Universidad de las Ciencias Informáticas

Facultad 7

Trabajo de diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Título: Generación de actualizaciones de software desde sistemas de control de versiones

Autores: Idayana Bastarreche Calistre

Luismel Del Valle Román

Tutor: MSc. Yasel Couce Sardiñas

Co Tutor: Ing. Beatriz Fernández Carmenate

La Habana, junio 2011

"Año 53 de la Revolución"

Declaración de Autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Idayana Bastarreche Calistre	Luismel Del Valle Román
Firma del Autor	**Firma del Autor**
Ing. Beatriz Fernández Carmenate	MSc. Yasel Couce Sardiñas
Firma del Tutor	**Firma del Tutor**

Datos de Contacto

Tutor:

- **MSc. Yasel Couce Sardiñas** (yaselc@uci.cu): Graduado de Licenciado en Ciencias de la Computación en el año 2005 en la Universidad Central de Las Villas. Posee categoría docente de Profesor Asistente. Ha impartido asignaturas de Sistemas Operativos y Seguridad Informática en la Facultad 7 desde el curso 2005 – 2006. Actualmente se desempeña como responsable del departamento de Sistemas de Apoyo a la Salud.

Co tutor:

- **Ing. Beatriz Fernández Carmenate** (bcarmenate@uci.cu): Graduado de Ingeniero en Ciencias Informáticas, en el año 2010, en la Universidad de las Ciencias Informáticas. Se encuentra trabajando como especialista de la calidad en el proyecto productivo Colaboración Médica.

Agradecimientos

Idayana

A toda mi familia por apoyarme y darme fuerzas para seguir adelante.

A mi papá Ibrahin y mis abuelos Zenaida e Ibrahin por todo el amor que me han dado y la confianza que han depositado en mí.

A mi hermano Iroel por estar siempre a mi lado, en los momentos buenos y malos, y por velar que yo tenga un futuro mejor.

A mi abuela Berta, mis tías María y Ori, mis tíos Osmany, Pavel y Víctor por toda la ayuda que me han dado para llegar hasta aquí.

A mi primita Indy por tomarme de ejemplo y hacer que me esfuerce cada día para ser mejor.

A todas mis amistades que de una forma u otra me han ayudado a lo largo de toda la carrera, en especial a mi amiga Aismé por estar siempre cuando la he necesitado y brindarme su amistad.

A mi compañero de tesis Luismel por haber trabajado sin descanso para que este sueño se convirtiera en una realidad.

A mis tutores Yasel y Beatriz por guiarnos en todo el transcurso de la tesis y confiar en nosotros desde el primer momento.

A mis suegros Raquel y Mojena y mi cuñado Javier por todo el aprecio que me han dado y acogerme en su familia como un miembro más.

A mi novio José por haber estado a mi lado durante los últimos 5 años, dándome ánimo cuando estaba triste, celebrando los momentos alegres, ayudándome en todo lo posible y apoyándome en todo momento.

A mi mamita querida, que aunque no se encuentre físicamente hoy aquí, ha sido la razón por la cual he salido adelante y sé que donde sea que esté, está muy contenta y orgullosa de su niña.

Luismel

Quiero agradecerles a las personas que han contribuido a este logro y que han puesto su grano de arena en el camino que se ha construido a lo largo de estos 5 años.

A mis compañeros de la universidad algunos conocidos desde primer año, algunos que ya no están pero que los seguimos recordando, con ellos he podido compartir todos los momentos de la vida en esta escuela, a Mojena, Raysel, Brian, Arrebato, Franklin, Vladimir, Renán, Guillermo, René, Melquiades, José Carlos, Alejandro y Javier.

A mis amigos de pinar Ussiel, José Carlos, Juan Carlos por siempre estar ahí en cualquier circunstancia.

Al profe Alexis que me ha ayudado a formarme no sólo como profesional sino también como persona, mostrándome en cada caso la decisión correcta a tomar.

A mis tutores Beatriz y Yasel un especial agradecimiento por modelar de una forma tan perfecta el resultado de toda una carrera que se refleja en este trabajo de diploma.

A mi toda mi familia, a mis tíos Enrique, Tato, Odalys y Olivia que se han preocupado por el de cursar de mi carrera en todo momento, a mis abuelos paternos Fela Y Guiyo que han sabido darme consejos importantes .A mi padrastro Osvaldo que se ha convertido en un amigo para mi. A mi hermano Osvaldito por hacerme recordar las cosas de la infancia que fácilmente olvidamos. A mi papá Luis que me ha enseñado a ser un persona responsable y preocupada. A mi tío Miguel que se ha convertido en un padre para mi apoyándome en los momentos más

difíciles, a mis dos madres Carmen y Julita que siempre han sabido estar ahí para mí y no tengo como agradecerle lo que han hecho por mí desde el momento que vi la luz por primera vez.

Por último y el agradecimiento principal y a quien va dedicada esta tesis a mi abuelo Miguel que aunque ya no está con nosotros ha sido la persona más importante en formación como persona y siempre me hizo saber que los más importante era mi formación con un buen profesional y revolucionario, para él todos mis agradecimientos.

Dedicatoria

Idayana

Dedico esta tesis a dos personas muy importantes en mi vida. A mi novio José por haber estado a mi lado durante toda la carrera y haberme hecho más fácil el camino para llegar hasta aquí y a mi mami Dora, que a pesar de no poder estar presente físicamente en es esta etapa de mi vida, ha sido mi inspiración para seguir adelante y sé que hoy hubiera sido un día muy especial para ella y que donde quiera que esté se siente muy orgullosa de mí.

Luismel

A mi papá Luis, a mi tío Miguel, a mi mamá Carmen, a mi abuela Julia y a mi abuelo Miguel que son las personas más importantes en mi vida va mi homenaje dedicándole este trabajo donde ellos también tiene parte de autoría.

Resumen

El objetivo de la presente investigación es integrar un diseñador de actualizaciones automáticas al sistema de control de versiones Subversion para facilitar el proceso de creación de las actualizaciones de los productos que se desarrollan en el Centro de Informática Médica y disminuir los errores que podrían introducirse en la realización de estos paquetes. Se diseñó una aplicación en ambiente de escritorio, se utilizó un sistema de control de versiones para crear el paquete de actualización y un repositorio, al cual se accede a través del protocolo FTP para publicar todos los recursos de actualización creados.

El sistema obtenido está soportado sobre tecnología .NET, específicamente sobre la versión 2.0 de este marco de trabajo y como lenguaje de programación C-Sharp. Para la implementación del mismo se utilizó como IDE de desarrollo el Visual Studio 2008 y RUP como metodología de desarrollo. La aplicación permite conectarse a un controlador de versiones para seleccionar los cambios que se van a incluir en la nueva actualización. Además brinda la posibilidad de crear tareas independientemente del tipo de plataforma a la cual va destinada. La integración del controlador de versiones con el diseñador de actualizaciones, le garantiza al proceso de empaquetamiento mayor gestión, agilidad y disminuye los errores por pérdida de información.

Palabras claves: *actualizaciones automáticas, control de versiones*.

Índice

Introducción

En la actualidad, la informática se ha convertido en una parte muy importante de la mayoría de los procesos que se desarrollan en el mundo, lo que implica la necesidad de preparar a las nuevas generaciones para la asimilación y utilización de dicha tecnología.

Una sociedad que aplique la informatización en todas sus esferas y procesos será más eficaz, eficiente y competitiva. Es evidente que para los países subdesarrollados resulta un reto el logro de este propósito, ya que su problemática fundamental está en lograr la supervivencia de sus pueblos. (1)

Con el desarrollo de la informática y las telecomunicaciones ha habido un gran auge en la industria del software, ya que la mayoría de los sectores de la sociedad se están informatizando y necesitan que todos los procesos sean automatizados.

Cuba, a pesar de contar con pocos recursos, no está ajena a este desarrollo tecnológico, ya se han comenzado a dar los primeros pasos para informatizar las principales áreas de la sociedad y una de ellas es el sector de la salud, debido a la estructura que presenta y el nivel de información que se maneja en el mismo.

En la Universidad de las Ciencias Informáticas, el Centro de Informática Médica (CESIM), es el encargado de desarrollar productos, servicios y soluciones informáticas para la optimización del trabajo y mejoramiento de la calidad de la atención médica.

Debido a la necesidad de mantener estos productos actualizados y funcionando correctamente, surge la idea de desarrollar una aplicación para diseñar los paquetes de actualización de forma automática y evitar así los errores que traería consigo la creación manual de estos paquetes.

Actualmente el Centro de Informática Médica cuenta con un diseñador de actualizaciones, pero que requiere la creación explícita de todas las acciones que se deben ejecutar en la actualización, lo cual se convierte en un trabajo largo y engorroso si el paquete que se desea conformar está compuesto por un número considerable de archivos.

Otro aspecto a tener en cuenta es que las tareas que se deben ejecutar cuando se va a actualizar un producto no están separadas por el tipo de plataforma a la que va destinada la actualización, lo cual podría introducir errores en el empaquetamiento.

Es importante señalar que no se puede registrar la estructura del árbol de directorios de la aplicación en el perfil del producto, lo que trae como consecuencia que se introduzcan errores en las direcciones de los archivos que contiene la actualización debido a que este proceso se realiza de forma manual.

Dada la situación anterior, el **problema** a resolver consiste en ¿cómo minimizar la introducción de errores en el proceso de diseño de las actualizaciones automáticas?

El **objeto de estudio** se centra en el proceso de gestión de la información de las aplicaciones informáticas y el **campo de acción** se enmarca en el proceso de gestión de la información de los diseñadores de actualizaciones automáticas.

Para dar solución al problema planteado se propone como **objetivo general** del trabajo: Integrar el Diseñador de Actualizaciones Automáticas a un controlador de versiones.

Para dar cumplimiento a este objetivo y resolver la problemática de este trabajo fueron establecidas las siguientes tareas:

- ✓ Realizar el análisis crítico del estado del arte de los exploradores de sistemas de control de versiones, para garantizar un producto acorde a las tendencias actuales.
- ✓ Analizar la primera versión del Diseñador de Actualizaciones Automáticas detectando las deficiencias que presenta, facilitando la ubicación organizada de la versión de un producto.
- ✓ Realizar el expediente de proyecto correspondiente a los flujos de trabajo: Requerimientos, Análisis y Diseño e Implementación, propuesto por la metodología de Desarrollo RUP, tomando como base el expediente realizado en la primera versión del sistema.
- ✓ Proponer modificaciones para el esquema de tareas, de forma que permita que se independicen las tareas de la plataforma.

- ✓ Diseñar la propuesta de solución, de manera que se garantice el uso de buenas prácticas en la implementación de la aplicación.
- ✓ Implementar la solución de manera que disminuyan los errores en el diseño del empaquetado y la pérdida de información.

Una vez finalizada la investigación, la solución desarrollada garantizará:

- Un proceso de diseño de actualizaciones automáticas más intuitivo y eficiente.
- Agilidad en el proceso de diseño de las actualizaciones automáticas.
- Mayor confiabilidad al proceso de soporte y mantenimiento de aplicaciones informáticas.

El presente trabajo consta de cuatro capítulos estructurados de la siguiente forma:

Capítulo 1 Fundamentación Teórica: Argumenta el estado del arte del tema a tratar a nivel internacional, nacional y de la universidad e incluye una explicación de las técnicas, tecnologías, metodologías y software empleados en la investigación para darle solución al problema. Se tratan los conceptos fundamentales para una correcta comprensión del tema.

Capítulo 2 Características del sistema: Define las características del sistema y se enmarca en el objeto de estudio. Se hace un análisis del dominio de la aplicación, se describen los procesos a automatizar para darle solución al problema y se generan documentos referentes a esta fase. También se definen requisitos no funcionales y un prototipo de interfaz externa.

Capítulo 3 Diseño del sistema: Se realiza el diseño del sistema donde se define el diagrama de clases del diseño por casos de uso así como la relación existente entre ellas. Posterior a esto se muestra la interacción entre los actores y el sistema mediante los diagramas de secuencia. Además se especifica la seguridad y el diseño de la interfaz de la aplicación.

Capítulo 4 Implementación: Presenta el modelo de implementación del Diseñador de Actualizaciones Automáticas, compuesto por el diagrama de componentes y el modelo de despliegue. Además se definen los estándares de codificación que se siguieron para la implementación.

Capítulo 1: Fundamentación Teórica

En este capítulo se hace referencia a una serie de conceptos que son de vital importancia para la comprensión y desarrollo de la investigación. Se realiza un análisis del estado de los diseñadores de actualizaciones automáticas y controladores de versiones tanto en el ámbito nacional como internacional y en la Universidad, además de exponer las principales características de las herramientas, tecnologías y metodologías utilizadas durante el desarrollo del sistema.

1.1 Conceptos asociados al dominio del problema

Para una mejor comprensión de los temas que se abordarán en el documento, se mostrarán una serie de conceptos identificados durante la investigación.

Una **versión**, revisión o edición de un producto, es el estado en el que se encuentra dicho producto en un momento dado de su desarrollo o modificación.

Un **sistema de control de versiones** (o sistema de control de revisiones) es una combinación de tecnologías y prácticas para seguir y controlar los cambios realizados en los ficheros del proyecto, en particular en el código fuente, en la documentación y en las páginas web. (2)

Subversion es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como **svn** por ser ese el nombre de la herramienta utilizada en la línea de órdenes. (3)

1.2 Antecedentes del sistema

Después de realizar una investigación y búsqueda de los sistemas informáticos que están integrados a controladores de versiones, se puede llegar a la conclusión de que no existe ningún diseñador de actualizaciones integrado a una herramienta de este tipo. Se encontraron aplicaciones en el ámbito internacional que sirven de referencia para comprender cómo se realiza la integración de un sistema de control de versiones a una aplicación informática. Un ejemplo de ello son los sistemas de gestión de

proyectos, los cuales tienen relación con los controladores de versiones para mantener los cambios realizados tanto en la documentación como en la implementación de los productos.

Sistemas existentes en el ámbito internacional.

Redmine

Redmine es una herramienta para la gestión de proyectos y el seguimiento de errores escrita usando el framework Ruby on Rails. Es software libre y de código abierto, disponible bajo la Licencia Pública General de GNU v2. (4)

Entre las principales características de Redmine se encuentran:

- Gestión de múltiples proyectos simultáneamente.

Permite gestionar múltiples proyectos desde una sola interfaz con una ventana de navegador. La navegación es muy sencilla y se puede saltar y cambiar de proyecto en cualquier momento. Además cada proyecto puede tener una configuración totalmente diferente y el usuario tener un rol distinto en cada uno. Los proyectos puedes definirse como privados, en los que el administrador debe dar acceso a cada miembro, o públicos, visibles para todo el mundo.

- Sistema de seguimiento flexible de temas relacionados.

Una de las mecánicas más útiles para el desarrollo de un proyecto en Redmine son las peticiones y su visualización. Estas peticiones se dividen en 3 tipos iniciales (errores, tareas y soporte) y pueden asignarse a un miembro del proyecto. Se puede indicar una fecha de inicio y fin para esa petición, e incluso llevar un control del tiempo y porcentaje realizado. También se le puede asignar una prioridad, enlazar con la subida de un fichero, y encajar en una categoría (que se pueden definir tantas como se quieran). Con todos estos datos, pueden visualizarse las peticiones de manera personalizada estableciendo filtros, y servir así de informes de tareas o incidencias. Además dentro de un proyecto pueden establecerse versiones y asignar tareas a determinadas versiones, así conforme se marquen tareas completadas, las versiones irán completando su porcentaje automáticamente.

- Uso de calendarios y diagrama de GANT.

Redmine incluye un calendario para visualizar todas las peticiones a lo largo de un mes elegido, marcando claramente el día de inicio y de fin de cada petición. Igualmente ocurre con la vista en diagrama de Gantt, que va marcando el porcentaje completado conforme avanzan los días. Las peticiones que se visualizan en ambos casos están sujetas a los filtros definidos por el usuario.

- Tiene integración SCM (Subversion, CVS, Git, Mercurial, Bazaar y Darcs).

Redmine puede integrarse con un repositorio de código (Subversion, Git, CVS, entre otros) que esté montado en la misma máquina, tan solo hay que indicarle el directorio local. La aplicación sirve así de interfaz web para el seguimiento del desarrollo de un proyecto. Pueden descargarse los ficheros, ver el historial, los cambios, e incluso descargar un archivo a modo de parche para aplicar a código desactualizado. Es un sistema de seguimiento de versiones, aunque no pueden actualizarse los ficheros directamente.

- Permite el envío de notificaciones y la creación de tareas vía correo electrónico.

Configurando previamente el servidor de correo SMTP, Redmine permite enviar notificaciones por correo electrónico en todos los proyectos, definiendo antes los eventos que activan estos avisos. Además cada usuario en su configuración puede elegir recibir notificaciones de cualquier evento, o solo las relacionadas con él (por ejemplo uno de los campos de las peticiones son las personas en seguimiento). Puede configurarse además el servidor de correo entrante, permitiendo así actualizar peticiones simplemente por email e incluso crear nuevas peticiones. (5)

Otras funcionalidades que incorpora son las siguientes:

- Soporta autenticación vía múltiples LDAP.
- Control de acceso basado en roles flexibles.
- Maneja archivos, documentos y noticias.
- Permite la creación de un wiki por proyecto.
- Permite la creación de foros por proyecto.

Trac

Trac es una herramienta para la gestión de proyectos y el seguimiento de errores escrita en Python, inspirado en CVSTrac. Su nombre original era svntrac, debido a su fuerte dependencia de Subversion. (6)

Trac es un sistema que integra un conjunto de componentes con capacidades suficientes para la gestión de proyectos de desarrollo de software.

Permite llevar una serie de utilidades propias para un proyecto, entre las cuales se encuentran:

- ✓ **Wiki:** Empleado para documentar cualquier aspecto del proyecto de modo colaborativo y sin necesidad de herramientas especiales.
- ✓ **Planificación (Roadmap):** Sistema para definir y visualizar el estado de los hitos de un proyecto (un hito incluye una descripción y una fecha y se usa como atributo de los tickets, que se asocian a hitos concretos).
- ✓ **Manejo de eventos (Timeline):** Sistema de seguimiento de eventos en el sistema:
    - Histórico de cambios en el wiki.
    - En el sistema de control de versiones.
    - En el sistema de gestión de incidencias o vencimiento de un hito.
- ✓ **Búsquedas:** Permite localizar páginas del wiki, comentarios dentro de los conjuntos de cambios o tickets en los que aparece una palabra.
- ✓ **Visor de Código:** Integrado con algún sistema de control de versiones asociado al proyecto, lo cual permite ver los cambios que se han producido en el programa de una forma visual (estado actual del repositorio, los cambios que se han ido produciendo, comparar distintas versiones de ficheros en línea).

Trac ha sido concebido de forma modular donde se pueden añadir plugins que proporcionan distintas funcionalidades. Entre sus características adicionales se encuentran:

- **Administración:** Personalización de entorno, manejo de usuarios, permisos y plugins.
- **Autenticación:** LDAP, BBDD o fichero.
- **Uso de VCS:** Subversion, Bazaar, GIT, Mercurial o Monotone.

Alfresco

Alfresco es la principal alternativa de código abierto para la gestión de contenidos empresariales (ECM, por sus siglas en inglés). La compañía integra la innovación del código abierto con la estabilidad de una verdadera plataforma de clase empresarial. (7)

El modelo de código abierto permite a Alfresco utilizar la mejor clase de tecnologías de código abierto y las contribuciones de la comunidad con el fin de obtener un software de alta calidad, producido con mayor rapidez y con un coste mucho menor.

Entre sus principales características se encuentran:

- Gestión de documentos.
- Gestión de contenido web (incluyendo aplicaciones web y virtualización de sesiones).
- Versionado a nivel de repositorio (similar a Subversion).
- Superposición transparente (similar a UnionFS).
- Gestión de registros.
- Gestión de imágenes.
- Acceso al repositorio vía CIFS/SMB, FTP y WebDAV.
- Soporte de varios idiomas.
- Empaquetamiento de aplicación portable.
- Soporte multiplataforma (oficialmente Windows, GNU/Linux y Solaris).
- Interfaz gráfica basada en navegadores de Internet (oficialmente Internet Explorer y Mozilla Firefox).

- Integración de escritorio con Microsoft Office y OpenOffice.Org.

Alfresco presenta varios beneficios:

- Ahorro presupuestario significativo en cuotas de mantenimiento de ECM propietario.
- Financiación de suscripción Alfresco incluida en los gastos de operación (OPEX).
- Mantenimiento constante del funcionamiento del negocio para los usuarios.

1.3 Tecnologías, lenguajes, metodologías y protocolos utilizados

1.3.1 Lenguaje de programación

En el mundo del desarrollo del software muchos programadores usan lenguajes de programación de alto nivel y orientados a objetos; la decisión del uso de uno u otro para el desarrollo de un software determinado está en las librerías que estos utilizan, en las cuales radica la verdadera riqueza del lenguaje. La elección final del lenguaje también dependerá del posible conocimiento que se tenga de la sintaxis del mismo y de su adaptación al medio para el que se quiere programar, ya que es diferente programar para red local, para Windows, Mac o Linux. A continuación se presentarán algunas características del lenguaje de programación seleccionado para trabajar.

Lenguaje C#

C# es un lenguaje de programación diseñado por Microsoft en 2001 como parte de su plataforma .NET. Combina el lenguaje de bajo nivel de C y la velocidad de la programación de alto nivel de Visual Basic. (8)

C# es un lenguaje orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, es simple pero eficaz y está diseñado para escribir aplicaciones empresariales. Presenta considerables mejoras e innovaciones en áreas como seguridad de tipos, control de versiones, eventos y recolección de elementos no utilizados (liberación de memoria).

Ventajas:

- ✓ Compila a código intermedio independiente del lenguaje en que haya sido escrita la aplicación e independiente de la máquina donde vaya a ejecutarse.

- ✓ Realiza la recolección automática de basura.
- ✓ Posee capacidades de reflexión.
- ✓ Es flexible en cuanto al orden de definición de las clases y las funciones.
- ✓ Soporta definición de clases dentro de otras.
- ✓ Todos los valores son inicializados antes de ser usados (automáticamente por defecto, o manualmente desde constructores estáticos).

XML (Extensible Markup Language)

Es un lenguaje de etiquetado extensible muy simple, pero estricto, que juega un papel fundamental en el intercambio de una gran variedad de datos. Sirve para estructurar, almacenar e intercambiar información. Es la base de los servicios Web, el contenido almacenado en un documento XML se puede transferir fácilmente a través de la red. Los servicios Web XML actúan de forma independiente y además permiten que las aplicaciones compartan información e invoquen funciones de otras aplicaciones independientemente del sistema operativo o la plataforma en que se ejecutan y los dispositivos utilizados para obtener acceso a ellos. (9)

XML es una tecnología que permite estructurar documentos (metalenguaje) para darle significado a su contenido. Esto se logra mediante el uso de etiquetas tipo HTML, pero definidas especialmente para el dominio de una aplicación en particular. El XML es un lenguaje en el cual cada desarrollador puede definir su propio conjunto de etiquetas y su principal objetivo, a diferencia de HTML, no es el despliegue del documento, sino la estructura y el significado de su contenido. (10)

XSchema XML

XSchema es un lenguaje cuyo objetivo principal es definir la estructura en bloques de un documento XML. El propósito de un esquema es definir y describir una clase de documentos XML usando estas construcciones para restringir y documentar el significado, uso y relaciones de las partes constituidas: tipo

de datos, elementos y su contenido, atributos y sus valores, entidades y su contenido, y anotaciones. Los esquemas documentan su propio significado, uso y función. (11)

XSchema dispone de una amplia gama de tipos básicos predefinidos para la definición de atributos y elementos. Estos tipos básicos están compuestos en su mayoría por tipos simples, aunque también se puede encontrar un tipo lista y un tipo unión. Además se pueden construir tipos de datos propios, extendiéndolos de los existentes o creando estructuras nuevas.

1.3.2 Lenguaje de modelado

UML (Unified Model Language)

El lenguaje para modelamiento unificado (UML), es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. (12)

UML no es un método de desarrollo, lo que significa que no sirve para determinar qué hacer o cómo diseñar el sistema, sino que simplemente ayuda a visualizar el diseño y a hacerlo más accesible para otros. Está controlado por el grupo de administración de objetos (OMG) y es el estándar de descripción de esquemas de software. Mediante UML es posible establecer una serie de requerimientos y estructuras necesarias para plasmar un sistema de software antes de iniciar el proceso de implementación del mismo.

1.3.3 Metodología de desarrollo

RUP (Rational Unified Process)

RUP es una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas probadas y una arquitectura configurable. (13) Es un proceso de desarrollo de software, es decir, un conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto.

RUP divide el proceso en cuatro fases: Inicio, Elaboración, Construcción y Transición. Además, brinda una serie de flujos de trabajos que son realizados durante las fases anteriormente mencionadas y que

garantizan el orden en que deben ser ejecutadas todas las tareas que establece esta metodología, ellos son: Modelo de negocio, Requerimiento, Análisis y diseño, Implementación, Prueba, Instalación, Ambiente, Administración de proyecto, Administración de configuración y cambios. Utiliza un Lenguaje Unificado de Modelado para preparar todos los esquemas de un sistema software.

RUP tres características fundamentales que marcan el desarrollo y adaptación de la metodología en un proyecto:

- Dirigido por casos: Posibilita que se defina la estructura fundamental del sistema y se utiliza para capturar los requisitos funcionales y definir los contenidos de las iteraciones.
- Centrado en la arquitectura: Incluye los aspectos estáticos y dinámicos más significativos del sistema, brindando la característica de que el desarrollo se vea marcado por una serie de construcciones organizadas por lo casos de uso, lo que posibilita una mejor organización y avance en el desarrollo.
- Iterativo e incremental: Ofrece la posibilidad de establecer las construcciones del sistema por partes y que a medida que sean terminadas dichas partes se vayan obteniendo incrementos en el desarrollo del software.

1.3.4 Herramientas utilizadas

Basado en los lineamientos de arquitectura del área temática SAS, para dar solución al problema planteado, se decide utilizar el Enterprise Architect 7.1 como herramienta de modelado. La implementación de todos los componentes y del servicio de actualización se realizará sobre la tecnología .Net de Microsoft, con el Microsoft .Net Framework y como entorno de desarrollo, el Microsoft Visual Studio .Net 2008. Se debe destacar que al no existir otra tecnología implicada, además del .Net Framework de Microsoft, también es posible usar estos componentes e incluso desarrollar extensiones para el proceso de actualización.

Visual Studio 2008

Visual Studio es un completo conjunto de herramientas para la creación tanto de aplicaciones de escritorio como de aplicaciones web. Aparte de generar aplicaciones de escritorio de alto rendimiento, se pueden

utilizar las eficaces herramientas de desarrollo basado en componentes y otras tecnologías de Visual Studio para simplificar el diseño, desarrollo e implementación en equipo de soluciones empresariales. (14)

Es un entorno de desarrollo integrado (IDE), que habilita el uso compartido de herramientas y hace más sencilla la creación de soluciones en varios lenguajes como por ejemplo: Visual Basic, Visual C# y Visual C++.

Enterprise Architect 7.1

Enterprise Architect 7.1 es una herramienta de construcción y modelado de software de alto rendimiento basado en el estándar de UML 2.1. Con una trazabilidad completa desde los requisitos iniciales hasta las decisiones de diseño de software, EA 7.1 provee el tipo de visualización y colaboración eficiente y robusta requerida en los entornos de desarrollo de software que actualmente son altamente demandantes. (15)

Algunas de las características claves de Enterprise Architect son:

- Generar código para el software que está construyendo.
- Crear elementos del modelo UML para un amplio alcance de objetivos.
- Ubicar esos elementos en diagramas y paquetes.
- Crear conectores entre elementos.
- Documentar los elementos que ha creado.
- Realizar ingeniería reversa del código existente en varios lenguajes.
- Importación/Exportación XMI 2.1.

Usando EA, puede realizar ingeniería directa y reversa de código C++, C#, Delphi, Java, Python, PHP, VB.NET y clases de Visual Basic, sincronizar códigos y elementos del modelo, diseñar y generar elementos de base de datos.

Enterprise Architect sustenta todos los diagramas y modelos UML. Puede modelar procesos de negocio, sitios web, interfaces de usuario, redes, configuraciones de hardware, mensajes y más. Estimar el tamaño

de su proyecto en esfuerzo de trabajo en horas. Capturar y trazar requisitos, recursos, planes de prueba, solicitudes de cambio y defectos. Desde los conceptos iníciales hasta el mantenimiento y soporte, Enterprise Architect tiene las características que precisa para diseñar y administrar su desarrollo e implementación.

Altova XMLSpy 2008 Professional Edition

Altova XMLSpy 2008Professional Edition es el editor XML líder de la industria y el ambiente de desarrollo de XML, que ofrece vistas de edición intuitivas y potentes utilidades XML para modelar, editar, transformar, y depurar las tecnologías relacionadas con XML de forma rápida y sencilla. (16)

XMLSpy 2008Professional Edition incluye un editor gráfico de esquema XML que le permite diseñar y documentar esquemas complejos con facilidad. También incluye un procesador de XSLT esquema y un depurador XSLT 1.0/2.0 para el perfeccionamiento de hojas de estilo, además de soporte para hojas de estilo que usan Java, C #, JavaScript y VBScript.

XMLSpy Professional Edition 2008 es compatible con las principales bases de datos relacionales, incluyendo IBM DB2, Microsoft SQL Server, Access, Oracle, MySQL, Sybase, entre otros. Puede conectarse y consultar una base de datos relacional, generar un esquema XML a partir de una base de datos, importación y exportación de datos basados en esquemas de bases de datos, generar bases de datos a partir de esquemas XML, y editar datos relacionales o XML.

1.3.5 Plataforma .NET

La plataforma .NET es un conjunto de tecnologías diseñadas para transformar Internet en una plataforma informática distribuida a escala completa. Proporciona nuevas formas de desarrollar aplicaciones a partir de colecciones de Servicios Web. La plataforma .NET soporta totalmente la infraestructura existente de Internet, incluyendo HTTP, XML y SOAP. (17)

La plataforma .NET proporciona:

- Un modelo de programación coherente e independiente del lenguaje para todas las capas o niveles de una aplicación.

- Una interoperabilidad transparente entre tecnologías.
- Una fácil migración desde tecnologías existentes.
- Un completo soporte de tecnologías de Internet independientes de la plataforma y basadas en estándares, incluyendo Hypertext Transfer Protocol (HTTP), Extensible Markup Language (XML) y Simple Object Access Protocol (SOAP).

Está diseñado para utilizar los servicios web con XML como mecanismo principal de comunicación entre aplicaciones. Posee avanzadas funciones en tiempo de ejecución lo que permite que cualquier aplicación pueda ser convertida en servicios web XML. Permite escribir programas en cualquiera de los lenguajes soportados por la plataforma, incluso utilizar simultáneamente varios lenguajes en un mismo programa. Entre los lenguajes que se han adherido a esta familia se encuentra C#, Microsoft Visual Basic, C++, Java, Pascal, entre otros conformando un grupo de más de veinte.

Framework 3.5

Microsoft .NET Framework 3.5 combina la eficacia de las API de .NET Framework 2.0 y 3.0 con nuevas tecnologías para crear aplicaciones que ofrecen interfaces de usuario atractivas, protegen la información de identidad personal de los clientes, permiten una comunicación segura y sin problemas y proporcionan la capacidad de modelar diversos procesos de negocio.

Microsoft .NET Framework 3.5 contiene una serie de características nuevas en distintas áreas tecnológicas que se han agregado como nuevos ensamblados para evitar cambios destacados. Algunas de estas características son:

- Integración total de LINQ (Language Integrated Query) y del reconocimiento de los datos. Esta nueva característica le permitirá escribir código en idiomas habilitados para LINQ para filtrar, enumerar y crear proyecciones de varios tipos de datos SQL, colecciones, XML y conjuntos de datos usando la misma sintaxis.
- ASP.NET AJAX le permite crear experiencias web más eficaces, más interactivas y con un gran índice de personalización que funcionan con los exploradores más usados.

- Compatibilidad total con las herramientas de Visual Studio 2008 para WF, WCF y WPF, incluida la nueva tecnología de servicios habilitados para flujos de trabajo.

1.3.6 Arquitectura del sistema

Modelo Vista Controlador es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

- El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.
- La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario.
- El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

Esta arquitectura separa la lógica de negocio (el modelo) y la presentación (la vista), por lo que se consigue un mantenimiento más sencillo de las aplicaciones.

En este capítulo se realizó un análisis de las ventajas de integrar un controlador de versiones a cualquier aplicación informática y se profundizó en las características que presentan los sistemas integrados a controladores de versiones que se utilizan a nivel internacional, para que sirvan de guía en el desarrollo de esta investigación. Se seleccionó dentro de un grupo de tecnologías, herramientas y metodologías las adecuadas para realizar el software que se requiere: C# como lenguaje de programación, Framework .NET de Microsoft, Enterprise Architect como herramienta CASE y RUP como metodología de desarrollo.

Capítulo 2: Características del Sistema

Este capítulo tiene como objetivo la descripción de las características del sistema así como la propuesta de solución para la integración del Diseñador de Actualizaciones Automáticas al controlador de versiones Subversion. Debido a la poca claridad de los procesos del negocio se realiza un Modelo de Dominio, además se hace un análisis de los requerimientos funcionales y no funcionales para lograr un mejor entendimiento del sistema.

2.1 Descripción del Sistema

Teniendo en cuenta las funcionalidades que brinda, el Diseñador de Actualizaciones Automáticas ha sido dividido en 3 partes fundamentales:

- La funcionalidad “Diseñador de Información” es el encargado de diseñar las actualizaciones de forma automática, permitiendo visualizar el estado de dicho proceso.
- La funcionalidad “Publicador de Información” es la encargada de la interacción con el responsable de la actualización, permitiéndole visualizar el estado del proceso general a través de una interfaz y de mantener publicadas las nuevas versiones existentes de los productos.
- La funcionalidad “Control de versiones” es la encargada de llevar la constancia de todas las versiones de los productos que se van publicando en el repositorio, así como los cambios que se le van realizando a las mismas.

2.2 Modelo de Dominio

Un Modelo de Dominio es una representación visual de clases conceptuales o de objetos reales en un dominio de interés. Representa clases conceptuales del dominio del problema, lo cual puede ser una idea o un objeto físico (símbolo, definición y extensión). (18)

El Modelo de Dominio es representado en UML con un Diagrama de Clases donde se muestran:

- Conceptos u objetos del dominio del problema: clases conceptuales.

- Asociaciones entre clases conceptuales.
- Atributos de las clases conceptuales.

El Modelo de Dominio se realiza si no se logran determinar los procesos de negocio con fronteras bien definidas, es decir, si no se puede ver claramente quiénes son las personas que realizan cada proceso de negocio, quiénes son los beneficiados con cada uno de estos procesos, pero además quiénes son las personas que desarrollan las actividades en cada uno de estos procesos.

El Diseñador de Actualizaciones Automáticas forma parte de un sistema de actualizaciones que no responde a las exigencias de un cliente, si no que se desarrolla basándose en las experiencias adquiridas por el hecho de las necesidades existentes en el Centro de Informática Médica.

2.2.1 Conceptos fundamentales del dominio

PA: La clase PA representa al Proveedor de Actualización que es el único encargado de supervisar todo el proceso de creación de los paquetes de actualizaciones y realizar las diferentes funcionalidades.

Recursos de Versión: Una clase recursos de versión no es más que aquella que tendrá el control de todas las versiones existentes, paralelo al desarrollo de una actualización.

Manifiesto: La clase manifiesto contiene un documento XML con el listado de la información de todos los recursos que el actualizador necesita para actualizar la aplicación con todas las acciones que este debe realizar y los pasos que llevará a cabo.

Paquete de Actualización: La clase paquete de actualización es la que contiene los ficheros y un documento XML asociado a una nueva versión, empaquetados y compactados en un fichero ZIP de la aplicación que se desea actualizar. Para cada uno de los paquetes de actualización de entrada existen reglas de validación.

Directorio: El directorio de archivos es el ordenador que se fije como repositorio, el mismo contendrá las versiones que se encuentran en una carpeta creada para guardar las mismas, en este directorio se realizarán las salvas continuas de las versiones.

Controlador de Versiones: El controlador de versiones es el repositorio que va a contener todos los cambios que se le han realizado a un determinado producto desde una versión específica.

2.2.2 Diagrama de Modelo de Dominio

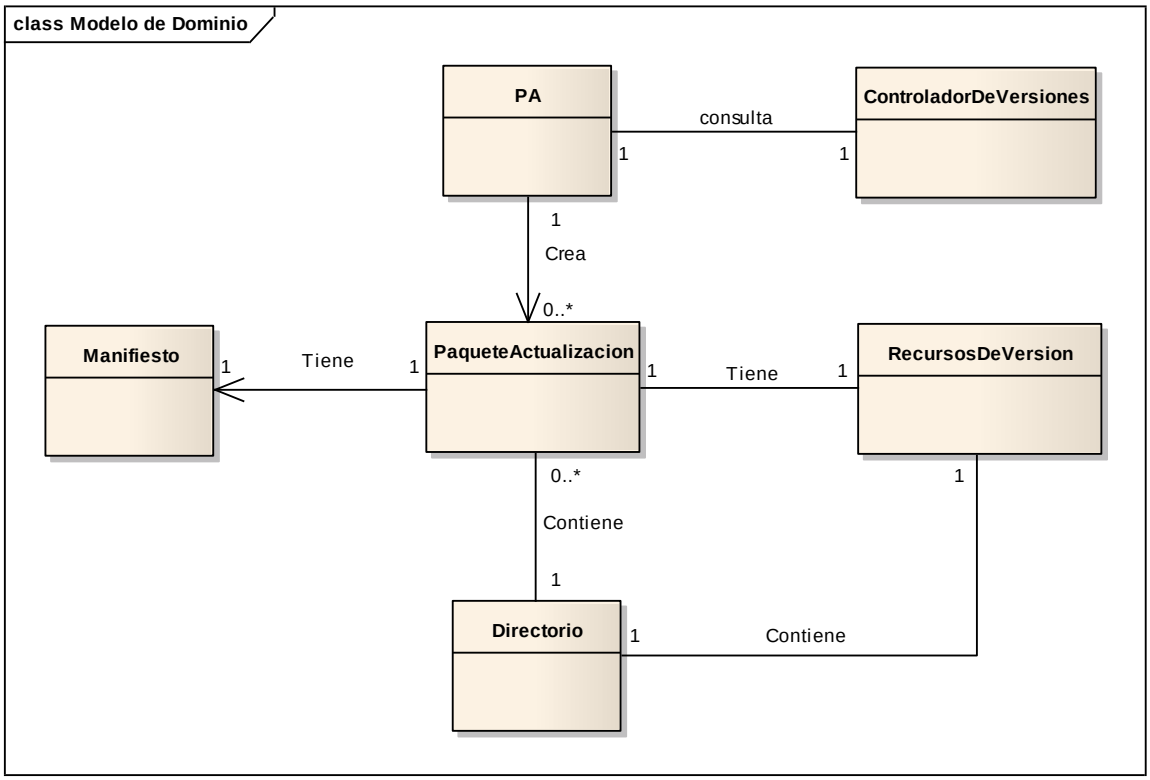


Fig. 1 Diagrama de Modelo de Dominio

2.3 Especificación de los requisitos de software

Los requisitos de software son condiciones o capacidades que debe cumplir un sistema para lograr que el mismo funcione adecuadamente. La obtención de los requisitos es de gran importancia debido a que es el hilo conductor de todo desarrollo de software. Una obtención de requisitos con calidad demuestra que el trabajo realizado culminará con éxito.

2.3.1 Requerimientos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Se mantienen invariables sin importar con qué propiedades o cualidades se relacionen.

A continuación se muestra un listado con los requisitos funcionales definidos para el sistema:

Requisitos Funcionales	
RF1 Crear actualización	RF7 Descargar archivos
RF2 Insertar producto	RF8 Ignorar cambios
RF3 Crear árbol de directorios	RF9 Crear perfil
RF4 Actualizar árbol de directorios	RF10 Eliminar Producto
RF5 Autenticar en el SVN	FR11 Listar publicación
RF6 Listar cambios desde el SVN	

Tabla 1 Requisitos Funcionales

2.3.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto usable, atractivo, confiable o rápido. Son fundamentales para lograr el éxito del producto y normalmente están vinculados a requerimientos funcionales.

Usabilidad

RNF1 El sistema podrá ser utilizado por un Proveedor de Actualizaciones, que es el único encargado del diseño y publicación de las actualizaciones.

RNF2 El sistema debe garantizar un acceso fácil y rápido, podrá ser usado por usuarios con conocimientos básicos de informática.

Eficiencia

RNF3 El Diseñador de Actualizaciones deberá ser rápido ante la solicitud de desarrollar la nueva versión del producto que se desea actualizar, el tiempo de respuesta para cada solicitud debe ser el mínimo posible.

Soporte

RNF4 Una vez terminada la integración del diseñador al controlador de versiones se realizarán las pruebas del mismo, capacitación del personal responsable de la aplicación y mantenimiento de software.

Restricciones de Diseño

RNF5 Para el desarrollo se utilizará C# como leguaje de programación.

RNF6 Se utilizará Visual Studio 2008 como entorno de desarrollo.

RNF7 Para realizar el modelado del software se utilizará la herramienta Enterprise Architect 7.1.

Requisitos para la documentación de usuarios en línea y ayuda del sistema.

RNF8 Se dispondrá de la documentación del sistema realizada con la metodología de desarrollo RUP y de un Manual de Usuario que indicará como interactuar con las funcionalidades del Diseñador de Actualización.

Interfaces de usuario

RNF9 El sistema debe tener una interfaz sencilla, agradable, legible y de fácil uso para el usuario. Debe permitir la ejecución de acciones de manera rápida.

RNF10 El contenido será mostrado de manera comprensible y fácil de leer.

Interfaces Hardware

RNF11 Requerimientos mínimos Ordenador Pentium IV o superior, 256 MB de memoria RAM o superior, capacidad en disco duro de 3 GB, copia de seguridad, velocidad del procesador de 512 MHz o superior.

Interfaces Software

RNF12 Garantizar que el diseñador se ejecute sobre Sistema Operativo Windows XP Profesional Service Pack 2 o superior.

Estándares aplicables

RNF13 Para la integración del diseñador de actualizaciones al controlador de versiones se deberá seguir los estándares de codificación y las pautas para la documentación, todos definidos por el Área Temática Sistemas de Apoyo a la Salud.

RNF14 Para las descripciones de casos de uso, y mensajes que debe emitir el sistema, se deben seguir las pautas de análisis definidas en el área temática.

2.4 Modelado del sistema

El modelo del sistema describe cuáles son las personas o grupos de personas, instituciones o sistemas relacionados ya existentes, que interactúan con los procesos que el sistema deberá realizar. Así como la relación que los mismos guardan entre sí.

2.4.1 Actores del sistema

A continuación se define y describen los actores que interactuarán con el sistema.

Actor	Descripción
PA	El PA (Proveedor de Actualizaciones) es el encargado de realizar todas las operaciones pertinentes a la creación y publicación de las actualizaciones.

Tabla 2 Actores del sistema

2.4.2 Diagrama de casos de uso del sistema

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. (19)

Un Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso del sistema. Representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa.

A continuación se muestra el diagrama de casos de uso del sistema para el Diseñador de Actualizaciones Automáticas, diferenciándose de color amarillo los casos de uso que se implementaron en la versión anterior y que serán modificados durante la implementación.

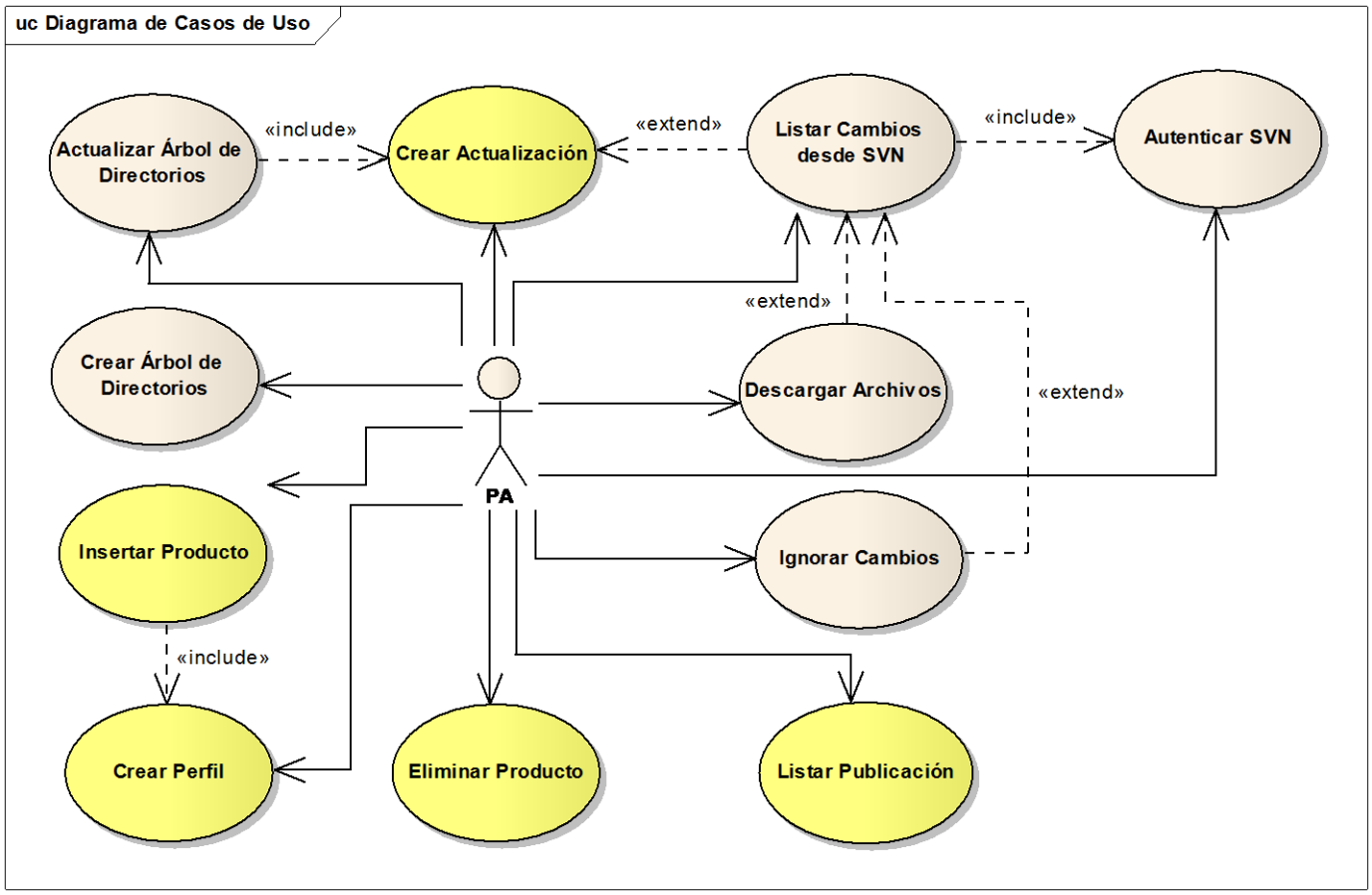


Fig. 2 Diagrama de casos de uso del sistema

2.4.3 Especificación de los casos de uso

Descripción del caso de uso: Insertar Producto

Objetivo	El objetivo que se persigue con este CU es gestionar el producto al que se le desea realizar la actualización.
Actores	Proveedor de actualizaciones: (Inicia) Inserta el producto al diseñador.
Resumen	El caso de uso inicia cuando el PA procede a Insertar un producto para realizar la nueva versión del producto, el sistema brinda la posibilidad de eliminar el producto si este no será utilizado, el caso de uso finaliza cuando el PA Inserta el producto.
Complejidad	Media
Prioridad	Crítico
Precondiciones	
Poscondiciones	Se creó un producto.

Descripción del caso de uso: Crear Árbol de Directorios

Objetivo	El objetivo que se persigue con este CU es crear el árbol de directorios de un producto para un mejor control de la dirección exacta donde se guardará la actualización y evitar la introducción de errores a dicha dirección.
Actores	Proveedor de actualizaciones: (Inicia) Crea el árbol de directorios.
Resumen	El caso de uso inicia cuando el PA accede a la opción de crear la nueva versión del producto y necesita adicionar las acciones que se deben ejecutar, paralelo a esto se va creando automáticamente la estructura del árbol de directorio que se requiere para la actualización.
Complejidad	Media
Prioridad	Crítico
Precondiciones	El producto ha sido insertado al diseñador.
Poscondiciones	Se creó la ruta para la publicación de la actualización.

Descripción del caso de uso: Listar cambios desde SVN

Objetivo	El objetivo de este caso de uso es dar la posibilidad de crear la nueva actualización de un producto mediante la descarga de la lista de cambios que se encuentre publicada con los detalles de la nueva versión que se desea crear.
Actores	Proveedor de actualizaciones: (Inicia) Lista los cambios.
Resumen	El caso de uso inicia cuando el PA accede a la opción de listar los cambios que se desean hacer para la nueva versión del producto, el sistema brinda la posibilidad de descargar los cambios o ignorar los mismos, si se selecciona descargar cambios estos se incluirán en la creación de la nueva actualización, si se ignoran la actualización será creada de forma manual, el caso de uso finaliza cuando el PA crea la actualización del producto.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	Existe el producto y se ha creado su perfil.
Poscondiciones	Se ha creado la actualización del producto.

En este capítulo se identificaron los principales conceptos del Diseñador de Actualizaciones Automáticas, se mostró además el modelo de dominio y la descripción de cada una de las clases que intervienen en el mismo. Se presentó un listado de los requerimientos funcionales y no funcionales que indican las capacidades o condiciones que el sistema debe cumplir y las propiedades o cualidades que el producto debe tener. Por último se conformó el diagrama de casos de uso del sistema, describiendo los actores que interactúan con el mismo, así como los casos de uso más significativos para el desarrollo de la aplicación.

Capítulo 3: Análisis y Diseño del Sistema

En este capítulo se realiza el análisis y diseño del sistema, donde se llevará a cabo todo el flujo de los procesos mediante diferentes diagramas. A través de los diagramas de interacción del sistema, se describe gráficamente la interacción entre los actores y la aplicación, quedando reflejados los mensajes que se transmiten entre los objetos. Además se describen las clases de diseño. Se plantea la seguridad en el sistema y la forma en que está concebida que se despliegue la aplicación.

3.1 Análisis del Sistema

En esta etapa se analizan los requisitos que fueron obtenidos con anterioridad, refinándolos y estructurándolos, con el objetivo de conseguir una comprensión más precisa de los mismos y una descripción que sea fácil de mantener y que ayude a estructurar todo el sistema. Se analizan soluciones alternativas y se asignan a diferentes elementos del software.

3.1.1 Diagramas de clases del análisis

Un Diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa el funcionamiento del mundo real, no de la implementación automatizada del mismo. (20)

El diagrama de clases del análisis describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos.

A continuación se presentan los diagramas de clases del análisis para cada caso de uso.

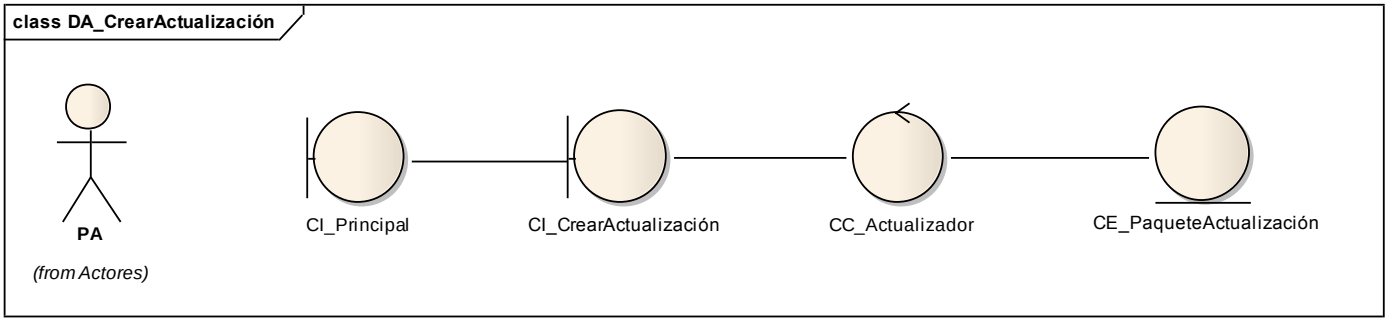


Fig. 3 Diagrama de clases del análisis del CUS Crear actualización

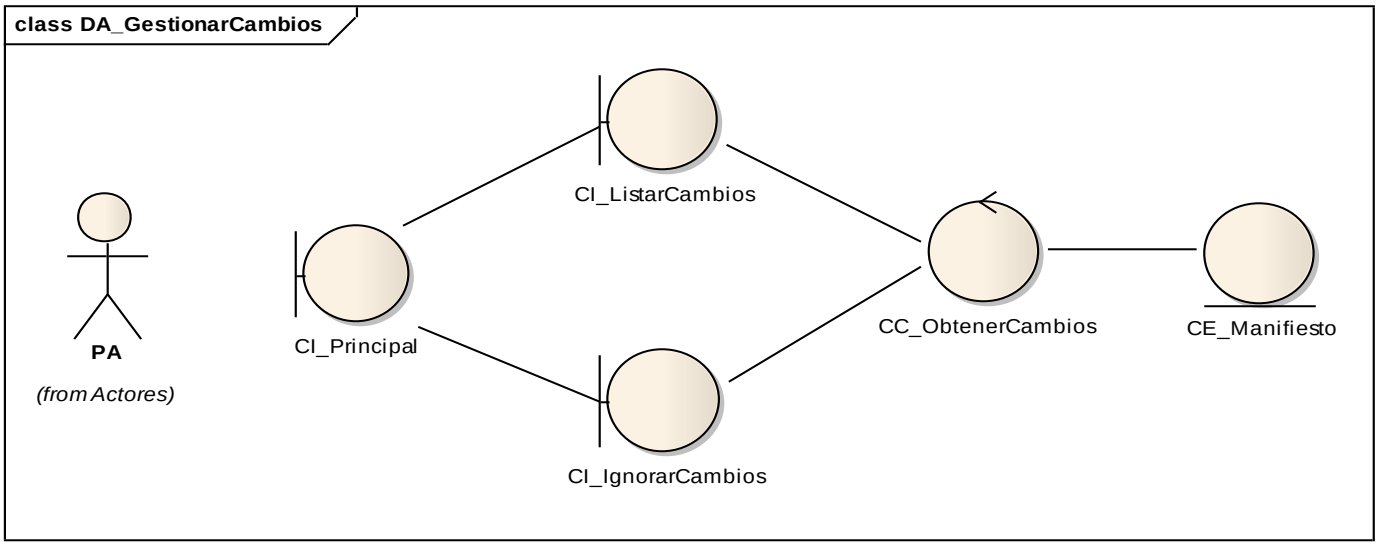


Fig. 4 Diagrama de clases del análisis del CUS Gestionar cambios

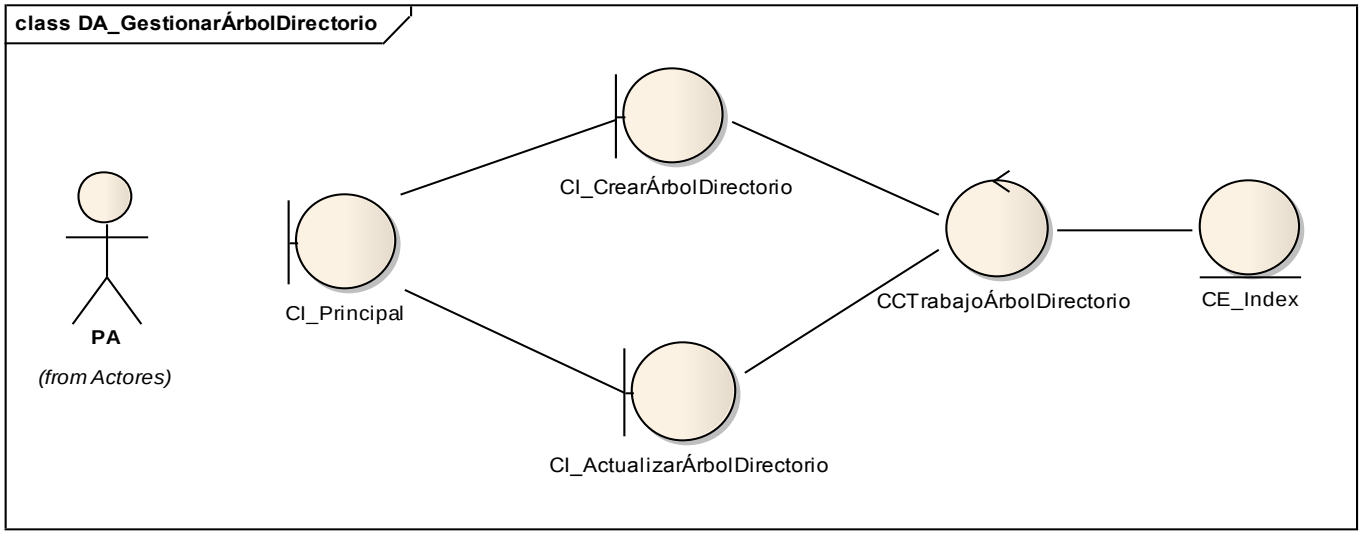


Fig. 5 Diagrama de clases del análisis del CUS Gestionar árbol de directorio

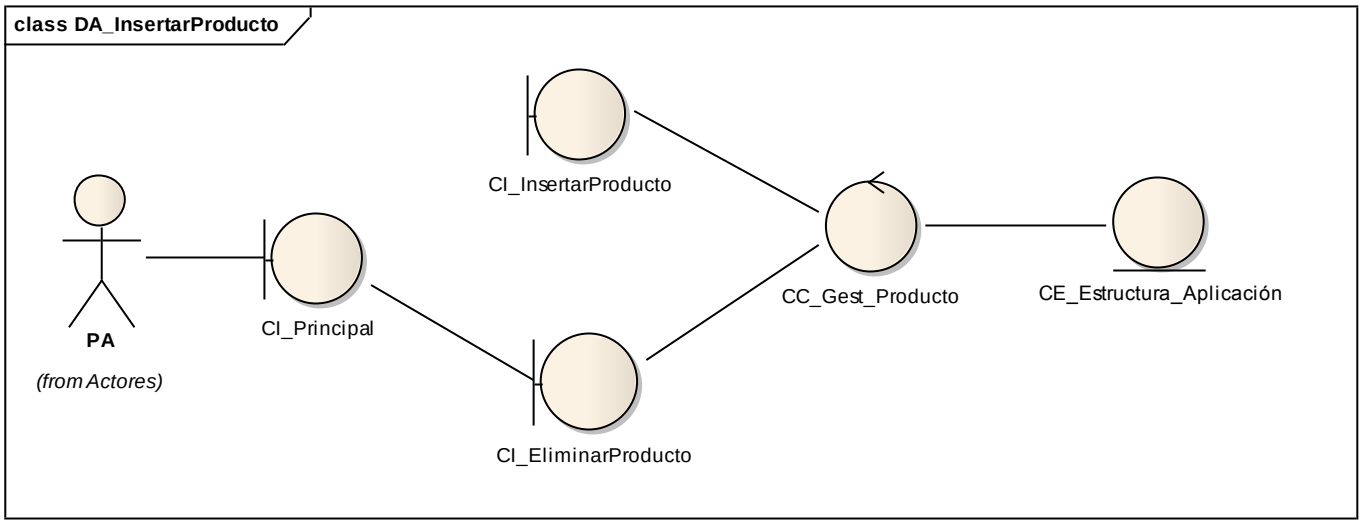


Fig. 6 Diagrama de clases del análisis del CUS Insertar producto

3.2 Estructura del Diseño

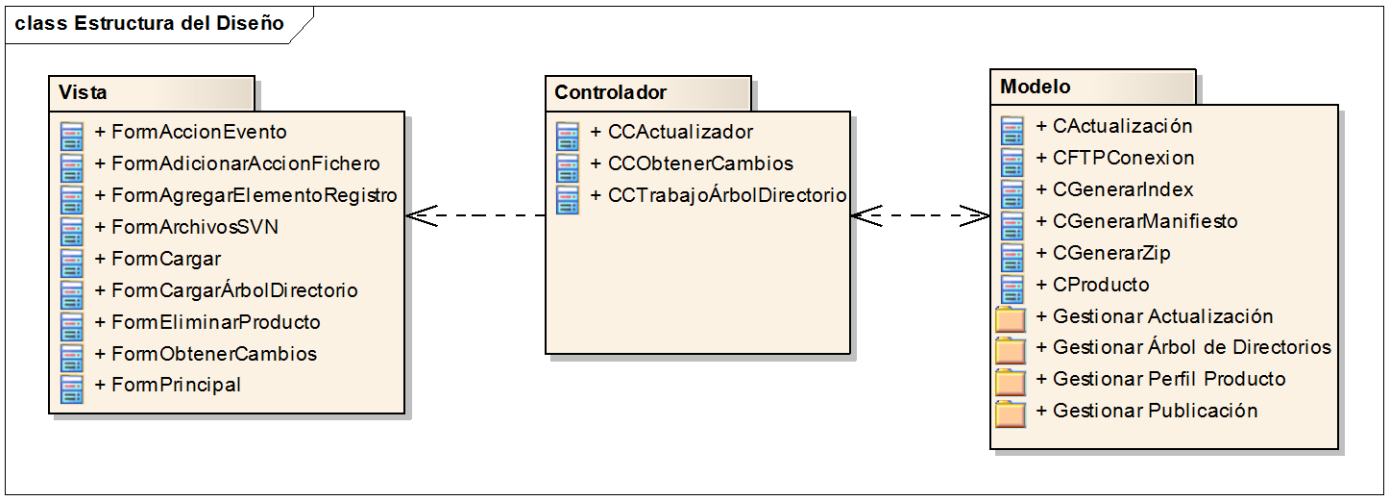


Fig. 7 Estructura del Diseño

Vista

La vista contiene los formularios con los cuales va a interactuar el Proveedor de Actualizaciones. También es conocida como interfaz gráfica y debe tener la característica de ser amigable para el usuario. Estas clases permiten al Proveedor de Actualizaciones insertar y visualizar datos que gestionados por las clases controladoras son finalmente reflejados en las clases modelos encargadas del acceso a ellos.

Modelo

En el modelo se encuentra la lógica de negocio y la capa de acceso a datos. En la capa lógica de negocio es donde se reciben las peticiones del Proveedor de Actualizaciones y se envían las respuestas tras el proceso. Se denomina capa de negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta se comunica con la vista a través de las clases controladoras, para recibir las solicitudes y presentar los resultados. En la capa de acceso a datos se encuentran las clases mediante las cuales el usuario puede guardar y obtener la información que necesite para el funcionamiento de la aplicación.

Controlador

En el controlador se encuentran las clases controladoras que son las encargadas de realizar las funcionalidades del Diseñador de Actualizaciones Automáticas. Estas clases tienen la misión de relacionar la vista y el modelo, es decir poseen funciones que permiten que los datos ingresados por el Proveedor de Actualizaciones sean procesados y finalmente sean guardados o mostrados utilizando las funciones de acceso a datos que presentan las clases modelos.

3.3 Diseño

En esta etapa es cuando se traducen los requerimientos funcionales y no funcionales en una representación de software. El diseño es el primer paso en la fase de desarrollo de cualquier producto o sistema de ingeniería y su objetivo es producir un modelo o representación de una entidad que se va a construir posteriormente.

3.3.1 Diagrama de clases del diseño por casos de uso

Los diagramas de clases del diseño describen gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Contienen las clases, atributos, métodos, navegabilidad y dependencias existentes entre ellas.

A continuación se representan los diagramas de clase del diseño por casos de uso del sistema.

CUS Crear actualización

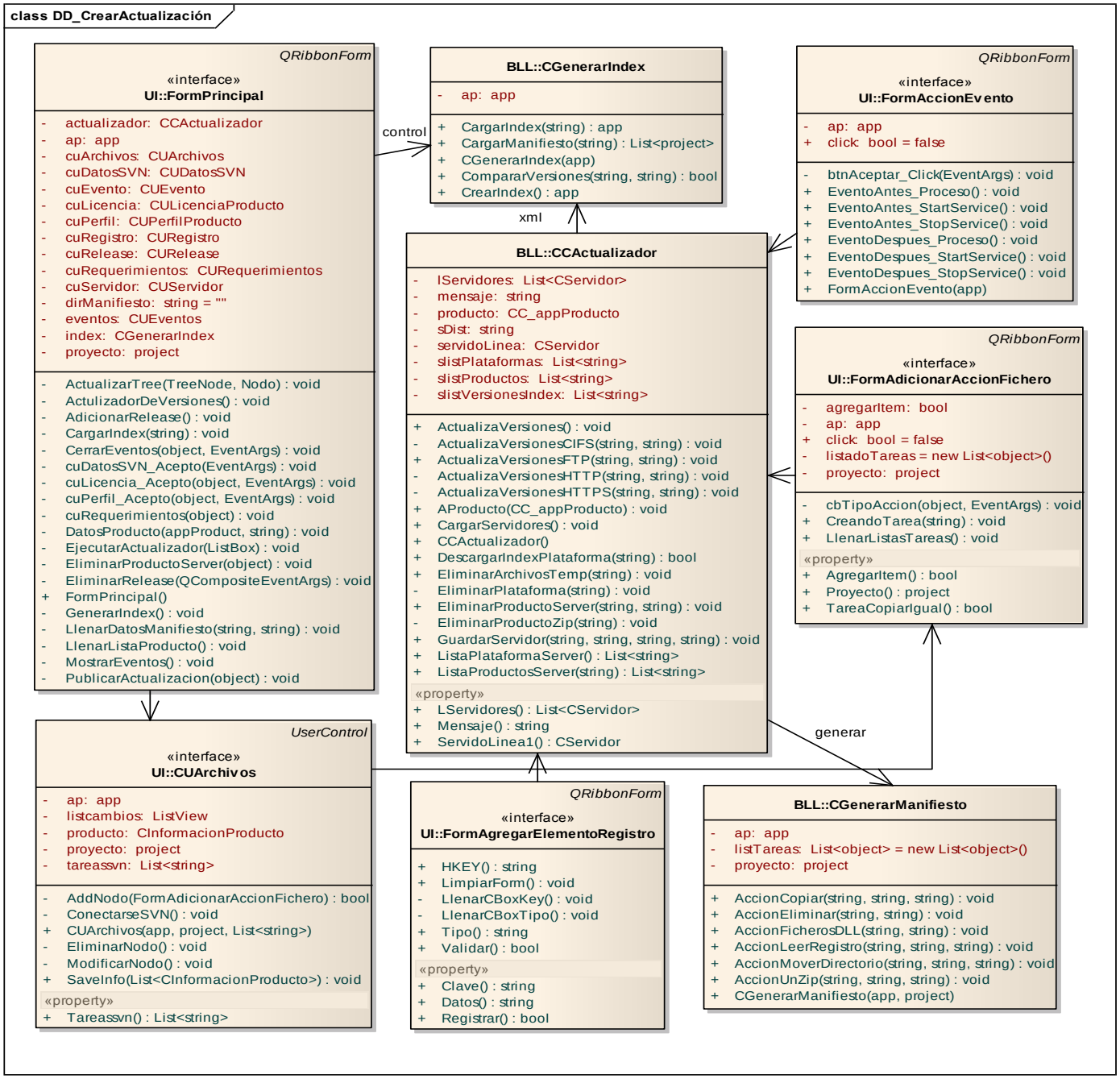


Fig. 8 Diagrama de clases del diseño del CUS Crear actualización

CUS Gestionar cambios

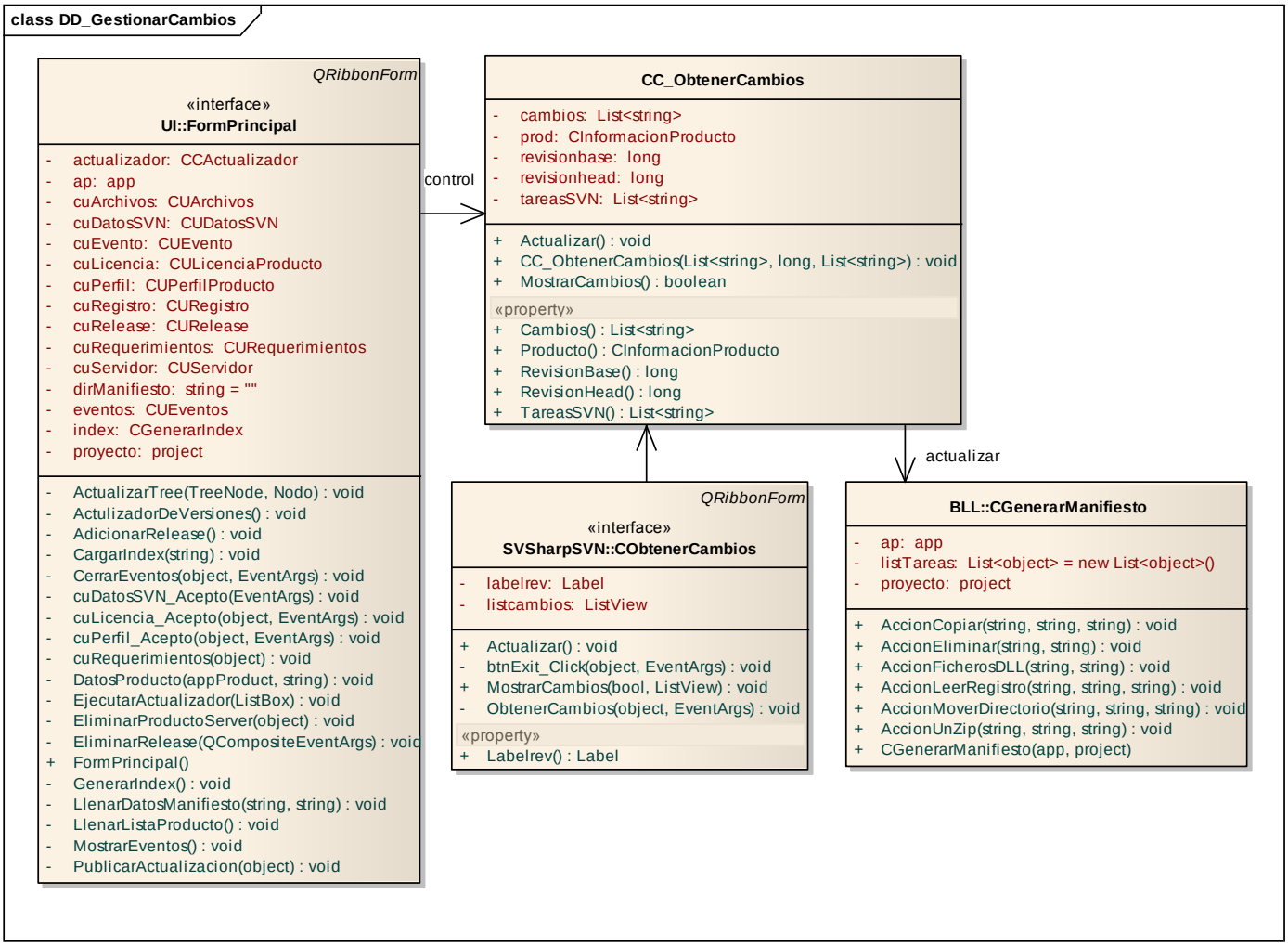


Fig. 9 Diagrama de clases del diseño del CUS Gestionar cambios

CUS Gestionar árbol de directorio

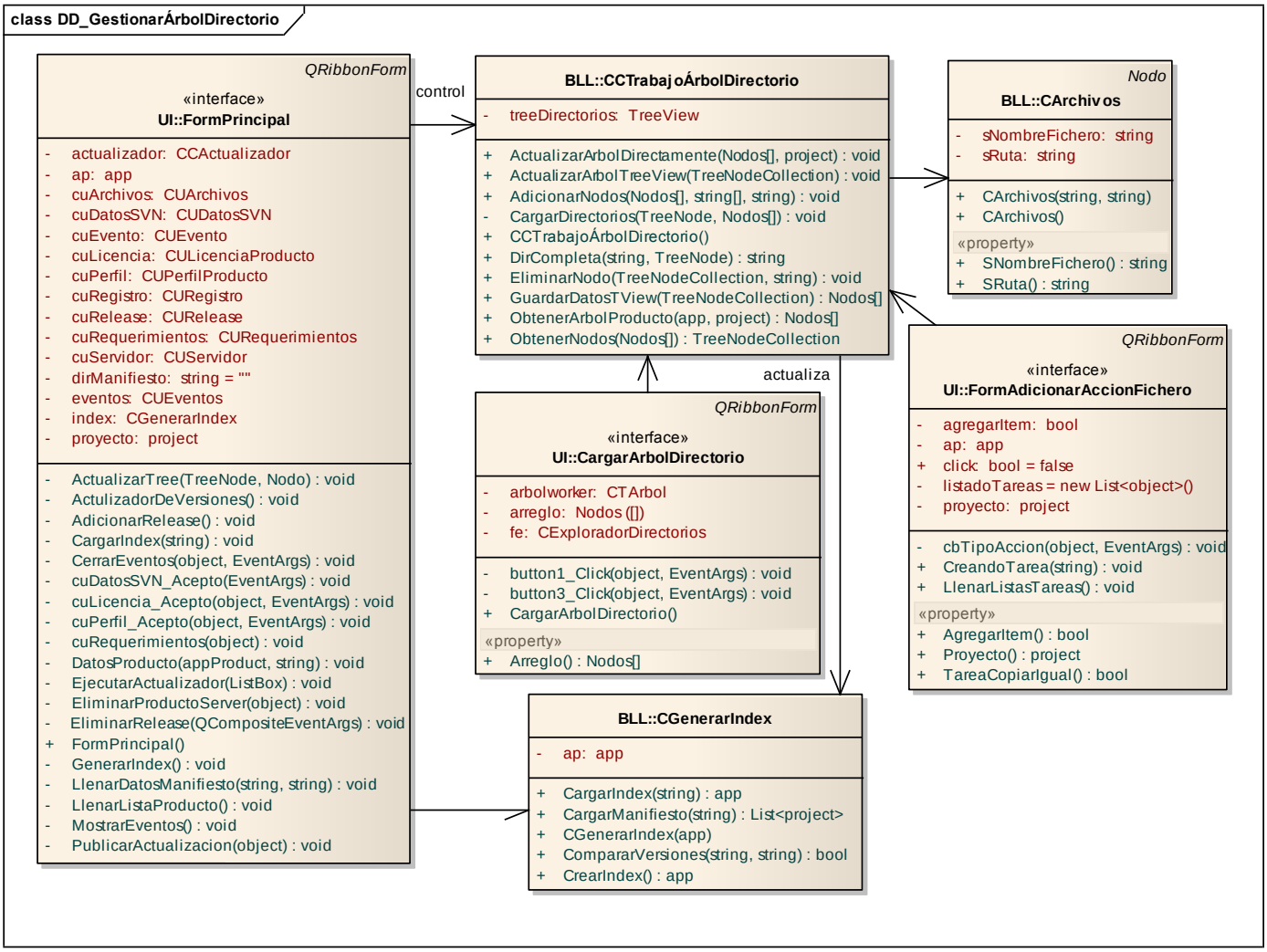


Fig. 10 Diagrama de clases del diseño del CUS Gestionar árbol de directorio

CUS Gestionar producto

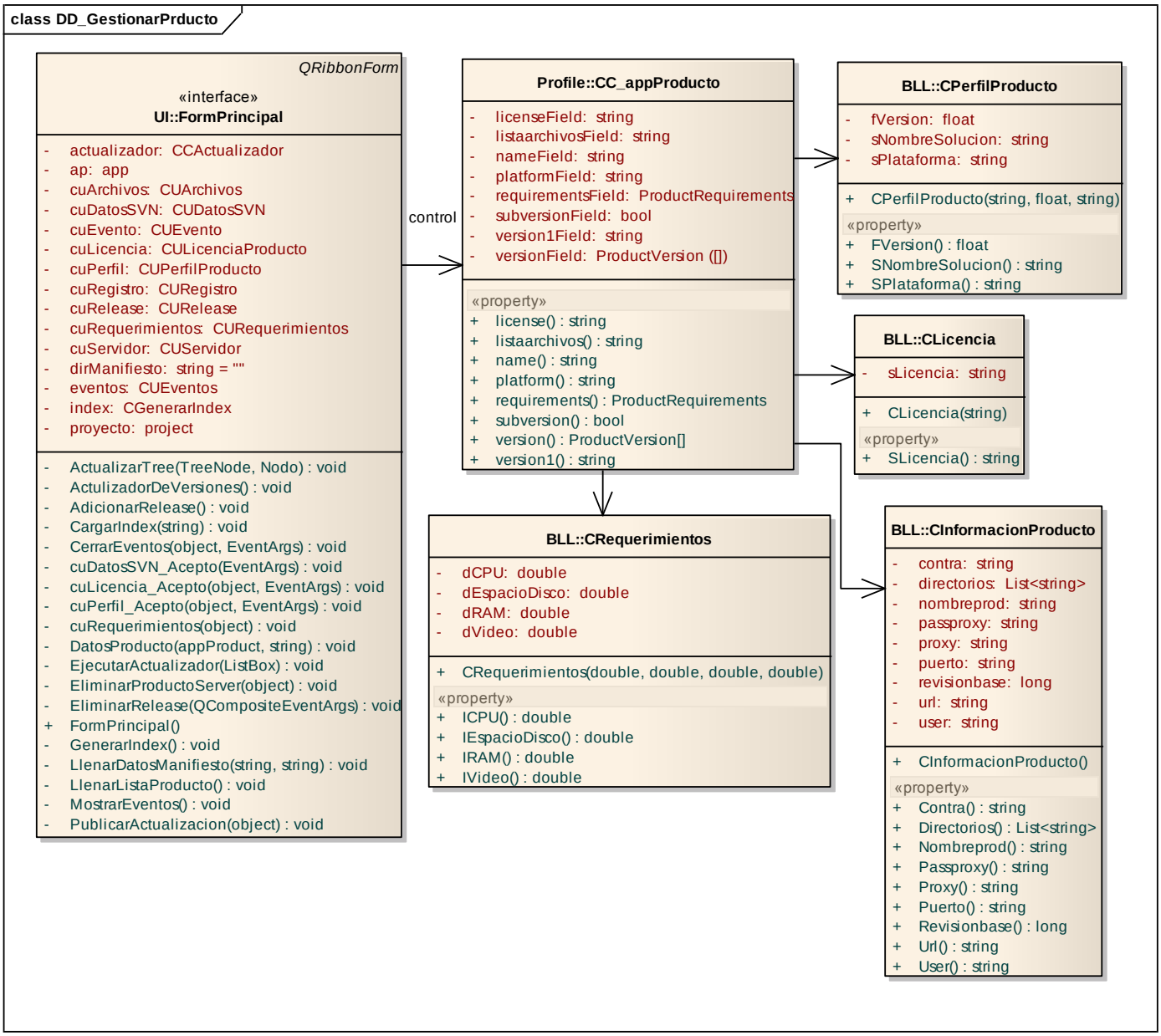


Fig. 11 Diagrama de clases del diseño del CUS Gestionar producto

3.3.2 Diagramas de secuencia

Los diagramas de secuencia muestran la interacción de un conjunto de objetos en una aplicación a través del tiempo. Se modelan para cada caso de uso y contienen detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el mismo, además de los mensajes intercambiados entre los objetos.

A continuación se muestran los diagramas de secuencia por casos de uso del sistema.

CUS Crear actualización

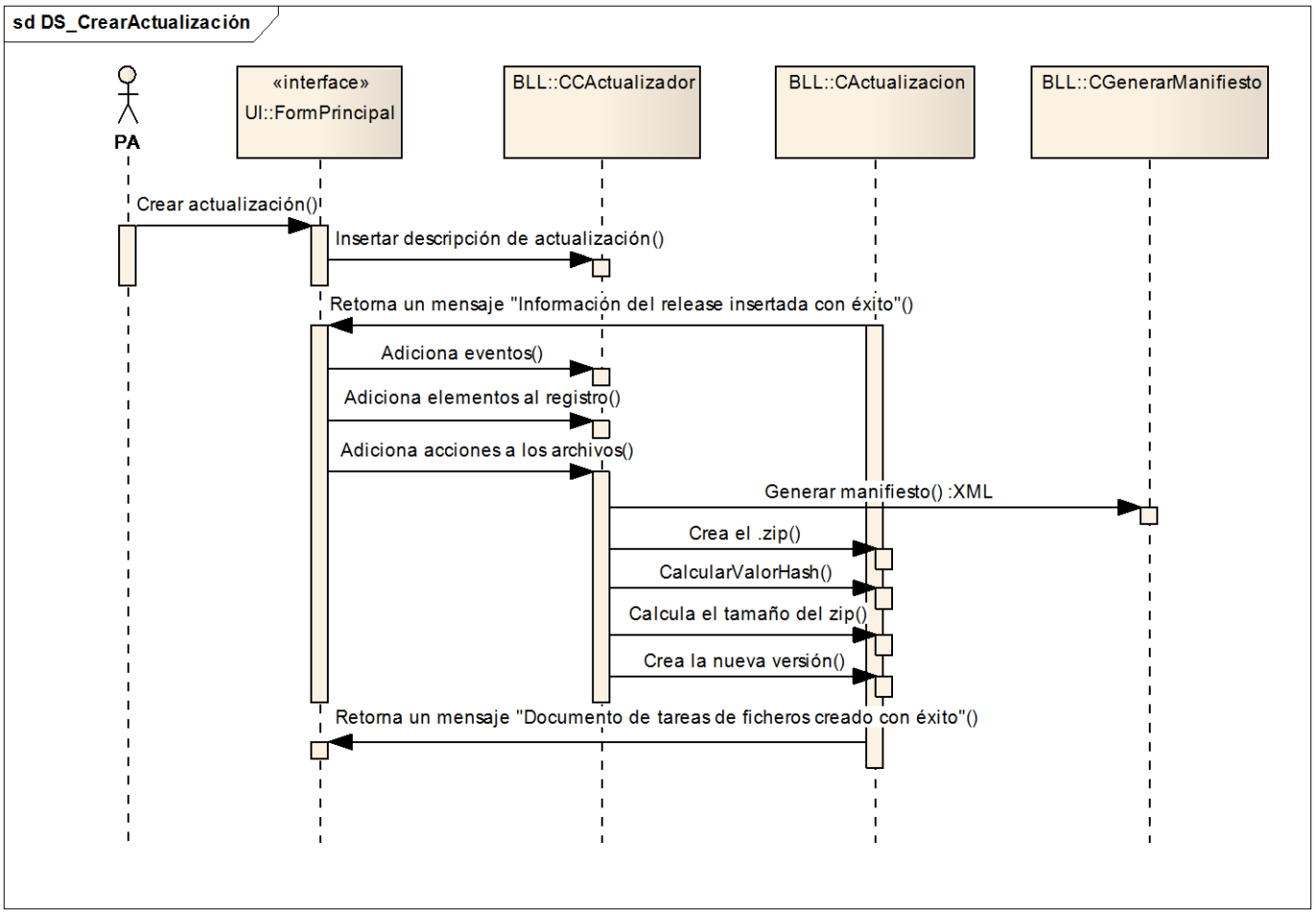


Fig. 12 Diagrama de secuencia del CUS Crear actualización

CUS Gestionar cambios

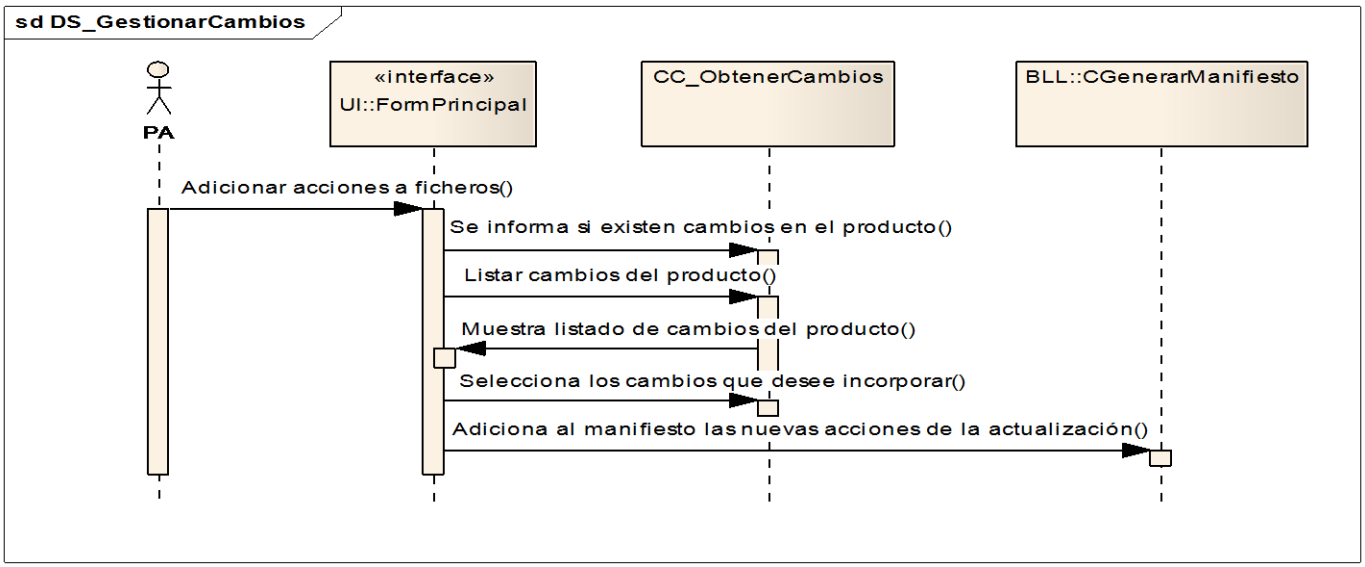


Fig. 13 Diagrama de secuencia del CUS Gestionar cambios

CUS Gestionar árbol de directorio

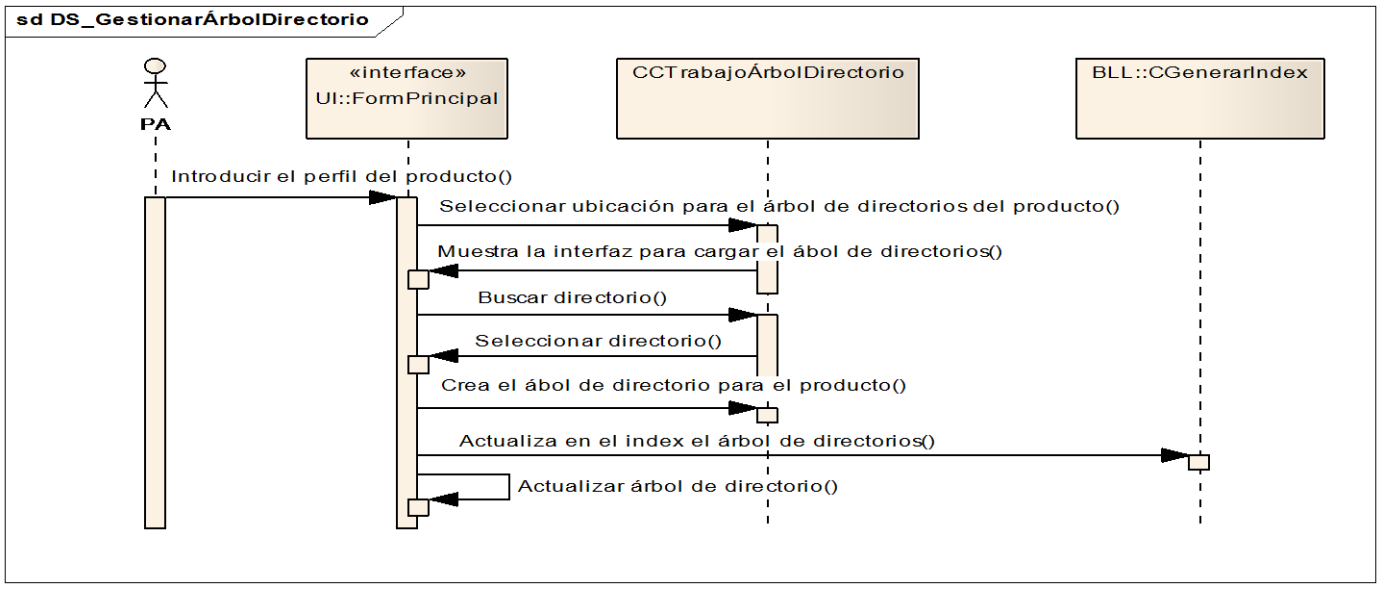


Fig. 14 Diagrama de secuencia del CUS Gestionar árbol de directorio

CUS Gestionar producto

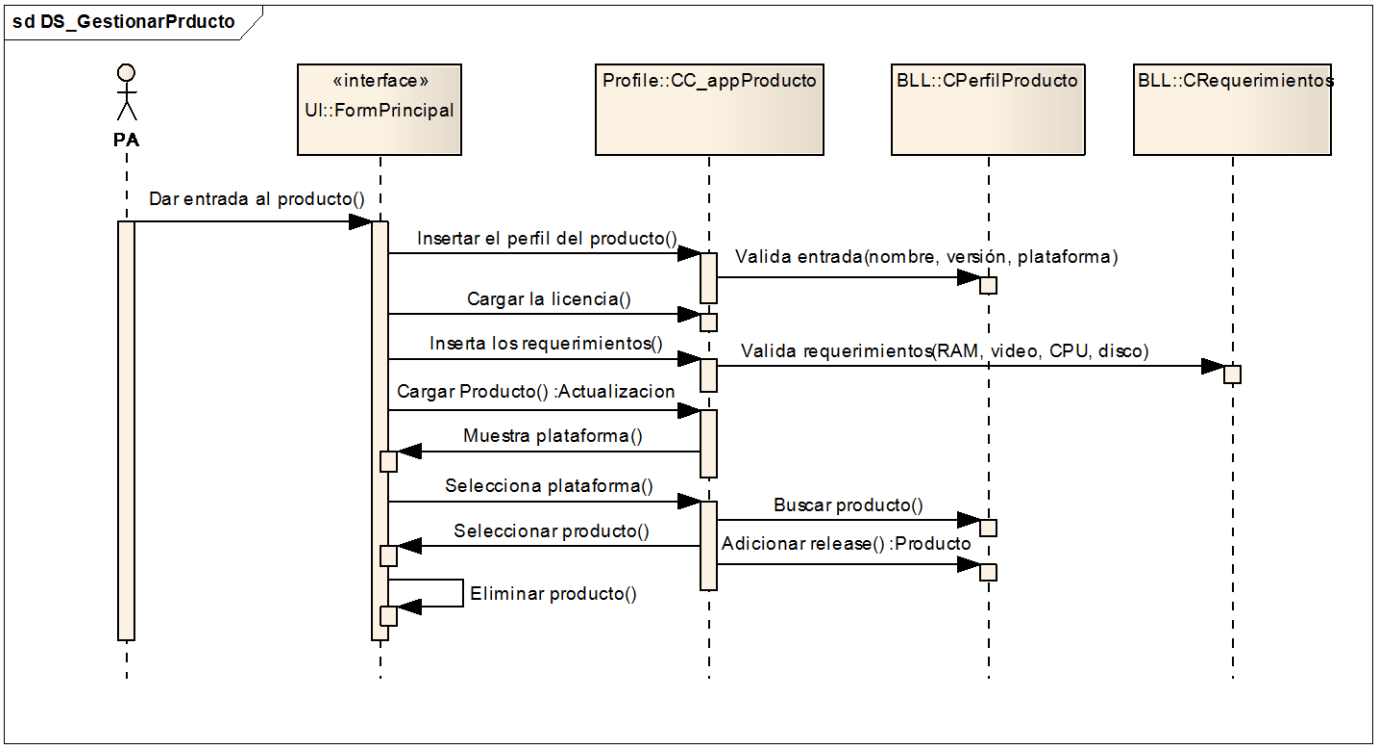


Fig. 15 Diagrama de secuencia del CUS Gestionar producto

3.4 Descripción de las clases

Capa de Presentación

La función principal de la capa de presentación es intercambiar información con los actores, por lo tanto es la única capa con la que interactúa directamente el usuario.

Nombre: FormPrincipal
Descripción: Esta es la interfaz principal del Diseñador de Actualizaciones Automáticas sobre la cual el usuario realizará todas las operaciones pertinentes a una actualización.

Nombre: FormObtenerCambios
Descripción: Esta interfaz muestra los cambios que han ocurrido en un producto publicado en un repositorio desde la revisión base del producto hasta la última revisión registrada en el mismo y permite seleccionar los que el usuario desea incluir en la actualización que está creando en ese momento.

Nombre: FormEliminarProducto
Descripción: Este formulario brinda la posibilidad de eliminar un producto que se encuentre publicado en el servidor FTP, para ello el usuario debe seleccionar el servidor donde se encuentra dicho producto, la plataforma del mismo y el producto en sí.

Nombre: FormCargar
Descripción: Este formulario permite cargar un producto existente en el repositorio para crearle una nueva actualización.

Nombre: FormCargarÁrbolDirectorio
Descripción: Este formulario permite seleccionar el árbol de directorios donde está instalado el producto al cual se le está creando una actualización.

Nombre: FormArchivosSVN
Descripción: Este formulario permite seleccionar los archivos de un producto a tener en cuenta a la hora de chequear los cambios, vale la aclaración que todos los archivos pertenecientes a un producto no son importantes a la hora de diseñar un paquete de actualizaciones.

Nombre: FormAgregarElementosRegistro
Descripción: Es el formulario donde el Proveedor de Actualizaciones introduce las nuevas llaves que se van a insertar al registro del sistema.

Nombre: FormAdicionarAcciónFichero
Descripción: Es el formulario donde el Proveedor de Actualizaciones realiza las acciones que se

incluyen dentro del paquete de actualización, es decir las acciones Eliminar, Copiar o Mover un archivo determinado, de igual manera se describe si se registran o no los archivos del tipo.dll o si se descomprimen los archivos .rar o .zip.

Nombre: FormAcciónEvento
Descripción: Es el formulario donde el proveedor de actualizaciones introduce los pasos necesarios que se deben realizar antes o después de una actualización de software, estos pasos son referentes a procesos que interactúan de alguna manera con el producto a actualizar.

Capa Lógica de Negocio

Se comunica con la capa de presentación y la capa de datos y es la encargada de recibir y responder cada petición de los usuarios.

Nombre: CActualización
Descripción: Esta clase contiene toda la información correspondiente a las actualizaciones de los productos.

Nombre: CCActualizador
Descripción: Esta clase es la que actualiza los datos de producto en el index, es decir permite guardar el index actualizado para su posterior publicación, de igual manera contiene algunas funcionalidades a utilizar cuando un producto sea eliminado.

Nombre: CCObtenerCambios
Descripción: Esta clase es la encargada de conectarse al repositorio SVN para obtener los cambios de un producto desde una versión determinada hasta la versión más reciente del mismo.

Nombre: CCTrabajoÁrbolDirectorio
Descripción: Esta clase es la encargada de realizar todas las operaciones correspondientes al árbol de directorio del producto al que se le está creando la actualización.

Nombre: CFTPConexion
Descripción: Esta clase se encarga de la conexión al servidor y sus funciones mediante el protocolo FTP.

Nombre: CGenerarIndex
Descripción: Esta clase permite generar el index de un producto o actualizar uno existente.

Nombre: CGenerarManifiesto
Descripción: Las funcionalidades de esta clase consisten en generar de un paquete de actualización el manifiesto con sus tareas correspondientes.

Nombre: CGenerarZip
Descripción: Esta clase permite generar el paquete de actualización en un archivo compactado (.zip) para que el usuario pueda utilizarlo posteriormente.

Nombre: CProducto
Descripción: En esta clase se gestiona todo lo referente al producto que se le desea realizar la actualización.

Con la realización de este capítulo se obtuvo el diagrama de clases del diseño para cada caso de uso del sistema, mediante los cuales se representaron las relaciones que se establecen entre las clases. Además quedaron definidos los diagramas de interacción y se realizó una descripción de las principales clases que estarán presentes en la implementación de la solución.

Capítulo 4: Implementación

El presente capítulo tiene como objetivo describir el modelo de implementación teniendo en cuenta el resultado obtenido en el diseño. El sistema es modelado en términos de componentes y este se organiza de acuerdo a los nodos específicos que conforman el modelo de despliegue. Se incluye en el desarrollo del capítulo los diagramas de componentes y despliegue.

4.1 Modelo de implementación

El modelo de implementación describe como los elementos del modelo de diseño y las clases, se implementan en términos de componentes, ficheros de código fuente, ejecutables, entre otros. Los diagramas de despliegue y componentes conforman lo que se conoce como un modelo de implementación, al describir los componentes y construir su organización y dependencia entre los nodos físicos en que funcionará la aplicación.

4.2 Componentes

Un componente es la parte modular de un sistema, desplegable y reemplazable que encapsula implementación, un conjunto de interfaces y proporciona la realización de los mismos. Un componente típicamente contiene clases y puede ser implementado por uno o más artefactos (ficheros ejecutables, binarios). Son las piezas reutilizables de alto nivel a partir de las cuales se pueden construir los sistemas. (21)

4.3 Diagrama de componentes

Los diagramas de componentes modelan la vista estática de un sistema. Tienen un nivel de abstracción más elevado que un diagrama de clase, usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, ya que eventualmente un componente puede comprender una gran porción de un sistema.

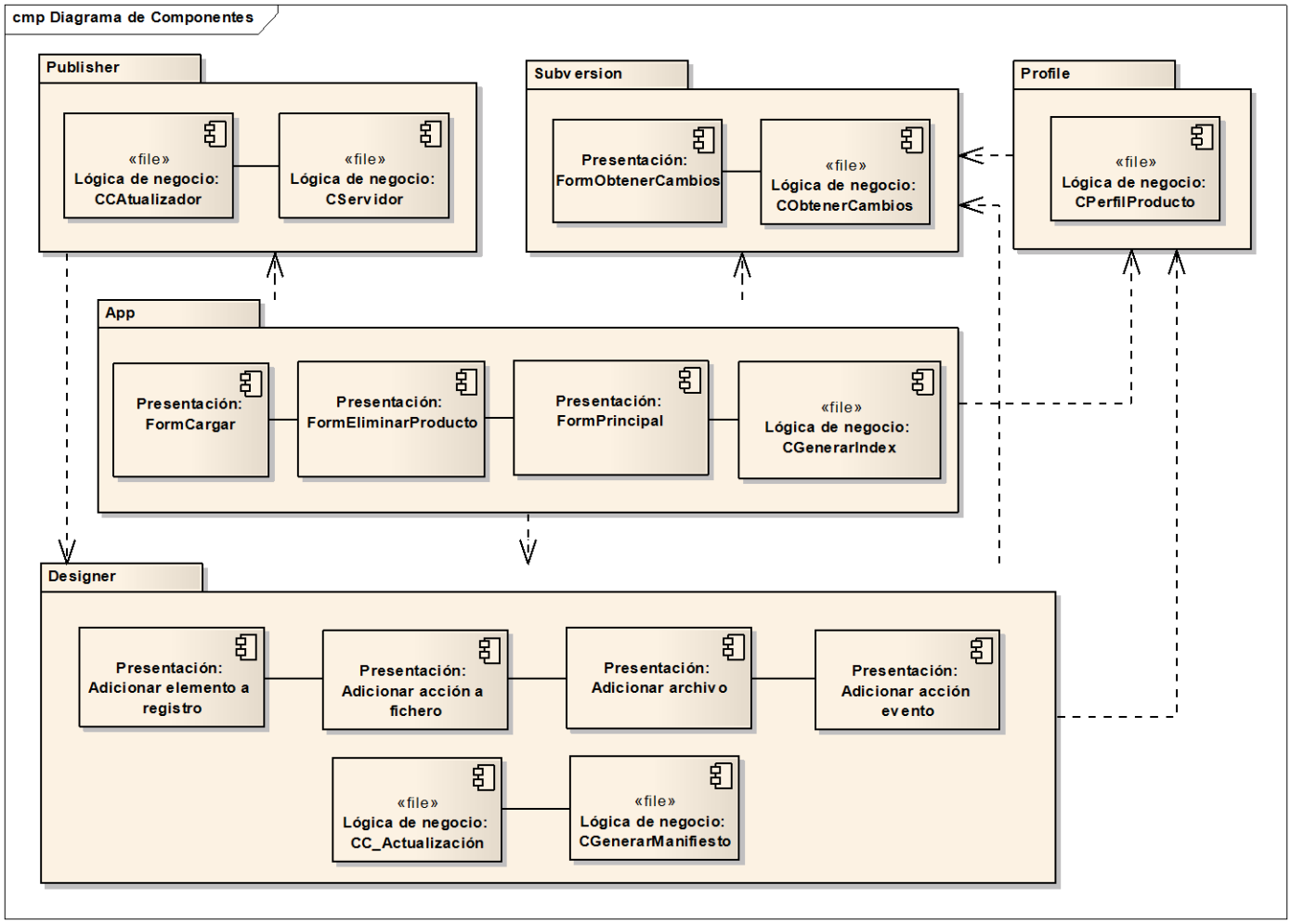


Fig. 16 Diagrama de componentes

4.4 Diagrama de despliegue

Un Diagrama de Despliegue muestra cómo y dónde se desplegará el sistema. Las máquinas físicas y los procesadores se representan como nodos. Como los artefactos se ubican en los nodos para modelar el despliegue del sistema, la ubicación es guiada por el uso de las especificaciones de despliegue.

Para desplegar el Diseñador de Actualizaciones Automáticas se requiere una PC cliente donde se instalará el Diseñador de actualizaciones y dos PCs servidoras, una de ellas para almacenar las

actualizaciones que han sido publicadas y la otra para mantener el historial de los cambios realizados a los productos en desarrollo.

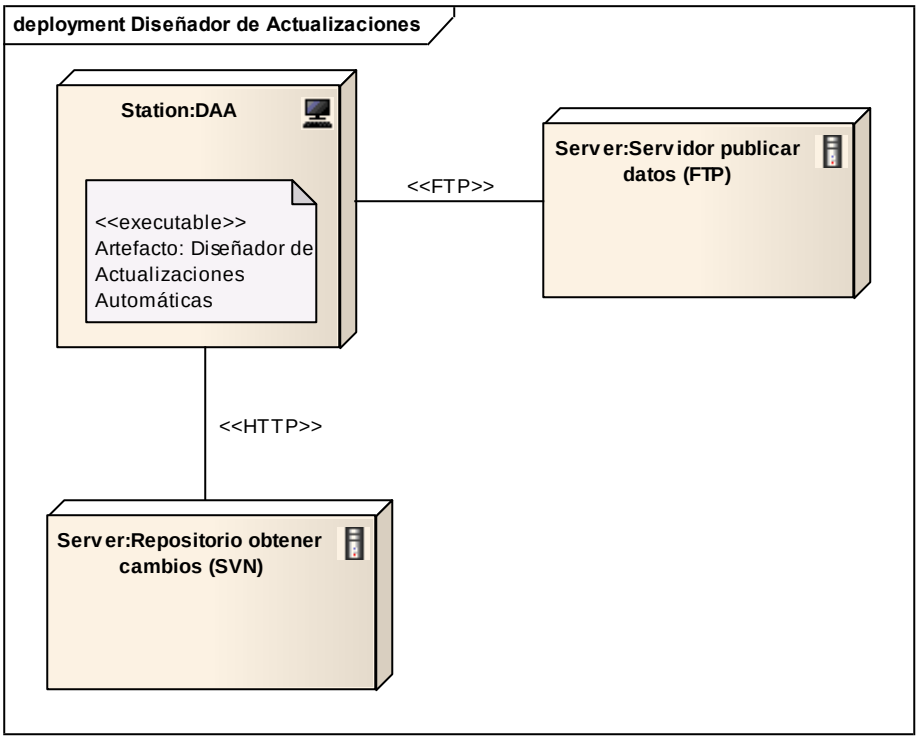
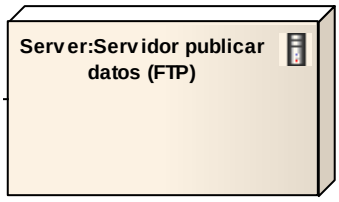


Fig. 17 Diagrama de despliegue

Descripción de los nodos

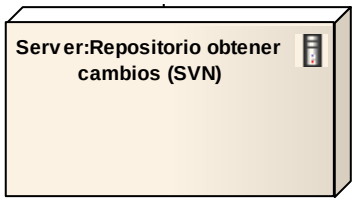
Server: Servidor Publicar Datos: Aquí estarán publicadas las distintas versiones de los productos para la actualización.



Requerimientos de hardware

Requerimientos mínimos Ordenador Pentium IV o superior, 1 GB de memoria RAM o superior, capacidad en disco duro de 500 GB, copia de seguridad, velocidad del procesador de 3.0 GHz o superior.

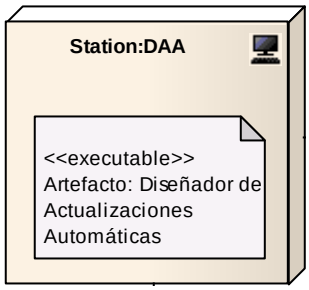
Server: Repositorio obtener cambios (SVN): Aquí estará el historial de los cambios realizados a los productos en desarrollo.



Requerimientos de hardware

Requerimientos mínimos Ordenador Pentium IV o superior, 1 GB de memoria RAM o superior, capacidad en disco duro de 500 GB, copia de seguridad, velocidad del procesador de 3.0 GHz o superior.

Station: Diseñador Actualizaciones Automáticas: Aquí estarán instalada la aplicación para llevar a cabo el diseño de las actualizaciones.



Requerimientos de hardware

Requerimientos mínimos Ordenador Pentium IV o superior, 256 MB de memoria RAM o superior, capacidad en disco duro de 8 GB, copia de seguridad, velocidad del procesador de 1.5 GHz o superior.

4.5 Restricciones de nomenclatura y Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código por lo que al comenzar un proyecto de software, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada.

A continuación se presentan una serie de normas que deben cumplir los programadores a la hora de implementar el Diseñador de Actualizaciones Automáticas.

Identación

La identación o sangría consiste en marginar hacia la derecha todas las instrucciones de un mismo módulo o bloque de instrucciones, de forma que se vea rápidamente cuáles pertenecen al bloque y cuáles no.

Identación ***Objetivo:* Lograr una estructura uniforme para los bloques de código así como para los diferentes niveles de anidamiento.**	
Inicio y fin de bloque	Se recomienda dejar dos espacios en blanco desde la instrucción anterior para el inicio y fin de bloque {}. Lo mismo sucede para el caso de las instrucciones if, else, for, while, do while, switch, foreach.
Aspectos Generales	El identado debe ser de dos espacios por bloque de código. No se debe usar el tabulador; ya que este puede variar según la PC o la configuración de dicha tecla. Los inicios (**{**) y cierre (**}**) de ámbito deber estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay sólo una instrucción. Nunca colocar **{**en la línea de un código cualquiera, esto requiere una línea propia.

Comentarios, líneas y espacios en blanco

Los comentarios representan la documentación interna de los programas. Estos deben ser breves y coherentes con el código, no deben ser obvios ni incluir divagaciones. Se deben utilizar cuando sea necesario para incrementar la legibilidad de un programa y deben tener un estilo uniforme, respetando una puntuación a lo largo de toda la aplicación.

Comentarios, líneas y espacios en blanco ***Objetivo:*** **Establecer un modo común para comentar el código de forma tal que sea comprensible con sólo leerlo una vez.**		
Ubicación de comentarios	Al inicio de cada clase o función y al final de cada bloque de código.	Se recomienda comentar al inicio de la clase o función especificando el objetivo de la misma así como los parámetros que usa (especificar tipos de dato, y objetivo del parámetro) entre otras cosas.
Líneas en blanco	Se emplean antes y después de métodos, clases y estructuras.	Se recomienda dejar una línea en blanco antes y después de la declaración de una clase o de una estructura y de la implementación de una función.
Espacios en blanco	Entre operadores lógicos y aritméticos.	Se recomienda usar espacios en blanco entre estos operadores para lograr una mayor legibilidad en el código. Ejemplo: producto = nombproducto
Aspectos generales	Sobre el comentario	Se debe evitar comentar cada línea de código. Cuando el comentario se aplica a un grupo de instrucciones debe estar seguido de una línea en blanco. En caso de que se necesite comentar una sola instrucción se suprime la línea en blanco o se escribe a continuación de la instrucción.
	Sobre los espacios en blanco	No se debe usar espacio en blanco: Después del corchete abierto y antes del cerrado de un arreglo. Después del paréntesis abierto y antes del cerrado. Antes de un punto y coma.

Identificadores

Un identificador es un nombre simbólico asociado a un elemento de un programa, que puede ser una variable, una constante, un módulo (procedimiento, función o método) o un archivo. El nombre debe ser conciso y describir lo más claramente posible al objeto que identifica.

Variables y constantes		
Apariencia de variables	Las variables tendrán un prefijo para el tipo de datos en minúscula.	El nombre que se le da a las variables debe comenzar con la primera letra en minúscula, la cual identificará el tipo de datos al que se refiere (tabla 3), en caso de que sea un nombre compuesto se empleará notación CamellCasing[1]. Ejemplo: sNombrePaciente
Apariencia de constantes	Todas sus letras en mayúsculas.	Se deben declarar las constantes con todas sus letras en mayúscula.
Aspectos generales	Nombres de las variables y constantes.	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.
Clases y Objeto ***Objetivo:* Nombrar las clases e instancias de forma estándar para todas las aplicaciones.**		
Apariencia de clases y objetos	Primera letra en mayúscula.	Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing[2]. Ejemplo: MiClase(). Para el caso de las instancias se comenzará con un prefijo que identificará el tipo de dato, este se escribirá en minúscula.
Apariencia de atributos	Primera letra en minúscula	El nombre que se le da a los atributos de las clases debe comenzar con la primera letra en minúscula, la cual estará en correspondencia al tipo de dato al que se refiere, en caso de que sea un nombre compuesto se empleará notación CamellCasing.
Apariencia de las	Primera letra en	Para nombrar las funciones se debe tratar de utilizar

[1] **Notación CamellCasing:** Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula excepto la primera palabra que debe iniciar con minúscula.
Ejemplo: notacionCamelCasing.

[2] **Notación PascalCasing:** Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula.
Ejemplo: NotacionPascalCasing

funciones	mayúscula	verbos que denoten la acción que hace la función. Se empleará notación PascalCasing. Ejemplo: function BuscarUnidad().
Declaración de parámetro en funciones	Agrupados por tipos Poner los string 1 numéricos 2, además, agrupar según valores por defecto.	Los parámetros que se le pasan a las funciones se recomienda sean declarados de forma tal que estén agrupados por el tipo de dato que contienen, especificando el tipo de datos (tabla 3).
Aspectos generales	Sobre las clases, los objetos, los atributos y las funciones.	El nombre empleado para las clases, objetos, atributos y funciones debe permitir que con sólo leerlo se conozca el propósito de los mismos.
Controles		
Apariencia de los controles.	Los controles tendrán un prefijo para el tipo de datos en minúscula.	El nombre que se le da a los controles deben comenzar con las primeras letras en minúscula, las cuales identificaran el tipo de datos al que se refiere (tabla 4), en caso de que sea un nombre compuesto se empleará notación CamellCasing. Ejemplo: btnAcepta. (22)

Tipo Datos	Prefijo	Ejemplo
Int	i	iCantPacientes
float	f	fPesoPaciente
double	d	dPesoCarro
bool	b	bPacienteActivo
string	s	sNombrePaciente
char	c	cLetra
De tipo enum	en	enSexo
byte	bt	btCantDiasPaciente
sbyte	sb	sbEdadPaciente

short	sh	shVariableShort
ushort	us	usVariableUshort
uint	ui	uiVariableUint
long	l	lVariableLong
ulong	ul	ulVariableUlong
decimal	dc	dcVariableDecimal
Objetos	o	oPacienteHistorico
Objetos de tipo Struct	st	stUnaStruct

Tabla 3 Tipos de datos

Control	Prefijo	Ejemplo
Botón	btn	btnAceptar
Etiqueta	lbl	lblNombre
Lista/Menú	mn	mnPrincipal
Campo de Texto	txt	txtFecha
Casilla de Verificación	chx	chxBorrar
Casilla de Selección	cbx	cbxSexo

Tabla 4 Prefijos de los controles

4.6 Comparación de la primera versión del Diseñador de Actualizaciones Automáticas con la solución desarrollada

Una de las deficiencias detectadas durante el análisis realizado a la primera versión de la aplicación, era que los usuarios cuando creaban una acción a ejecutar en la actualización, ya sea copiar o mover un archivo, debían introducir la dirección de destino de dicho archivo de forma manual. Esto traía como consecuencia que se introdujeran errores en este proceso, ya que el formato para escribir la dirección no era correcto.

Este problema se solucionó con la implementación del árbol de directorios de la aplicación, lo que posibilita que el usuario no tenga que introducir el destino de un archivo de forma manual y pueda seleccionar de la carpeta donde se encuentra instalado el producto, el lugar donde desea copiarlo.

Otro de los problemas que presentaba la primera versión del Diseñador de Actualizaciones Automáticas, era que la confección de una actualización se convertía en un proceso largo y engorroso si la misma estaba compuesta por un número considerable de archivos, ya que cada acción que se debía ejecutar al actualizar una aplicación tenía que ser creada manualmente.

Con el desarrollo de la nueva versión, este problema quedó erradicado pues la integración al controlador de versiones Subversion da la posibilidad al usuario de listar todos los cambios ocurridos y seleccionar los que desee incluir en la actualización que está creando. Todo ello agiliza el proceso de creación de las actualizaciones y demuestra que hubo un paso de avance en el desarrollo de la versión actual del Diseñador de Actualizaciones Automáticas y que la integración del mismo al controlador de versiones Subversion disminuye considerablemente los errores por pérdida de información.

En este capítulo, se obtuvieron los diagramas de componentes y de despliegue, mostrándose cómo se encuentra divido por paquetes de clase el desarrollo del Diseñador de Actualizaciones Automáticas agrupando clases a fines según la capa a la que pertenecen. Además se definen los estándares de codificación que se siguieron para la implementación.

Conclusiones

Una vez finalizada la investigación para el desarrollo del presente trabajo se arribaron a las siguientes conclusiones:

- El análisis de la primera versión del Diseñador de Actualizaciones Automáticas, evidenció que aún no estaba listo para su explotación en ambientes de producción real.
- La identificación y corrección de las deficiencias del Diseñador de Actualizaciones Automáticas permitió obtener una primera versión estable del producto.
- La integración del Diseñador de Actualizaciones Automáticas a un controlador de versiones, representa una fortaleza para este producto, además de constituir una característica novedosa en aplicaciones de este dominio.
- La nueva versión del producto elimina la necesidad de introducir manualmente algún cambio en la fase de diseño de la actualización, lo que garantiza la reducción de los errores introducidos en esta etapa.
- La solución desarrollada establece la base para la extensión, soporte y mantenimiento de los productos desarrollados en el Centro de Informática Médica.

Recomendaciones

- Implementar la compatibilidad del Diseñador de Actualizaciones Automáticas con otros sistemas de control de versiones, para brindar soporte a equipos de desarrollos que no utilicen el Subversion.
- Incluir en el perfil del producto la información de los servicios que utiliza la aplicación, de forma que se emplee un nomenclador a la hora de conformar las tareas de actualización y eliminar así la posibilidad de introducir errores en los nombre de dichos servicios.
- Extender el proceso de publicación de las actualizaciones, utilizando en lugar del protocolo FTP, la tecnología de Servicios WEB, brindando así una solución para el acceso concurrente y seguro de varios usuarios al servicio de publicación.

Referencias Bibliográficas

1. Informatización de la sociedad cubana. [En línea] [Citado el: 11 de noviembre de 2010.] http://www.mic.gov.cu/hinfosoc.aspx.

2. Control de versiones. [En línea] [Citado el: 6 de octubre de 2010.] http://producingoss.com/es/vc.html/.

3. Subversion - EcuRed. [En línea] [Citado el: 15 de febrero de 2011.] http://www.ecured.cu/index.php/Subversion.

4. Redmine. [En línea] [Citado el: 4 de noviembre de 2010.] http://www.redmine.org/.

5. Análisis de aplicación: Redmine. [En línea] [Citado el: 12 de mayo de 2011.] http://www.ceslcam.com/noticias/noticia/articulo/analisis-de-aplicacion-redmine/.

6. Proyecto Trac. [En línea] [Citado el: 16 de noviembre de 2010.] http://trac.edgewall.org/.

7. Alfresco. [En línea] [Citado el: 2 de diciembre de 2010.] http://www.alfresco.com/es/about.

8. Definición de C#. [En línea] [Citado el: 4 de octubre de 2010.] http://www.hard-h2o.com/diccionario-informatico.html.

9. Tecnología XML. [En línea] [Citado el: 27 de noviembre de 2010.] http://www.sparxsystems.com.ar/EAUserGuide/ea.html.

10. XML. Donde empiezan los servicios web. [En línea] [Citado el: 28 de noviembre de 2010.] http://www.eveliux.com/mx/xml-donde-empiezan-los-servicios-web.php.

11. Definición de XML Schema. [En línea] [Citado el: 5 de diciembre de 2010.] http://pergaminovirtual.com.ar/definicion/XMLSchema.html.

12. Definición de UML. [En línea] [Citado el: 11 de enero de 2011.] http://tecnologia.glosario.net/terminos-tecnicos-internet/uml-1655.html.

13. Rational Unified Process. [En línea] [Citado el: 06 de diciembre de 2010.] http://www.rational.com.ar/herramientas/rup.html.

14. Visual Studio. [En línea] [Citado el: 4 de octubre de 2010.] http://msdn.microsoft.com/es-es/library/kx37x362%28v=VS.90%29.aspx.

15. Enterprise Architect. [En línea] [Citado el: 4 de octubre de 2010.] http://www.sparxsystems.com.ar/new/products/.

16. Altova XMLSpy 2008 Professional Edition. [En línea] [Citado el: 5 de diciembre de 2010.] http://www.altova-xmlspy-professional-edition.software.informer.com/2008.

17. Introducción a la plataforma Microsoft .Net. [En línea] [Citado el: 4 de octubre de 2010.] http://fians.uat.edu.mx/catedraticos/lgarciae/1.-%20Introduccion%20a%20la%20plataforma%20Microsoft%20.NET.pdf.

18. Modelo de Dominio. [En línea] [Citado el: 18 de enero de 2011.] http://www.ldc.usb.ve/~martinez/cursos/ci3715/clase6_AJ2010.pdf.

19. Tutorial UML, Diagrama de Casos de Uso. [En línea] [Citado el: 12 de diciembre de 2010.] http://www.clikear.com/manuales/uml/diagramascasouso.aspx.

20. Introduccion_a_la_Disciplina_Analisis_y_Diseno.pdf. [En línea] [Citado el: 1 de marzo de 2011.] http://eva.uci.cu/file.php/102/Curso_2010-2011/Clases/Semana_10/Conferencia_10/Materiales_complementarios/Introduccion_a_la_Disciplina_Analisis_y_Diseno.pdf.

21. Conferencia_Implementacion.pdf. [En línea] [Citado el: 1 de marzo de 2011.] http://eva.uci.cu/file.php/259/Curso_2010-2011/Semana_8/Conferencia_6/Materiales_Basicos/Conferencia_Implementacion.pdf.

22. Gómez Velázquez, Karel Ing., Collada de la Rosa, Ricardo Ing. *Documento de Arquitectura de Software.* 2009.

Bibliografía

1. Alfresco. [En línea] [Citado el: 2 de diciembre de 2010.] http://www.alfresco.com/es/about.
2. Altova XMLSpy 2008 Professional Edition. [En línea] [Citado el: 5 de diciembre de 2010.] http://www.altova-xmlspy-professional-edition.software.informer.com/2008.
3. Análisis de aplicación: Redmine. [En línea] [Citado el: 12 de mayo de 2011.] http://www.ceslcam.com/noticias/noticia/articulo/analisis-de-aplicacion-redmine/.
4. Caso de Uso - EcuRed. [En línea] [Citado el: 13 de Diciembre de 2010.] http://www.ecured.cu/index.php/Caso_de_uso.
5. Conferencia_Implementacion.pdf. [En línea] [Citado el: 1 de marzo de 2011.] http://eva.uci.cu/file.php/259/Curso_2010-2011/Semana_8/Conferencia_6/Materiales_Basicos/Conferencia_Implementacion.pdf.
6. Control de versiones. [En línea] [Citado el: 6 de octubre de 2010.] http://producingoss.com/es/vc.html/.
7. Definición de C#. [En línea] [Citado el: 4 de octubre de 2010.] http://www.hard-h2o.com/diccionario-informatico.html.
8. Definición de UML. [En línea] [Citado el: 11 de enero de 2011.] http://tecnologia.glosario.net/terminos-tecnicos-internet/uml-1655.html.
9. Definición de XML Schema. [En línea] [Citado el: 5 de diciembre de 2010.] http://pergaminovirtual.com.ar/definicion/XMLSchema.html.
10. Enterprise Architect. [En línea] [Citado el: 4 de octubre de 2010.] http://www.sparxsystems.com.ar/new/products/.
11. Enterprise Architect - EcuRed. [En línea] [Citado el: 4 de Octubre de 2010.] http://www.ecured.cu/index.php/Enterprise_Architect.

12. Flujo de Trabajo Requerimiento - EcuRed [En línea] [Citado el: 13 de Diciembre de 2010.] http://www.ecured.cu/index.php/Flujo_de_Trabajo_Requerimiento.

13. Garcerant, Iván. Tecnología y Synergix. [En línea] [Citado el: 18 de Enero de 2011.] http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

14. Gómez Velázquez, Karel Ing., Collada de la Rosa, Ricardo Ing. *Documento de Arquitectura de Software.* 2009.

15. Informatización de la sociedad cubana. [En línea] [Citado el: 11 de noviembre de 2010.] http://www.mic.gov.cu/hinfosoc.aspx.

16. Introducción a la plataforma Microsoft .Net. [En línea] [Citado el: 4 de octubre de 2010.] http://fians.uat.edu.mx/catedraticos/lgarciae/1.-%20Introduccion%20a%20la%20plataforma%20Microsoft%20.NET.pdf.

17. Introducción_a_la_Disciplina_Análisis_y_Diseño.pdf. [En línea] [Citado el: 1 de marzo de 2011.] http://eva.uci.cu/file.php/102/Curso_2010-2011/Clases/Semana_10/Conferencia_10/Materiales_complementarios/Introduccion_a_la_Disciplina_Analisis_y_Diseno.pdf.

18. Introduction to XML Schema. [En línea] [Citado el: 19 de noviembre de 2010.] http://www.w3schools.com/schema/schema_intro.asp.

19. Modelo de Dominio. [En línea] [Citado el: 18 de enero de 2011.] http://www.ldc.usb.ve/~martinez/cursos/ci3715/clase6_AJ2010.pdf.

20. .Net – EcuRed. [En línea] [Citado el: 4 de Octubre de 2010.] http://www.ecured.cu/index.php/.Net.

21. Por qué utilizar C#. [En línea] [Citado el: 16 de noviembre de 2010.] http://msdn.microsoft.com/es-es/library/aa287554%28v=VS.71%29.aspx.

22. Proyecto Trac. [En línea] [Citado el: 16 de noviembre de 2010.] http://trac.edgewall.org/.

23. Rational Unified Process. [En línea] [Citado el: 06 de diciembre de 2010.] http://www.rational.com.ar/herramientas/rup.html.

24. Redmine. [En línea] [Citado el: 4 de noviembre de 2010.] http://www.redmine.org/.

25. Sparx Systems – Tutorial UML 2 - Diagrama de Secuencia. [En línea] [Citado el: 1 de marzo de 2011.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_sequencediagram.html.

26. Source Control for Visual Studio 2008: VisualSVN Server, TortoiseSVN, & AnkhSVN. [En línea] [Citado el: 19 de Enero de 2010.] http://www.codeproject.com/KB/dotnet/VistaVisualSourceControl.aspx.

27. Subversion - EcuRed. [En línea] [Citado el: 15 de febrero de 2011.] http://www.ecured.cu/index.php/Subversion.

28. Tecnología XML. [En línea] [Citado el: 27 de noviembre de 2010.] http://www.sparxsystems.com.ar/EAUserGuide/ea.html.

29. Tutorial UML, Diagrama de Casos de Uso. [En línea] [Citado el: 12 de diciembre de 2010.] http://www.clikear.com/manuales/uml/diagramascasouso.aspx.

30. Visual Studio. [En línea] [Citado el: 4 de octubre de 2010.] http://msdn.microsoft.com/es-es/library/kx37x362%28v=VS.90%29.aspx.

31. Visual Studio 2008 – EcuRed. [En línea] [Citado el: 4 de Octubre de 2010.] http://www.ecured.cu/index.php/Visual_Studio_2008.

32. XML. [En línea] [Citado el: 17 de noviembre de 2010.] http://linux.about.com/cs/linux101/g/xmllparextensib.htm.

33. XML. Donde empiezan los servicios web. [En línea] [Citado el: 28 de noviembre de 2010.] http://www.eveliux.com/mx/xml-donde-empiezan-los-servicios-web.php.

34. XMLSpy Editor. [En línea] [Citado el: 16 de noviembre de 2010.] http://www.altova.com/xml-editor/.

Anexos

1- Descripción de casos de uso

Descripción del Caso de Uso: Autenticar en el SVN

Objetivo	El objetivo de este caso de uso es permitir el acceso al repositorio a los usuarios que interactúan con el sistema para poder ver las versiones que han sido publicadas y descargar los cambios que desee incluir en una nueva versión de un producto.	
Actores	Proveedor de actualizaciones: (Inicia) Autentica en el SVN.	
Resumen	El caso de uso inicia cuando el PA accede a la opción de listar los cambios desde el repositorio e inmediatamente el sistema pide que introduzca sus datos para permitirle el acceso al mismo, el caso de uso finaliza cuando el PA accede al repositorio.	
Complejidad	Media	
Prioridad	Secundario	
Precondiciones	El usuario debe tener los permisos necesarios para poder acceder al repositorio.	
Postcondiciones	El usuario se encuentra autenticado en el sistema.	
Flujo de eventos		
Flujo básico Autenticar en el SVN		
	Actor	**Sistema**
1.	El caso de uso inicia cuando el PA accede a la opción de listar los cambios desde el repositorio.	
2.		Brinda la posibilidad de introducir los datos de acceso al repositorio: • Usuario • Contraseña

		• Dirección URL Y permite: Aceptar para autenticarse.
3.	Introduce los datos de acceso al repositorio: • Usuario • Contraseña • Dirección URL Selecciona la opción de aceptar.	Comprueba la validez de los datos, si son correctos muestra el listado de las versiones publicadas en el repositorio.
4.		Si los datos son incorrectos el sistema muestra un mensaje notificándolo: **"Los datos introducidos son incorrectos"**.
5.		En caso de que deje campos vacíos el sistema debe mostrar un mensaje de error: **"Los datos son obligatorios para la autenticación en el repositorio".**
6.		El CU termina cuando el sistema muestra un mensaje **"Ha accedido correctamente al repositorio".**
Relaciones	CU Incluidos	
	CU Extendidos	
Requisitos no funcionales	1,2,5,6	

Descripción del Caso de Uso: Crear Perfil

Objetivo	El objetivo que se persigue con este CU es crear el perfil del producto para que se pueda llevar a cabo la nueva actualización con los requisitos

	específicos.
Actores	Proveedor de actualizaciones: (Inicia) Crea el perfil del producto.
Resumen	El caso de uso inicia cuando el PA accede a la opción de Crear perfil del producto, el sistema brinda la opción de introducir los datos del perfil, el sistema crea el perfil, el caso de uso finaliza.
Complejidad	Media
Prioridad	Crítico
Precondiciones	Al producto ha sido insertado al diseñador.
Postcondiciones	Se creó el perfil del producto.
Flujo de eventos	
Flujo básico Crear perfil	

	Actor	**Sistema**
1.	El caso de uso inicia cuando el PA accede a la opción de crear el perfil.	
2.		El sistema muestra los datos predeterminados de perfil de producto y las especificaciones del dicho perfil. • Perfil • Licencia • Requerimiento Brinda la posibilidad de introducirlos datos de: • Nombre de la solución • Versión • Licencia Seleccionar: • Plataforma • Velocidad mínima de CPU

		• RAM • Mínimo de video • Disco Y permite aceptar cada una de las especificaciones del producto.
3.	El actor introduce lo datos de: • Nombre de la solución • Versión • Licencia Selecciona: • Plataforma • Velocidad mínima de CPU • RAM • Mínimo de video • Disco Selecciona la opción de aceptar.	
4.		El sistema muestra que su perfil es completo.
5.		El CU termina cuando el sistema muestra el perfil de producto con todos los datos listo para realizar la nueva versión.

Relaciones	CU Incluidos	Este es un caso de uso incluido ya que no se podrá ejecutar al menos que se halla insertado el producto y siempre que este sea insertado se creará su respectivo perfil. Crear perfil. Ver CU Insertar producto.
	CU Extendidos	No tiene
Requisitos no funcionales	1,2,5,6	

Asuntos pendientes	Permitir que al crear el perfil se gestiones más especificidades del producto.

Descripción del Caso de Uso: Listar publicación

Objetivo	El objetivo que se persigue con este CU es listar todas las actualizaciones que han sido publicadas en el servidor FTP.
Actores	Proveedor de actualizaciones: (Inicia) listar publicación.
Resumen	El caso de uso inicia cuando el PA accede a Modificar Actualización o Eliminar Actualización, ya que para cualquiera de las dos opciones es necesario que el sistema liste las actualizaciones que han sido publicadas y así el usuario pueda escoger la que desea.
Complejidad	Baja
Prioridad	Auxiliar
Precondiciones	Existe la actualización del producto.
Postcondiciones	Se listaron las actualizaciones publicadas.
Flujo de eventos	
Flujo básico descripción de actualización	

	Actor	**Sistema**
	El caso de uso inicia cuando el PA selecciona la opción de Modificar Actualización o Eliminar Actualización.	
		En ambos casos el sistema brinda la posibilidad de escoger el tipo de plataforma para la cual se creó la actualización.
	Selecciona la plataforma de la actualización.	
		El CU termina cuando el sistema muestra un listado con todas las actualizaciones que han sido publicadas en el servidor FTP.

Relaciones	CU Incluidos	
	CU Extendidos	
Requisitos no funcionales	1,2,3,4,5	
Asuntos pendientes	La actualización ha de ser publicada.	

Glosario de Términos

- Árbol de directorio: Árbol que contiene la distribución de los archivos de un producto.
- Clases: Conjunto de objetos que comparten atributos, operaciones, relaciones y semántica, las mismas representan los conceptos fundamentales del sistema.
- Diagrama de Clases: Representación de los conceptos de importancia en el área de la aplicación, así como de las relaciones entre estos.
- Extensible Markup Language (XML): Un lenguaje de marcado extensible que puede usarse para almacenar datos en un formato estructurado, basado en texto y definido por el usuario.
- Fichero ZIP: Extensión de los ficheros comprimidos con el algoritmo ZIP.
- FTP: Son las siglas de File Transfer Protocol, el nombre del protocolo estándar de transferencia de ficheros. Su misión es permitir a los usuarios recibir y enviar ficheros de todas las máquinas que sean servidores FTP.
- Manifiesto: La clase manifiesto contiene un documento XML con el listado de la información de todos los recursos que el Actualizador necesita para actualizar la aplicación con las acciones que este debe realizar y pasos que llevará a cabo.
- Paquete de Actualización: Serie de ficheros que juntos forman una versión de un software y están asociados a acciones que se describen en un documento XML. Estos ficheros y el documento XML conforman el paquete de actualización.
- Tareas: Cada una de las acciones que se incluyen en un paquete de actualización, pueden ser: Copiar, Eliminar, Mover Directorio y Descomprimir Archivo.