas: El entrevistador "investigador" y el entrevistado.

¿Que es la introspección?
Esta técnica recomienda que el ingeniero de requisitos se ponga en el lugar del cliente y trate de imaginar como desearía él el sistema. Y en base a estas suposiciones comenzar a recomendar al cliente sobre la funcionalidad que debería presentar el sistema.

2.14 Requerimientos funcionales y no funcionales.

Con la utilización de las técnicas expuestas anteriormente se obtuvieron los siguientes requisitos:

2.14.1 Requisitos funcionales.

R1 – Autenticar usuarios.

R1.1 – Verificar autenticidad de usuarios y contraseñas.

R1.2 – Verificar nivel de acceso de los usuarios.

R2 – Crear Planilla de descarga de software de la facultad (PF).

R2.1 – Datos generales de la facultad.

R2.1.1 – Facultad.

R2.1.2 – Número de la solicitud.

R2.1.3 – Fecha de la solicitud.

R2.1.4 – Nombre del asesor de arquitectura y tecnología.

R2.2 – Datos generales de la solicitud.

R2.2.1 – Id de la solicitud.

R2.2.2 – Nombre del software.

R2.2.3 – Código del proyecto.

R2.3 – Nomenclatura del proyecto productivo.

R2.3.1 – Código del proyecto productivo.

R2.3.2 – Nombre del proyecto productivo.

R2.4 – Datos generales del software a descargar.

R2.4.1 – Id del software.

R2.4.2 – Descripción.

R2.4.3 – Fabricante.

R2.4.4 – Versión del software.

R2.4.5 – URL.

R2.4.6 – Plataforma.

R2.4.6.1 – Existencia en versión para Windows.

R2.4.6.2 – Existencia en versión para Linux.

R2.4.7 – Versión de la licencia.

R2.4.7.1 – Versión comunitaria.

R2.4.7.2 – Versión comercial.

R2.4.8 – Producto.

R2.4.8.1 – Nacional.

R2.4.8.2 – Exportación.

R2.4.9 – Tamaño del producto a descargar.

R3 – Modificar Planilla de descarga de software de la facultad.

R4 – Eliminar Planilla de descarga de software de la facultad.

R5 – Revisar Planilla de descarga de software de la facultad.

R6 – Crear Planilla de descarga de software de la dirección de la infraestructura productiva (PIP).

R6.1 – Datos generales de la dirección.

R6.1.1 – Dirección.

R6.1.2 – Número de la solicitud.

R6.1.3 – Fecha de la solicitud.

R6.1.4 – Nombre del especialista.

R6.1.5 – Nombre del director de la dirección.

R6.2 – Datos generales de la solicitud.

R6.2.1 – Id de la solicitud.

R6.2.2 – Nombre del software.

R6.3 – Datos generales del software a descargar.

R6.3.1 – Id del software.

R6.3.2 – Descripción.

R6.3.3 – Fabricante.

R6.3.4 – Versión del software.

R6.3.5 – URL.

R6.3.6 – Plataforma.

R6.3.6.1 – Existencia en versión para Windows.

R6.3.6.2 – Existencia en versión para Linux.

R6.3.7 – Versión de la licencia.

R6.3.7.1 – Versión comunitaria.

R6.3.7.2 – Versión comercial.

R6.3.8 – Producto.

R6.3.8.1 – Nacional.

R6.3.8.2 – Exportación.

R6.3.9 – Tamaño del producto a descargar.

R7 – Modificar Planilla de descarga de software de la dirección de la infraestructura productiva.

R8 – Eliminar Planilla de descarga de software de la dirección de la infraestructura productiva.

R9 – Revisar Planilla de descarga de software de la dirección de la infraestructura productiva.

R10 – Crear Planilla de descarga de software UCI.

R10.1 – Datos generales de la dirección técnica.

R10.1.1 – Número de la solicitud.

R10.1.2 – Fecha de cierre.

R10.1.3 – Grupo de soporte al desarrollo.

R10.1.4 – Director de la dirección técnica IP.

R10.2 – Datos generales de descarga de software UCI - Facultades.

R10.2.1 – Id del software.

R10.2.2 – Nombre del software.

R10.2.3 – Tamaño del software.

R10.2.4 – Facultad.

R10.3 – Datos generales de descarga de software UCI –Direcciones IP.

R10.3.1 – Id del software.

R10.3.2 – Nombre del software.

R10.3.3 – Tamaño del software.

R10.3.4 – Facultad.

R11 – Eliminar Planilla de descarga de software UCI.

R12 – Modificar Planilla de descarga de software UCI.

R13 – Realizar Búsqueda de información de productos.

R14 – Descargar Información.

R15 – Comprobar descarga.

R16 – Crear informe.

R17 – Modificar Informe.

R18 – Eliminar Informe.

R19 – Elaborar Acta de aceptación (AA).

R19.1 – Fecha de la conciliación.

R19.2 – Nombre del asesor de arquitectura y tecnología.

R19.3 – Nombre del especialista grupo de soporte al desarrollo.

R19.4 – Observaciones del proceso.

R19.5 – Firmas.

R19.5.1 – Nombre Asesor de Arquitecturas y Tecnologías.

R19.5.2 – Especialista Grupo de Soporte al Desarrollo DT.

R19.5.3 – Director Dirección Técnica IP.

R19.6 – Enviar AA a los especialistas.

R20 – Chequear Licencias.

R21 – Actualizar estado del software.

R22 – Cargar Planillas.

2.14.2 Requisitos no funcionales.

- **Requisito de apariencia o interfaz externa.**
  - El diseño debe ser sencillo.
  - Los colores empleados para el desarrollo de nuestra aplicación deben tener tonos claros preferiblemente azules y verdes para que se logre un producto agradable a la vista del usuario.
  - El sistema debe tener una interfaz amigable y fácil de usar.
- **Requisito de usabilidad.**
  - El sistema podrá ser usado por personas con conocimientos mínimos en la rama de la informática.
- **Requisito de rendimiento.**
  - El tiempo de carga del sitio debe ser el más optimo posible. Para ello se usarán imágenes justas y optimizadas con formato .gif o .jpg.
  - Debe soportar tantos usuarios como permita el gestor de bases de datos PostgreSQL.
- **Requisito de soporte.**
  - El sistema debe tener un manual de ayuda en línea que se podrá consultar desde cualquier página donde el usuario se encuentre navegando.
- **Requisito de portabilidad.**
  - El sistema debe ser multiplataforma.

- El sistema debe ser de fácil instalación.

➢ **Requisito de seguridad y privacidad.**

- El sistema debe garantizar la confidencialidad de la información, la información debe solo ser vista por personas autorizadas.
- El sistema debe garantizar la integridad de la información, el contenido debe ser solo alterable por personal autorizado.
- El sistema debe estar disponible a todos los autorizados, todo el tiempo.
- El sistema debe ofrecer confirmación a tareas que alteren información como por ejemplo (Eliminar, modificar, etc.).

➢ **Requisitos políticos y culturales.**

- En este sistema se usará y utilizará información que este acorde con los principios revolucionarios y los valores éticos y morales de nuestra sociedad.

➢ **Requisitos legales.**

- El uso y empleo del sistema estará regido por las normas y reglamentaciones establecidas en la UCI.

➢ **Requisito de hardware.**

Para el desarrollo del sistema:

- Pentium de 133 MHz o superior.
- 128 MB de memoria RAM mínima.

Para la explotación del sistema:

Servidor:

- Pentium (4) CPU 3.00GHz, 3.00GHz.
- 504 MB de RAM.
- 154 GB de disco duro.

Cliente:

- Pentium de 133 MHz o superior.
- 128 MB de memoria RAM mínima, 256 MB de memoria RAM recomendada.

➢ **Requisito de software.**

Para el desarrollo del sistema:

- Sistema operativo Linux o Windows.
- Un navegador web (Internet Explorer o Mozilla Firefox).
- Visual Paradigm.
- Zend Studio.
- Dreamweaver.

Para la explotación del sistema:

- Sistema operativo Linux o Windows.
- Un navegador web (Internet Explorer o Mozilla Firefox).
- Servidor de Base de datos PostgreSQL.

2.15 Definición de los casos de uso del sistema.

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema.

2.15.1 Definición de los actores del sistema.

Los actores del sistema pueden representar un rol que juega una o varias personas, un equipo o un sistema automatizado, pueden ser un recipiente pasivo de información e intercambiar información con él pero no son parte del sistema.

Actores	Justificación
Usuario Avanzado	Es un Usuario responsable de crear el informe de estadísticas y visualizar el informe.
Usuario	Es toda aquella persona que va a tener acceso al sistema.
Asesor de arquitectura y tecnología	Es un Usuario que crea la PF y la envía al EGSD.
Especialista del grupo de soporte y apoyo al	Es un Usuario del Sistema responsable de verificar las solicitudes, crea la PUCI, realiza búsqueda de información de los productos,

desarrollo	manda a ejecutar descarga de información e informa sobre la descarga a los Asesores de Tecnología, UCISTORE y a DATALAB.
Gestor de descarga	Tercero que interactúa con el sistema cuya funcionalidad es ejecutar la descarga de Internet.
Activista de actas	Es una especialización de los asesores de arquitectura y tecnología y de los especialistas de la dirección de la IP cuya función es elaborar un acta de aceptación sobre el servicio que se brinda.
Dirección de Servicios Legales (DSL).	Es el grupo encargado de revisar las planillas del pedido de las descargas a nivel central.

Tabla 2. 3 Descripción de los actores del sistema.

➢ **Representación de los actores del sistema.**

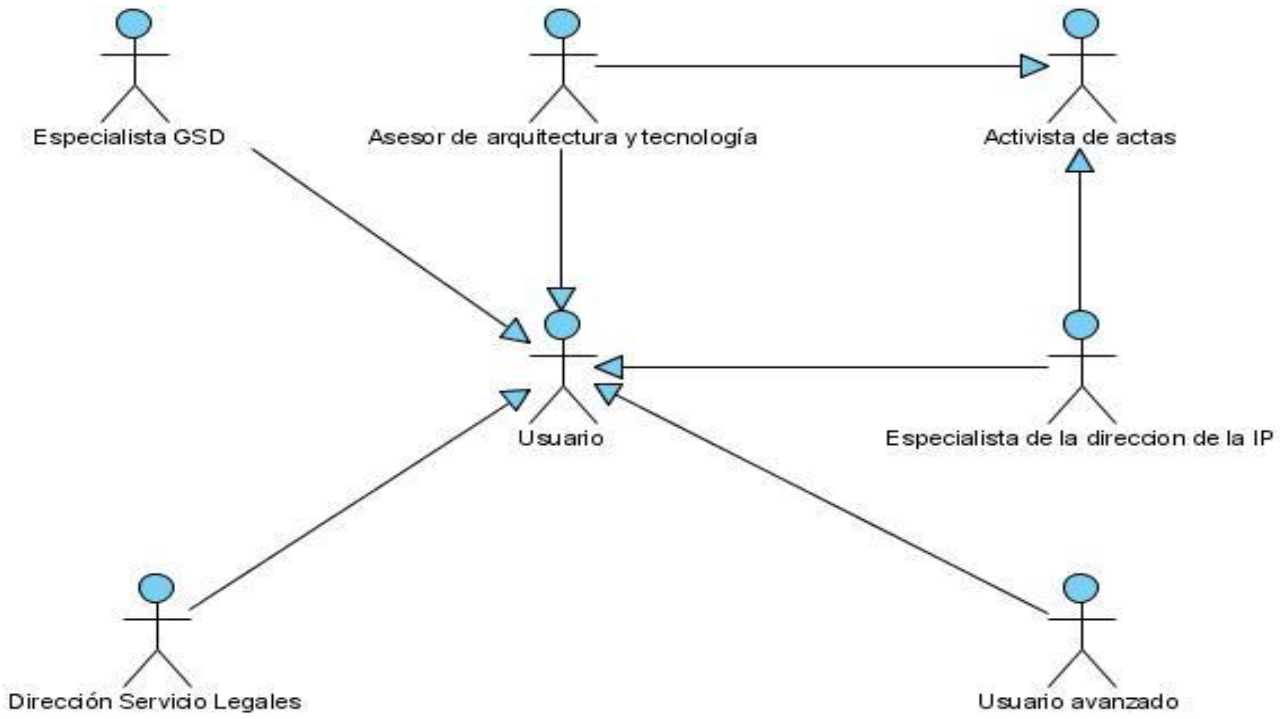

Figura 2. 4 Actores del sistema.

2.15.2 Listado de casos de uso.

CU – 1	Autenticar Usuario.
Actor	Usuario.

Descripción	Cuando un Usuario decide autentificarse, escribe su nombre, contraseña, pulsa el botón Login, si los datos entrados son correctos el sistema mostrará un mensaje de bienvenida sino un mensaje de error.
Referencia	R1.

CU – 2	Gestionar PF
Actor	AAT.
Descripción	Permite que el Asesor de Arquitectura y Tecnología cree, modifique y elimine la planilla de solicitud de descarga de la facultad.
Referencia	R2, R3, R4.

CU – 3	Verificar Planillas.
Actor	EGSD.
Descripción	Permite al EGSD revisar las solicitudes de descarga de software, visualizándole las mismas y dándole la opción de aceptar o denegar cualquiera de ellas.
Referencia	R5, R9.

CU – 4	Gestionar PIP.
Actor	EDIP.
Descripción	Permite que se elabore, modifique o se elimine una planilla de solicitud de descarga de las direcciones de la infraestructura productiva.
Referencia	R6, R7, R8.

CU – 5	Gestionar PUCI.
Actor	EGSD.
Descripción	Permite crear, modificar o eliminar una planilla correspondiente a

	todas las descargas que se han hecho hasta el momento en la Universidad.
Referencia	R10, R11, R12.

CU – 6	Realizar Búsqueda de Información de los productos.
Actor	EGSD.
Descripción	Permite al EGSD realizar una búsqueda en la base de datos para verificar si el software que se solicita fue bajado o no con anterioridad.
Referencia	R13.

CU – 7	Descargar Información.
Actor	EGSD.
Descripción	Este caso de uso es donde se procede a ejecutar la descarga directamente de Internet utilizando un gestor de descarga.
Referencia	R14.

CU – 8	Comprobar descarga.
Actor	EGSD.
Descripción	Permite al EGSD chequear si el software se descargo o no.
Referencia	R15.

CU – 9	Gestionar Informe.
Actor	Usuario Avanzado.
Descripción	Permite al Usuario Avanzado crear, modificar y eliminar un informe creado con los datos seleccionados por él, el cual será generado por el sistema.
Referencia	R16, R17, R18.

CU – 10	Elaborar AA.
Actor	Activista de actas.
Descripción	Permite al activista de actas elaborar un acta donde se refleje si está conforme con el servicio prestado o no. La cual enviará al especialista para que este proceda a copiar el software en el repositorio o sea eliminado de no existir conformidad con el producto descargado.
Referencia	R19.

CU – 11	Chequear Licencias.
Actor	DSL.
Descripción	Permite al DSL chequear en las planillas el software que se va a descargar.
Referencia	R20.

CU – 12	Actualizar Estado del Software.
Actor	EGSD.
Descripción	Permite al EGSD agregar información del software que se descargó al sistema o eliminar el software del sistema.
Referencia	R21.

CU – 13	Cargar Planillas.
Actor	Usuario.
Descripción	Permite a los usuarios cargar las planillas de solicitudes que se le hayan hecho.
Referencia	R22.

2.15.3 Diagrama de casos de uso.

Para lograr una mayor claridad a la hora de representar el sistema mediante casos de uso y facilitar la comprensión de los mismos se crearon 7 paquetes.

A continuación se representan las estructuras y relaciones de los paquetes.

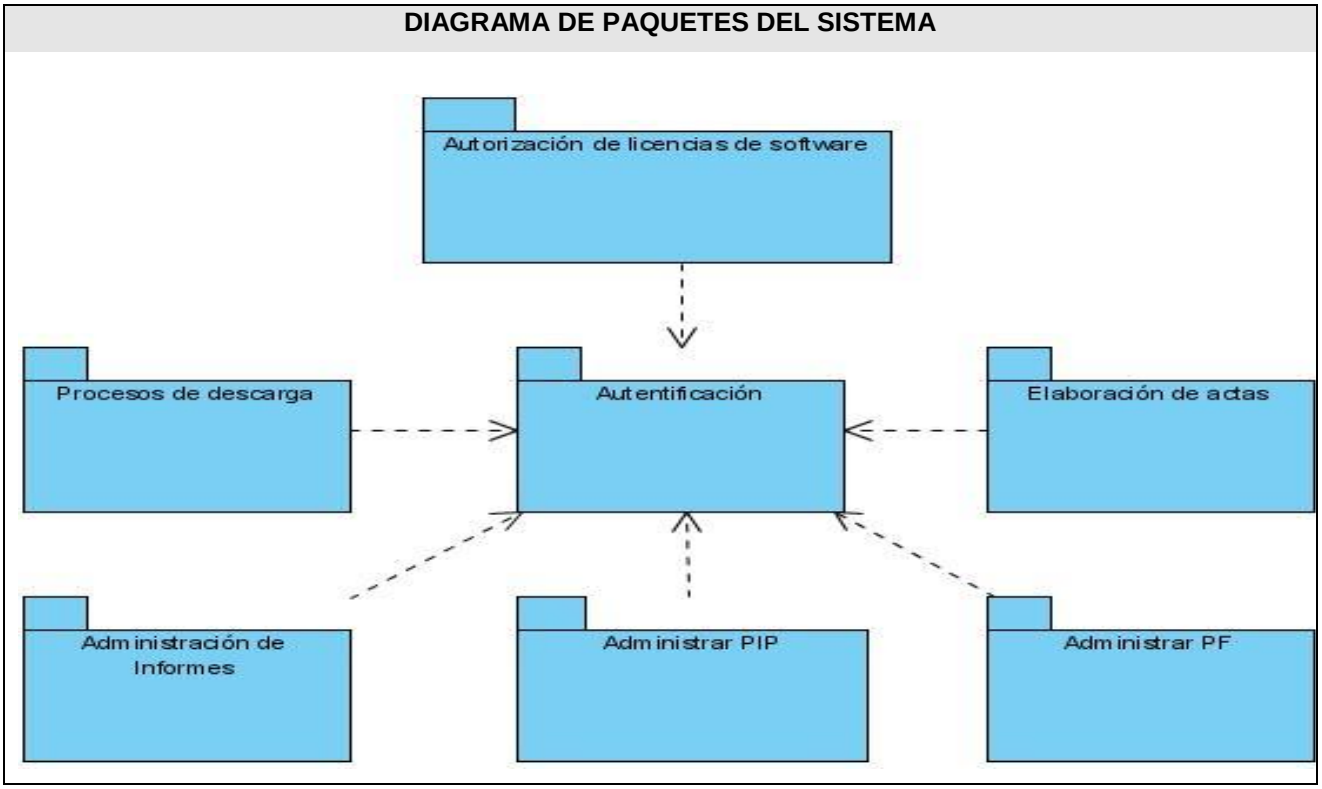

Figura 2. 5 Diagrama de paquetes del sistema.

A continuación se muestra la representación grafica de los casos de uso correspondiente a cada paquete.

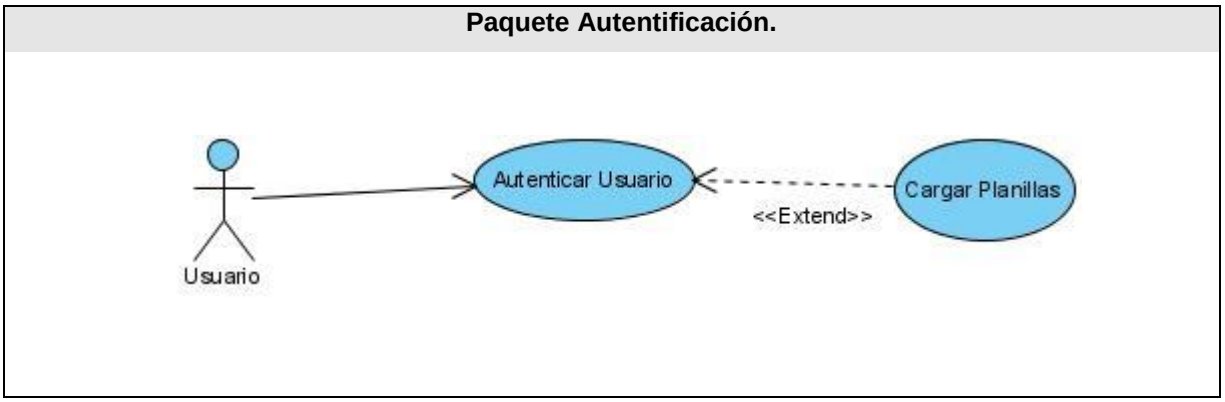

Figura 2. 6 Diagrama de CU <Paquete Autentificación>.

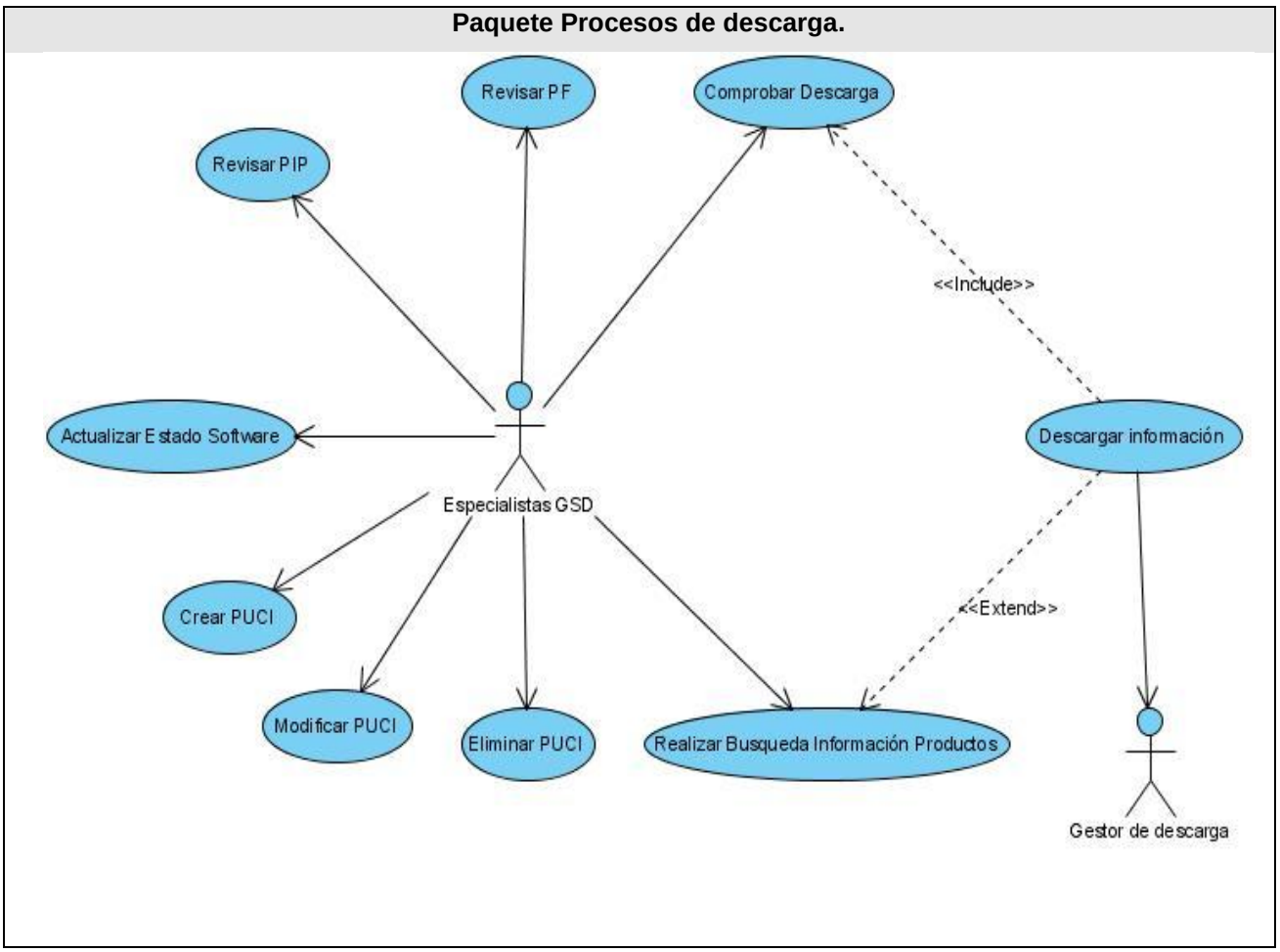

Figura 2. 7 Diagrama de CU <Paquete Procesos de descarga>.

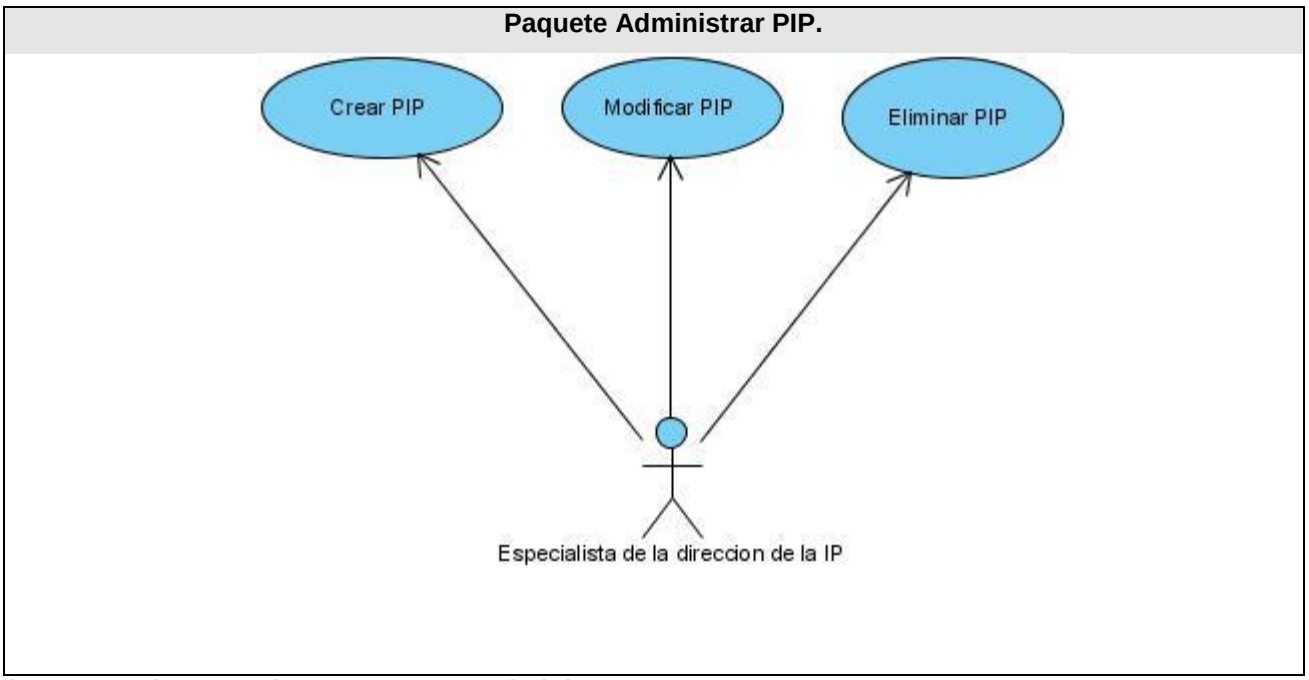

Figura 2. 8 Diagrama de CU < Paquete Administrar PIP >.

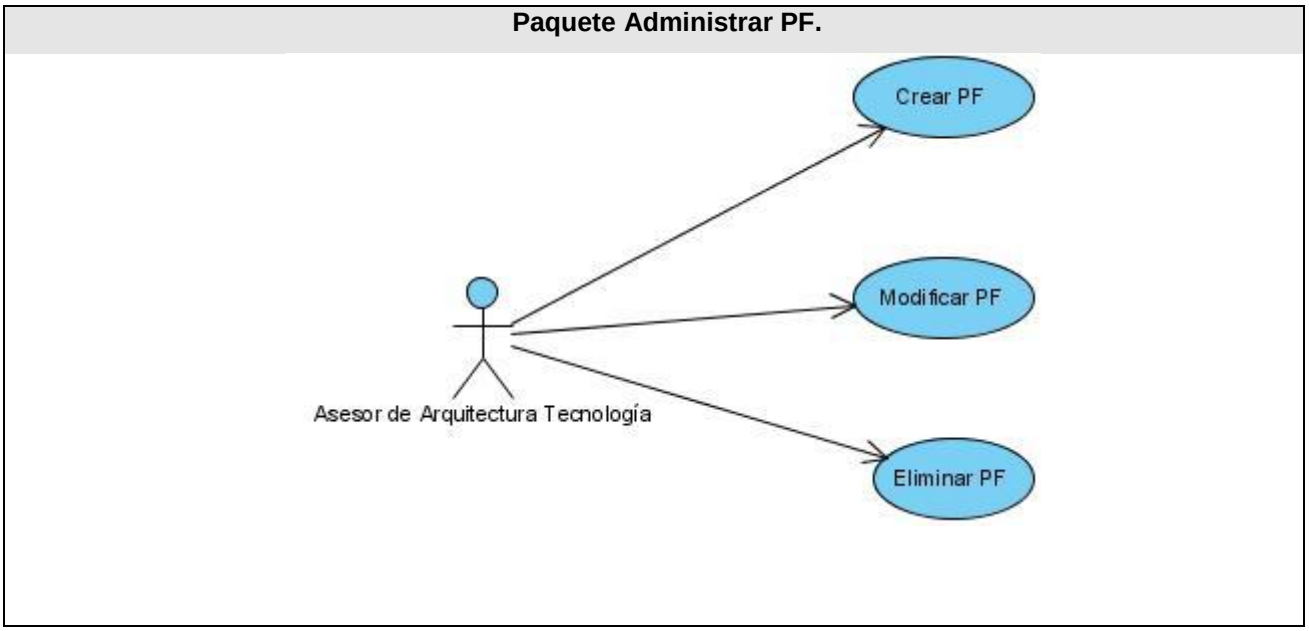

Figura 2. 9 Diagrama de CU < Paquete Administrar PF >.

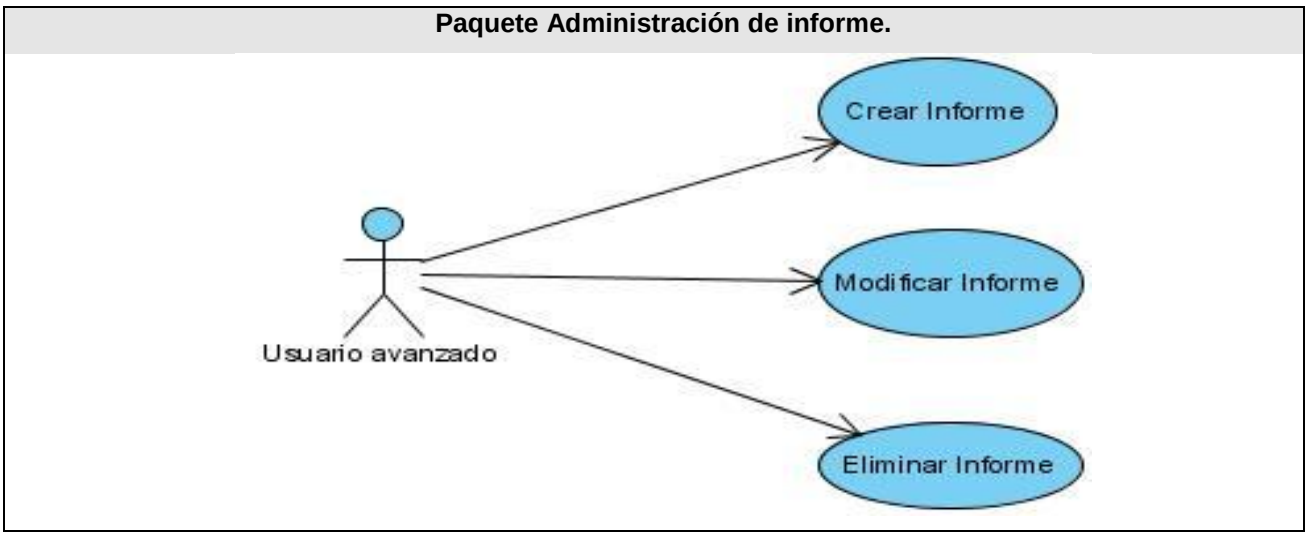

Figura 2. 10 Diagrama de CU < Paquete Administración de informe >.

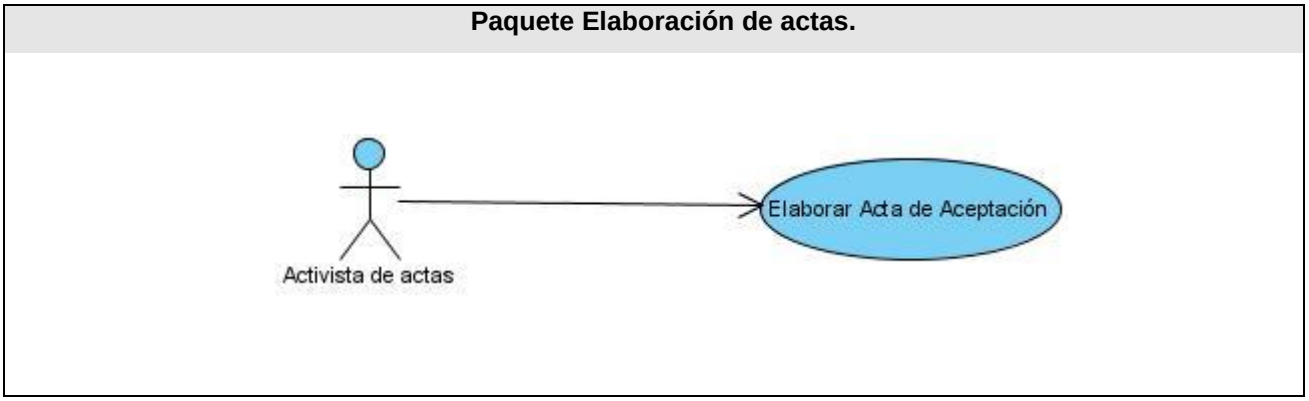

Figura 2. 11 Diagrama de CU < Paquete Elaboración de actas >.

Figura 2. 12 Diagrama de CU < Paquete Autorización de licencias de software>.

2.16 Casos de uso expandidos.

En el Anexo 5 se muestran las descripciones textuales de los restantes casos de usos descritos en los diagramas con las interfaces correspondientes.

Caso de uso	
CU-3	Verificar Planillas.
Propósito	Permite al EGSD revisar las PF y PIP.
Actores: EGSD.	
Resumen: El caso de uso se inicia cuando el EGSD entra al sistema y decide verificar las PF y las PIP que se han hecho hasta el momento.	
Referencias: R5**,** R9.	
Precondiciones:	• El EGSD debe haberse autenticado en el sistema. • El AAT debe haber creado la PF. • El EDIP debe haber creado la PIP.
Poscondiciones	• Planilla PF aceptada. • Planilla PF denegada. • Planilla PIP aceptada. • Planilla PIP denegada.

Acción del actor	**Respuesta del sistema**
1- Escoge la planilla a verificar.	2- El sistema muestra la planilla correspondiente con la posibilidad de Aceptar o Denegar la solicitud.
3- El EGSD revisa la planilla y escoge una de las opciones mostradas.	4- Si el EGSD escoge la opción: Aceptar ir a sección Aprobada. Denegar ir sección Denegada.
Sección<<Aprobada>>	
	1- El sistema pone el campo “estado_EGSD” de la tabla de la base

	de datos correspondiente a **1(Aprobada)** en dependencia de la planilla que se haya revisado.
Sección<<Denegada>>	
	1- El sistema pone el campo "estado_EGSD" de la tabla de la base de datos a **0(Denegada)**, correspondiente a la planilla que se haya revisado.
Flujo alternativo: Sección<<Denegada>>	
Acción del actor	**Respuesta del sistema**
Interfaz: Ver Anexo 8	

Caso de uso	
CU-6	Realizar búsqueda de información de productos.
Propósito	Permite al EGSD realizar una búsqueda en la base de datos para verificar si el software que se solicita fue descargado o no con anterioridad.
Actores: EGSD.	
Resumen: El caso de uso se inicia cuando el EGSD decide realizar una búsqueda en la base de datos del software que se pide para descargar.	
Referencias: R13.	
Precondiciones:	• El EGSD debe haberse autenticado. • Debe de existir una solicitud de descarga de información. • Debe de haber revisado al menos una planilla de solicitud de descarga de información.

Poscondiciones	• Se muestra el programa buscado o posibles semejantes a él.

Acción del actor	**Respuesta del sistema**
1- El EGSD selecciona la planilla que desee comprobar la existencia del software en el repositorio.	2- Muestra planilla.
3- El EGSD da clic sobre el botón buscar.	4- El sistema busca y muestra los nombres y las URL de los softwares encontrados.
5- El EGSD comprueba la existencia del software en los resultados mostrados.	
6- Si se encuentra el software en los resultados mostrados el EGSD presionará en el botón ok.	7- El sistema levanta la interfaz para notificar al AAT que el software ya ha sido descargado y el lugar donde se encuentra el producto descargado.
8- El EGSD introduce los datos de la notificación. • Datos generales de la notificación: ✓ Destino. ✓ Asunto. ✓ Descripción.	9- El sistema comprueba la validez de los datos.
10- Si no se encuentra el software en los resultados mostrados el EGSD procederá a ejecutar la descarga.	11- El sistema invoca el **CU Descargar Información.**
Flujo alternativo: Realizar búsqueda de información de productos.	
Acción del actor	**Respuesta del sistema**

	1- El sistema no muestra ningún resultado de la búsqueda.
	9- El sistema emite mensaje de error “Datos incorrectos”.
Interfaz: Ver Anexo 9	

Caso de uso	
CU-7	Descargar Información.
Propósito	Permite al EGSD realizar la descarga directamente desde Internet.
Actores: EGSD.	
Resumen: Este caso de uso se inicia cuando el EGSD decide proceder a ejecutar la descarga directamente de Internet utilizando un gestor de descarga para manejar esta información.	
Referencias: R14.	
Precondiciones:	• Se haya realizado la búsqueda en el repositorio y el software no se encuentre.
Poscondiciones	• Se descarga el software. • No se descarga el software.

Acción del actor	**Respuesta del sistema**
1- El EGSD copiará la URL y procederá a ejecutar la descarga utilizando el FlashGet.	
2- El EGSD presionará en el botón descargar.	3- El sistema salva el nombre del producto en la Base de datos.
	4- Actualiza estado del producto (descargado) que se esta descargando en la base de datos

	a 0(en descarga).
5- El EGSD Invocará el **CU Comprobar descarga** cuando desee saber si el software se descargó completamente o no.	
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
Interfaz: Ver Anexo 10	

Caso de uso	
CU-8	Comprobar descarga.
Propósito	Permite al EGSD chequear si el software se descargó o no.
Actores: EGSD.	
Resumen: El caso de uso se inicializa cuando el EGSD decide chequear la descarga de información.	
Referencias: R15.	
Precondiciones:	• Que se haya mandado a ejecutar la descarga.
Poscondiciones	• Se descargó completamente el software. • No se ha descargado el software.
Acción del actor	**Respuesta del sistema**
1- El EGSD da clic en el botón comprobar descarga.	2- El sistema levanta una interfaz con todos los softwares que se estén descargando en ese momento.
3- El EGSD selecciona el software que desee comprobar.	4- El sistema comprueba si el software se ha descargado completamente.

	5- El sistema informa al EGSD que el software se ha descargado completamente.
	6- El sistema levanta la interfaz para notificar al AAT que se ha descargado correctamente el software y la ubicación del mismo para que este pueda copiarlo. Se informará a los responsables de UCISTORE y DATALAB para que comprueben si este software existe en sus servidores.
7- El EGSD introduce los datos y pulsa sobre el botón enviar. • Datos generales de la notificación. ✓ Asunto. ✓ Responsable de UCISTORE. ✓ Responsable de DATALAB. ✓ AAT. ✓ Descripción.	8- El sistema comprueba la validez de los datos y envía el mensaje.
Flujo alternativo: Comprobar descarga	
Acción del actor	**Respuesta del sistema**
	4- El sistema emite un mensaje comunicándole al EGSD que el software no se ha descargado completamente.

	4.1- Si el software se descargó completamente actualiza el estado correspondiente al estado de la descarga (descargado) en la base de datos a 1(producto descargado).
	8- El sistema emite un mensaje al EGSD “Debe llenar todos los campos correctamente”.
Interfaz: Ver Anexo 11	

Conclusiones

En el capítulo anterior se ha realizado el modelado del negocio de nuestro sistema, identificado los actores, casos de uso y trabajadores del negocio. Se definieron los requisitos funcionales y no funcionales, se creó el diagrama de paquetes de los casos de uso del sistema, y los casos de uso fueron descritos, dándole una visión del sistema que se desarrollará.

CAPÍTULO 3. ANÁLISIS Y DISEÑO DEL SISTEMA.

3.1 Introducción.

En el presente capítulo se muestran los modelos correspondientes al análisis y al diseño permitiendo comprender y refinar los requisitos capturados en el capítulo anterior, así como crear la base estructural del sistema a implementar. También se muestran los diagramas de clases y de interacción (secuencia), modelo de datos, además de mostrar la interfaz de usuario del sistema.

3.2 Descripción del framework a utilizar, patrones y arquitectura.

3.2.1. ¿Que es un patrón?

Según Larman:

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

O sea es una forma de unificar todo aquel contenido que se repite en más de una ocasión para dar solución a un problema determinado, garantizando así la reutilización del código que hoy en día tiene una gran repercusión a nivel mundial por las facilidades que ofrece a los desarrolladores a la hora de implementar sistemas nuevos.

Para el análisis y diseño del sistema que se propone se han empleado una serie de patrones, tales como:

- Modelo-Vista-Controlador (MVC). (Figura 3.1)
- Bajo acoplamiento.
- Alta cohesión.
- Experto.
- Creador.

3.2.2. ¿Qué es un framework?

Según muchos autores casi todo el mundo que se dedica al desarrollo de software utiliza, conoce, o como mínimo se ha tropezado con el concepto de framework, cuya traducción aproximada sería "marco de trabajo".

Sin embargo el concepto de framework no es tan sencillo y mucho menos fácil de definir.

"…diseño abstracto orientado a objetos para un determinado tipo de aplicación, que se compone de una clase abstracta para cada componente principal del diseño; contendrá normalmente una librería de subclases que pueden ser utilizadas como componentes del diseño…" [18]

Después de un estudio se decide utilizar como framework para el desarrollo de la aplicación web el Symfony ya que es uno de los frameworks más utilizados a nivel mundial, cuenta con una amplia documentación y comunidad que lo respalda, lo que nos facilitaría el desarrollo de aplicaciones web con código legible, sencillo y organizado y con gran facilidad a la hora de darle mantenimiento dichas aplicaciones.

Symfony está basado en un patrón clásico del diseño web conocido como MVC, que está formado por tres niveles:

- El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

Debido al uso del patrón MVC en el análisis y diseño del sistema que se propone se ha definido para la realización del mismo una arquitectura MVC la cual separa el controlador, la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes del tipo de gestor de bases de datos utilizado por la aplicación.

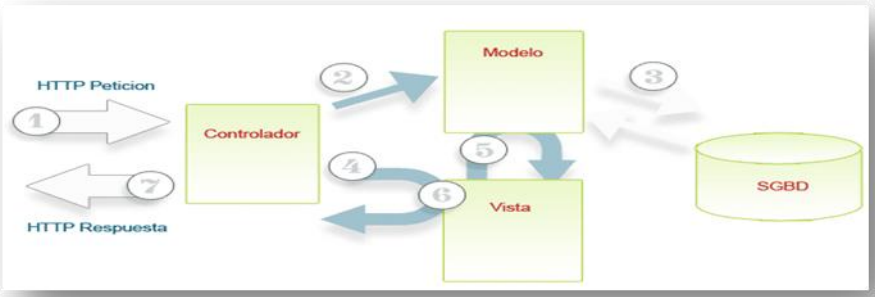

Figura 3. 1 Modelo-Vista-Controlador.

3.3 Modelo de Análisis.

Durante el análisis se desarrolla un amplio estudio de los requisitos para extraer gran cantidad de información de ellos, se adentra en lo que sería el funcionamiento interno del sistema por lo que se emplea en la descripción del análisis un lenguaje más cercano al de los desarrolladores.

"Durante el análisis, analizamos los requisitos que se describen en la captura de requerimientos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que nos ayude a estructurar el sistema entero, incluyendo su arquitectura" [19].

El modelo de análisis persigue como objetivos fundamentales lograr una comprensión más exacta de los requisitos del sistema, describir lo que requiere el cliente y establecer una aproximación a lo que sería el modelo de diseño.

3.4 Diagrama de clases del análisis.

El diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema [19].

Estos diagramas representan una vista estática del sistema. Existen tres estereotipos de clases estandarizados en UML y se utilizan para ayudar a los desarrolladores a distinguir el ámbito de las diferentes clases. Estas clases son:

Interfaz*:* Modelan la interacción entre el sistema y sus actores.

Entidad*:* Modelan información que posee larga vida y que por lo general es persistente.

Control: Coordinan la realización de un caso de uso controlando las actividades de los objetos que implementan su funcionalidad.

Seguidamente se presentan los diagramas de clases del análisis:

Diagrama de clases del análisis: CU Autenticar usuario.

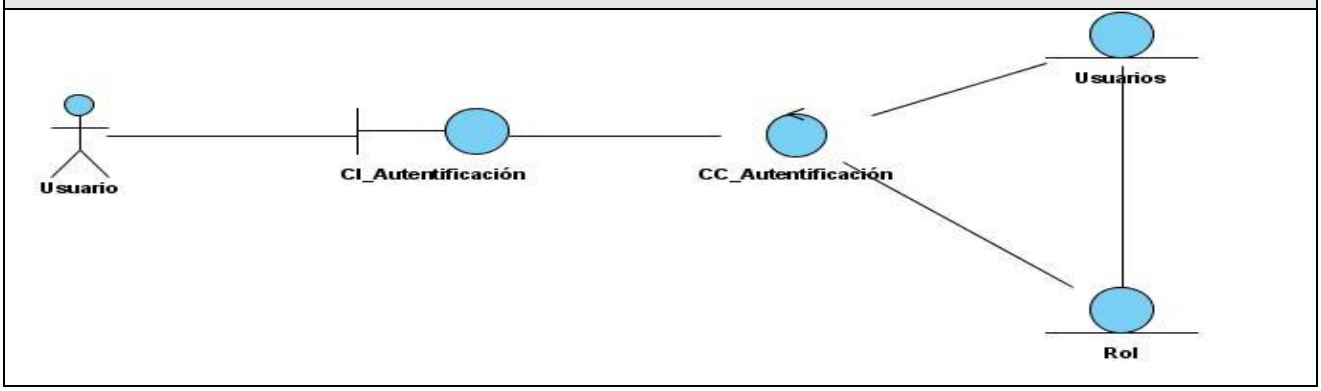

Figura 3. 2 DCA CU Autenticar usuario.

Diagrama de clases del análisis: CU Cargar Planillas.

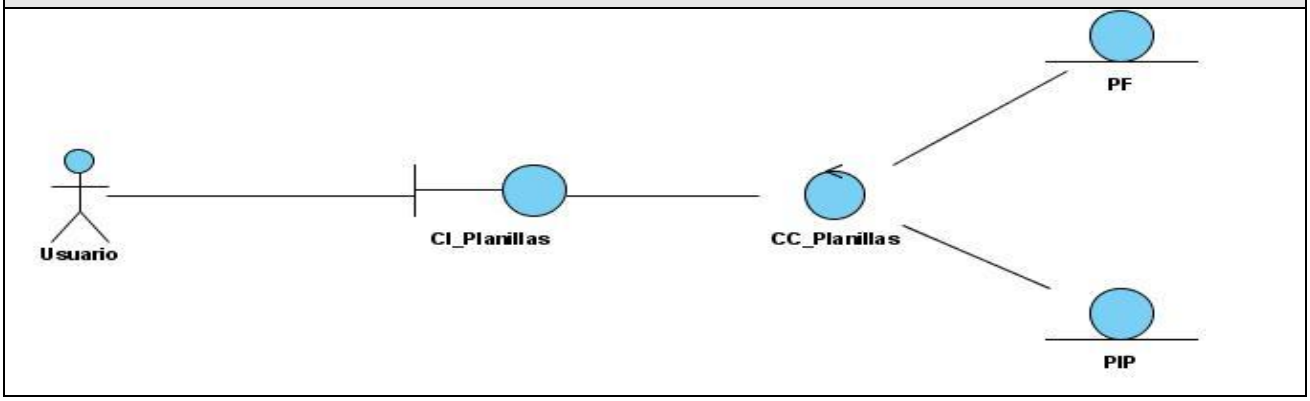

Figura 3. 3 DCA CU Cargar planillas.

Diagrama de clases del análisis: CU Realizar búsqueda información de productos.

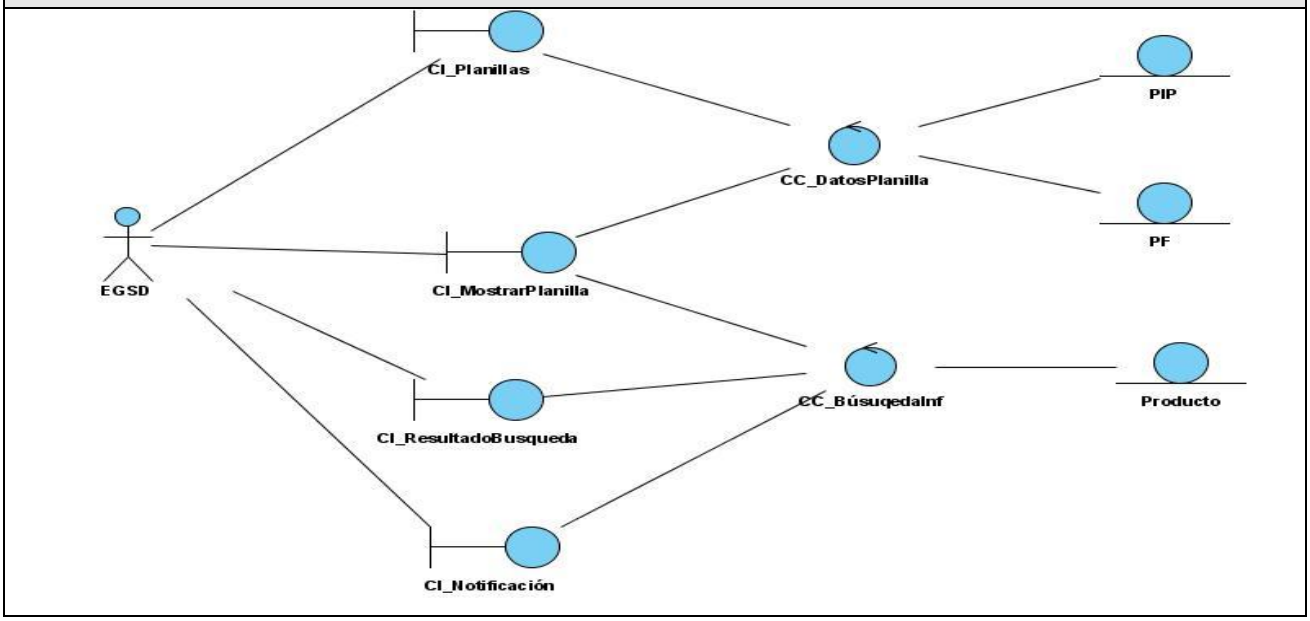

Figura 3. 4 DCA CU Realizar búsqueda información de productos

Diagrama de clases del análisis: CU Verificar planillas.

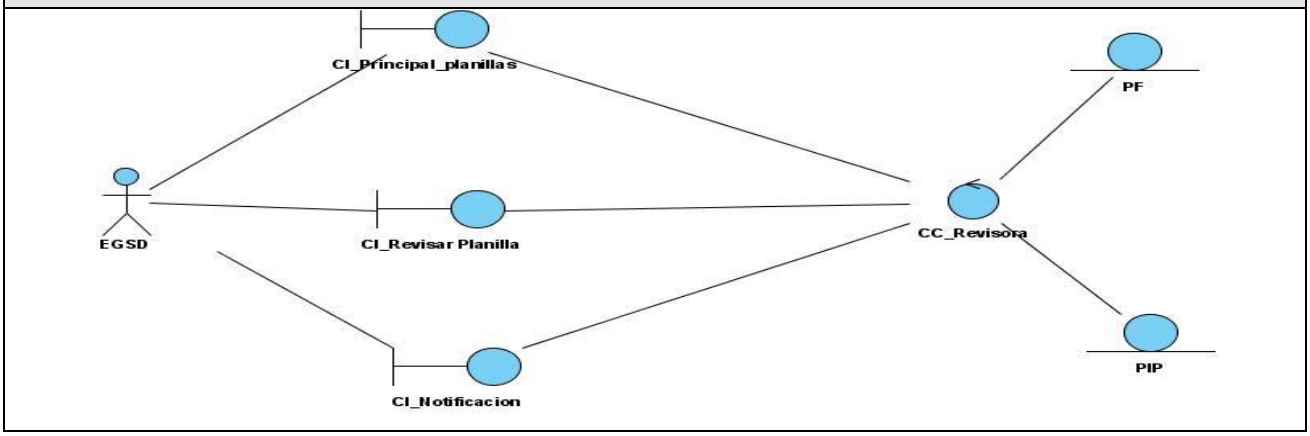

Figura 3. 5 DCA CU Verificar planillas.

Diagrama de clases del análisis: CU Comprobar descarga.

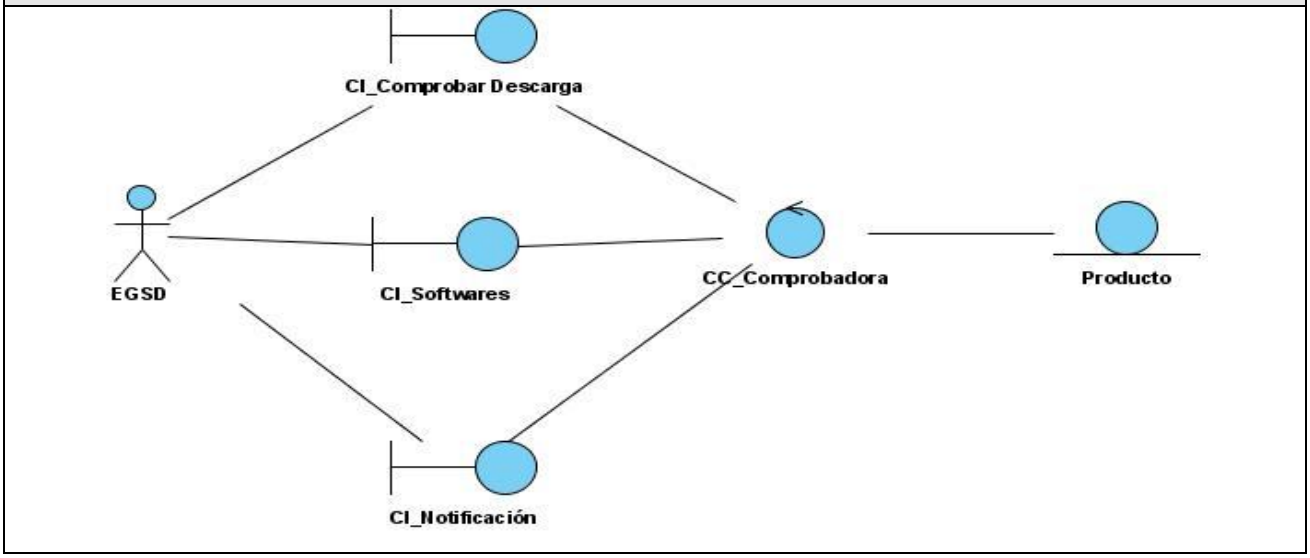

Figura 3. 6 DCA CU Comprobar descarga.

Diagrama de clases del análisis: CU Descargar información.

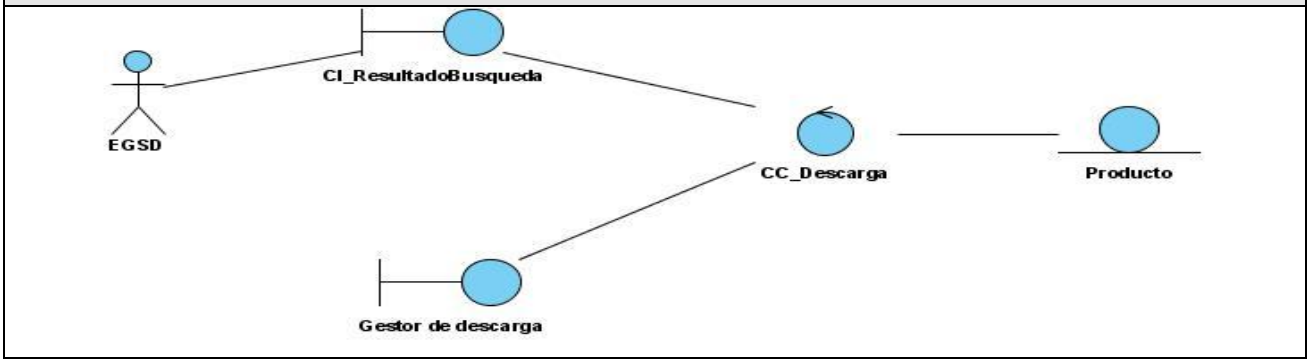

Figura 3. 7 DCA CU Descargar información.

Diagrama de clases del análisis: CU Actualizar estado del software.

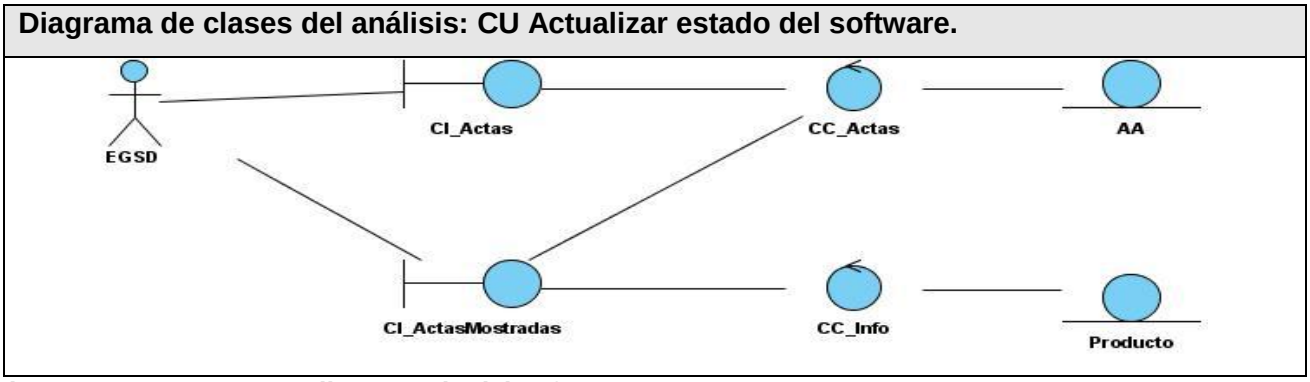

Figura 3. 8 DCA CU Actualizar estado del software.

Diagrama de clases del análisis: CU Chequear licencias.

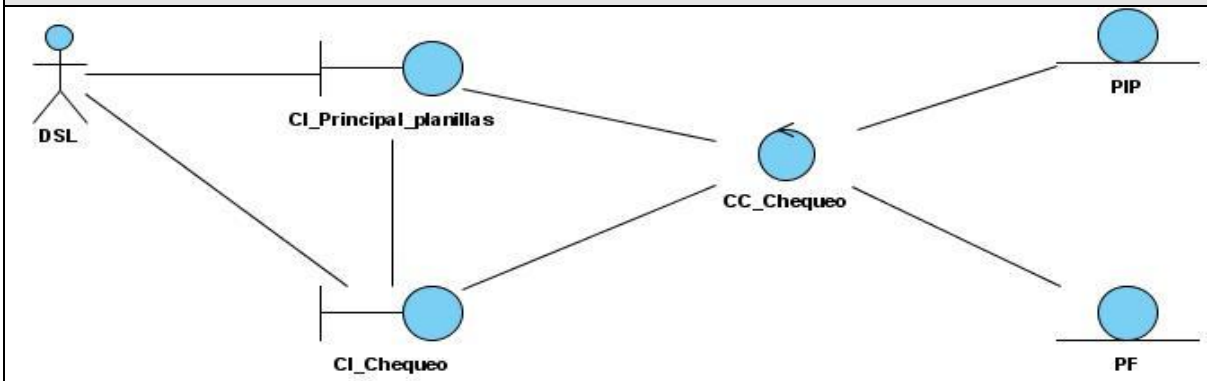

Figura 3. 9 DCA CU Chequear licencias.

Diagrama de clases del análisis: CU Gestionar Informe.

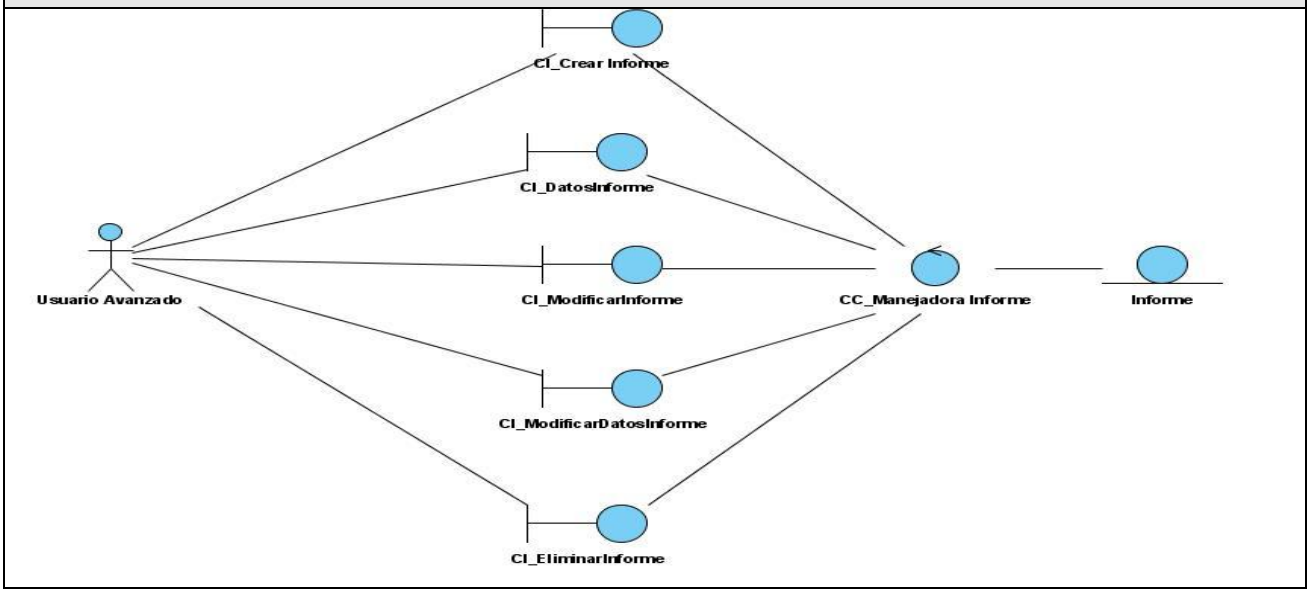

Figura 3. 10 DCA CU Gestionar Informe.

Diagrama de clases del análisis: CU Gestionar PF.

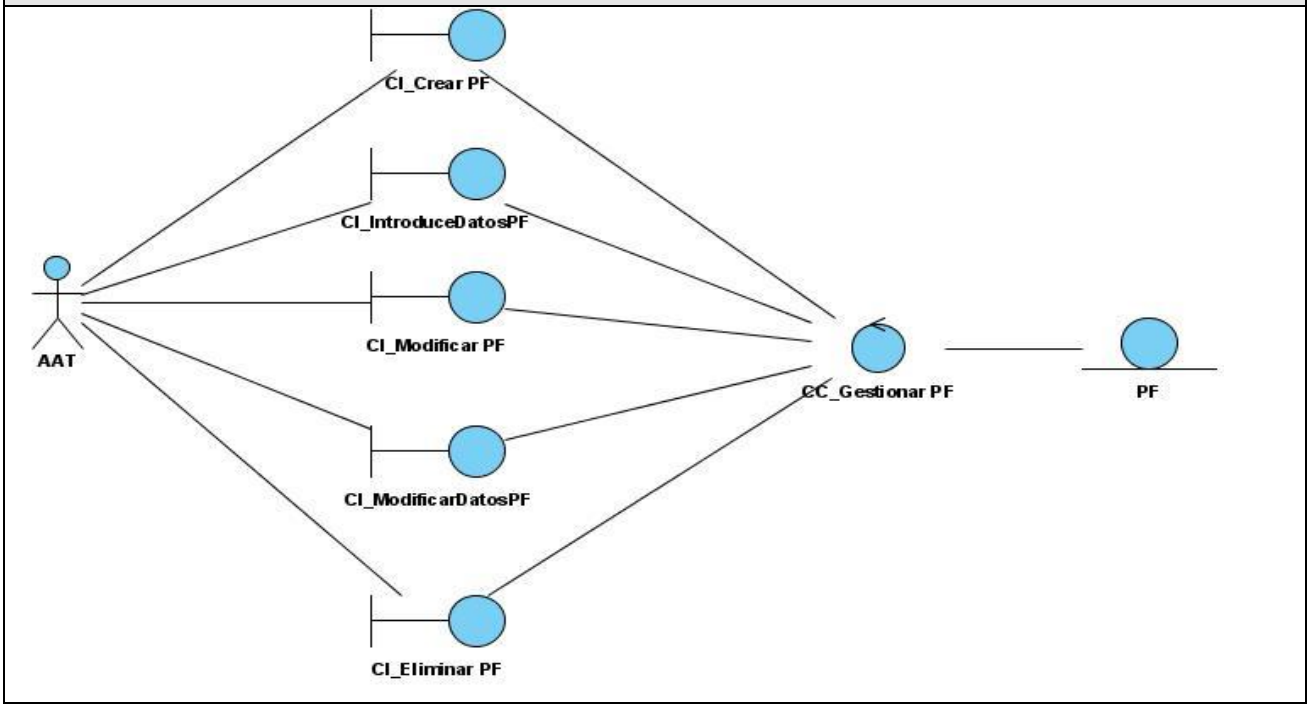

Figura 3. 11 DCA CU Gestionar PF.

Diagrama de clases del análisis: CU Gestionar PIP.

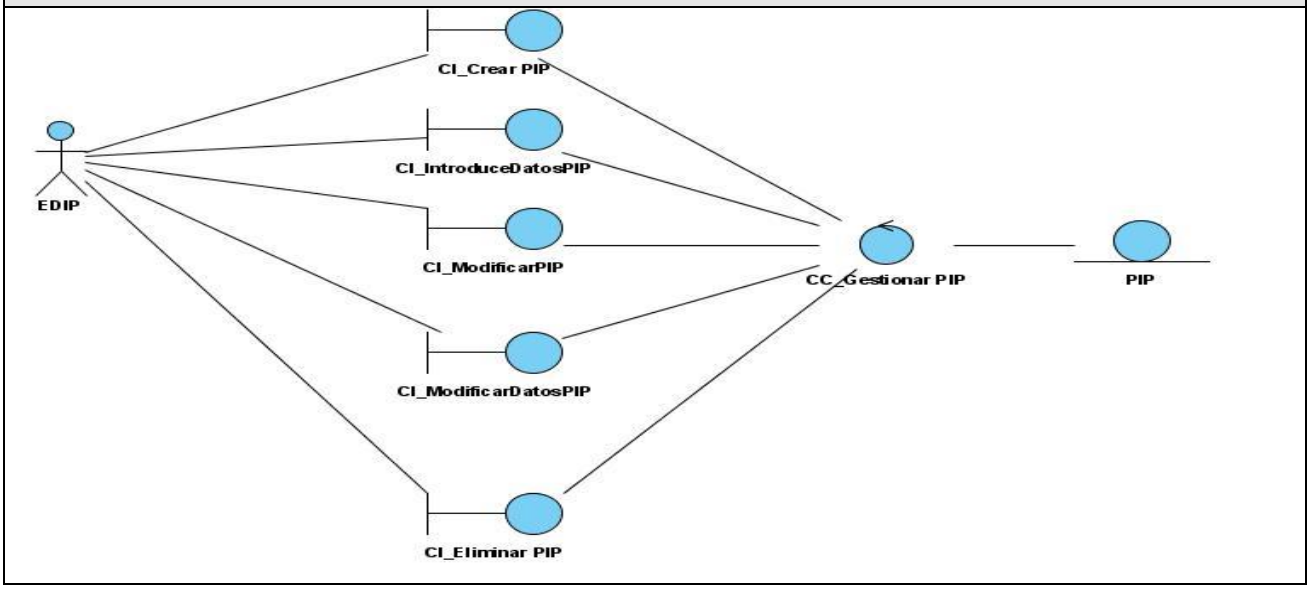

Figura 3. 12 DCA CU Gestionar PIP.

Diagrama de clases del análisis: CU Gestionar PUCI.

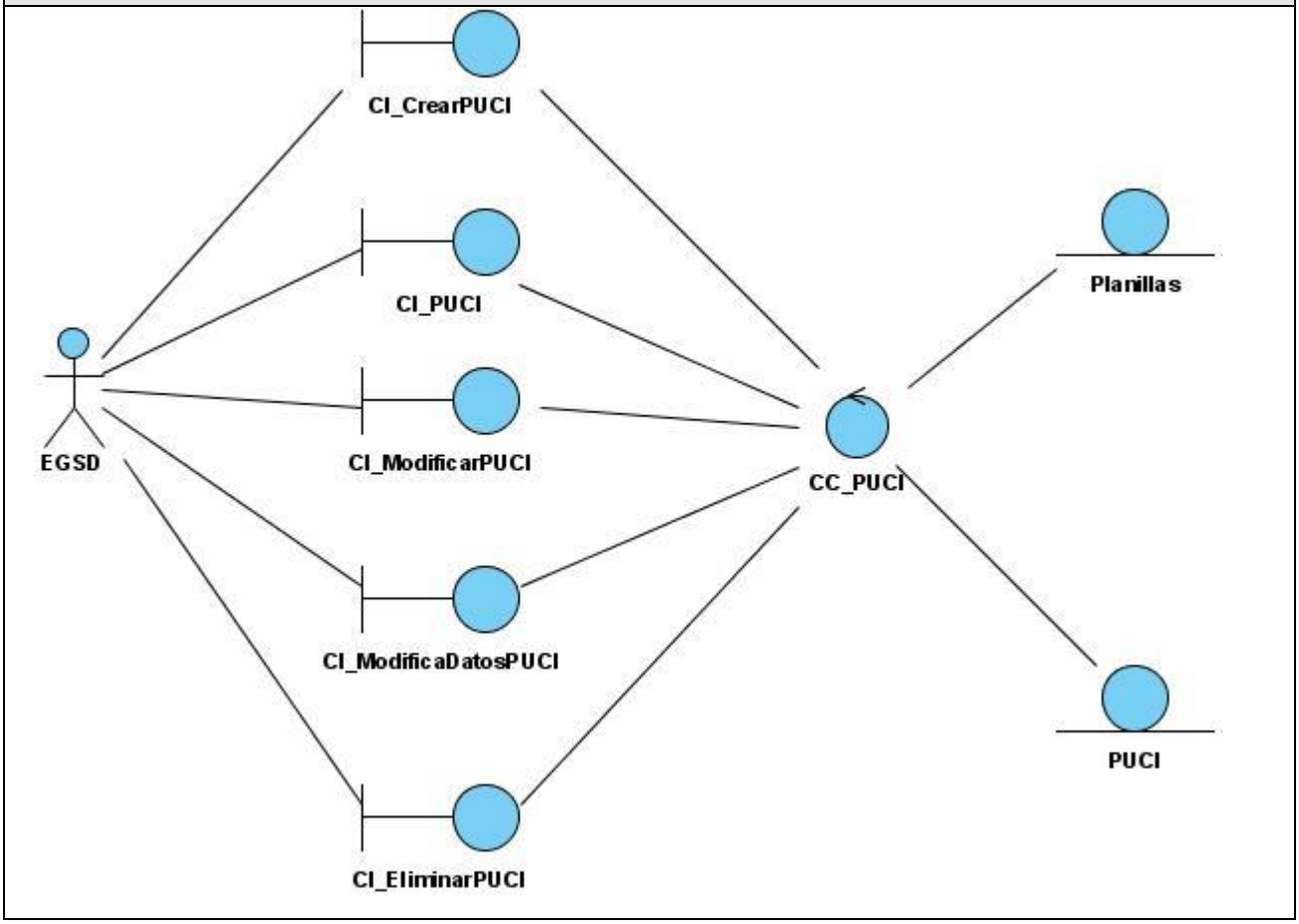

Figura 3. 13 DCA CU Gestionar PUCI.

Diagrama de clases del análisis: CU Elaborar actas.

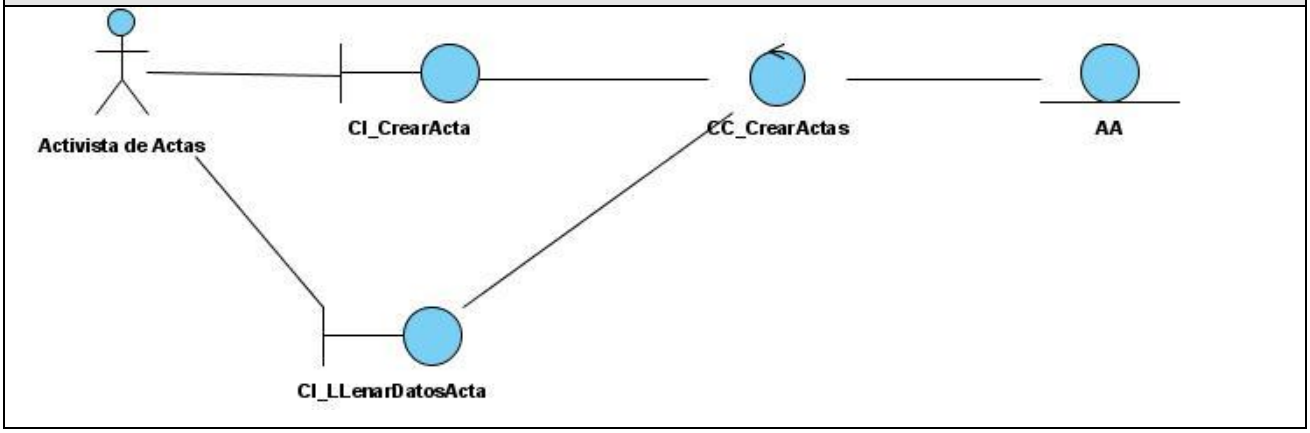

Figura 3. 14 DCA CU Elaborar actas.

3.6 Diagrama de colaboración del análisis.

En el Anexo 6 se muestran los diagramas de colaboración del análisis de los casos de usos.

3.7 Modelo de diseño.

El modelo de diseño es la representación de algo que se va a desarrollar, en esta fase es donde se modela el sistema para que soporte todos los requisitos tanto funcionales como no funcionales.

El modelo de diseño es el modelo de objetos que describe la realización de los CU, y sirve como una abstracción del modelo de implementación y el código fuente. Es usado como entrada inicial en la actividad de implementación.

3.8 Diagramas de clase del diseño.

Diagrama de clases del diseño: CU Autenticar usuario.

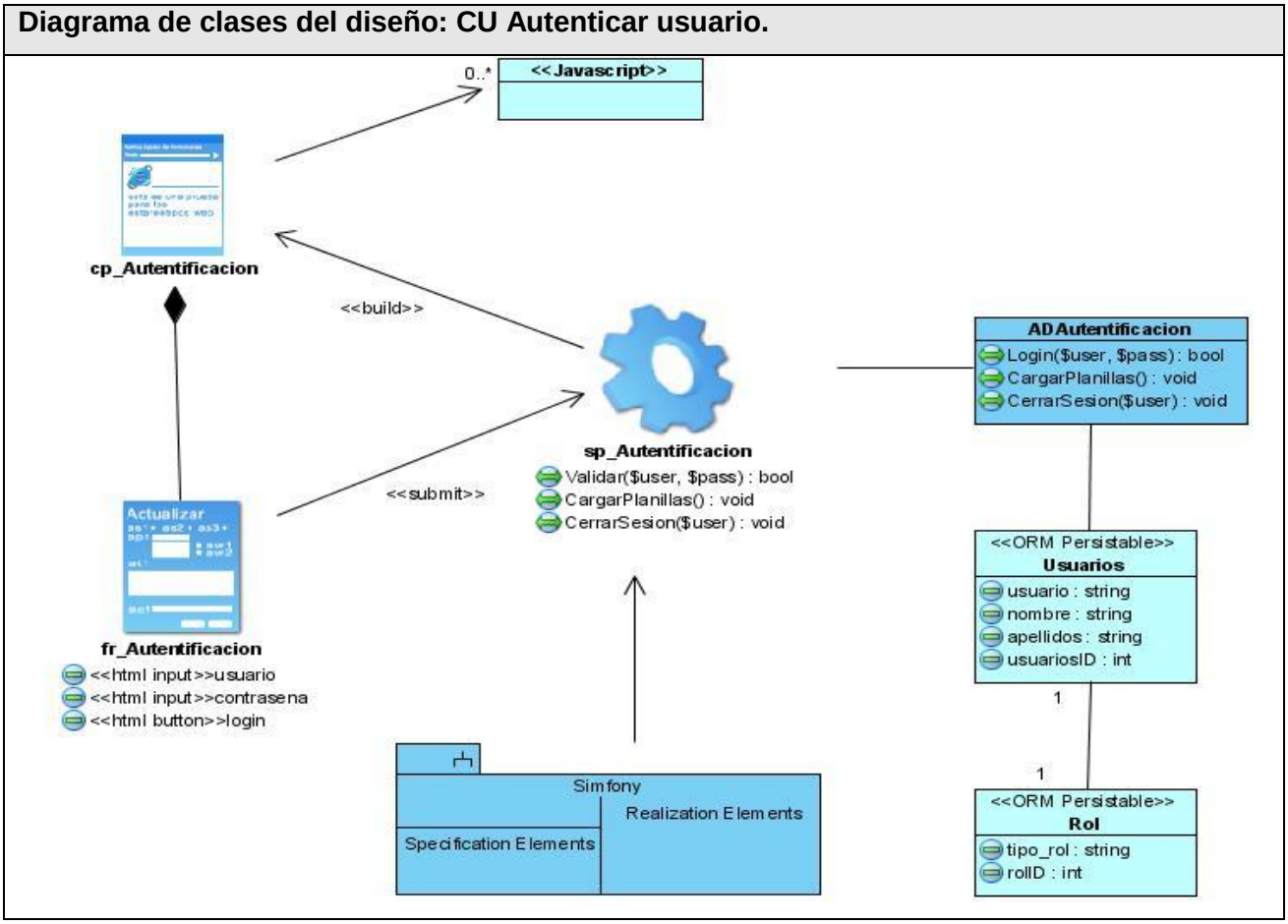

Figura 3. 15 DCD CU Autenticar usuario.

Diagrama de clases del diseño: CU Cargar Planillas.

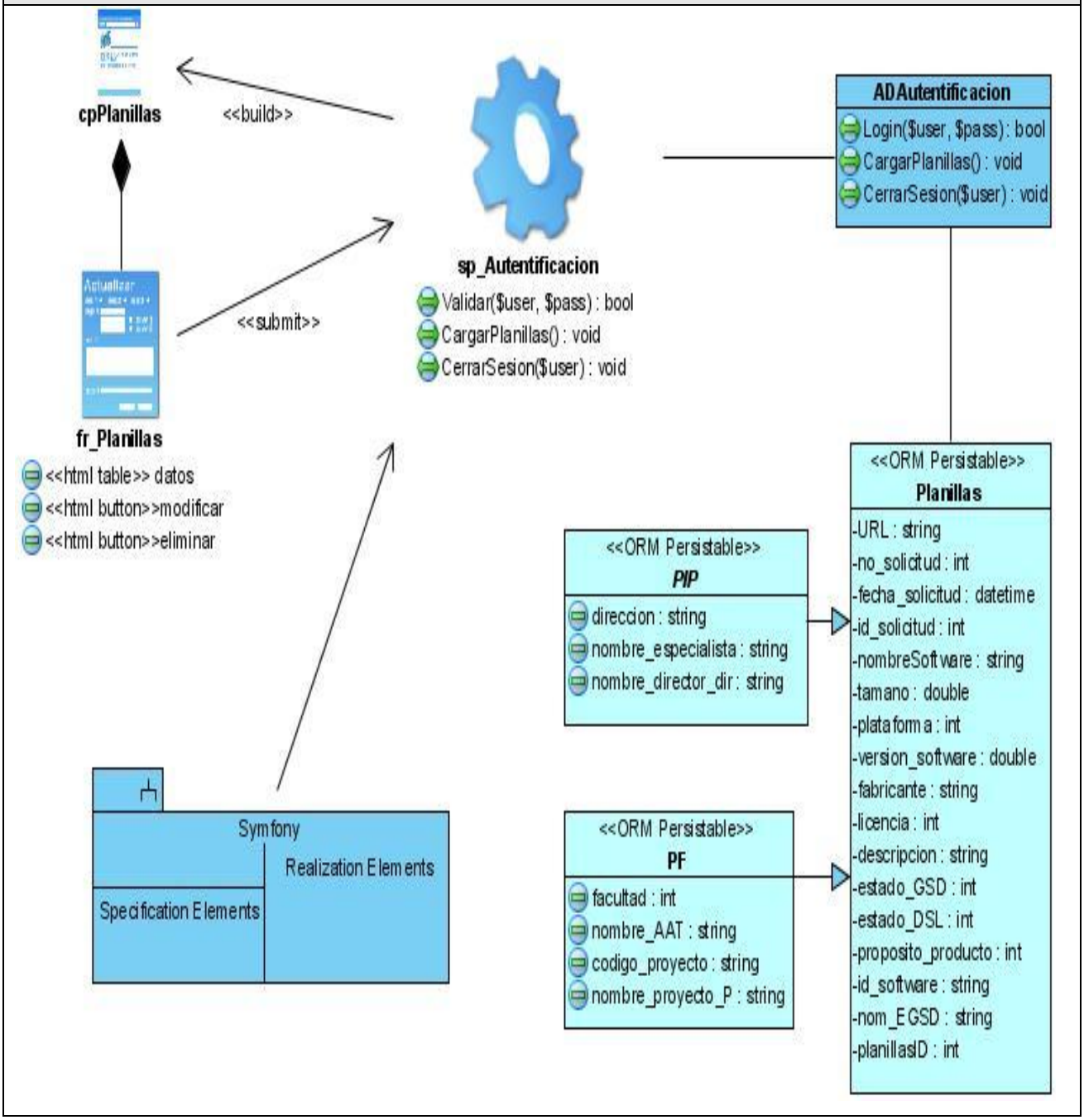

Figura 3. 16 DCD CU Cargar Planillas.

Diagrama de clases del diseño: CU Realizar búsqueda información de productos.

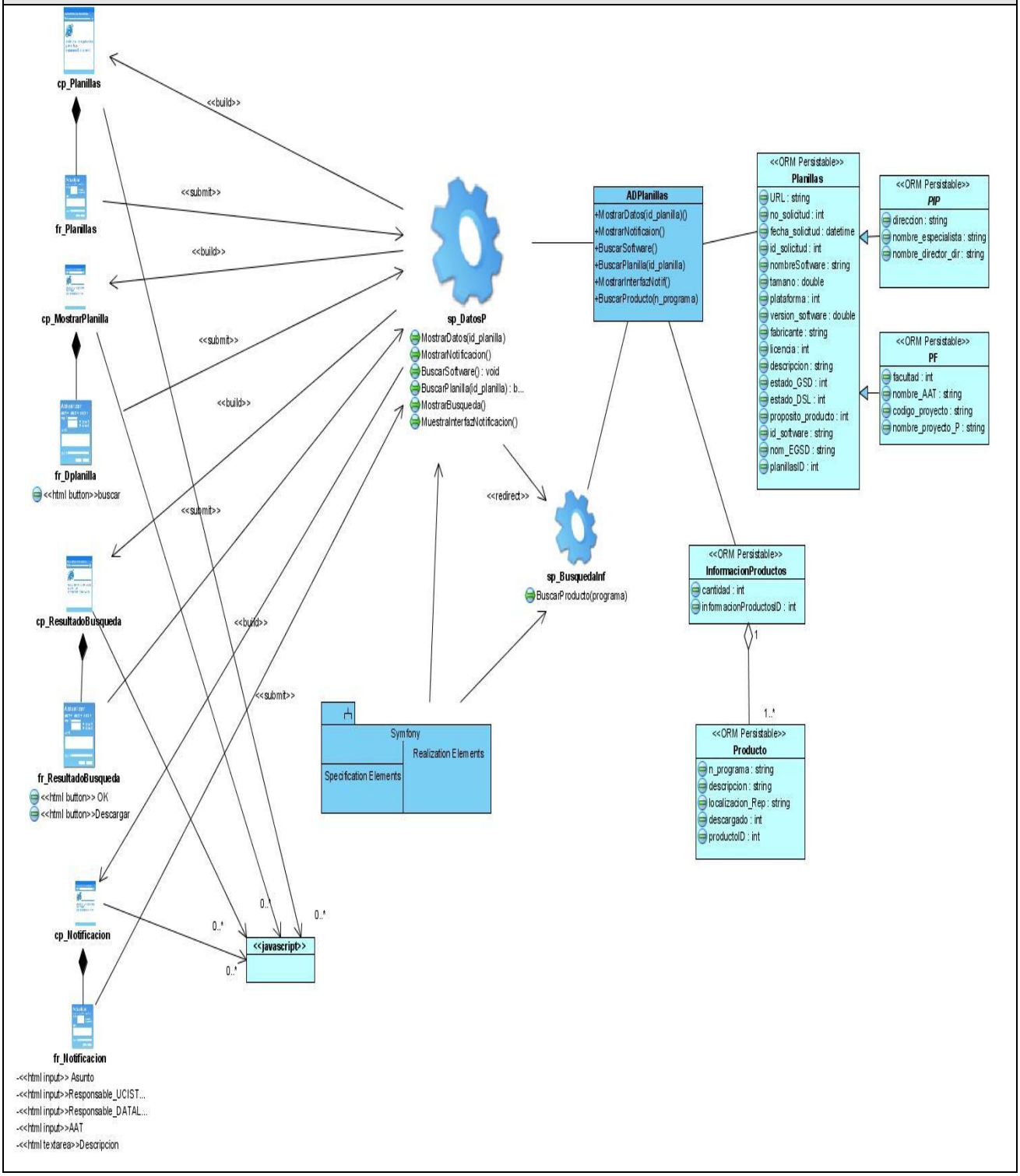

Figura 3. 17 DCD CU Realizar búsqueda información de productos.

Diagrama de clases del diseño: CU Verificar planillas.

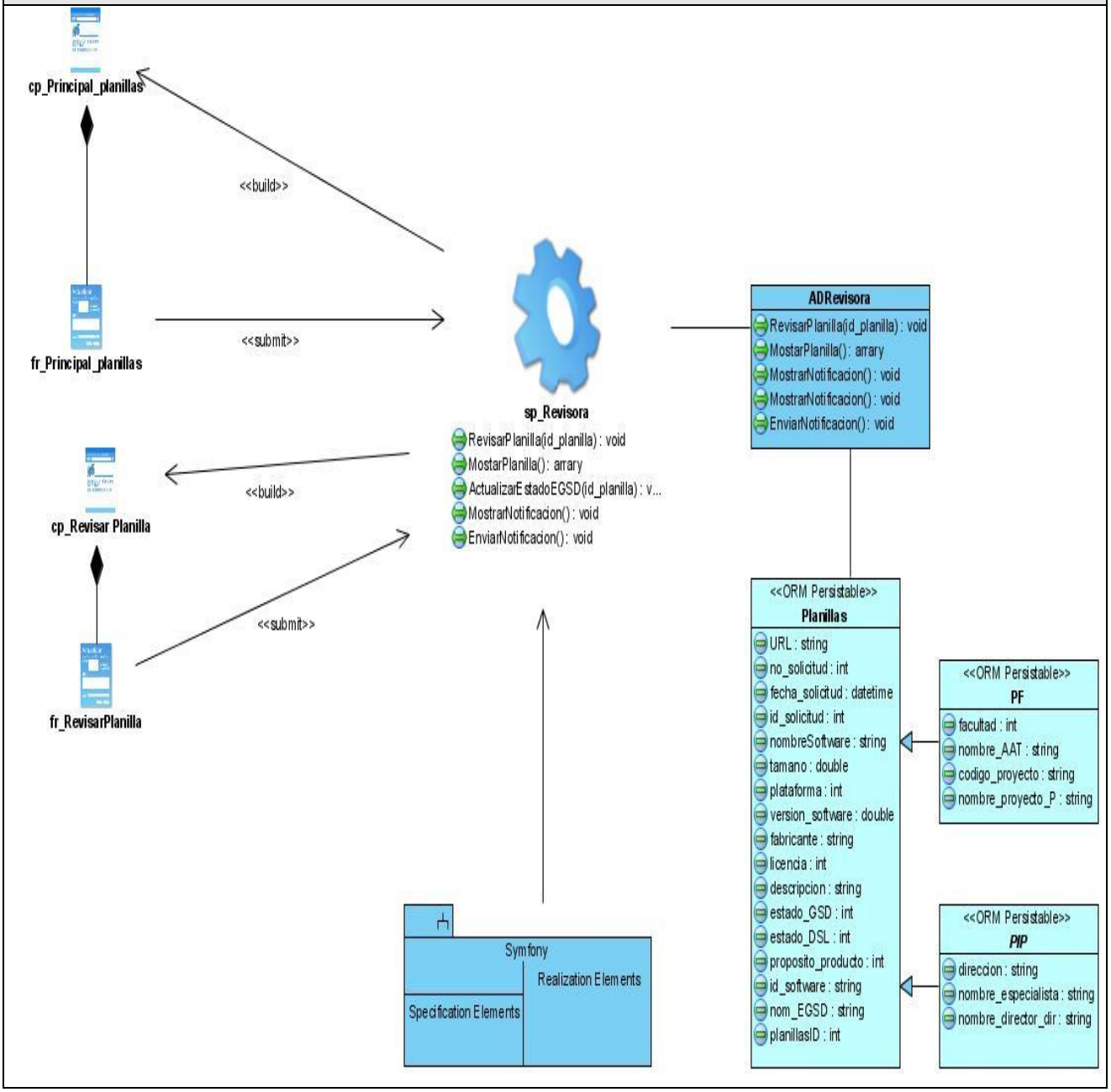

Figura 3. 18 DCD CU Verificar planillas.

Diagrama de clases del diseño: CU Comprobar descarga.

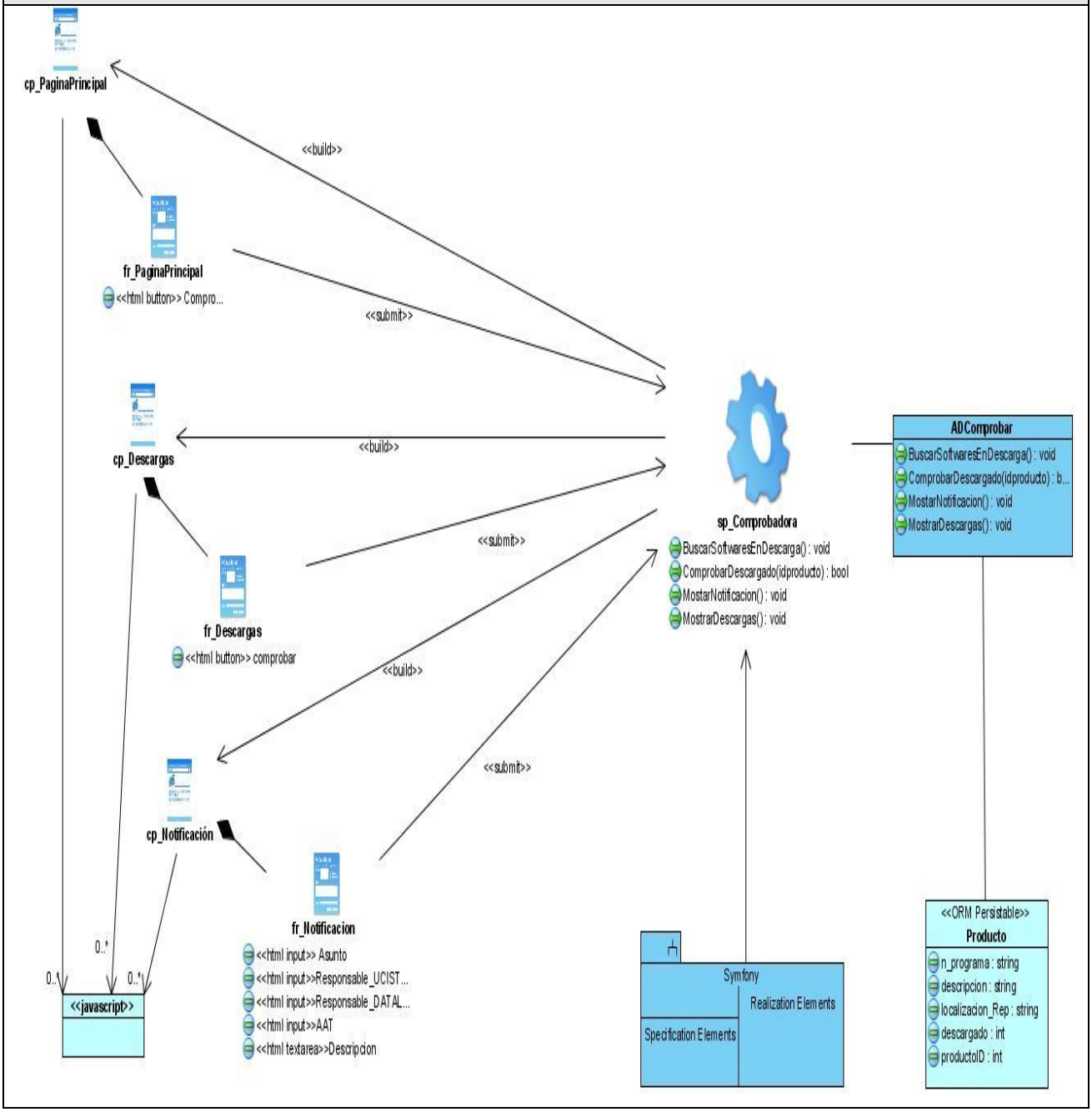

Figura 3. 19 DCD CU Comprobar descarga.

Diagrama de clases del diseño: CU Descargar información.

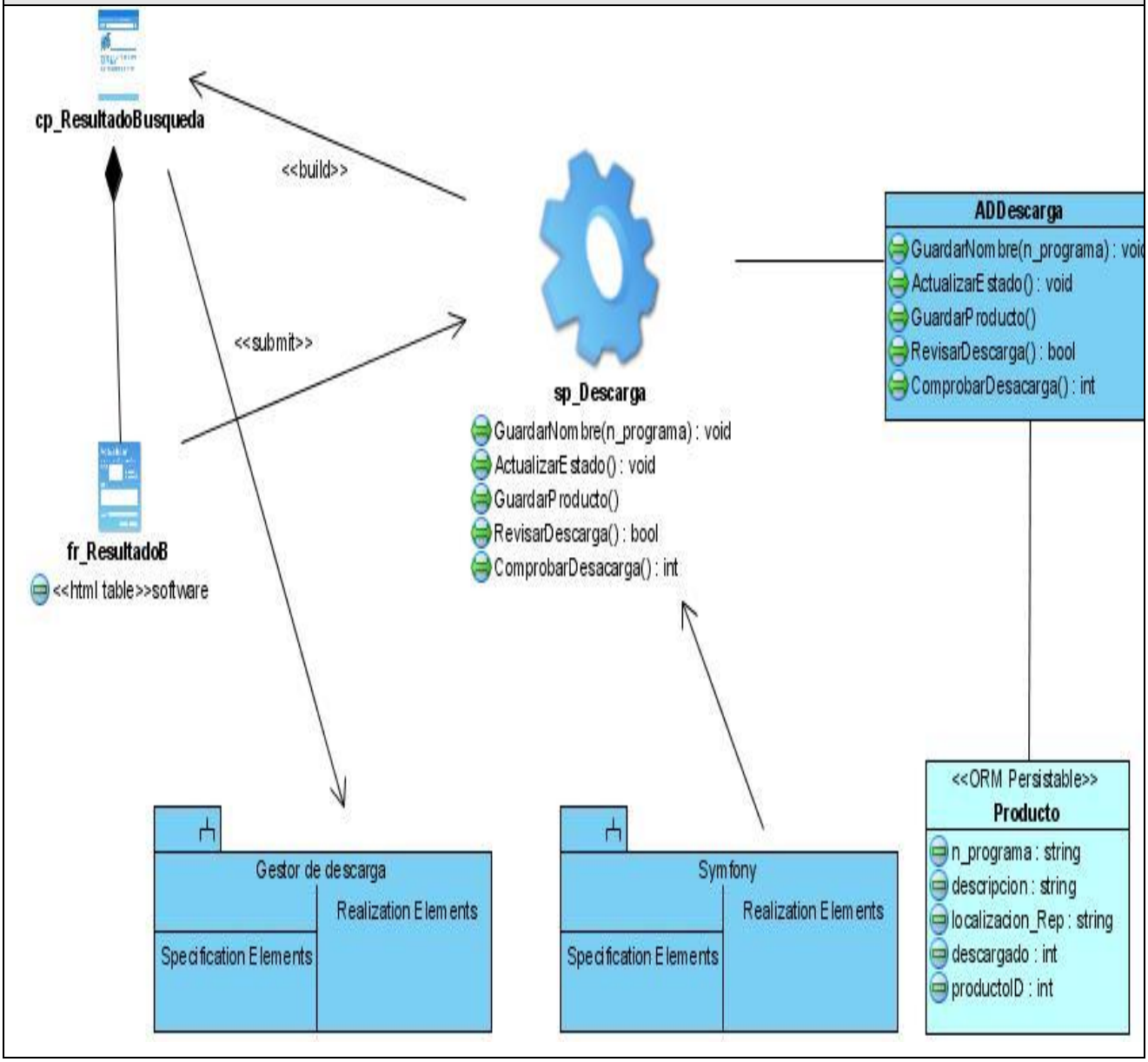

Figura 3. 20 DCD CU Descargar información.

Diagrama de clases del diseño: CU Actualizar estado del software.

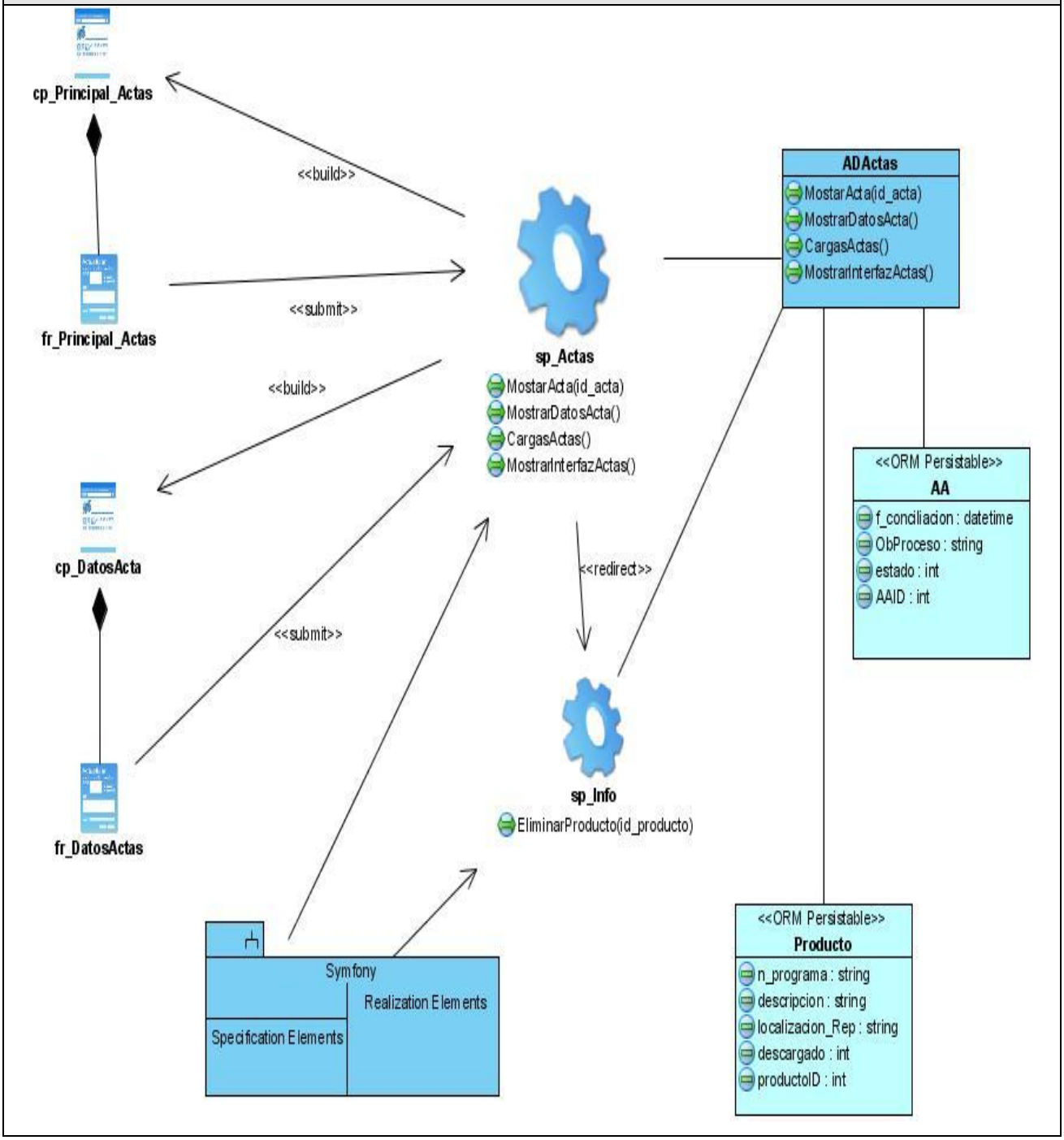

Figura 3. 21 DCD CU Actualizar estado del software.

Diagrama de clases del diseño: CU Chequear licencias.

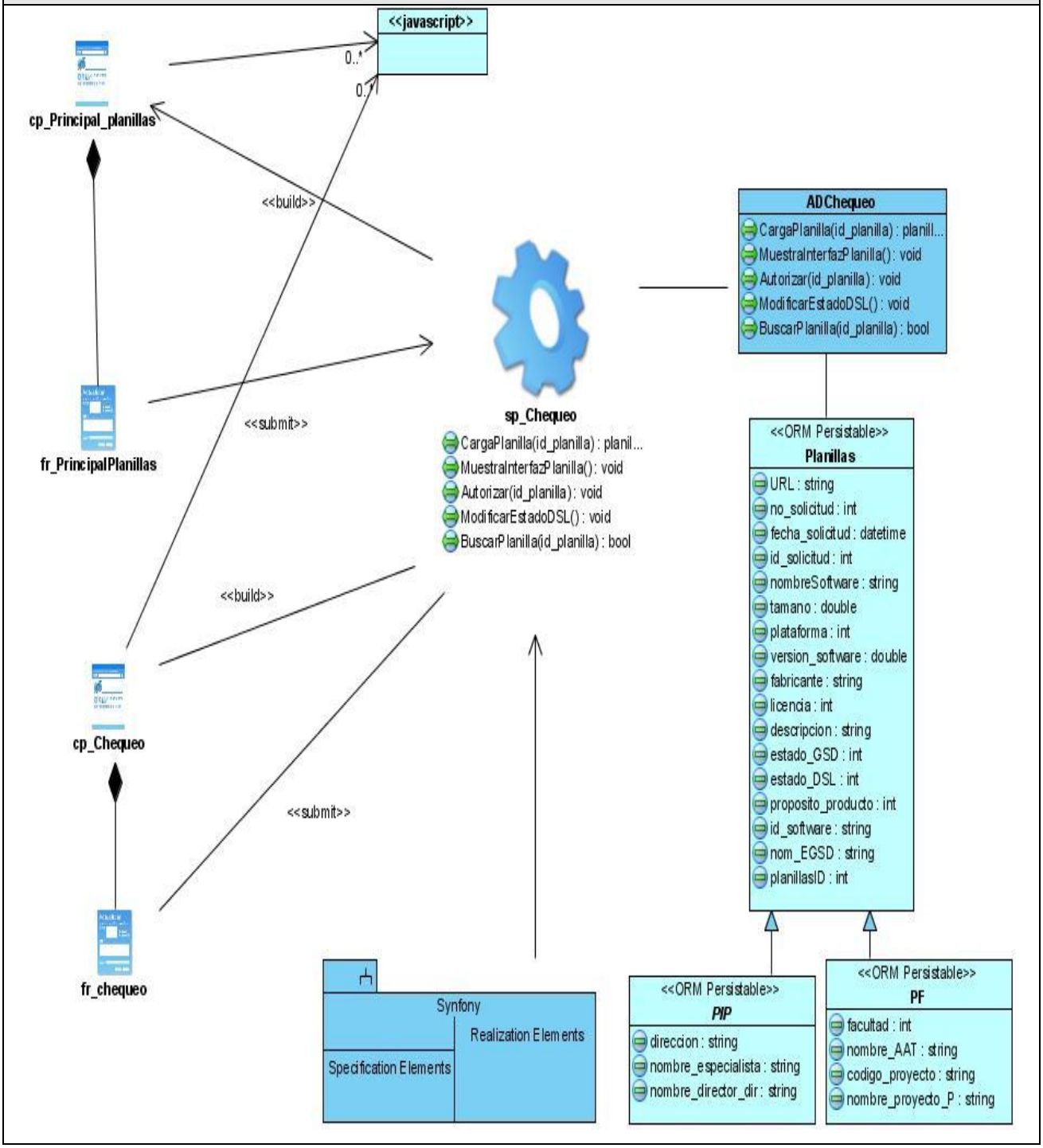

Figura 3. 22 DCD CU Chequear licencias.

Diagrama de clases del diseño: CU Gestionar Informe.

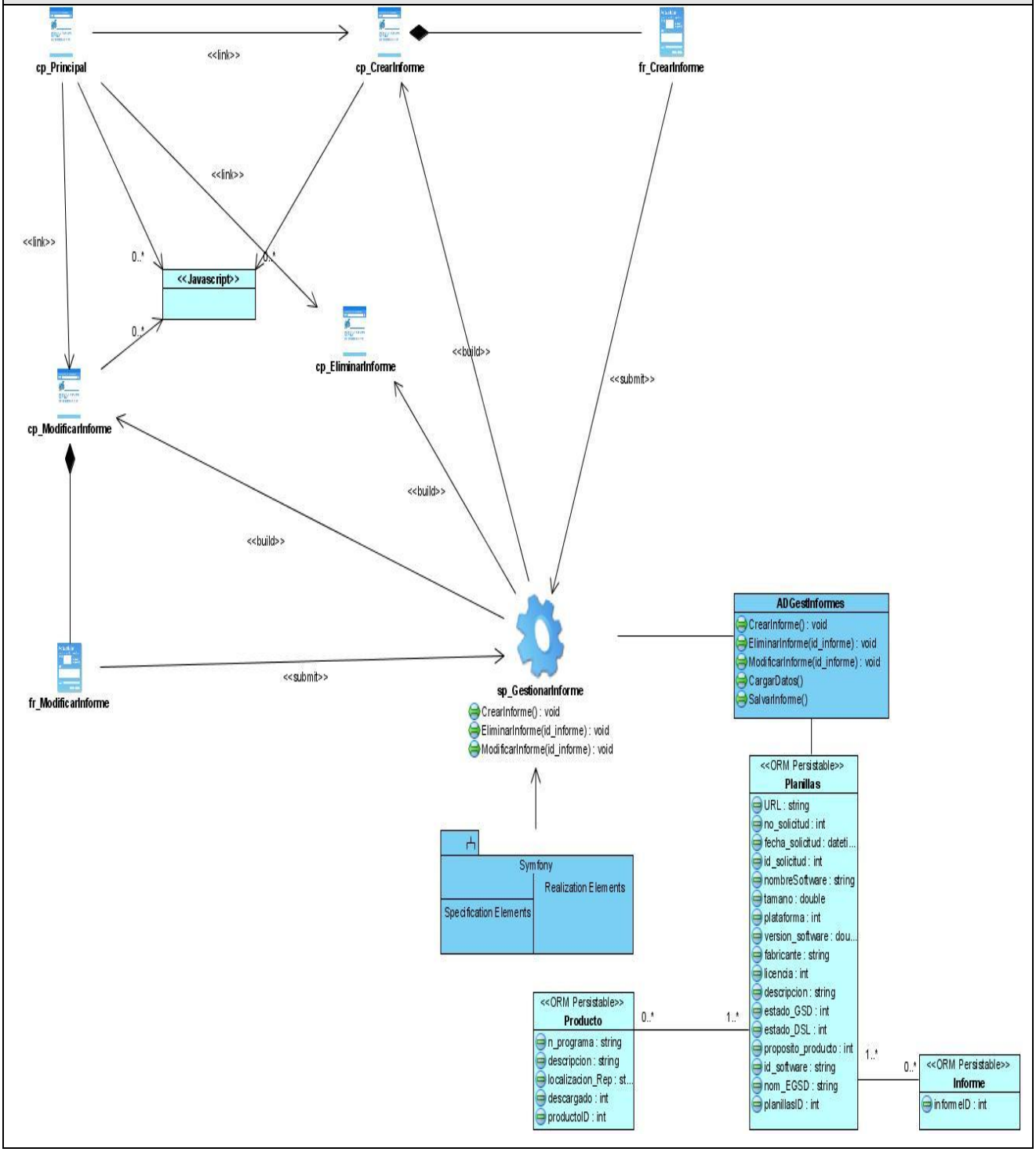

Figura 3. 23 DCD CU Gestionar Informe.

Diagrama de clases del diseño: CU Gestionar PF.

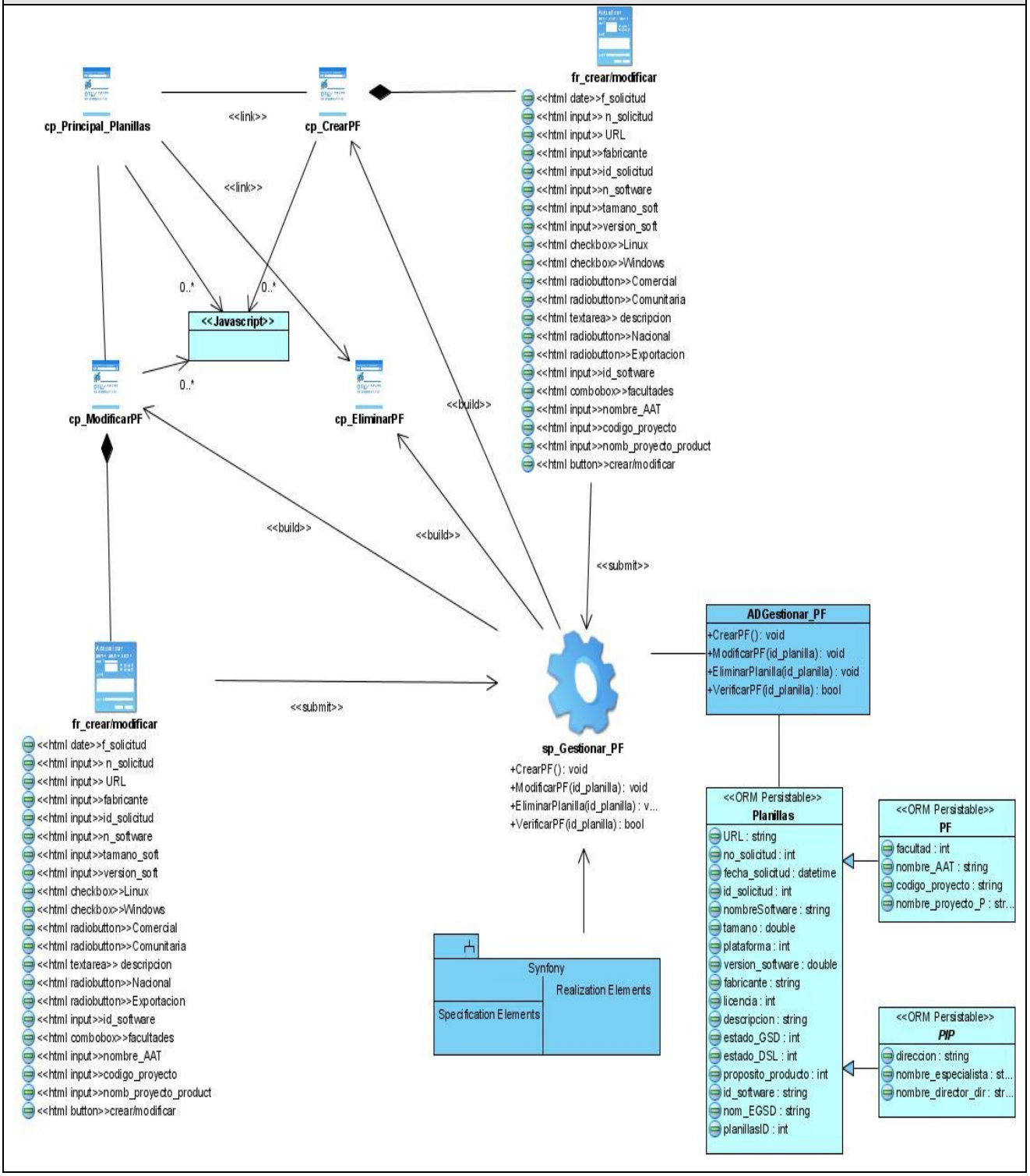

Figura 3. 24 DCD CU Gestionar PF.

Diagrama de clases del diseño: CU Gestionar PIP.

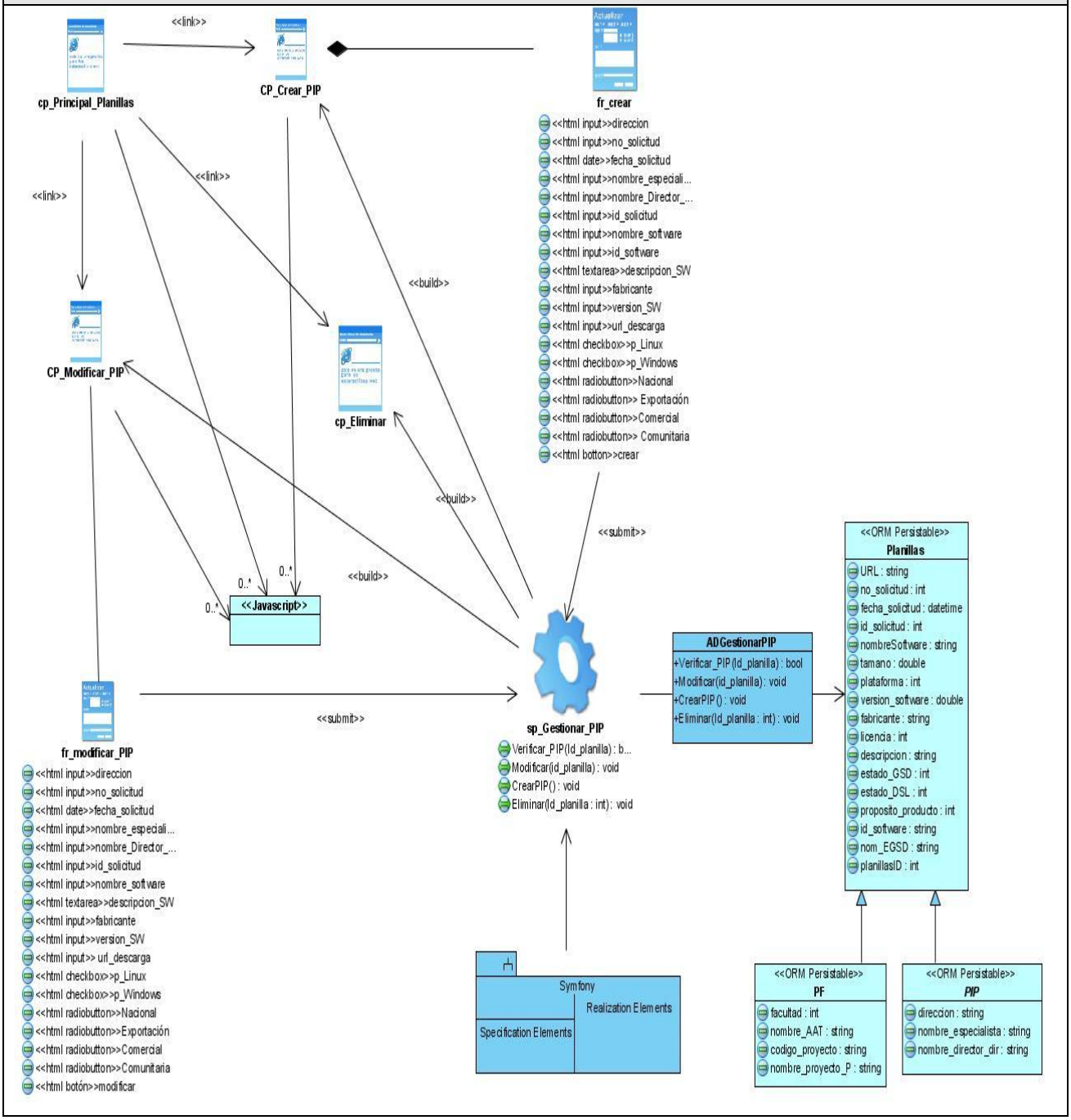

Figura 3. 25 DCD CU Gestionar PIP.

Diagrama de clases del diseño: CU Gestionar PUCI.

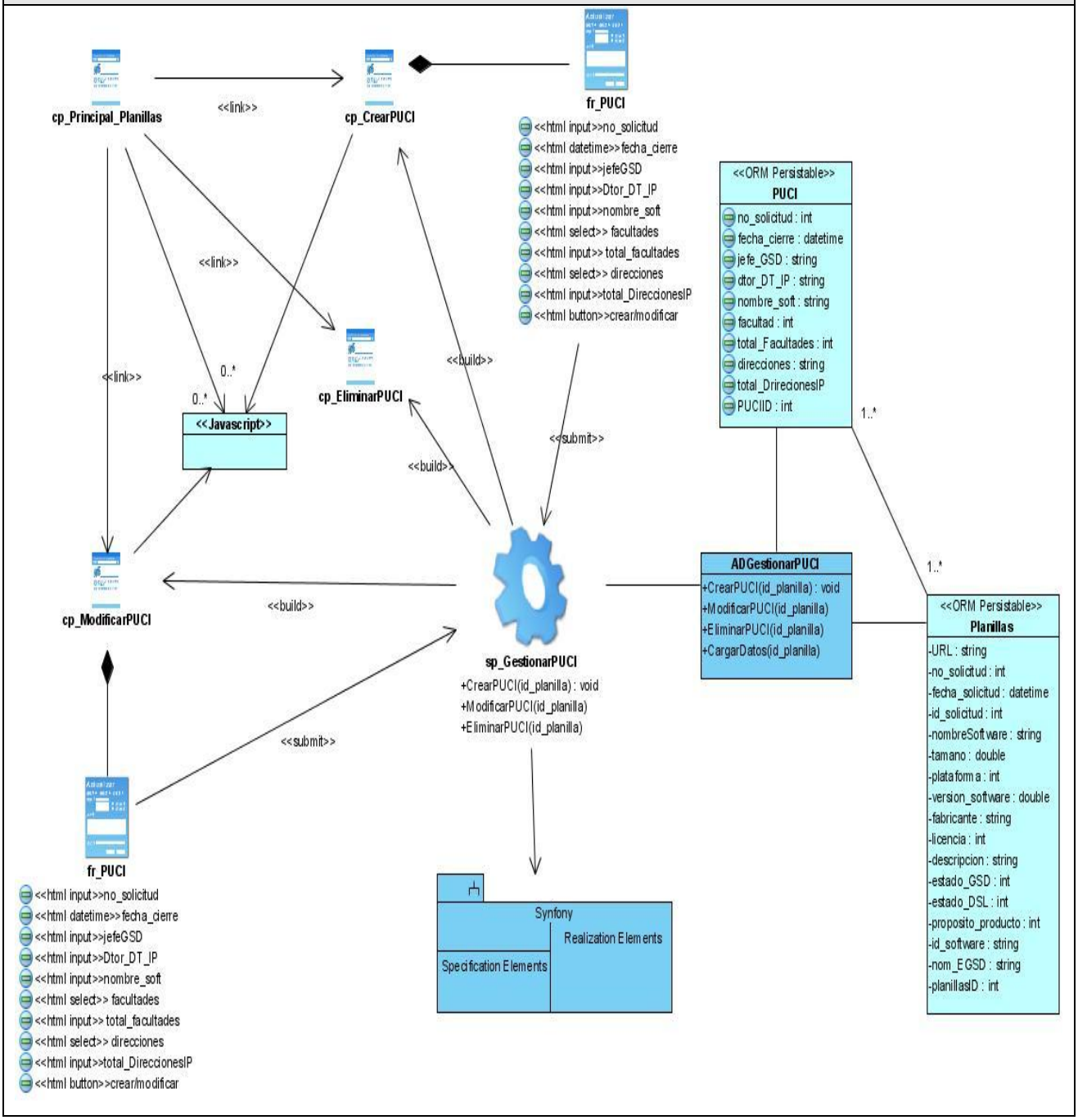

Figura 3. 26 DCD CU Gestionar PUCI.

Diagrama de clases del diseño: CU Elaborar actas.

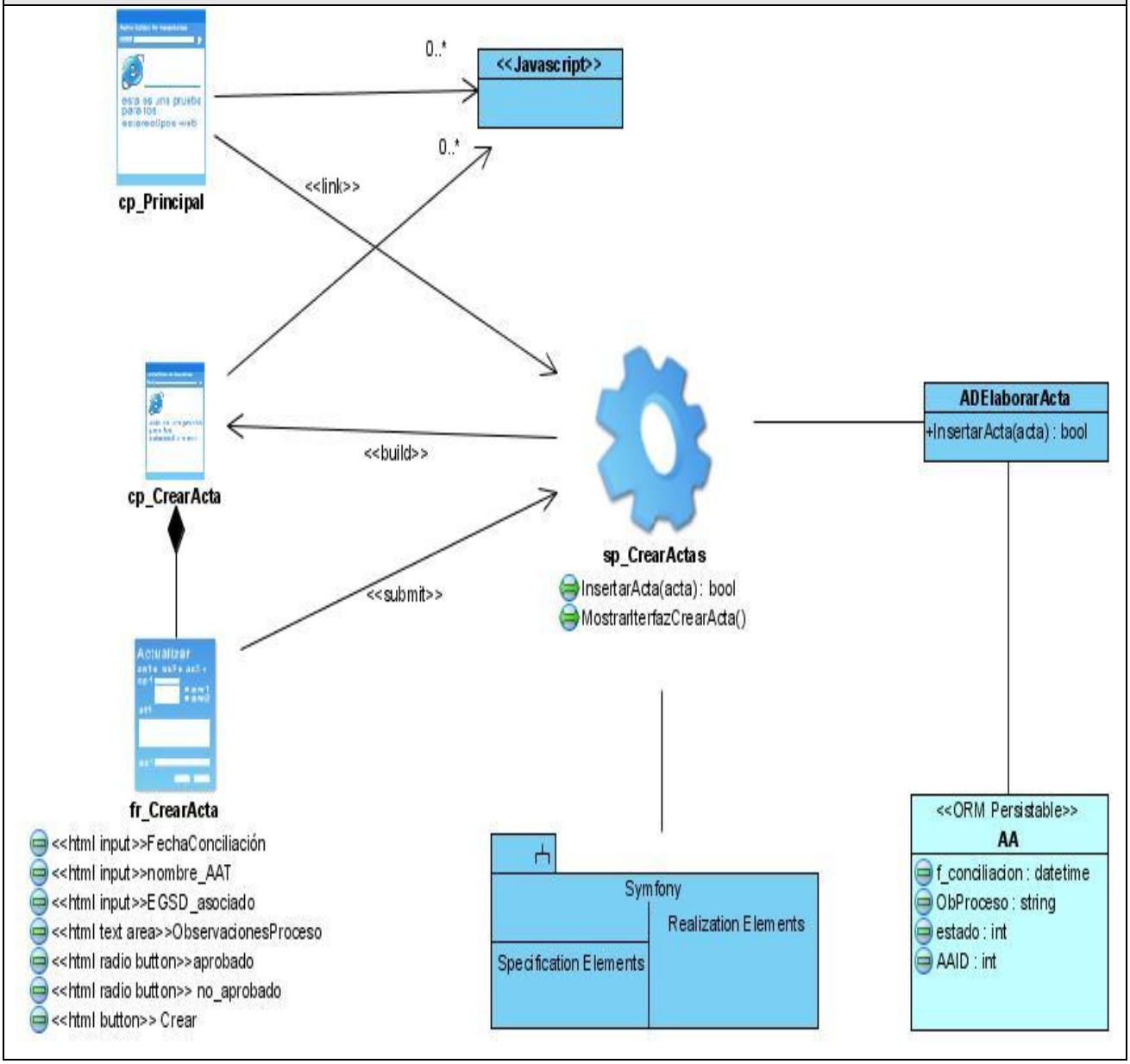

Figura 3. 27 DCD CU Elaborar actas.

3.9 Diagrama de secuencia del diseño.

Los diagramas de secuencia del diseño se pueden ver en el Anexo 7**.**

3.10 Diseño de la base de datos.

Diagrama de clases persistentes.

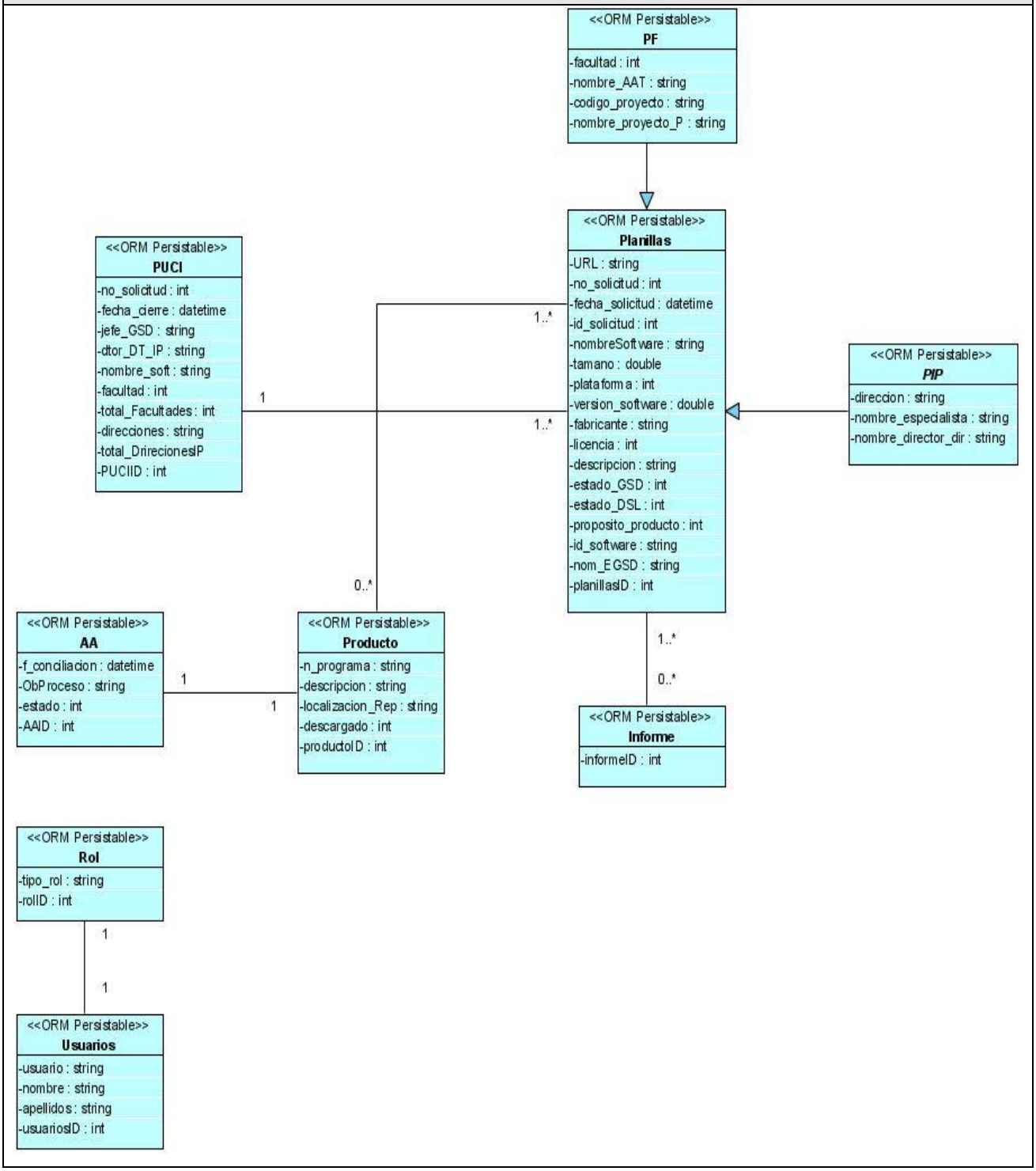

Figura 3. 28 Diagrama de clases persistentes.

Diagrama entidad relación.

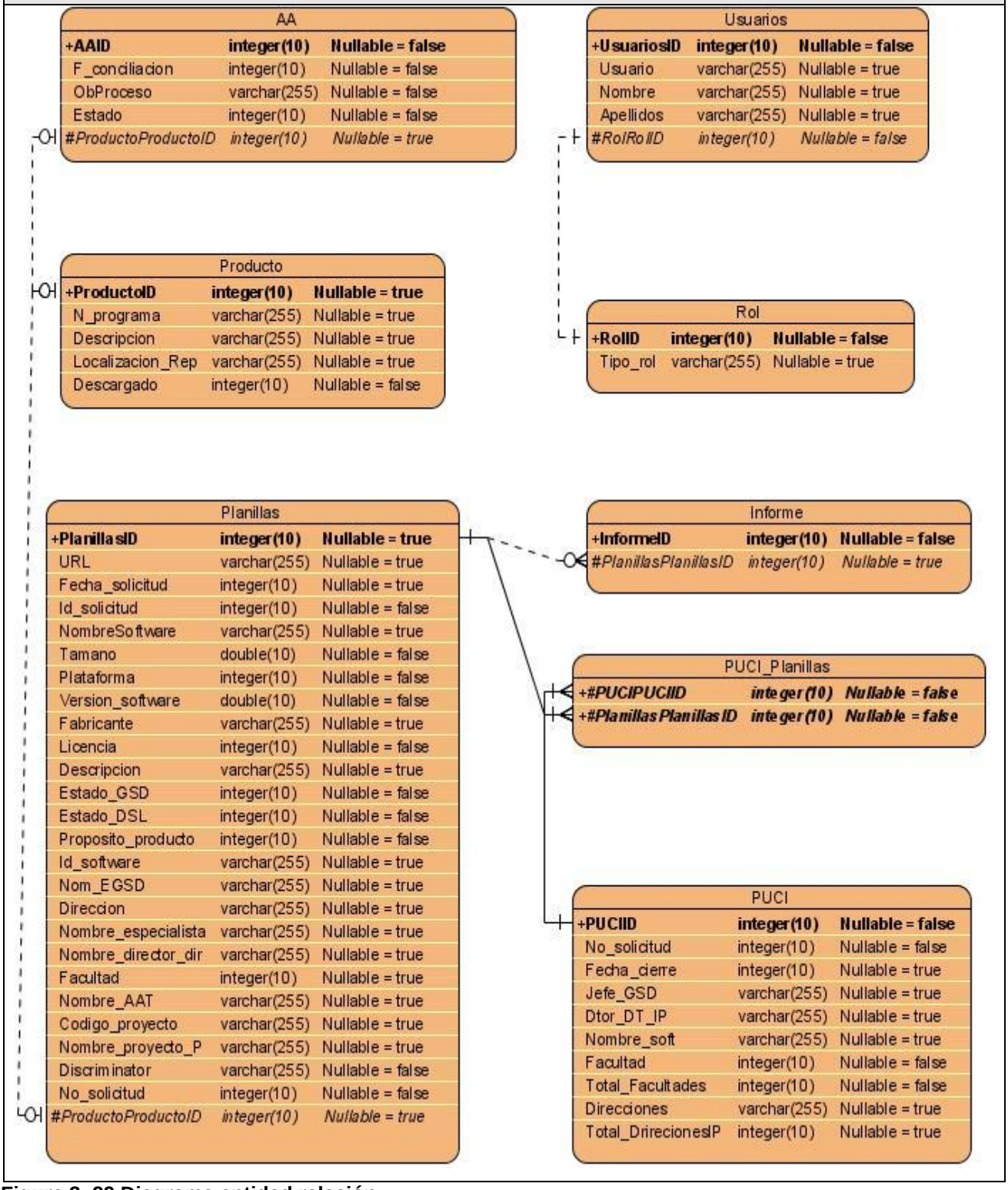

Figura 3. 29 Diagrama entidad relación.

3.11 Diagrama de despliegue.

Diagrama de despliegue.

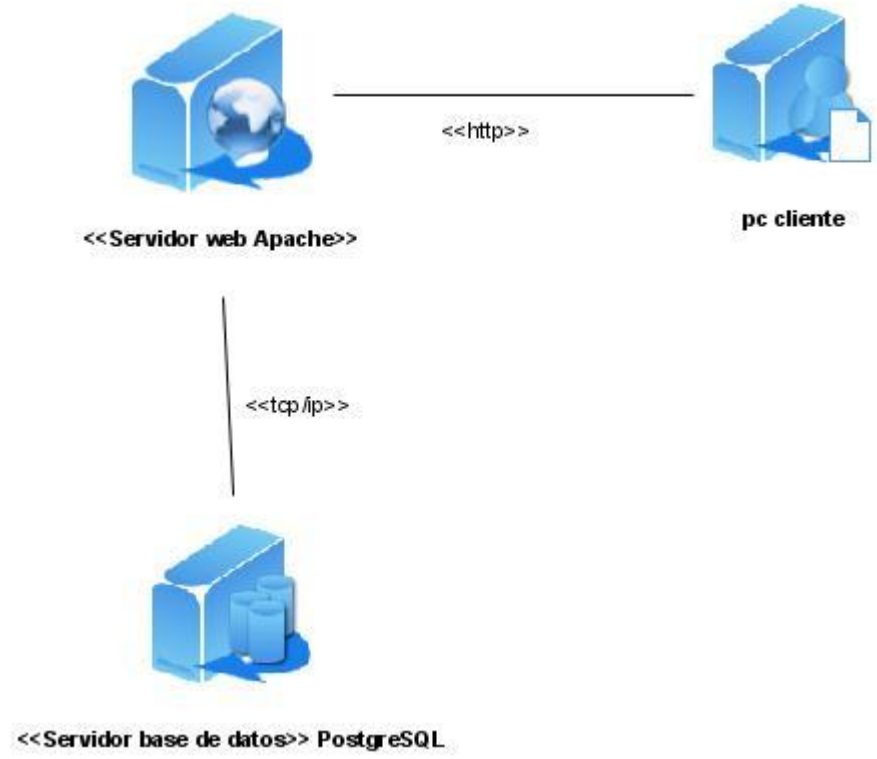

Figura 3. 30 Diagrama de despliegue.

Conclusiones

En este capítulo se desarrollo el modelo de análisis y diseño lográndose una comprensión del sistema a desarrollar, se seleccionó el Symfony como Framework de desarrollo a utilizar en el sistema de gestión de descargas, además obtuvimos los diagramas de clases del análisis y el diseño permitiendo modelar la solución después de comprender los requisitos funcionales y no funcionales.

CAPITULO 4. ESTUDIO DE FACTIBILIDAD.

4.1 Introducción.

Estimar el tiempo de desarrollo, el costo y los beneficios de cualquier obra humana es de vital importancia, nadie puede darse el lujo de realizar algo que no reporte beneficios o que su tiempo de desarrollo y costo sean muy elevados.

Por eso en este capítulo se realiza una estimación del esfuerzo de desarrollo del sistema gestor de descarga utilizándose el Análisis de Puntos de Casos de Uso para un mejor estudio de los beneficios tangibles e intangibles que aportará el sistema gestor de descarga una vez desarrollado**.**

4.1 Análisis de Puntos de Casos de Uso.

La estimación utilizando Análisis de Puntos de Casos de Uso es un método propuesto por Gustav Karner de Objectory y refinado por otros autores. Estima el tiempo de desarrollo de un proyecto asignando pesos a ciertos factores que afectan la realización del mismo. [20] [21]

4.2 Pasos a seguir para la aplicación del método.

El primer paso para la estimación consiste en el cálculo de los Puntos de Casos de Uso sin Ajustar utilizando la siguiente ecuación:

UUCP = UAW + UUCW

Donde:

- **UUCP**: Puntos de Casos de Uso sin Ajustar.
- **UAW**: Factor de Peso de los Actores sin Ajustar.
- **UUCW**: Factor de Peso de los Casos de Uso sin Ajustar.

4.2.1 Factor de Peso de los Actores sin Ajustar (UAW).

Este valor se calcula a través de un análisis de la cantidad de actores presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los actores se establece teniendo en cuenta:

- En primer lugar si se trata de una persona o de otro sistema.
- En segundo lugar, la forma en la que el actor interactúa con el sistema.

Los criterios se muestran en la siguiente tabla y los valores que toman para nuestro caso de estudio:

Tipo de actor	Descripción	Factor de peso	Cant * Peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Application Programming Interface).	1	1*1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2	0*2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3	6*3
Total			19

Tabla 4. 1 Tipos de actores.

UAW = ∑ (Cantidad actor * peso)

UAW = 6*3 + 0*2 + 1*1

UAW = 19

4.2.2 Factor de Peso de los Casos de Uso sin ajustar (UUCW).

Se calcula realizando un análisis de la cantidad de casos de uso presentes en el sistema y la complejidad de cada uno de ellos.

La complejidad de los casos de uso se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo, donde una transacción se entiende como una secuencia de actividades atómica, es decir, se efectúa la secuencia de actividades completa, o no se efectúa ninguna de las actividades de la secuencia.

Para nuestro sistema los valores se muestran en la siguiente tabla:

Tipo de Caso de uso	Descripción	Factor de peso	Cant * Peso
Simple	El Caso de Uso contiene de 1 a 3 transacciones.	5	2*5
Medio	El Caso de Uso contiene de 4 a 7 transacciones.	10	6*10
Complejo	El Caso de Uso contiene más de 8 transacciones.	15	6*15
Total			160

Tabla 4. 2 Complejidad de los CU

UUCW = ∑ (Cant CU * peso)

UUCW = 2*5 + 6*10 + 6*15 = 10 + 60 + 90

UUCW = 160

UUCP = UAW + UUCW

UUCP = 19 + 160

UUCP = 179

4.3 Cálculo de Puntos de Casos de Uso ajustados.

Una vez que se tienen los Puntos de Casos de Uso sin ajustar, se debe ajustar éste valor mediante la siguiente ecuación:

UCP = UUCP x TCF x EF

Donde:

- **UCP**: Puntos de Casos de Uso ajustados.
- **UUCP**: Puntos de Casos de Uso sin ajustar.
- **TCF**: Factor de complejidad técnica.

- **EF**: Factor de ambiente.

4.3.1 Factor de complejidad técnica (TCF).

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante. En la siguiente tabla se muestra el peso de cada uno de éstos factores para el sistema:

Factor	Descripción	Peso	Valor	Comentario	Σ ($Peso_i$ * $Valor_i$)
T1	Sistema distribuído.	2	0	El sistema es centralizado.	0
T2	Objetivos de performance o tiempo de respuesta.	1	1	La velocidad es limitada por las entradas provistas por el usuario	1
T3	Eficiencia del usuario final.	1	1	Escasas restricciones de eficiencia.	1
T4	Procesamiento interno complejo.	1	1	No hay cálculos complejos.	1
T5	El código debe ser reutilizable.	1	1	No se requiere que el código sea reutilizable.	1
T6	Facilidad de instalación.	0.5	1	La facilidad de instalación debe ser la normal.	0.5
T7	Facilidad de uso.	0.5	3	Normal.	1.5
T8	Portabilidad.	2	2	No se requiere que el sistema sea portable.	4
T9	Facilidad de cambio.	1	2	Se requiere un costo moderado de mantenimiento.	2

T10	Concurrencia.	1	0	No hay concurrencia.	0
T11	Incluye objetivos especiales de seguridad.	1	3	Seguridad normal.	3
T12	Provee acceso directo a terceras partes.	1	0	Los usuarios web no tienen acceso directo.	0
T13	Se requieren facilidades especiales de entrenamiento a los usuarios.	1	1	Pocos usuarios internos, sistema fácil de usar.	1
Total					16

Tabla 4. 3 Factores de complejidad técnica del sistema.

El Factor de complejidad técnica se calcula mediante la siguiente ecuación:

$TCF = 0.6 + 0.01 * \Sigma (Peso_i * Valor_i)$

$TCF = 0.6 + 0.01 * 16$

$TCF = 0.76$

4.3.2 Factor de ambiente (EF).

El grupo involucrado en el desarrollo tiene un gran impacto en las estimaciones de tiempo, este tiene habilidades y cierto entrenamiento que afectan el cálculo del Factor de Ambiente. Se trata de un conjunto de factores que se cuantifican con valores de 0 a 5. En la siguiente tabla se muestra el significado y el peso de cada uno de éstos factores para el sistema.

Factor	**Descripción**	**Peso**	**Valor**	**Comentario**	**$\Sigma (Peso_i * Valor_i)$**
E1	Familiaridad con el modelo de proyecto utilizado.	1.5	3	El grupo está bastante familiarizado con el modelo.	4.5
E2	Experiencia en la aplicación.	0.5	4	La mayoría del grupo ha trabajado.	2

E3	Experiencia en orientación a objetos.	1	4	La mayoría del grupo tiene conocimientos en programación a objetos.	4
E4	Capacidad del analista líder.	0.5	3	No se contrató a ningún especialista.	1.5
E5	Motivación	1	5	El grupo está motivado.	5
E6	Estabilidad de los requerimientos.	2	3	Se esperan cambios en el proyecto.	6
E7	Personal part-time.	-1	3	Todo el grupo no trabaja a tiempo completo.	-3
E8	Dificultad del lenguaje de programación.	-1	2	Se usará lenguaje PHP que es lenguaje fácil de usar.	-2
Total					18

Tabla 4. 4 Factor de ambiente

- Para los factores E1 al E4, un valor asignado de 0 significa sin experiencia, 3 experiencia media y 5 amplia experiencia (experto).
- Para el factor E5, 0 significa sin motivación para el proyecto, 3 motivación media y 5 alta motivación.
- Para el factor E6, 0 significa requerimientos extremadamente inestables, 3 estabilidad media y 5 requerimientos estables sin posibilidad de cambios.
- Para el factor E7, 0 significa que no hay personal part-time (es decir todos son full-time), 3 significa mitad y mitad, y 5 significa que todo el personal es part-time (nadie es full-time).
- Para el factor E8, 0 significa que el lenguaje de programación es fácil de usar, 3 medio y 5 que el lenguaje es extremadamente difícil.

El Factor de ambiente se calcula mediante la siguiente ecuación:

$EF = 1.4 - 0.03 * \Sigma (Peso_i * Valor_i)$

$EF = 1.4 - 0.03 * 18$

$EF= 0.86$

Finalmente :

UCP = UUCP * TCF * EF

UCP = 179 * 0.76 * 0.86

UCP = 116.9944

4.4. Cálculo del esfuerzo de implementación-Estimación Final.

E = UCP * CF

Donde:

- **E:** Esfuerzo estimado en horas-hombre
- **UCP:** Puntos de Casos de Uso ajustados
- **CF:** Factor de conversión

CF = 20 horas-hombre→(si Total EF ≤ 2)

CF = 28 horas-hombre→(si Total EF = 3 ó Total EF = 4)

CF = abandonar o cambiar proyecto→(si Total EF ≥ 5)

Total EF = Cant EF < 3 (entre E1 – E6) + Cant EF > 3 (entre E7, E8)

Total EF = 2 + 0

Total EF = 2

Como Total EF = 2 entonces CF = 20 horas-hombres.

E = UCP * CF

E = 116.9944* 20

E = 2339.888

Se debe tener en cuenta que éste método proporciona una estimación del esfuerzo en horas-hombre contemplando sólo el desarrollo de la funcionalidad especificada en los casos de uso.

Finalmente, para una estimación más completa de la duración total del proyecto, hay que agregar a la estimación del esfuerzo obtenida por los Puntos de Casos de Uso, las estimaciones de esfuerzo de las demás actividades relacionadas con el desarrollo de software. Para esto se considera un criterio estadísticamente aceptable que se muestra a continuación conjuntamente con los resultados para nuestro sistema.

Después de realizar los cálculos de forma manual se utilizó el ESTIMAC para comprobar los cálculos y los resultados fueron idénticos.

Actividad	Porcentaje	Horas-Hombre
Análisis	10.00%	584.972
Diseño	20.00%	1169.944
Programación	40.00%	2339.888
Pruebas	15.00%	877.458
Sobrecarga(otras actividades)	15.00%	877.458
Total	100%	5849.72

Tabla 4. 5 Resultados

Suponemos que la infraestructura productiva de la Universidad decide realizar la implementación del sistema, tomando para esto 10 estudiantes que trabajen 25 días al mes 5 horas diarias en aproximadamente 4 meses y 15 días el sistema estaría desarrollado como se puede observar en la Figura 4.1 de acuerdo a cálculos del ESTIMAC.

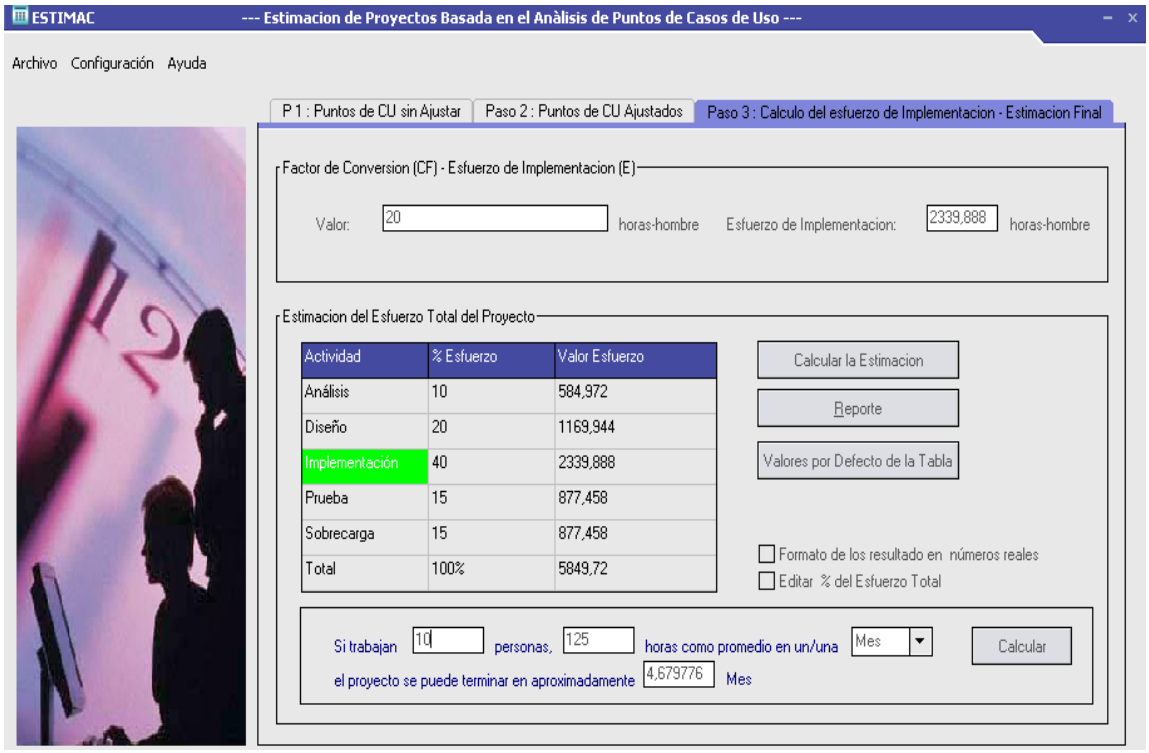

Figura 4. 1 Cálculo ESTIMAC.

4.5 Beneficios tangibles e intangibles.

El sistema gestor de descarga permitirá a los asesores de arquitectura y tecnología de la Universidad o Vicedecano de Producción realizar de una manera correcta, organizada, ágil y segura la solicitud de descarga de software. El servicio del sistema gestor de descarga será de vital importancia para poder brindar la calidad requerida de la manera más rápida posible a cada una de las solicitudes y necesidades existentes en la producción de la Universidad, evitará que se realicen descargas innecesarias de software haciendo un uso racional de la cuota de Internet.

Los beneficios son generalmente intangibles:

- ✓ Disminución del tiempo y esfuerzo que se invierte en esta tarea que se realiza, hasta ahora de forma manual con ayuda del correo electrónico.
- ✓ Disminución de materiales impresos (Planillas) relacionados con los procesos de de solicitud de descarga.
- ✓ Mayor seguridad en de todo el proceso de gestión.
- ✓ Solicitud de descarga de software desde cualquier PC con acceso a la Intranet.
- ✓ Ahorro de materiales de oficina.

4.6 Análisis de costos y beneficios.

Casi todas las tecnologías utilizadas para el desarrollo del sistema son libres por lo que no hay que incurrir en grandes gastos por concepto de licencias, además estos estarían compensados con los beneficios reportados.
Dado el costo y los beneficios que reportará el sistema gestor de descarga se propone que se realice la implementación del mismo.

Conclusiones.

En este capítulo se realizó un análisis de la factibilidad del sistema gestor de descarga utilizando el método de Análisis de Puntos de Caso de Uso llegando a la conclusión de que es factible el desarrollo del sistema para lograr organización, ahorrar materiales y de forma general hacer más fácil el proceso de solicitud de descarga.

Conclusiones generales.

Al culminar el trabajo de diploma análisis y diseño de un sistema para la gestión de descargas de softwares de los proyectos productivos en la UCI concluimos que los objetivos fueron cumplidos. Se realizó un estudio de las principales tecnologías, lenguajes y herramientas que se deben utilizar en la realización del sistema, argumentándose en cada caso el porque de su uso. Muchas de ellas ya fueron utilizadas con éxito en ésta primera etapa.

Utilizando la metodología de desarrollo de software RUP se realizó el análisis y diseño del sistema, se realizó el modelo del negocio lo que permitió una buena comprensión del sistema desarrollar. Además se determinaron los requisitos funcionales y no funcionales para obtener las responsabilidades que tendrá el sistema una vez desarrollado. Se describieron los casos de uso del negocio y los casos de uso del sistema, se diseño el prototipo de interfaz de usuario.

Se diseñaron las clases que deben ser implementadas conjuntamente con la base de datos del sistema.

Finalmente se realizó un estudio de la factibilidad del sistema mostrándonos que es factible la implementación y puesta en marcha del sistema, ya que se gestionaran las descargas que se realizan de Internet.

Recomendaciones.

Cumplido el principal objetivo, la realización del análisis y diseño del sistema de gestión de descarga, se recomienda:

- Realizar la implementación del sistema utilizando las herramientas, lenguaje de programación y metodología propuesta.
- Formular las pruebas que validarán el correcto funcionamiento del sistema.
- Elaborar la documentación de ayuda de usuario.
- Instalar el sistema, asignando algún personal que de soporte y mantenimiento al mismo.

Referencias Bibliográficas.

1. Rosa Mirían E. CubaDebate Contra el terrorismo mediático, [Online] 2007. http://www.cubadebate.cu/index.php?newsid_obj_id=9369&tpl=design/especiales.tpl.html.
2. Alvaro. UptoDown, [Online] 2007. http://flashget.uptodown.com/.
3. Gestiona tus descargas con Flashget, [Online] 2007. http://www.faqoff.org/aprende/internet/flashget-01.htm.
4. Worsley J, Drake J. PostgreSQL Práctico [Online] http://www.sobl.org/traducciones/practical-postgres/practical-postgres.html.
5. PostgreSQL, [Online] 2008. http://es.tldp.org/Postgresql-es/web/navegable/todopostgresql/app-pgadmin.html
6. Axmark D y COL. Principales Características de MySQL [Online] 2008. http://www.dev.mysql.com/doc/refman/5.0/es/features.html.
7. Introducción a la Ingeniería del software, [Online] 2008. http://teleformacion.uci.cu/file.php/42/Clases_Curso_2007-2008/conferencias/Conferencia_1/Profesores/Conferencia_1.pdf.
8. Ferré Grau X, Isabel Sánchez Segura M. Desarrollo Orientado a Objetos con UML. [Online] http://www.itox.mx/Comunidad/Librero/umlTotal.pdf.
9. Guía Breve de CSS, [Online] 2008. http://www.w3c.es/divulgacion/guiasbreves/HojasEstilo.
10. Netcraft Noticias, [Online] 2008. http://news.netcraft.com/archives/web_server_survey.html
11. Boullón Garzón R, Rodríguez Laredo P. Proyecto Final de Carrera 2008, [Online]: http://www.intitec.com/varios/ComparativaSQL.pdf.
12. Zend Studio, [Online] 2008. http://www.desarrolloweb.com/articulos/1178.php.
13. Ciudad Ricardo F, Soto López N. Conferencia Ingeniería del software I, Fase de Inicio. Modelo del Negocio, 2007.
14. Jacobson I, Booch G, Rumbaugh J. El proceso unificado de desarrollo de software. Ciudad de La Habana, Félix Varela, 2004, 438.
15. Técnicas de investigación documental, [Online] 2008. http://html.rincondelvago.com/tecnicas-de-la-investigacion-documental.html.
16. Análisis de los Requerimientos de Información, [Online] 2008. http://148.202.148.5/cursos/cc321/fundamentos/unidad3/tema3_3_1.html.

17. Técnica de recopilación de información, [Online] 2008. http://www.mitecnologico.com/Main/TecnicasDeRecopilacionDeInformacion.
18. Johnson, Ralph E. and Brian Foote. Designing Reusable Classes. Journal of Object-Oriented Programming, 1988.
19. Jacobson I, Booch G, Rumbaugh J. El proceso unificado de desarrollo de software. Ciudad de La Habana, Félix Varela, 2004, 438.
20. Peralta M. ESTIMACIÓN DEL ESFUERZO BASADA EN CASOS DE USO, [Online] 2008 http://teleformacion.uci.cu/file.php/42/Clases_Curso_2007-2008/conferencias/Conferencia_5/Estudiantes/Estimacion_del_esfuerzo_basada_en_casos_de_usos.pdf]
21. Planificación y Estimación de Proyectos, [Online] 2008. http://teleformacion.uci.cu/file.php/42/Clases_Curso_2007-2008/conferencias/Conferencia_5/Estudiantes/Conferencia_Gestion_Proyectos_Estudiantes.pdf

Bibliografía

1. Larman C. UML Y PATRONES, introducción al análisis y diseño orientado a objetos, Tomo 1, Felix Varela, 2004.
2. Pressman R. Ingeniería del Software, Un enfoque práctico, Tomo 2, Felix Varela, La Habana, 2005.
3. Larman C. UML Y PATRONES, introducción al análisis y diseño orientado a objetos, Tomo 2, Felix Varela, 2004.
4. [Online] 19/02/2008, Metodología. http://www.encamina.com/boletines/ENCAMINA%20y%20las%20metodolog%C3%ADas%20software.htm
5. [Online] 26/04/2008, GUTMANS A. and others, PHP 5 Power Programming. http://bibliodoc.uci.cu/pdf/reg04073.pdf
6. Pressman R. Ingeniería del Software, Un enfoque práctico, Tomo 1, Felix Varela, La Habana, 2005.
7. [Online] 8/05/2008, Castro E. HTML con XHTML y CSS, Todo el código para crear sitios web efectivos y originales. http://bibliodoc.uci.cu/pdf/reg02147.pdf
8. [Online] 10/05/2008, RAMOS MONSO M. Sitios web dinámicos e interactivos. http://bibliodoc.uci.cu/pdf/reg03729.pdf
9. TORRES S, J. L. Especificación de Requisitos en Ingeniería de Software, 2008.
10. [Online] 13/05/2008, Schmuller Joseph, Aprendiendo UML en 24 horas. http://bibliodoc.uci.cu/pdf/reg00004.pdf
11. [Online] 05/01/2008 Rozic, Sergio E., Bases de datos y su aplicación con SQL, Buenos Aires, MP Ediciones, 2004. http://bibliodoc.uci.cu/pdf/reg03438.pdf.
12. [Online] 08/01/2008 Crumlish, Christian / Madrid, Anaya, 2003 , 992 p. http://bibliodoc.uci.cu/pdf/reg02104.pdf

Glosario de Términos.

AAT: Asesor de arquitectura y tecnología.

ARCOS o Red ARCOS: (Americas Region Caribbean Optical-ring System**)** Interconecta los Estados Unidos y 18 países en la región Pan Caribe y Centro América, brindando la única red anillada de cable submarino protegido, totalmente redundante**.**

CU: Caso de uso.

DATALAB: Repositorio de software e información de los laboratorios docentes.

DCA: Diagrama de clases del análisis.

DCD: Diagrama de clases del diseño.

Descarga: Este término es utilizado para nombrar de forma única a toda la documentación, software, herramientas, licencias, etc. que se baja de Internet.

DSL: Dirección de Servicios Legales.

EDIP: Especialista de la Dirección de la IP.

EGSD: Especialista del Grupo de Soporte al Desarrollo de la Dirección Técnica.

Gigabytes: Múltiplo del byte: un gigabyte es 1.024 MegaBytes, cerca de 1.000 millones de bytes.

GSD: Grupo de Soporte al Desarrollo de la Dirección Técnica.

IDEs: Entornos de desarrollo integrados.

Internet: Red de ordenadores a nivel mundial. Ofrece distintos servicios, como el envío y recepción de correo electrónico (e-mail), la posibilidad de ver información en las páginas web, de participar en foros de discusión (News), de enviar y recibir ficheros mediante FTP, de charlar en tiempo real mediante IRC, etc.

IP: Infraestructura Productiva.

Megabytes (mb): Múltiplo del byte: un megabyte son 1.024 Kilobytes, cerca de un millón de bytes.

Megabytes/seg: Velocidad de transferencia de información a través de una red.

ORDMS: Sistema manejador de bases de datos objeto relacional.

PF: Planilla de solicitud de descarga de software de la facultad.

PIP: Planilla de solicitud de descarga de software de la Dirección de la IP.

PP: Planilla de solicitud de descarga de software del proyecto productivo.

PUCI: Planilla de solicitud de descarga de software de la UCI.

RAM: Siglas de random access memory, un tipo de memoria de ordenador a la que se puede acceder aleatoriamente; es decir, se puede acceder a cualquier byte de memoria sin acceder a los bytes precedentes.

Seg: Unidad de tiempo en el Sistema Internacional de Unidades.

UCI: Universidad de las Ciencias Informáticas.

UCISTORE: Repositorio de software e información de la UCI.