UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Sistema para la gestión de actualizaciones en las aplicaciones de escritorio que se desarrollan en el centro GEYSED.

AUTORES: Elizabeth Herrera Almira

Eliener Roche Santos

TUTORES: MSc. Yanet Espinol Martín

Ing. Yunet Gasca Suárez

Ing. Aimé Esther Guzmán Ramírez

CO-TUTOR: Lic. Gretchen Guillermo Hernández

La Habana, Julio 2015

"Año 57 de la Revolución"

"El hombre debe transformarse al mismo tiempo que la producción progresa; no realizaríamos una tarea adecuada si fuéramos tan sólo productores de artículos, de materias primas y no fuéramos al mismo tiempo productores de hombres."

Ernesto Che Guevara

Declaración de Autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos al centro Geoinformática y Señales Digitales (GEYSED) de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de junio del año 2015.

Elizabeth Herrera Almira

_____

Firma Autor

Eliener Roche Santos

_____

Firma Autor

Ing. Yunet Gasca Suárez

_____

Firma Tutor

MSc. Yanet Espinal Martín

_____

Firma Tutor

Ing. Aimé Esther Guzmán Ramírez

_____

Firma Tutor

Datos de Contacto

Tutora: Msc. Yanet Espinal Martín (yanete@uci.cu)

Graduada en Ciencias de la Computación en la Universidad de Oriente (UO), en el año 2005. Profesora del Departamento de Técnicas de Programación y Sistemas Digitales.

Tutor: Ing. Yunet Gasca Suárez (ygasca@uci.cu)

Graduada de Ingeniera en Ciencias informáticas en la UCI en el año 2013. Trabajadora del centro GEYSED en el proyecto Calidad. Desempeña el rol como administradora de calidad.

Tutor: Ing. Aimé Esther Guzmán Ramírez (aguzman@uci.cu)

Graduada de Ingeniera en Ciencias informáticas en la UCI en el año 2010. Trabajadora del centro GEYSED en el proyecto Calidad. Desempeña el rol como administradora de calidad.

Autor: Elizabeth Herrera Almira (ealmira@etudiantes.uci.cu)

Estudiante perteneciente al centro GEYSED, específicamente al proyecto Calidad. Desempeña el rol de Probador.

Autor: Eliener Roche Santos (eroche@estudiantes.uci.cu)

Estudiante perteneciente al centro GEYSED, específicamente al proyecto Calidad. Desempeña el rol de Probador.

Elizabeth

Agradecer primero que todo a mi familia en especial a mi abuela, mi papá, mi abuelo, mi hermano, los cuales no importa la situación en la que me encuentre siempre son capaces de sacarme adelante.

A mis amigos y compañeros de los cuales me llevo los mejores recuerdos de estos 5 años que transite por la universidad, Anet, Laura, con las que solamente conviví durante dos años y aun así siempre estuvieron presentes cuando las necesitaba.

A mis compañeras de apartamento que supieron soportarme y aunque algunas veces tuvieran ganas de matarme, les agradezco que no lo hicieran. Gracias.

A mis compañeros de aula que siempre me apoyaron con las pruebas y me dieron ánimos. Gracias

A los profesores que poco a poco me enseñaron tantas cosas y me prepararon para convertirme en una futura ingeniera.

A mi compañero de tesis por ayudarme con el trabajo.

Y por último pero no por eso menos importantes a mis tutoras por apoyarme y sacarme adelante en los últimos momentos del transcurso de este viaje.

Eliener

Agradecer primero a todas esas personas que durante estos 5 años de la carrera me han apoyado y me han ayudado para poder llegar a este momento.

Agradecerle también a mi compañera de tesis por ser el hilo conductor de este trabajo de diploma.

También a mis tutoras Aíme, Yunet y Yanet por su confianza durante el año.

También quiero agradecer a mi líder de proyecto Solangel por apoyarme tanto profesional como emocionalmente.

Y para terminar y no por ser los últimos los menos importantes, agradecer a mis hermanos del alma Rojas, Rainer y Maxwell por su apoyo incondicional hacia mí en la realización de esta tarea.

Elizabeth

Dedico este trabajo de diploma a toda mi familia quienes han sido siempre el pilar de apoyo en mi vida tanto personal como profesional.

A mis amigos que siempre han estado conmigo compartiendo buenos y malos momentos.

A todos gracias de corazón.

Eliener

Dedico este trabajo de diploma primeramente a mis padres Aida y Elías por ser el motor impulsor de que este sueño se haya hecho realidad y por haberme apoyado y brindarme toda su confianza. También dedico este trabajo de diploma a mis abuelos Juanico, Florentino y en especial a mi abuela Flora y a mi abuela Nélida que no se encuentra mi lado pero estoy seguro que estaría orgullosa de mí. A mis hermanas Yenis y Arazay y a mi cuñado Manolo por sus consejos cotidianos de la vida.

A todos muchas gracias.

Resumen

Las actualizaciones constituyen un proceso importante y efectivo para el mantenimiento y desarrollo de todo tipo de software. Están orientadas a corregir vulnerabilidades y fallos de seguridad, además de incorporar nuevas funcionalidades a un determinado sistema. Para lograr el mantenimiento de los diversos software que se desarrollan en el centro Geoinformática y Señales Digitales (GEYSED) de la Universidad de las Ciencias Informáticas (UCI), es necesario fomentar el uso del proceso de actualización debido a las ventajas que les proporciona.

Una vez examinada la necesidad explicada anteriormente, se propuso realizar un sistema con el objetivo de gestionar actualizaciones para aplicaciones de escritorio desarrolladas en el centro GEYSED. De esta forma se garantizó que las actualizaciones de cada aplicación, se almacenen en un servidor, del cual los usuarios accederán a través de un sistema, brindándole de esta manera, la facilidad de actualizar dichas aplicaciones.

A través de métodos teóricos se realizó un estudio y análisis de las características principales de herramientas que posibilitan actualizar un software. RUP, Visual Paradigm, NetBeans, Java y el servidor FileZilla forman parte de la selección de tecnologías, herramientas y metodología que posibilitaron la obtención de un sistema que da solución a la problemática existente dentro del departamento. Las pruebas realizadas a la plataforma obtuvieron resultados satisfactorios, lo que garantiza el correcto funcionamiento de la aplicación y la satisfacción del cliente.

PALABRA CLAVE: actualizaciones, servidor.

Abstract

The updates are an important and effective process for the development and maintenance of any kind of software. They are oriented to correct vulnerabilities and security failures. They also have the abilities to add new functionalities to software. In order to obtaining the maintenance of different software that are developed in the center Geoinformatics and Digital Signal (GEYSED) of University of Informatics Sciences (UIC), is necessary to encourage the use of this type of update on it, given the benefits that provides them.

Having examined the need previously explained, it was proposed a system with the objective of manage updates for desktop applications developed in the center GEYSED. This ensured that each application updates, are stored on a server, which users access through a system, thus providing the facility to update those applications.

Through a study of theoretical methods, a study was made about the main characteristics of tools that allows update software. RUP, Visual Paradigm, NetBeans, Java and FileZilla server are the tools and methodologies that allowed the creation of a system that provides a solution to the existing problems on the center. Tests performed at the system were successful, ensuring the correct functioning of the application and customer satisfaction.

KEYWORDS: updates, server.

Índice

Índice

Índice de Figuras

Índice de Tablas

Introducción

El avance tecnológico ofrece a la sociedad herramientas que permiten una mejor comunicación, facilidad de asociación, reducción del tiempo empleado en el desarrollo de actividades e incluso mejoras en la tecnología ya existente. Los usuarios son cada vez más exigentes en cuanto a la obtención de aplicaciones o sistemas que satisfagan sus necesidades, tanto laborales como personales. Esto trae consigo la creación de nuevas actualizaciones, que permitan el mejoramiento de productos existentes o solucionar problemas específicos del software.

Estas actualizaciones se pueden realizar de forma manual o automática. Cuando el usuario es el encargado de buscar en la red, descargarla e instalarla en la aplicación o sistema a la cual corresponda dichas actualizaciones, es cuando se efectúan de forma manual. Otro procedimiento sería trasladándose hacia la entidad u ordenador donde se encuentre dicha actualización, con el fin de obtenerla e instalarla en su equipo. Sin embargo, puede existir la posibilidad de que la actualización que se necesita posea dependencias de otros componentes, de los cuales el usuario no tiene conocimiento, provocando que el sistema o aplicación no se ejecute correctamente. (Garcia, 2008)

El proceso de actualización de forma automática surge con la necesidad de dar solución a las dificultades que trae consigo llevar a cabo el proceso de forma manual. Una de sus ventajas es que solamente sería necesario tener conocimiento de la dirección electrónica donde se encuentren almacenadas estas actualizaciones. Es el propio *software* el encargado de realizar los procesos de obtención e instalación del archivo, así como de sus dependencias. En muchas ocasiones las aplicaciones y sistemas ya vienen configurados con dicha dirección. Esto constituye una vía para mejorar funcionalidades, eliminar fallos de seguridad y corregir errores de una aplicación.

En la actualidad la mayoría de las empresas y compañías gestionan las actualizaciones de forma automática, debido a que estas están desarrolladas en un único sistema operativo o por la misma institución, permitiendo que el sistema este conformado en una misma arquitectura. Sin embargo existen casos donde las empresas contratan clientes para desarrollar productos que resuelvan una necesidad real.

Muchas entidades aun utilizan la forma manual debido a que no poseen los recursos necesarios para realizar el proceso automáticamente. Ya sea que no posean acceso a un servidor al que se pueda acceder de forma remota, donde estén almacenadas las actualizaciones o el sistema que poseen no es

capaz de realizar el proceso de actualización por sí mismo. Además los usuarios se ven en la necesidad de comprobar manual y periódicamente si la versión utilizada es la última disponible y, por tanto, la más segura. (Microsoft, 2009)

"Los mecanismos de actualización, automáticos o no, están orientados a resolver los problemas de soporte y mantenimiento de los sistemas y aplicaciones, para hacer un uso efectivo de las redes de información y evitar costosos traslados de personal calificado." (Alegsa, 2011)

En el centro Geoinformática y Señales Digitales (GEYSED) de la Universidad de las Ciencias Informáticas (UCI) el mecanismo de actualización para los sistemas que desarrolla es de forma manual. Para ello es necesario recompilar el sistema completo creándose un instalador o compilando directamente bibliotecas de las que depende y reemplazándolas en el lugar donde estén instaladas. En dichos sistemas pueden surgir nuevas funcionalidades, modificaciones en sus procesos, así como en la corrección de deficiencias detectadas con respecto a su funcionamiento. Es posible que estas acciones conlleven a la pérdida de configuraciones, errores de dependencias entre bibliotecas, disminución de la productividad y aumento del tiempo empleado por parte de los desarrolladores, debido a que tengan que transportarse hasta cada una de las estaciones de trabajo de los proyectos o entidades donde se necesite una actualización. Esto no asegura que los involucrados en este proceso porten el paquete de actualización completo y libre de errores.

Teniendo en cuenta la problemática anteriormente planteada se define el siguiente **problema de la investigación:** ¿Cómo disminuir el tiempo empleado en las actualizaciones de los sistemas desarrollados en el Centro GEYSED y desplegados en sus proyectos o entidades clientes y las probabilidades de error a partir de su ejecución?

El **objeto de estudio** que se establece para darle solución a la problemática planteada anteriormente es: el proceso de actualización de los sistemas de software, determinando como **campo de acción:** el proceso de actualizaciones en aplicaciones de escritorio desarrolladas en el centro GEYSED.

Para dar solución al problema propuesto se define como **objetivo general:** desarrollar un sistema que permita la gestión de actualizaciones para las aplicaciones de escritorio del centro GEYSED.

Se definen las siguientes **preguntas científicas** a las que se dará respuesta durante el desarrollo de la investigación.

- ¿Cómo se realiza el proceso de actualización de los sistemas informáticos del centro GEYSED?
- ¿Qué características debe cumplir el *software* a desarrollar para que posibilite el proceso de actualización de aplicaciones de escritorio desarrolladas en el centro GEYSED?
- ¿Qué tipo de ficheros se actualizan en los proyectos que desarrollan aplicaciones de escritorio?
- ¿Cómo garantizar que el sistema cumpla con las funcionalidades definidas para posibilitar el proceso de actualización de aplicaciones escritorio desarrolladas en el centro GEYSED?

Con el fin de darle cumplimiento al objetivo general y al problema anteriormente planteado se trazaron las siguientes **tareas de la investigación:**

- Identificación de los procesos necesarios para la gestión de actualizaciones.
- Caracterización del estado del arte a nivel nacional e internacional de soluciones que brinden servicios de actualizaciones.
- Caracterización de las herramientas y tecnologías a utilizar en el desarrollo de la solución.
- Diseño de los artefactos que se involucran en el desarrollo de la solución.
- Implementación de la propuesta de solución.

Con el propósito de garantizar el correcto desarrollo de la investigación se utilizaron los siguientes métodos científicos:

Métodos Teóricos:

- **Histórico – Lógico:** Se utilizó para el análisis de la evolución histórica de las herramientas similares al sistema a desarrollar en cuanto a las actualizaciones, para valerse de punto de referencia y comparación. Para ello se realiza un estudio de la información existente en el periodo que comprende desde el año 2000 hasta la fecha.
- **Analítico – Sintético:** Se utilizó en el análisis de bibliografías especializadas en el tema de actualizaciones, para lograr la extracción de los elementos más importantes que se relacionen con el objeto de estudio planteado.

Técnica de recopilación de información:

- **Entrevista:** Se utilizó con el propósito de obtener información acerca del proceso de actualización de los sistemas en desarrollo de los proyectos del centro GEYSED. Se llevó a cabo mediante la realización de preguntas a los jefes de los proyectos, para determinar los inconvenientes que trae consigo la actualización de forma manual de los sistemas desarrollados. Esta técnica proporcionará los elementos necesarios para realizar el levantamiento de requisitos funcionales con los que debe cumplir la solución. Se utiliza el tipo de entrevista no estructurada, debido a que el entrevistador que realiza las preguntas puede cambiarlas de acuerdo a la persona que esté entrevistando.

Estructura del Documento:

CAPÍTULO 1: Fundamentación Teórica asociada a la utilización de servicios de actualizaciones.

En este capítulo se analizan los principales conceptos asociados a los sistemas de actualizaciones. Se efectúa el estudio de diferentes sistemas que realizan actualizaciones, se caracterizaron y seleccionaron las herramientas y tecnologías que se consideraron adecuadas para el desarrollo de la solución propuesta.

CAPÍTULO 2: Presentación de la solución propuesta.

En este capítulo se realiza el análisis y diseño del sistema a desarrollar. Se especifican los requisitos, funcionales y no funcionales, con los que debe cumplir el sistema. Además se establecen los patrones y artefactos que sirven como punto de partida para la implementación del sistema.

CAPÍTULO 3: Implementación y Prueba.

En este capítulo se realiza la implementación del sistema de actualizaciones y se lleva a cabo el proceso de pruebas para validar su funcionamiento, en función del objetivo trazado.

Capítulo 1: Fundamentación teórica asociada a la utilización de servicios de actualizaciones.

Introducción

En este capítulo se establecen los conceptos asociados a los sistemas de actualizaciones, con el propósito de brindar al lector un mayor dominio del problema. Por otra parte se realiza el análisis de herramientas que realizan procesos similares al sistema que se desea desarrollar, que servirán como punto de partida para el desarrollo de la solución. Además se realiza un análisis de las tecnologías y herramientas actuales, que se utilizan en el desarrollo de sistemas que permiten realizar el proceso de actualización, haciendo énfasis en las seleccionadas para el desarrollo de la propuesta de la solución. Para ello, se tiene en cuenta la estandarización de tecnologías que se lleva a cabo en el departamento. Por ende se abordan elementos relacionados con la metodología de desarrollo, el lenguaje de modelado, las herramientas y lenguaje de programación a utilizar.

1.1 Conceptos asociados al dominio del problema

A continuación se brindan varios conceptos que se consideran indispensables para el correcto entendimiento de la investigación realizada.

Actualización Informática

En *software* la acción de cambiar o alterar una aplicación o sistema operativo por una versión más actual es denominada actualización. Es la tarea o actividad que supone la puesta al día y/o sustitución de una información contenida en un registro o archivo por otra más reciente. Dicha actualización puede ser capaz de corregir defectos, mejorar un programa o ponerlo al día. (Alegsa, 2011)

Como parte del dominio del problema el término tratado anteriormente no es más que la operación llevada a cabo por los proyectos del Centro, en función de sustituir versiones obsoletas de aplicaciones por otras más recientes desarrolladas sobre tecnologías que según sus características puedan recibir soporte. Otras de las razones para actualizar una aplicación sería la adicción de nuevas funcionalidades, para cumplir con nuevos requisitos solicitados por un cliente o corregir errores detectados durante su uso. Una

actualización puede estar constituida por diferentes archivos, ya sean documentos, directorios, ejecutables o bibliotecas dinámicas.

Servicios informáticos

Son los medios para entregar valor a los clientes, facilitando sus tareas para obtener resultados, sin que los clientes deban asumir los costos específicos ni los riesgos asociados. (Martínez, 2010)

Conjunto de actividades (planeamiento, análisis, diseño, programación, operación, entrada de datos, autoedición) asociadas al manejo automatizado de la información que satisfacen las necesidades de los usuarios de este recurso.

Es necesario que el sistema a desarrollar brinde los servicios de obtención, actualización e instalación de cada uno de los archivos almacenados en el servidor.

Servidor

Un servidor, es un ordenador o máquina informática que está al servicio de otras máquinas, ordenadores o personas llamadas clientes y que le suministran a estos, todo tipo de información. (García, 2013)

Los servidores suelen utilizarse para almacenar archivos digitales. Los clientes, por lo tanto, se conectan a través de la red con el servidor y acceden a dicha información. En ocasiones, un ordenador puede cumplir con las funciones de servidor y de cliente de manera simultánea. (García, 2013)

Servidor FTP[1]

Un servidor FTP es un programa especial que se ejecuta en un servidor conectado normalmente en Internet (aunque puede estar conectado en otros tipos de redes, LAN, MAN, etc.). La función del mismo es permitir el desplazamiento de datos entre diferentes servidores / ordenadores. (ServidorFTP, 2011)

El servidor FTP en el caso del sistema a desarrollar será el ordenador donde estarán almacenadas las actualizaciones que serán gestionadas por el administrador. Además es donde se le darán los permisos a los usuarios, ya sean clientes como administradores para tener acceso a dichas actualizaciones.

[1] *File Transfer Protocol* (Protocolo de transferencia de archivos).

Capítulo 1: Fundamentación teórica asociada a la utilización de servicios de actualizaciones.

1.2 Descripción general del objeto de estudio

El número de empresas que utilizan sistemas operativos y aplicaciones con servicios de actualizaciones automáticas ha ido incrementando considerablemente. Ejemplo de ello son Microsoft con Windows, Apple con Mac OS X y Ubuntu con Gestor de Actualizaciones. Por otra parte existen navegadores que realizan el proceso de actualización de forma automática; como Chrome, Firefox, Opera. En términos de seguridad, algunos antivirus como Avira y Kaspersky realizan dicho proceso para mantener la integridad de los equipos que protegen. (inteco cert, 2009)

Los procesos de actualización de sistemas informáticos buscan corregir en gran medida las vulnerabilidades intrínsecas del *software* inicialmente instalado. Con ello se evita que puedan ser utilizados por cibercriminales[2] para obtener datos confidenciales del usuario e instalar *software* maliciosos con el fin de dañar un determinado sistema.

El hecho de poder actualizar aplicaciones informáticas desde la propia estación de trabajo ofrece la gran ventaja de que el *software* empleado cuente con nuevas prestaciones que ayuden a la seguridad del mismo y a su manipulación. Esto permite al usuario adquirir, según sus necesidades, las funcionalidades que mejor se adapten al *software* que utiliza. (Alegsa, 2011)

Un *software* instalado está en constante evolución a través de extensiones, mantenimiento, cambios en los requisitos o en la configuración. Manejar esta evolución es un problema complejo y frecuentemente subestimado, que puede ser causa de dificultades tanto para el cliente, como para el proveedor del *software*. (Garcia, 2008)

La acelerada evolución de las tecnologías de información, la creciente complejidad de los requisitos de negocio, los continuos cambios en el contexto de desarrollo, entre otros aspectos, ocasiona una rápida obsolescencia del *software*. El proceso de actualización de *software* puede ser visto como adición, eliminación, reemplazo o reconfiguración de las funcionalidades del *software*. (Garcia, 2008)

Entre las formas utilizadas para realizar el proceso de actualización de un *software* se encuentran: manual y automática. Con respecto a la manual, es indispensable que el usuario invierta tiempo en conectarse a

[2] Personas que realizan actividades delictivas en internet; ejemplo de ello, robar información, acceder a redes privadas, estafas.

los servidores o páginas web de los fabricantes en la red, en búsqueda del archivo del programa que considere necesario para actualizar una determinada aplicación o sistema. Aunque la ejecución de dicho proceso es muy simple se pueden encontrarse problemas con la cantidad de tiempo que se invierte en la realización de las operaciones que engloba, además cuando una actualización depende de algún otro archivo y el usuario no está consciente de esta necesidad pueden existir fallos en la ejecución del programa o incluso cuando este conoce sobre estas dependencias tendrá que realizar todo el proceso de búsqueda nuevamente para adquirirlas.

Aun sabiendo los inconvenientes que presenta el proceso de actualización de forma manual, es elegido por algunos usuarios. En cambio otros prefieren realizarlo de forma automática, pues debido a los avances de la tecnología informática, el propio *software* a actualizar es el encargado de buscar las actualizaciones, descargarlas e instalarlas.

En algunos casos el *software* no tiende a actualizarse por sí mismo, sino que necesitan de herramientas que permitan reemplazar su versión obsoleta. Un sistema de actualizaciones automáticas debe estar compuesto por al menos dos aplicaciones: un servidor y un cliente para realizar la transmisión de los datos que se desean actualizar de forma automática, sin necesidad de que exista un personal capacitado atendiendo este proceso. (Microsoft, 2009)

Se han creado herramientas con el propósito de encargarse de las actividades relacionadas con la actualización del *software*. La creación de una actualización y su publicación dio paso a la instauración de sistemas de gestión de actualizaciones con el fin de mantener los programas actualizados.

Un sistema para la gestión de actualizaciones, está conformado por diferentes aplicaciones destinadas a automatizar el proceso de instalación, actualización, configuración y eliminación de paquetes de *software*. El *software* se distribuye en forma de paquetes, frecuentemente encapsulado en un solo archivo. Estas actualizaciones incluyen información importante, por ejemplo su nombre completo, una descripción de su funcionalidad, la versión y una lista de otras actualizaciones requeridas para el correcto funcionamiento del *software*. (UCI, 2012)

Las actualizaciones se ponen a disposición de los usuarios en los repositorios o servidores, con el fin de proporcionar un control sobre los diferentes tipos de *software* que van a ser instalados en su sistema.

Capítulo 1: Fundamentación teórica asociada a la utilización de servicios de actualizaciones.

Los sistemas encargados de realizar el proceso de actualización están enmarcados en la utilización de gestores de actualizaciones, los cuales poseen como principal objetivo el correcto funcionamiento de los programas instalados en el ordenador. Para llevar a cabo el proceso inicialmente mencionado se debe partir de las siguientes tareas:

- Comprobación de la diferencia entre la versión existente de la actualización y la última a almacenar en el servidor.
- Instalación, actualización y eliminación simple de actualizaciones.
- Obtención de dependencias para garantizar que el *software* funcione correctamente.
- Agrupamiento de las actualizaciones según su función para evitar la confusión al instalarlas y realizar el mantenimiento de los programas.

1.3 Situación Problemática

La actualización de forma manual de las aplicaciones de escritorio desarrolladas por los proyectos del centro GEYSED es un problema vigente. Una vez desplegado el *software* en la entidad del cliente o en el mismo proyecto donde fue desarrollado, si ocurre algún error o problemas con la instalación, es reportado al centro lo ocurrido y los programadores proceden a realizar una actualización del sistema en cuestión.

En caso de que el informe de error sea emitido por una entidad cliente es necesario trasladarse hacia ella, en función de instalar la actualización creada o reinstalar el sistema si es requerido, solucionando el error ocurrido. Sin embargo, si esto sucede a nivel de proyecto este procedimiento se realizaría de igual forma, lo que garantizando un enfoque más rápido y sencillo. Esto se debe a la constante interacción del jefe de proyecto y su equipo con el software desarrollado.

Los desarrolladores, al trasladarse una y otra vez hacia donde está instalado el *software,* generan pérdida de tiempo, el cual podría ser empleado en otras tareas asociadas al desarrollo de otros sistemas. Además no existe una forma de verificar si el *software* tendrá otras dificultades o errores, generados a partir de los que fueron reportados por la entidad cliente. La necesidad de actualizar componentes específicos de un sistema, que por sus características dependan de un conjunto de otros componentes, es considerado uno de los errores más comunes. En la mayoría de los casos esto se debe a que no existe una gestión de dependencias entre los componentes, por lo que la necesidad de un sistema capaz de realizar esta

operación, sería de gran ayuda, no solo para los programadores, sino para las propias entidades donde estos sistemas son desplegados.

1.4 Análisis de las herramientas similares

A continuación se realiza un estudio de herramientas similares al sistema a desarrollar, con el propósito de tomar como base sus principales características y funcionalidades.

Updater Application Block

Updater Application Block, fue creado por Microsoft, es una biblioteca que puede agregarse a su aplicación para administrar la descarga de las partes de la aplicación a través de HTTP[3]. Se puede utilizar para detectar**,** descargar e instalar las actualizaciones de la aplicación cliente desplegada en una ubicación central. Las actualizaciones son procesadas y se muestran todos los archivos de la nueva versión de la aplicación. Al utilizarse se puede mantener las aplicaciones clientes al día con poca o ninguna intervención del usuario. También se puede extender para utilizar clases personalizadas para la descarga de archivos y para realizar tareas de configuración posteriores a la implementación**.**

La herramienta presenta las siguientes ventajas primordiales:

- Implementa un "modelo de extracción" para descargar automáticamente las actualizaciones para aplicaciones que utilizan o han sido desarrolladas sobre el *framework .NET.*
- Realiza las tareas de configuración posteriores a la transferencia directa sin necesidad de intervención del usuario. (Microsoft, 2009)

De la herramienta Updater Application Block se toma como referencia la característica que tiene de obtener actualizaciones almacenadas en un servidor central y listarlas para los clientes presentandoles la obcion de escoger la que desea actualizar. Pero no se utiliza como solución para la problemática debido a que:

- Es una aplicación que solo se utiliza en el sistema operativo Windows.

[3] *Hypertext Transfer Protocol* o HTTP (en español protocolo de transferencia de hipertexto)

- Es solo para descargar automáticamente las actualizaciones para aplicaciones que utilizan o han sido desarrolladas sobre el *framework .NET* o sea es solamente orientada a un producto.
- Expone las actualizaciones sobre el protocolo HTTP.

Kaspersky Administration Kit

Es la herramienta utilizada para realizar el proceso de actualización del programa Kaspersky. Las actualizaciones regulares de las bases de datos y las aplicaciones de protección contra amenazas, implican que Kaspersky Administration Kit realice verificaciones constantes y garantiza la actualización permanente de su infraestructura de seguridad. (ZAO, 2013)

Desarrollada por Kaspersky Lab, *Kaspersky Administration Kit* es una herramienta que facilita el trabajo del administrador ya que permite organizar y monitorizar de forma centralizada la protección de toda una red corporativa, consolidando diferentes capas de protección en un sistema integrado. Está diseñado para controlar y mantener de forma remota y centralizada los sistemas de protección antivirus de Kaspersky Lab. Permite entre otras actividades administrar de forma centralizada la descarga de actualizaciones para bases de datos y módulos de aplicaciones y distribuirlas a todos los equipos de red.

Este servicio contiene un agente de actualización que consiste en un equipo especial dentro de la red del servidor de administración. Su destino es guardar y distribuir las bases actualizadas, los paquetes de instalación, las tareas y directivas de grupo. El servicio puede usar la multidifusión IP (*Internet Protocol*) para distribuir los paquetes de instalación, así como para enviar un paquete de instalación a todos los equipos del grupo que aún no han sido asignados para el establecimiento del producto. Kaspersky también utiliza los beneficios de los protocolos FTP y HTTP para la descarga de las actualizaciones automáticas. (Kaspersky Lab, 2011)

Ventajas de la herramienta:

- Mayor seguridad.
- Instalación remota.
- Fácil administración de la protección.
- Control continuo (supervisión de la protección antivirus).

Capítulo 1: Fundamentación teórica asociada a la utilización de servicios de actualizaciones.

De la herramienta Kaspersky Administration Kit se toma como referencia el mecanismo de conexión con un servidor centralizado para obtener e instalar la actualización en el producto Kaspersky que se tenga desplegado en el equipo cliente. Permitiendo que sea el cliente quien decida cuándo realizar el proceso de actualización. Pero no se utiliza como solución para la problemática debido a que:

- Es un sistema orientado a la actualización de una gama de productos de la empresa Kaspersky Lab.
- Poseen una arquitectura definida para todos sus productos.

Debian Package Management

El sistema de paquetería de Debian se compone por un conjunto de paquetes que contienen los ficheros de configuración, la documentación y una lista con sus dependencias. Es importante comprender que las dependencias son archivos que dependen de otros para su correcto funcionamiento. Debian Package (Dpkg[4]) es la base del sistema de gestión de paquetes de Debian GNU/Linux[5]. Se utiliza para instalar, actualizar, quitar y proporcionar información acerca de los paquetes .deb. Utiliza un front-end[6] para resolver conflictos complejos en las dependencias de paquetes (APT[7]).

APT es un sistema de gestión de paquetes creado por el proyecto Debian que simplifica en gran medida la instalación y eliminación de programas. Dicho sistema buscará en su base de datos para encontrar la versión más reciente del paquete y lo descargará del servidor correspondiente. Si este paquete necesitara otro para funcionar, APT resolverá las dependencias e instalará los paquetes necesarios. Ejemplos de programas basados en APT son: *Synaptic Package Manager*, Gestor de paquetes de Adept. El usuario interactúa con estos programas con el propósito de buscar las actualizaciones que deseen e instalarlas. (Dassen, y otros, 2012)

[4] Dpkg (Debian Package - Paquetería en Debian)

[5] Sistemas operativos basados en GNU con núcleo Linux.

[6] El *front-end* es la parte del software que interactúa con el o los usuarios.

[7] APT (Advanced Packaging Tool - Herramienta Avanzada de Empaquetado)

Capítulo 1: Fundamentación teórica asociada a la utilización de servicios de actualizaciones.

De la herramienta Debian Packege Management se toma como referencia la característica en que el sistema de actualizacion que posee almacena las actualizaciones en un servidor, para luego descargarlas con un gestor de actualizaciones. Pero no se utiliza como solución para la problemática debido a que:

- Utiliza un sistema de comandos para administrar las actualizaciones que se van a almacenar. Aunque es fácil aprender a trabajar en consola por comandos para los futuros administradores les es mejor hacerlo de manera visual.
- Es solamente para los sistemas basados en Linux.

Sistema Operativo Nova

El proyecto Nova fue lanzado oficialmente el 15 de febrero de 2011. Es una distribución de GNU/Linux desarrollada en la UCI por estudiantes y profesores y con la participación de miembros de otras instituciones, con razón de apoyar la migración del país a tecnologías de *Software* Libre y Código Abierto. NOVA es una distribución de GNU/Linux, conjunto de programas desarrollado por la comunidad internacional, que comparte la característica de ejercer códigos abiertos o *software* libres, y permite su modificación y redistribución bajo determinadas condiciones, que ceden los derechos patrimoniales a quienes los usan. (UCI, 2012)

El sistema operativo Nova cuenta con dos Gestores de actualizaciones (el Nova Escritorio y el Nova Ligero). Ambos son sistemas similares encargados de verificar todos los paquetes instalados en el ordenador y comparar con el repositorio (que ya está configurado), si la versión almacenada existente es superior a la que se tiene instalada en dicho ordenador. Estas versiones son listadas permitiendo al usuario seleccionar los paquetes deseados. Para actualizar simplemente se descarga la versión nueva y es eliminada la existente.

En cuanto a la actualización del repositorio, se realiza mediante los ficheros de configuración, en el que se define la dirección del mismo. Para poder actualizarlo se ejecuta un comando que permite la conexión a este mediante la dirección entrada. En la raíz del repositorio existe un fichero de paquetes (package) donde se encuentra la lista de todos los paquetes existentes, con sus índices, compuestos por el nombre, versión y dirección. Lo siguiente que se realiza es descargar ese fichero y empieza a revisar cada uno de esos índices, formando una base de datos con todos los paquetes que existen en el repositorio, para luego subir esas nuevas actualizaciones al repositorio.

De la herramienta Nova se toma como referencia la característica de que el sistema de actualizacion que posee es capaz de almacenar en un repositorio las actualizaciones, para luego descargarlas con un gestor de actualizaciones. Pero no se utiliza como solución para la problemática debido a que:

- Utiliza un sistema de comandos para administrar las actualizaciones que se van a almacenar. Aunque es fácil aprender a trabajar en consola por comandos para los futuros administradores les es mejor hacerlo de manera visual.
- No es un sistema multiplataforma.

Después de realizado dicho análisis se comprueba, a partir del porqué de la no utilización de los sistemas existentes, que para darle solución a la problemática que se tiene en el centro GEYSED, es necesario realizar un sistema capaz de gestionar y descargar las actualizaciones para las aplicaciones de escritorio que se desarrollan en dicho centro.

Para el desarrollo de la propuesta de solución, se hizo necesario realizar el estudio y selección de las metodologías, herramientas y tecnologías que se pueden utilizar para obtener un producto con la calidad requerida, con un buen funcionamiento y que satisfaga las necesidades del cliente.

1.5 Metodología, herramientas y tecnologías empleadas en el desarrollo de la solución

Para el desarrollo del *software* se realizan diferentes tareas, dentro de las que se encuentran el estudio de las tendencias actuales, herramientas y tecnologías con el objetivo de encaminar el proceso de desarrollo de la solución.

1.5.1 Metodología de software

Una metodología de desarrollo de *software* es un conjunto de actividades con un orden lógico, que haciendo uso de herramientas, garantizan que cuando termine la última actividad se tenga un producto, que cumpla con los requisitos que estableció el cliente al inicio del desarrollo de *software*.

El Proceso Unificado de Desarrollo de Software (RUP)

RUP constituye un marco de trabajo o metodología estándar de desarrollo capaz de soportar el ciclo de vida de un *software*. Contiene un proceso iterativo e incremental, dirigido por requisitos o casos de uso, centrado en la arquitectura y enfocado a la gestión del riesgo y posee características adaptables a las organizaciones y proyectos de *software*. Utiliza el lenguaje de modelado UML[8] para preparar todos los esquemas de un sistema. (Jacobson, y otros, 2010)

- **Iterativo e incremental**, lo que permite identificar y procesar un conjunto de artefactos por fase que se liberan como resultado de una iteración, para trazar los documentos y modelados de la solución que se desea desarrollar.
- **Dirigido por requisitos o casos de uso**, lo que permite la utilización de casos de uso en el proceso, desde su conceptualización hasta su despliegue, permitiendo satisfacer las necesidades del cliente en cada una de las funcionalidades deseadas.

Justificación de la metodología seleccionada

Se selecciona RUP como guía para el desarrollo del sistema, con el fin de lograr una buena estandarización de metodologías con el centro, puesto que es la metodología que se utiliza. Además el equipo de desarrollo está familiarizado con la metodología, lo que constituye una ventaja a la hora de trabajar.

1.5.2 El Lenguaje Unificado de Modelado (UML) como soporte de la modelación de la solución propuesta

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software*. Captura decisiones y conocimientos sobre los sistemas que se deben construir. Este indica cómo crear y leer los modelos, pero no dice cómo crearlos. Lenguaje cuyo vocabulario y reglas se centran en la representación conceptual y física de un sistema. (Jacobson, y otros, 2010)

Las funciones de UML:

- **Visualizar:** permite expresar de forma gráfica un sistema, de forma que otro lo pueda entender.

[8] Lenguaje Unificado de Modelado.

- **Especificar:** permite especificar cuáles son las características de un sistema antes de su construcción.
- **Construir:** a partir de los modelos especificados se pueden construir los sistemas diseñados.
- **Documentar:** los propios elementos gráficos sirven como documentación del sistema desarrollado y pueden servir para su futura revisión.

Justificación del lenguaje modelado seleccionado

Se decidió utilizar UML en su versión 2.0 para el desarrollo de los diagramas, debido a que facilita la visualización de los procesos con que contará el sistema. Además, los diagramas generados sirven como artefactos para la documentación que sustenta el sistema a desarrollar y pueden ser utilizados como base para el desarrollo de futuras versiones.

1.5.3 Herramienta CASE (Computer Aided Software Engineering)

Las herramientas CASE son un conjunto de métodos, utilidades y técnicas que facilitan el modelado de un *software*.

Visual Paradigm for UML 8.0

Permite aplicar la ingeniería tanto directa como inversa, además de ser una herramienta colaborativa por lo que es posible que varios usuarios trabajen sobre el mismo proyecto. La herramienta facilita la visualización y manipulación del proyecto de modelado. Posee una gran cantidad de módulos para facilitar el trabajo de desarrollo de un *software* y garantizar la calidad del producto final. Una de sus características principales es que está disponible para múltiples sistemas operativos, por lo que se pudiera decir que es multiplataforma. También es compatible con los cambios específicos en el código fuente de algunos lenguajes de programación como C + + y Java. (VPository, 2010)

Justificación de la herramienta CASE seleccionada

Se selecciona Visual Paradigm for UML 8.0 debido a que, como su nombre lo indica, utiliza el lenguaje de modelado UML. Al ser una herramienta multiplataforma permite su utilización en varios sistemas operativos. Es compatible con la metodología seleccionada y permite generar documentación para la

elaboración de la propuesta de solución, haciéndola más entendible. Es fácil de manejar y se utiliza en la todos los proyectos del centro.

1.5.4 Entorno Integrado de Desarrollo (IDE)

El Entorno Integrado de Desarrollo o en inglés *Integrated Development Environment* (IDE) es un programa compuesto por un conjunto de herramientas para un programador. Proporcionan un marco de trabajo amigable para la mayoría de los lenguajes de programación. Un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. Algunos IDEs pueden funcionar con varios lenguajes de programación y estos pueden ser multiplataforma. (Maldonado, 2007)

1.5.4.1 NetBeans 8.0

Es un entorno de desarrollo gratuito y de código abierto, soporte a casi todas las novedades en el lenguaje Java. Simplifica la gestión de grandes proyectos con el uso de diferentes vistas, asistentes de ayuda, y estructurando la visualización de manera ordenada, lo que ayuda en el trabajo diario. Permite desarrollar las aplicaciones mediante módulos, lo que facilita la posterior extensión de dichas aplicaciones. Ofrece autocompletado de código, marcado de error para el lenguaje Java. Realiza las tareas asociadas a la programación: editar el código, compilarlo, ejecutarlo, depurarlo[9]. (Oracle Corporation, 2013)

Justificación del IDE seleccionado

Después de haber realizado el estudio del IDE se llega a la conclusión de que presenta las características necesarias para dar cumplimiento al objetivo general trazado. Esto se debe a que permite construir una interfaz gráfica, con las que el usuario puede identificarse, además de estar diseñado para realizar aplicaciones capaces de desplegarse en varios sistemas operativos. Cuenta con abundante bibliografía para aprender a trabajar con él.

Bibliotecas utilizadas

Jdk-7

[9] El debuggin o depuración es el proceso metodológico para encontrar y reducir bugs (errores) o defectos en un programa informático o en una pieza de hardware.

Java Development Kit o (JDK), es un *software* que provee herramientas de desarrollo para la creación de programas en java. Puede instalarse en una computadora local o en una unidad de red. El JDK cuenta con un compilador que permite convertir código fuente en bytecode, es decir, el código "máquina" de la máquina virtual de Java. (Java, 2008)

Apache Poi 3.9

Apache POI se utiliza en aplicaciones de extracción de texto, tales como arañas web, constructores de índices, y los sistemas de gestión de contenidos. Proporciona bibliotecas Java para archivos de lectura y escritura en los formatos de Microsoft Office, como Word, PowerPoint y Excel. (Foundation, 2013)

Edtftpj-pro 4.7.0

EdtFTPj/PRO permite transferir archivos de cualquier tipo a través de internet de manera segura utilizando FTPS, es decir FTP ejecutado en SSL. Es una librería Java que se encarga de realizar este tipo de transferencias para evitar que los datos puedan ser interceptados por usuarios maliciosos que buscan realizar acciones fraudulentas. Es completamente compatible con la mayoría de los servidores FTP que se pueden encontrar en la red, de esta manera asegura una versatilidad a toda prueba. El sistema de seguridad de esta librería es bastante completo y ofrece soporte para cortafuegos lo que garantiza una transmisión cifrada de los datos. (Technologies, 2012)

JdirectoryChooser

JdirectoryChooser proporciona GUI[10] fáciles de usar para la manipulación de directorios en Java. El usuario puede simplemente elegir un directorio existente de estructura de árbol o editar archivos de los directorios del sistema. Algunas de sus funcionalidades son: (JTechLabs, 2006)

- Crear nuevos directorios.
- Eliminar directorios existentes.
- Cambiar el nombre de los directorios.
- Copiar directorios.

[10] La interfaz gráfica de usuario, conocida también como GUI (del inglés graphical user interface)

- Mover directorios.

1.5.5 Lenguaje de Programación

Java

Java es un lenguaje orientado a objetos. Es compilado, las clases que genera son interpretadas por la máquina virtual de Java, la cual mantiene el control sobre las clases que se estén ejecutando. Es un lenguaje multiplataforma, el mismo código Java que funciona en un sistema operativo, funcionará en cualquier otro que tenga instalada la máquina virtual. La máquina virtual al ejecutar el código, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros. (Java, 2009)

Java elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el reciclador de memoria dinámica. Reduce en un 50% los errores más comunes de programación con lenguajes como C y C++ al eliminar muchas de las características de éstos. (Java, 2008)

Justificación del lenguaje de programación seleccionado

Se utiliza Java como lenguaje de programación debido a que es un lenguaje relativamente fácil de aprender y se puede hacer de manera gradual. Cuenta con una gran comunidad de desarrolladores y foros lo que posibilita la aclaración de dudas con bastante rapidez. Además, por su gran utilización en el mundo, la bibliografía crece cada día más. Es un lenguaje que permite crear sistemas multiplataforma.

1.5.6 Servidor FileZilla versión 0.9

El servidor FileZilla es un programa gratuito para dotar al sistema de capacidades para la distribución de archivos por medio de FTP (*File Transfer Protocol*). Es, tanto un programa cliente de FTP, con el que se puede conectar con otros servidores para descargar o subir ficheros, como un servidor de FTP encargado de almacenar ficheros. Es una aplicación de código abierto, posee administración remota. A pesar de estar únicamente disponible para Windows y en el lenguaje inglés, sigue contando con una sencilla interfaz gráfica. (Magazine, 2009)

Justificación del servidor seleccionado

Se utiliza este servidor porque permite la transferencia de todo tipo de archivos vía FTP, es sencillo de utilizar gracias a que posee una interfaz gráfica entendible. Cuenta con un alto nivel de seguridad debido a la restricción que presenta con respecto al acceso a datos por parte de usuarios no autorizados.

1.6 Conclusiones parciales

Concluido este capítulo se determina que el estudio realizado acerca de los principales conceptos asociados al dominio del problema, permitió un mayor entendimiento por parte del equipo de trabajo y sentó las bases para iniciar el proceso de desarrollo del sistema. Se llevó a cabo una investigación acerca de herramientas similares al sistema a desarrollar que permitió tomar como punto de partida para el desarrollo de la solución, la forma en que dichos sistemas lograban la conexión con el servidor, que el usuario fuera capaz de conocer las actualizaciones disponibles y posteriormente poder actualizarlas en la estación de trabajo que desee.

La metodología RUP será la encargada de guiar el ciclo de vida de la propuesta de solución durante su desarrollo y contará para ello con las herramientas y tecnologías seleccionadas: UML en su versión 2.0 como lenguaje de modelado, Visual Paradigm como Herramienta CASE en su versión 8.0, como lenguaje de programación Java, NetBeans en su versión 8.0 como IDE y FileZilla en su versión 0.9 como servidor FTP.

Capítulo 2: Presentación de la solución propuesta.

Introducción

En este capítulo se realiza el análisis y diseño de la solución a desarrollar teniendo como guía la metodología especificada en el capítulo anterior. Se realiza el levantamiento de los requisitos con los que debe cumplir el sistema, además de la descripción de los casos de usos con sus descripciones correspondientes. También se construyen los diagramas de clases del diseño y los diagramas de secuencia, con el propósito de obtener una visión más detallada de las clases que conforman el sistema, además de conocer el flujo de los eventos de casos de uso o de secciones que contenga el *software*.

2.1 Modelo de Dominio

El modelo de dominio no representa el sistema a desarrollar, permite un mayor entendimiento de la situación existente en un entorno determinado. *"La etapa orientada a objetos esencial del análisis o investigación es la descomposición de un dominio de interés en clases conceptuales individuales u objetos. El modelo de dominio describe conceptos del mundo real, no componentes software."* (Larman, 2003)

Se decide utilizar el modelo de dominio debido a que no se encuentran bien definidos los procesos referentes a la forma de actualización de los sistemas desarrollados por los proyectos del centro GEYSED ni quienes son las personas encargas de llevar a cabo este proceso. Por otra parte no se cuenta con un cliente real, lo que imposibilita conocer, con un nivel amplio de detalle, los procedimientos existentes para la actualización de un sistema determinado.

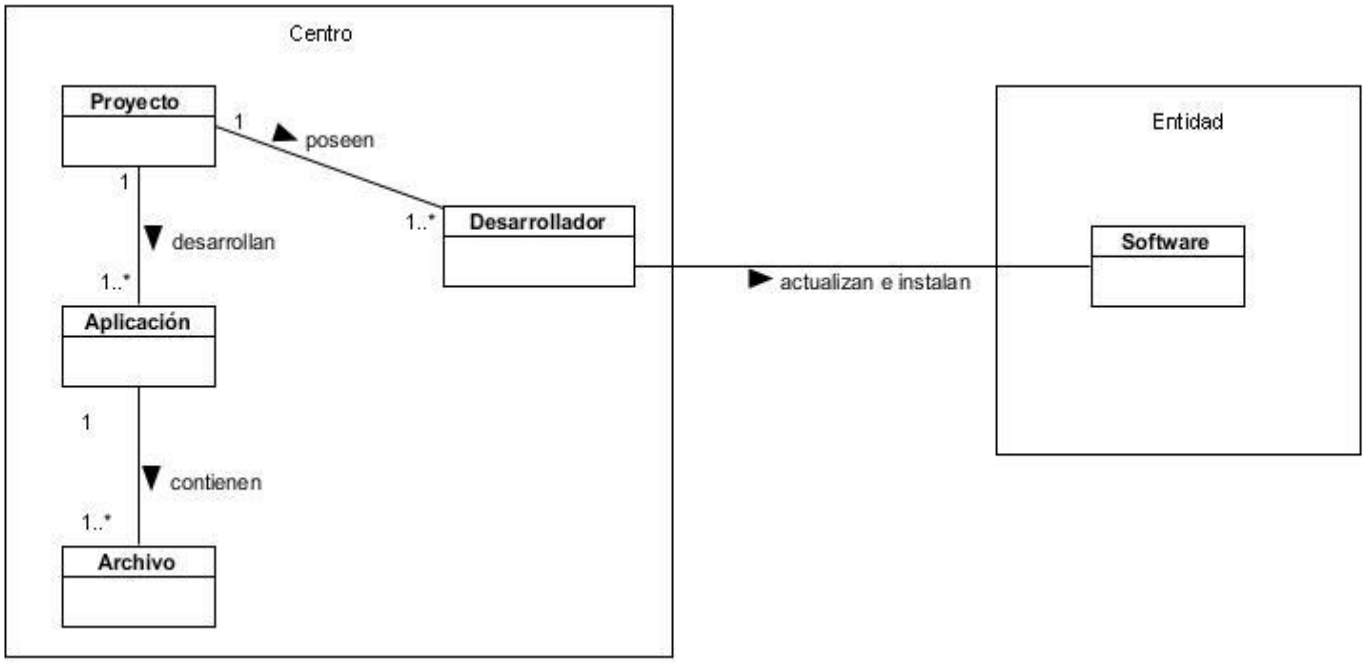

Fig. 1: Modelo de Dominio

2.1.1 Descripción del dominio

Cuando se crean nuevas funcionalidades, existen fallos en el sistema o se desarrollan nuevas versiones de un *software*, los proyectos del centro proceden a desarrollar aplicaciones, las cuales contienen los ficheros necesarios para dar solución a los problemas que se presentan. Es un desarrollador del proyecto el encargado de trasladarse hacia la entidad donde se encuentra desplegado el sistema. Una vez llegado a la entidad el desarrollador comienza el proceso de actualización e instalación del *software*.

2.1.2 Glosario del dominio del problema

A continuación se presentará una descripción de las clases asociadas al dominio del problema representadas en la Fig.2:

- **Centro:** Organización dirigida a desarrollar productos, servicios y soluciones informáticas en el campo del procesamiento del centro GEYSED.
- **Proyecto:** Organización integrada por personas que desempeñan diferentes roles asociados al desarrollo de *software*.

- **Aplicación:** *Software* desarrollado por los proyectos.
- **Archivo:** Archivos que se encuentran dentro de las aplicaciones, ya sean ejecutables, documentos, bibliotecas dinámicas.
- **Entidad:** Organización donde se despliega el sistema desarrollado.
- **Software:** Sistema que permite descargar y actualizar las actualizaciones de las aplicaciones.

2.2 Especificación de requisitos del sistema

2.2.1 Requisitos Funcionales (RF)

Los requisitos funcionales describen las funciones que el *software* a desarrollar debe cumplir. **(Swebok, 2004)**

Tabla. 1: Especificación de Requisitos utilizados en la sistema.

No.	Requisitos	Descripción
RF1	Cargar actualización al servidor.	El sistema debe permitir cargar las actualizaciones necesarias para el cliente en el servidor. Estas actualizaciones pueden ser lo mismo bibliotecas dinámicas, ejecutables, documentos. Esto permite añadir y eliminar de la actualización elementos de los cuales depende para su correcto funcionamiento. Cuando se va a descargar una actualización, se descargan también los elementos de los que esta depende.
RF2	Actualizar fichero.	El sistema debe permitir actualizar los programas que se necesiten para cada uno de los clientes.
RF3	Eliminar actualización del servidor.	El sistema debe permitir eliminar las actualizaciones que ya no son necesarias en el servidor.
RF4	Autenticar usuario.	El sistema debe permitir al usuario configurar las aplicaciones instaladas en su máquina, con el fin de establecer: usuario, contraseña, la dirección y el

		puerto por el cual se va a conectar al servidor.
RF5	Listar dependencias.	El sistema debe mostrar una lista de las dependencias correspondientes a la actualización. Estas dependencias son las que se agregaron anteriormente a la misma.
FR6	Modificar Actualización.	El sistema debe permitir modificar una actualización una vez creada.
RF7	Crear Proyecto.	El sistema debe permitir crear los proyectos que se desean actualizar.
RF8	Modificar Proyecto.	El sistema debe permitir modificar los proyectos que se desean actualizar.
RF9	Eliminar Proyecto.	El sistema debe permitir eliminar los proyectos que se desean actualizar.
RF10	Crear Producto.	El sistema debe permitir crear los productos que se desean actualizar.
RF11	Modificar Producto.	El sistema debe permitir modificar los productos que se desean actualizar.
RF12	Eliminar Producto.	El sistema debe permitir eliminar los productos que se desean actualizar.
RF13	Insertar Usuario.	El administrador inserta los usuarios con sus respectivos permisos en la herramienta que provee el servicio FTP.
RF14	Modificar Usuario.	El administrador modifica los usuarios en la herramienta que provee el servicio FTP.
RF15	Eliminar Usuario	El administrador elimina los usuarios de la herramienta que provee el servicio FTP.
RF16	Configuración subsistema cliente	El usuario configura el sistema cliente para que se descarguen las actualizaciones en una misma

		carpeta.

2.2.2 Requisitos No Funcionales (RNF)

Los requisitos no funcionales se conocen a veces como requisitos de calidad. Son los requisitos que no describen ninguna función a realizar en el sistema. Son las cualidades que el producto debe poseer. **(Swebok, 2004)**

RNF1: Software

El sistema deberá ejecutarse en los sistemas operativos Ubuntu v. 11.4 y Windows v.7 o superior.

RNF2: Hardware

Servidor:

- Procesador Core 2 Duo o Dual Core de 1.5 GHz o superior
- 1 GB de RAM o superior.
- Disco Duro de 80 GB o superior

Sistema desarrollado:

- Procesador Core 2 Duo o Dual Core de 1.0 GHz o superior
- 1 GB de RAM o superior.
- Disco Duro 40 GB o superior.

RNF3: Seguridad

Los usuarios, tanto clientes como administrador, solo tendrán acceso a las carpetas correspondientes al proyecto al que pertenezcan.

2.3 Definición de casos de uso en el sistema

2.3.1 Actores del sistema

Tabla. 2: Descripción de actores del sistema.

Actores	Descripción
Administrador	Representa a la persona que interactúa directamente con el subsistema administrador, con el fin de mantener en el servidor las últimas actualizaciones desarrolladas por los proyectos.
Usuario	Representa a la persona que interactúa con subsistema cliente, o sea es el que elegirá del servidor cuáles son las actualizaciones que necesita.

2.3.2 Diagramas de casos de uso del sistema

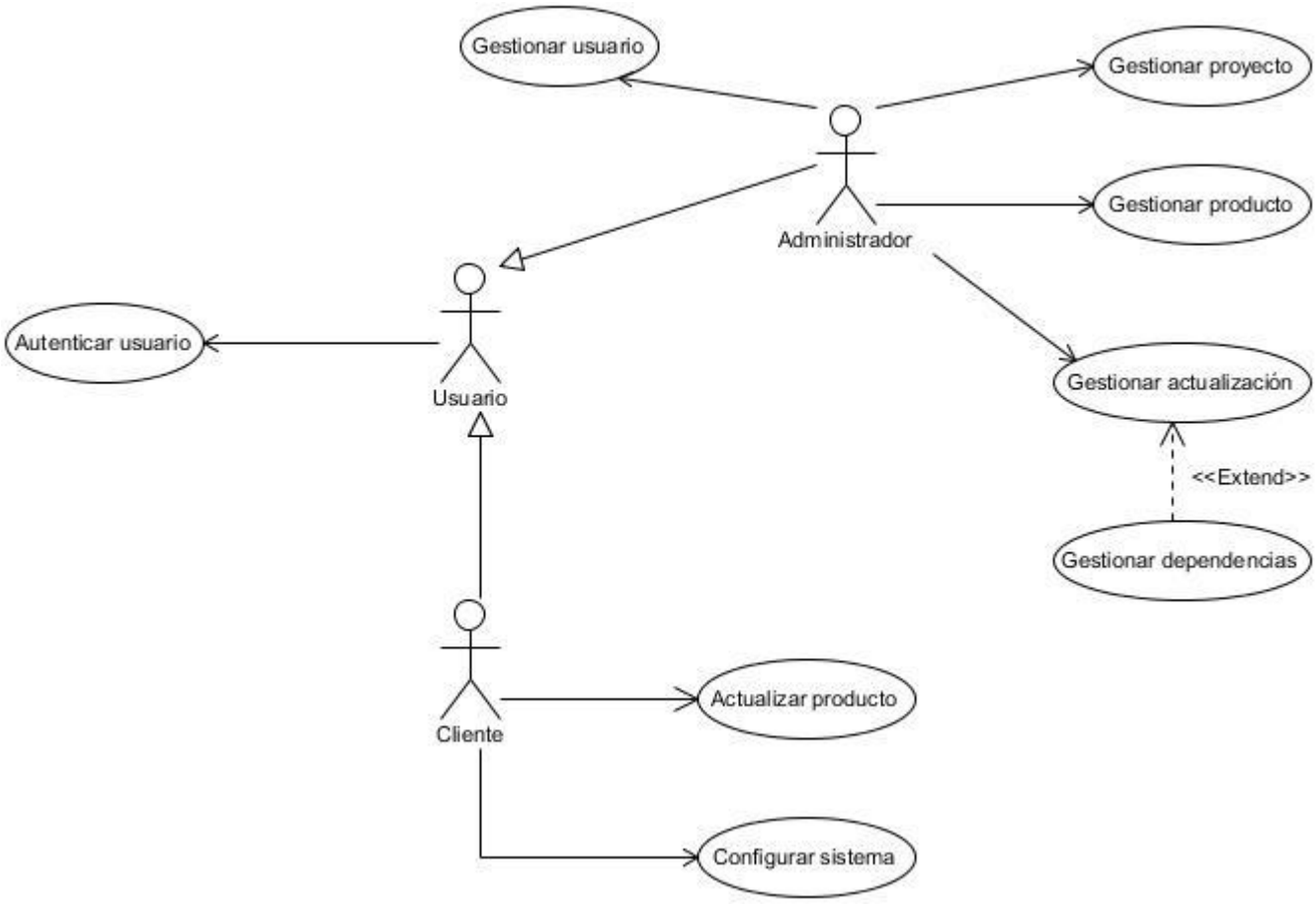

Fig. 2: Representación de Casos de Usos del sistema

2.3.3 Descripción de casos de usos del sistema

Tabla. 3: Descripción del Caso de Uso Actualizar producto

Objetivo:	Mantener los programas actualizados correctamente.	
Actor:	Usuario	
Resumen:	El caso de uso inicia cuando el usuario selecciona la actualización que desea descargar. Finaliza cuando se actualiza el fichero en el ordenador.	
Complejidad:	Alta	
Prioridad:	Crítico	
Referencia:	RF2	
Precondiciones:	- La aplicación debe estar conectada al servidor. - Deben existir actualizaciones nuevas en el servidor. - Debe estar configurada la conexión con el servidor.	
Postcondiciones:	Quedan actualizados los ficheros.	
Flujo de eventos		
Flujo básico: "Actualizar Ficheros"		
	Actor	**Sistema**
1.	Selecciona la actualización que desea descargar y presiona el botón "Descargar".	
2.		Muestra una ventana para buscar la dirección donde se va a descargar la actualización.
3.	Busca donde se va a descargar la actualización y presiona el botón "Aceptar" para descargar la actualización.	
4.		Descarga la actualización con sus dependencias y muestra una ventana con un mensaje: "Se ha

		descargado la actualización correctamente.”
5.		Termina el caso de uso.

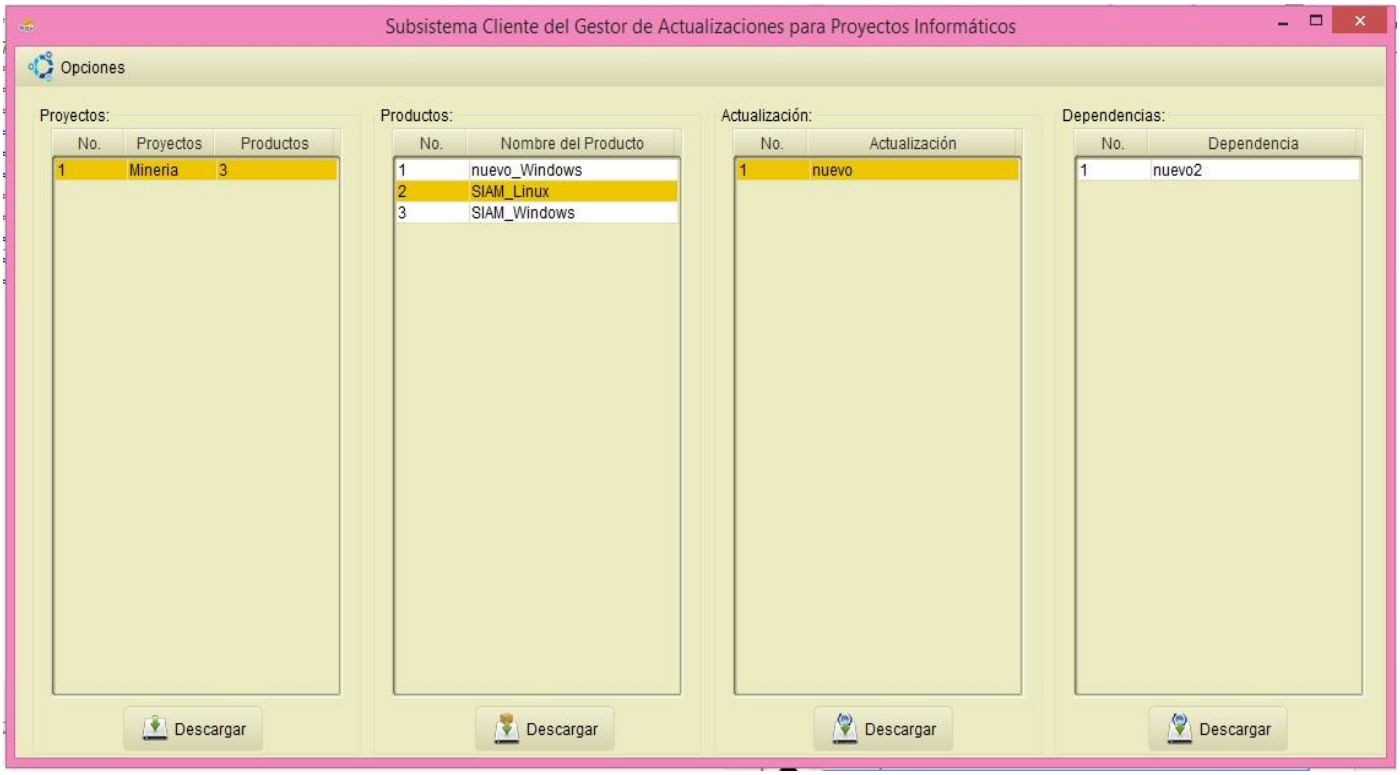

Fig. 3: Prototipo de Interfaz para el caso de uso Actualizar Ficheros

Relaciones:	**CU Incluido**	No existe
	CU Extendido	No existe

Tabla. 4: Caso de uso Gestionar proyecto.

Objetivo:	Crear un proyecto para llevar a cabo el proceso de actualizaciones.
Actor:	Administrador (Inicia)
Resumen:	El caso de uso inicia cuando el administrador crea los proyectos a los cuales se les va a realizar la actualización. Finaliza cuando se crea la actualización en el

	servidor.	
Complejidad:	Alta	
Prioridad:	Crítico	
Referencia:	RF7, RF8, RF9.	
Precondiciones:	Iniciar la aplicación.	
Postcondiciones:	Quedan creados los proyectos.	
Flujo de eventos		
Flujo básico: "Gestión de Proyectos"		
	Actor	**Sistema**
1.	Selecciona del formulario las opciones "Registrar Proyecto (A)", "Modificar Proyecto (B)", "Eliminar Proyecto C", "Gestionar producto (Ver caso de uso)".	
3.		Gestiona los proyectos según la opción seleccionada por el usuario.
4.		Termina el caso de uso.
Sección 1: Registrar Proyecto		
	Actor	**Sistema**
1.	Llena el campo "Nombre del Proyecto" y presiona el Botón A.	
2.		Agrega el nuevo proyecto a la lista existente en el servidor.
Sección 2: Modificar Proyecto		
	Actor	**Sistema**
1.	Selecciona el proyecto a modificar.	
2.	Escribe el nuevo nombre del proyecto.	

3.	Selecciona el campo B.	
4.		Modifica el nombre del proyecto en el servidor.
Sección 3: Eliminar Proyecto		
	Actor	**Sistema**
1.	Selecciona el proyecto que desea eliminar.	
2.	Presiona el botón C	
		Elimina el proyecto del servidor

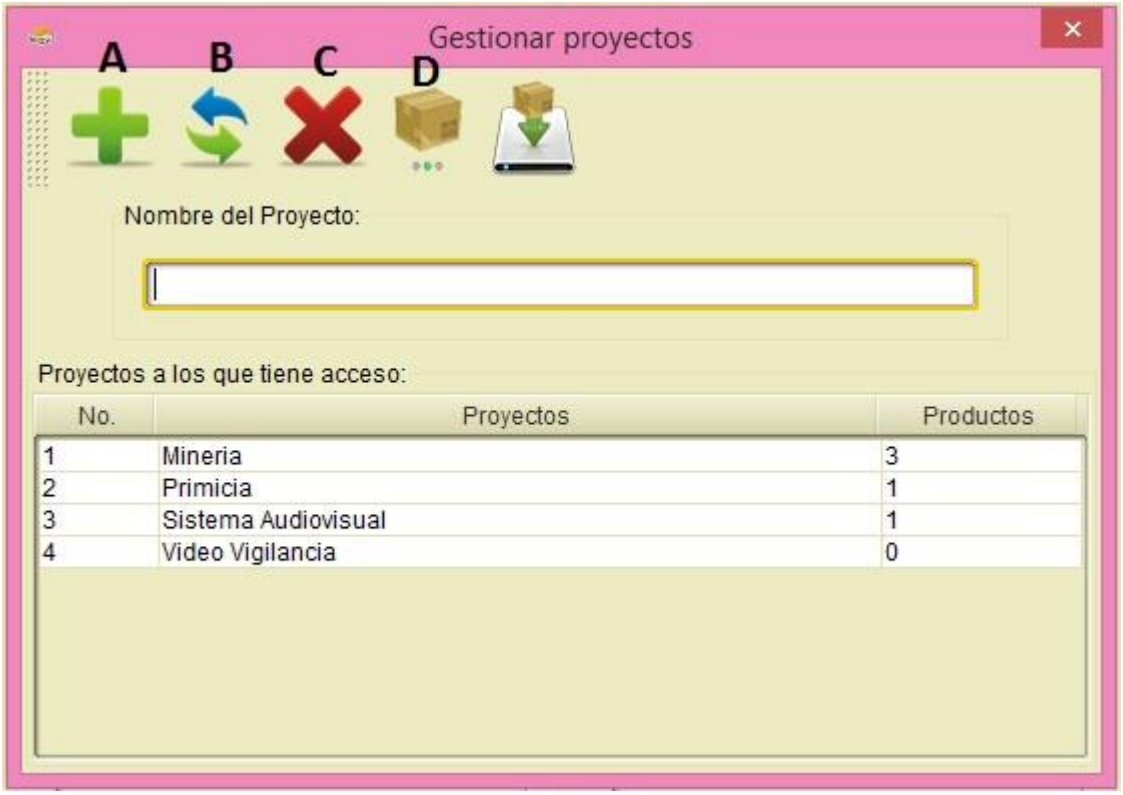

Fig. 4: Prototipo de interfaz del caso de uso Gestionar proyectos

Relaciones:	**CU Incluido**	No existe
	CU Extendido	No existe

Tabla. 5: Caso de uso Gestionar productos

Objetivo:	Crear los productos asociados a un proyecto para llevar a cabo el proceso de actualizaciones.	
Actor:	Administrador (Inicia)	
Resumen:	El caso de uso inicia cuando el administrador crea los productos a los cuales se les va a realizar la actualización. Finaliza cuando se crea la actualización en el servidor.	
Complejidad:	Alta	
Prioridad:	Crítico	
Referencia:	RF10, RF11, RF12.	
Precondiciones:	Existan los proyectos a los cuales se le van a agregar los productos.	
Postcondiciones:	Quedan creados los productos.	
Flujo de eventos		
Flujo básico: "Gestión de Productos"		
	Actor	**Sistema**
1.	Selecciona del formulario las opciones "Registrar Producto (A)", "Modificar Producto (B)", "Eliminar Producto (C)" "Gestionar proyecto (D) (Ver caso de uso)", "Gestionar actualización (E) (Ver caso de uso)".	
2.		Gestiona el producto según la opción seleccionada por el usuario.
3.		Termina el caso de uso.
Sección 1: Registrar Producto		
	Actor	**Sistema**

1.	Selecciona el proyecto al que pertenece.	
	Llena el campo “Nombre del Producto”	
1.	Selecciona la plataforma en la que va a correr.	
	Selecciona el botón A	
2.		Agrega el nuevo producto a la lista existente en el servidor.
Sección 2: Modificar Producto		
	Actor	**Sistema**
1.	Selecciona el producto a modificar.	
2.	Escribe el nuevo nombre del producto.	
3.	Selecciona el campo B.	
4.		Modifica el nombre del producto en el servidor.
Sección 3: Eliminar Producto		
	Actor	**Sistema**
1.	Selecciona el producto que desea eliminar.	
2.	Presiona el botón C	
		Elimina el producto del servidor

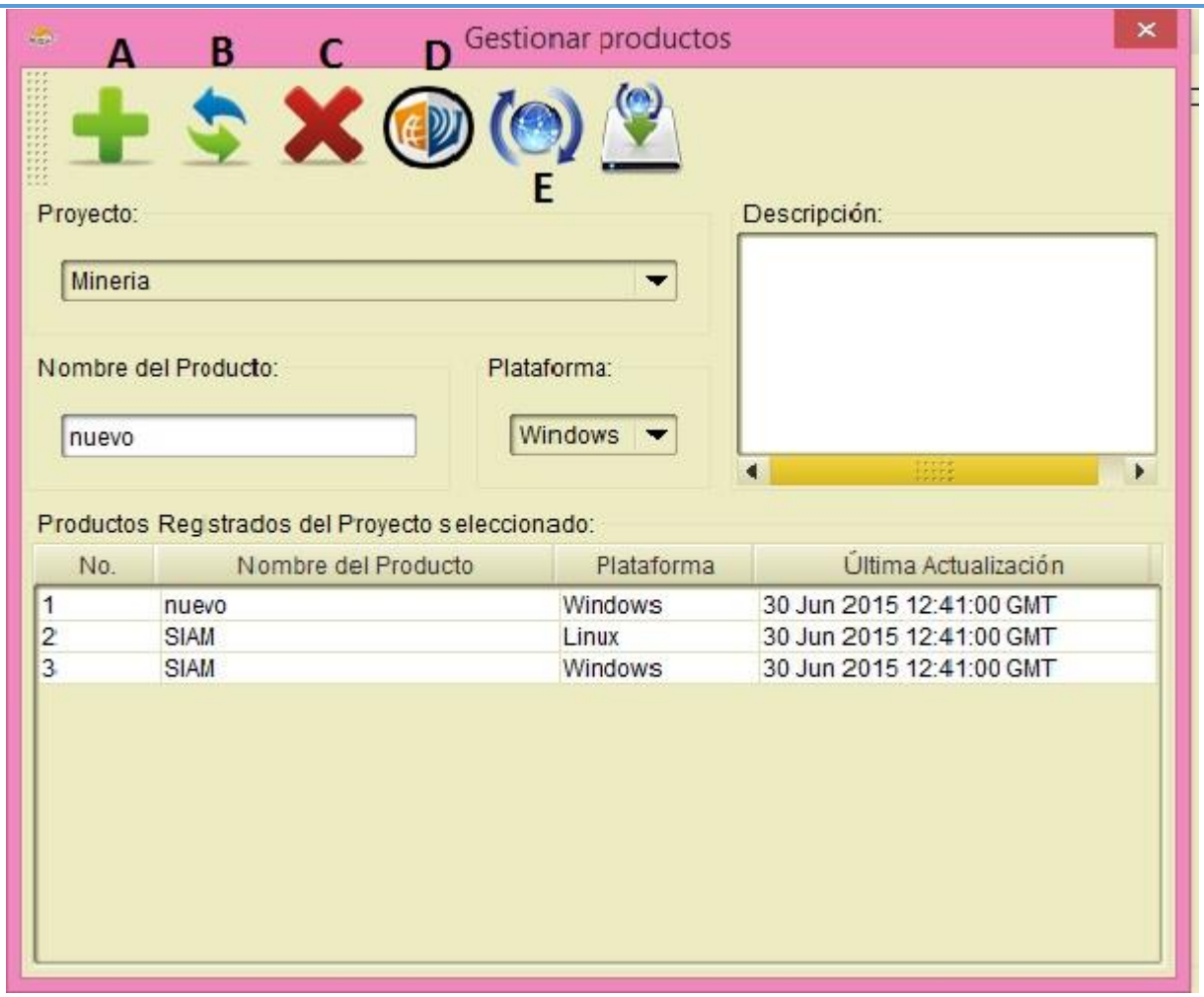

Fig. 5: Prototipo de interfaz del caso de uso Gestionar productos

Relaciones:	**CU Incluido**	No existe
	CU Extendido	No existe

Tabla. 6: Caso de Uso Gestionar actualizaciones

Objetivo:	Posibilitar a los usuarios la adquisición de las actualizaciones. Permitir agregar o eliminar dependencias a una actualización.
Actor:	Administrador (Inicia)

Resumen:	El caso de uso inicia cuando el administrador selecciona la actualización que desea subir al servidor. Finaliza cuando la actualización queda almacenada en el servidor.
Complejidad:	Alta
Prioridad:	Crítico
Referencia:	RF1, RF3, RF6.
Precondiciones:	Debe existir el proyecto y el producto a la cual corresponde la actualización.
Postcondiciones:	Quedan almacenadas o eliminadas las actualizaciones.

Flujo de eventos

Flujo básico: "Gestión de Actualizaciones"

	Actor	**Sistema**
1.	Selecciona del formulario las opciones "Registrar Actualización (A)", "Modificar Actualización (B)", "Eliminar Actualización (C)".	
2.		Gestiona el producto según la opción seleccionada por el usuario.
3.		Termina el caso de uso.

Sección 1: Registrar Actualización

	Actor	**Sistema**
1.	Selecciona el proyecto al que pertenece.	
2.	Selecciona el producto al que pertenece.	
3.	Llena el campo "Nombre de la Actualización"	
4.	Selecciona el "Tipo de Actualización" deseada: - Paquete de instalación. - Fichero (script, rar, zip) - Directorio	

5.	Selecciona la “Recomendación” deseada: - Descargar. - Instalar. - Descargar e instalar.	
6.	Selecciona el botón “Cargar Actualización (D)”	
7.		Muestra una ventana de búsqueda en la raíz principal de la máquina.
8.	Escoge la actualización que desea cargar.	
9.	Selecciona las dependencias de la actualización si las necesita (Ver caso de uso Gestionar dependencias).	
10.	Busca la dirección donde se va a descargar	
11.	Selecciona la fecha perteneciente a la actualización, ya sea de forma manual o automática.	
12.		Agrega la nueva actualización a la lista existente en el servidor.
Sección 2: Modificar Actualización		
	Actor	**Sistema**
1.	Selecciona la actualización a modificar.	
2.	Escribe el nuevo nombre de la actualización.	
3.	Selecciona la “Recomendación” que desea modificar: - Descargar. - Instalar. - Descargar e instalar.	
4.	Selecciona las dependencias de la actualización que desee agregar. (Ver caso de uso Gestionar dependencias).	
5.	Selecciona el campo B.	

6.		Queda modificada la actualización en el servidor.
Sección 3: Eliminar Actualización		
	Actor	**Sistema**
1.	Selecciona la actualización que desea eliminar.	
2.	Presiona el botón C	
		Elimina la actualización del servidor

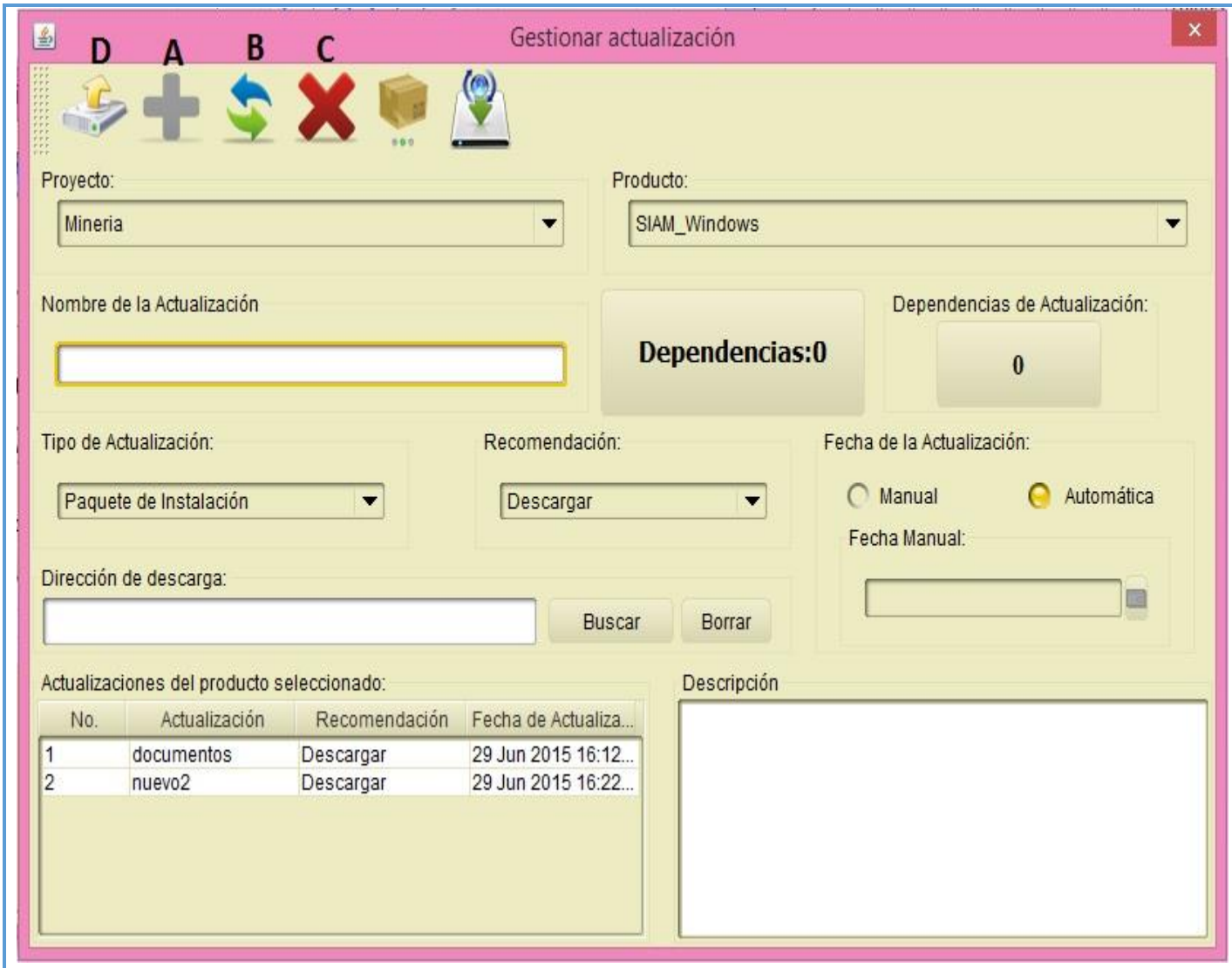

Fig. 6: Prototipo de interfaz del caso de uso Gestionar actualización.

Relaciones:	**CU Incluido**	No existe
	CU Extendido	No existe

Tabla. 7: Descripción de Caso de Uso Autenticar usuario

Objetivo:	Proveer a la aplicación los datos necesarios para establecer la conexión al

	servidor.
Actor:	Usuario (Inicia)
Resumen:	El caso de uso inicia cuando el usuario configura el sistema para establecer la conexión al servidor. Finaliza cuando se desconecta del servidor.
Complejidad:	Alta
Prioridad:	Crítico
Referencia:	RF4
Precondiciones:	Iniciar la aplicación.
Postcondiciones:	La aplicación se conectó al servidor.

Flujo de eventos

Flujo básico: "Autenticar usuario"

	Actor	**Sistema**
1.		Muestra un formulario con los campos: - Administración (A) - Cliente (B)
2.	Selecciona la opción A o la B.	
3.		Muestra un formulario con los campos: - Dirección servidor - Puerto - Usuario - Contraseña
4.	Llena los campos establecidos en la ventana principal. Selecciona el botón "Aceptar"	

5.		Establece la conexión con el servidor e inicializa la aplicación.
6.		Termina el caso de uso.

Fig. 7: Prototipo Interfaz del caso de uso Autenticar usuario.

Relaciones:	**CU Incluido**	No existe
	CU Extendido	No existe

Tabla. 8: Descripción de Caso de Uso Gestionar dependencia.

Objetivo:	Asignar dependencias por cada una de las actualizaciones asociadas a un proyecto.
Actor:	Administrador (Inicia)
Resumen:	El caso de uso inicia cuando el administrador asigna las dependencias a la actualización. Finaliza cuando son agregadas las dependencias.
Complejidad:	Alta
Prioridad:	Crítico

Referencia:	RF5	
Precondiciones:	Debe existir la actualización a la que se le va a crear la dependencia.	
Postcondiciones:	Quedan agregadas las dependencias.	
Flujo de eventos		
Flujo básico: "Gestionar dependencias"		
	Actor	**Sistema**
1.	Selecciona del formulario las opciones "Cargar dependencia (A)", "Modificar dependencia (B)", "Eliminar dependencia (C)".	
3.		Gestiona las dependencias según la opción seleccionada por el usuario.
4.		Termina el caso de uso.
Sección 1: Cargar dependencia		
	Actor	**Sistema**
1.	Llena el campo "Nombre de la Dependencia".	
2.	Selecciona el "Tipo de Dependencia": - Paquete de instalación. - Fichero (script, rar, zip) - Directorio	
3.	Selecciona la "Recomendación": - Descargar. - Instalar. - Descargar e instalar.	
4.	Busca la dirección donde se va a descargar.	
5.	Selecciona la fecha perteneciente a la actualización, ya sea de forma manual o automática.	
6.	Selecciona el botón "Cargar dependencia (A)".	

		Muestra una ventana de búsqueda en la raíz principal de la máquina.
	Escoge la dependencia que desea cargar.	
7.		Agrega la nueva dependencia a la actualización correspondiente.
Sección 2: Modificar dependencia		
	Actor	**Sistema**
1.	Selecciona la dependencia a modificar.	
2.	Escribe el nuevo nombre de la dependencia.	
3.	Selecciona la “Recomendación” que desea modificar: - Descargar. - Instalar. - Descargar e instalar.	
4.	Selecciona el campo B.	
5.		Queda modificada la dependencia en el servidor.
Sección 3: Eliminar dependencia		
	Actor	**Sistema**
1.	Selecciona la dependencia que desea eliminar.	
2.	Presiona el botón C	
		Elimina la dependencia del servidor

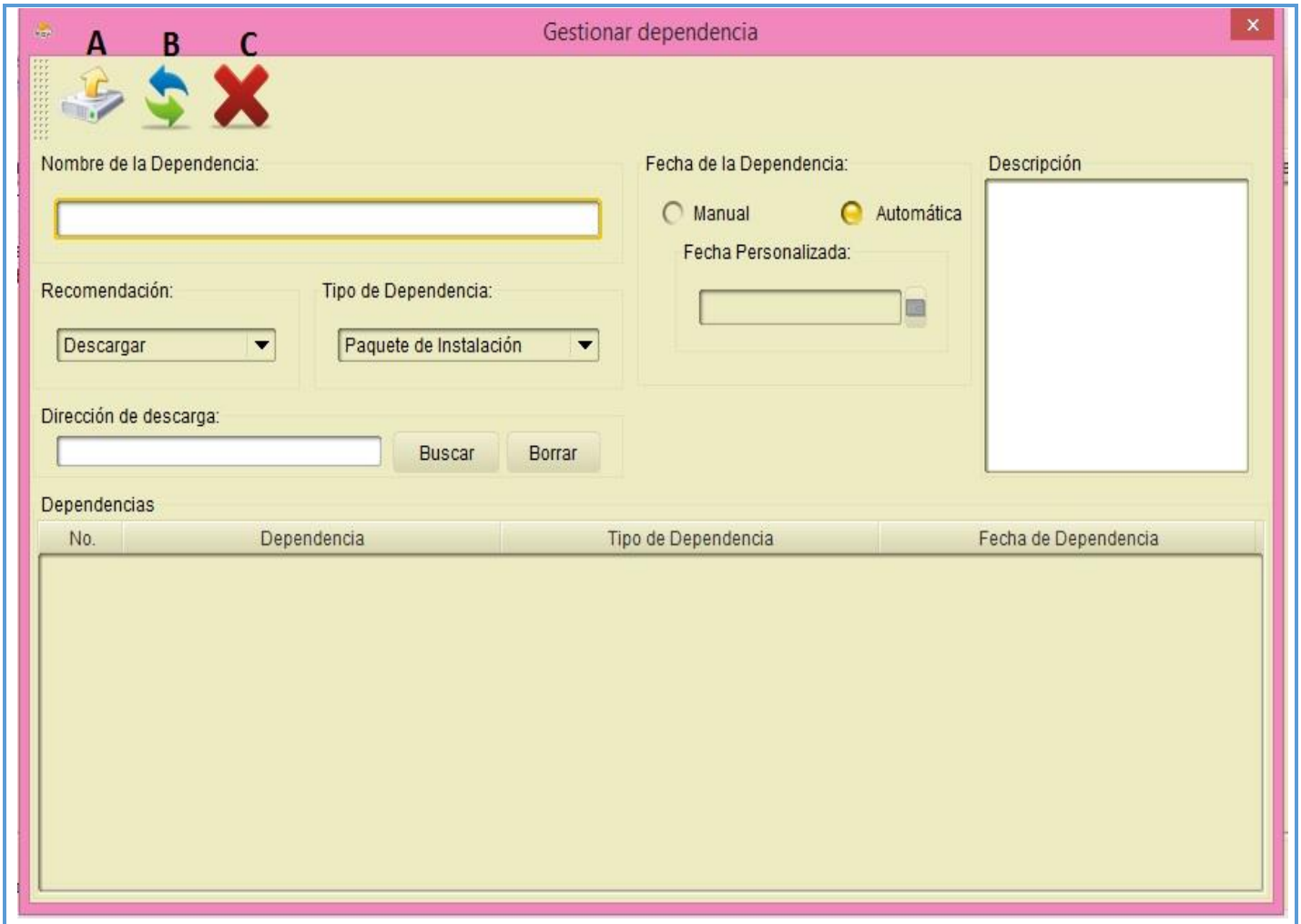

Fig. 8:Prototipo de interfaz del caso de uso Gestionar dependencia

Relaciones:	**CU Incluido**	No existe
	CU Extendido	No existe

2.4 Patrón Arquitectónico

El patrón arquitectónico en capas organiza la estructura lógica de gran escala de un sistema en capas separadas de responsabilidades distintas y relacionadas. Mantienen una separación clara y cohesiva de intereses como que las capas más bajas son servicios generales de bajo nivel, y las capas más altas son

más específicas de la aplicación. La colaboración y el acoplamiento son desde las capas más altas hacia las más bajas; se evita el acoplamiento de las capas más bajas a las más altas. En general, existe una separación de intereses, una separación entre los servicios de alto y bajo nivel, y de servicios específicos y generales de la aplicación. Esto reduce el acoplamiento y las dependencias, mejora la cohesión, incrementa la claridad. (Larman, 2003)

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que ocurra algún cambio, solo es necesario cambiar el código del nivel requerido, sin que se afecten los demás.

Se utiliza la arquitectura de N capas. Se compone por la Presentación y Lógica de Negocio. Según Pressman: La capa de presentación es responsable de las interfaces con las que se visualizará la aplicación, la capa de lógica de negocio es la responsable del procesamiento que tiene lugar en la aplicación y la capa de la base de datos contiene los datos de la aplicación. (Pressman, 2011)

Como base para el desarrollo del sistema se escoge la arquitectura en dos capas ya que simplifica su comprensión y organización. Además, con su utilización se reduciría la dependencia ya que las capas bajas no son conscientes de los detalles que se encuentran en las capas superiores.

2.5 Patrones de Diseño

Los patrones de diseño son los que permiten describir fragmentos y reutilizar ideas de diseño, ayudando a beneficiarse de la experiencia de otros. Comunican los estilos y soluciones consideradas como buenas prácticas, que los expertos en el diseño orientado a objetos utilizan para la creación de sistemas. (Larman, 2003)

GRASP[11]

Los patrones GRASP, más que patrones propiamente dichos, son una serie de “buenas prácticas” de aplicación recomendable en el diseño de software. En el desarrollo del sistema se hizo uso de los patrones Experto, Creador, Controlador, Alta cohesión y Bajo acoplamiento porque guardan directa relación con la creación y asignación de responsabilidades a los objetos. Unos de sus principales objetivos

[11] Patrones de Software para la asignación de responsabilidades

es permitirle al diseñador conseguir un correcto diseño, para de esta manera mejorar la documentación y mantenimiento de un sistema existente así como hacer de este reutilizable. (Larman, 2003)

A continuación se presentan los cinco patrones GRASP:

- **Experto:**

Es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele ser útil en el diseño orientado a objetos. Este soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. Con la utilización de este patrón cada clase contiene la información necesaria para llevar a cabo una función específica dentro del sistema.

Ejemplo donde se evidencia:

En la sistema se hace uso de este patrón con la utilización de la librería FTP que contiene una serie de clases (*FileTransferClient* experta en la conexión con el servidor, *FTPClient* experta en eliminar y crear directorios *FTPTransferType* encargada del tipo de trasferencia que se hace) que fueron de utilidad para el manejo de la conexión con el servidor, la creación, eliminación y trasferencia de directorios y ficheros hacia el mismo.

- **Creador:**

Guía la asignación de responsabilidades relacionadas con la creación de objetos. La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. La utilización del mismo favorece el bajo acoplamiento del sistema.

Ejemplo donde se evidencia:

En el sistema el uso de este patrón se evidencia entre las clases de la capa Lógica de Negocio, Actualización y Dependencia. Una actualización puede o no estar compuesta por un conjunto de dependencias por lo que se crea una lista de dependencia (List<Dependencia>) en la clase Actualización.

- **Bajo Acoplamiento y Alta Cohesión:**

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras. Un acoplamiento bajo significa que una clase no depende de muchas otras. El patrón Alta cohesión determina cuan relacionadas y adecuadas están las responsabilidades de una clase, de manera que no realice un trabajo excesivo; Las clases solo contiene los parámetros y funcionalidades que están estrechamente relacionados con ella. Se hizo necesario la utilización de este patrón en el sistema con el fin de controlar la complejidad de cada clase utilizada para mantener un buen comportamiento de las mismas. El nivel de cohesión no se puede ver de manera aislada a otras responsabilidades y otros principios como el patrón

experto y bajo acoplamiento. Una mala cohesión causa normalmente un mal acoplamiento y viceversa. Se evidencian en todas las clases interfaces y entidades repartiéndose las tareas por las fueron creadas y siendo lo más autónomas posible ejemplo (la clase de interfaz GestionarActualizacion tiene como objetivo operar los eventos donde se obtienen los datos únicos de una actualización que posteriormente son transferidos a la clase Controladora).

- **Controlador:**

Está diseñada para ser capaz de controlar el flujo de los eventos del sistema para clases determinadas. Se evidencia en la clase Controladora, encargada de controlar el acceso a la información establecida en las capas de lógica de negocio.

Patrón GoF*[12]*

Los patrones de diseño, conocidos como GoF se clasifican en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento. A continuación son detallados los que han sido utilizados en el diseño de la solución.

- **Fachada (*Facade*):**

Patrón estructural que trata de simplificar la interfaz entre dos sistemas o componentes de software. Oculta un sistema complejo detrás de una clase que funciona como pantalla o fachada. La idea principal es la de ocultar todo lo posible la complejidad de un sistema, el conjunto de clases o componentes que lo definen, de forma que solo se ofrezca un punto de entrada al sistema cubierto por la fachada. Su uso aísla los posibles cambios que se puedan producir en alguna de las partes. (Mühlrad, 2008)

En la solución del sistema se evidencia en la clase interfaz Principal donde se crea una interfaz compuesta por dos subsistemas Subsistema administración que a su vez tiene contenido varias interfaces y Subsistema cliente.

2.6 Diseño del software

En el diseño según Jacobson: "*se modela el sistema y se encuentran sus formas para que soporte todos los requisitos (funcionales y no funcionales).*" (Jacobson, y otros, 2010)

[12] Pandilla de cuatro por su siglas en inglés *Gang of Four*

En el diseño, según Larman: *"se pone énfasis en una solución conceptual que satisface los requisitos, en vez de ponerlo en la implementación."* (Larman, 2003)

2.6.1 Diagrama de clases del diseño

Los diagramas de clases del diseño se utilizan para visualizar las relaciones entre las clases existentes en el sistema.

A continuación se muestra el diagrama de clases que conforman el sistema implementado.

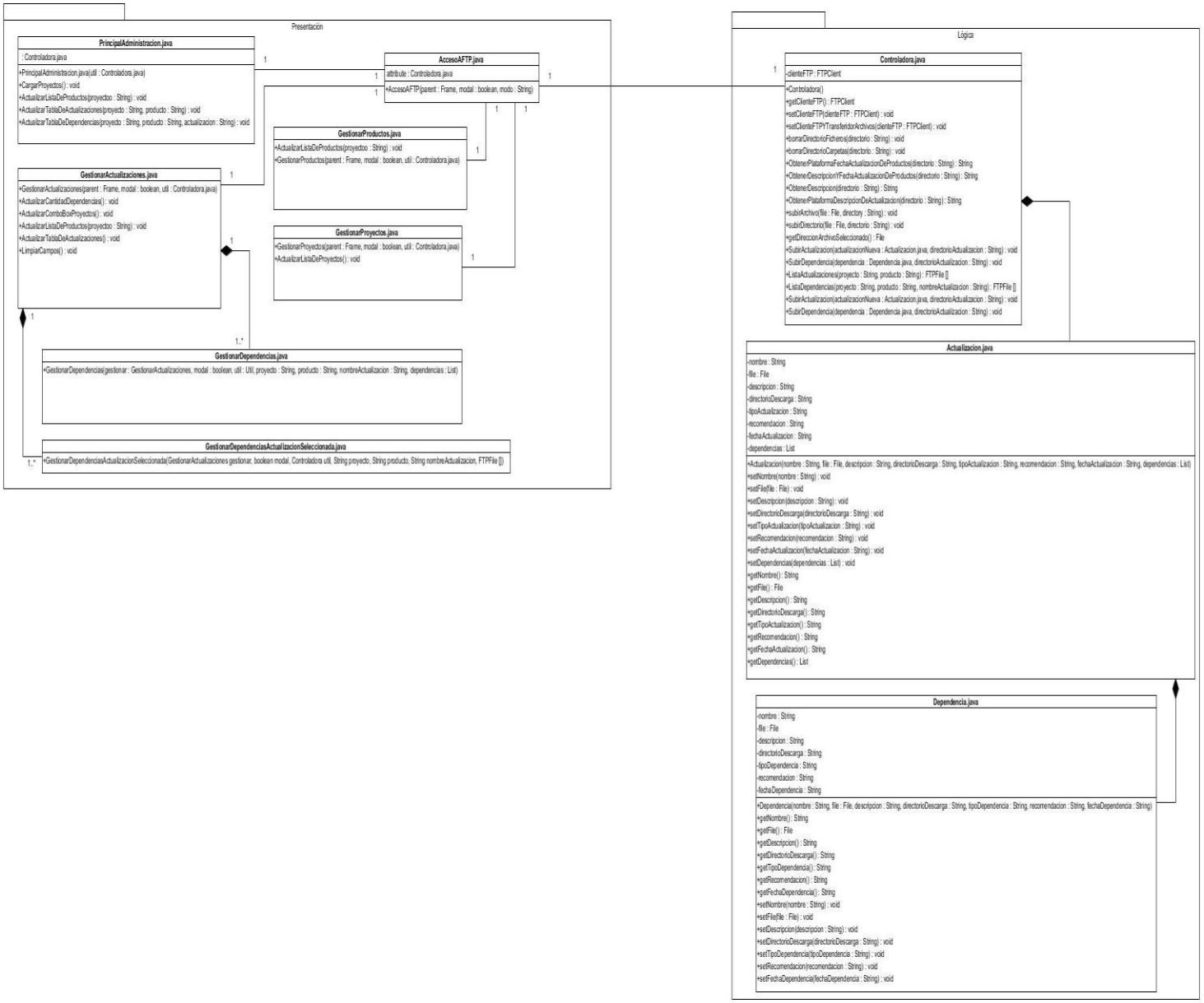

Fig. 9: Diagrama de clases del diseño de la solución subsistema Administrador.

2.7 Diagrama de Secuencia

Un diagrama de secuencia del sistema (DSS) es un dibujo que muestra, para un escenario específico de un caso de uso, los eventos que generan los actores externos, el orden y los eventos entre los sistemas.

(Larman, 2003). A continuación se muestra uno de los diagramas de secuencia de uno de los casos de uso anteriormente descritos.

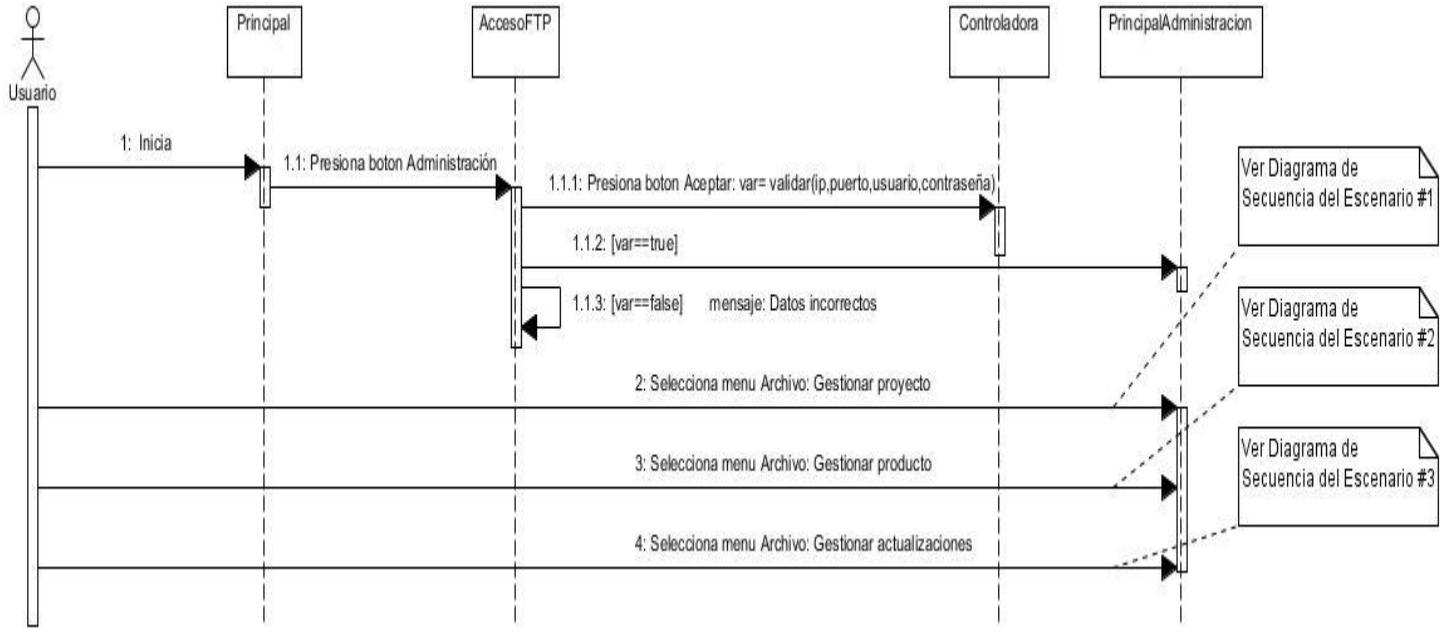

Fig. 10: Diagrama de Secuencia del sistema.

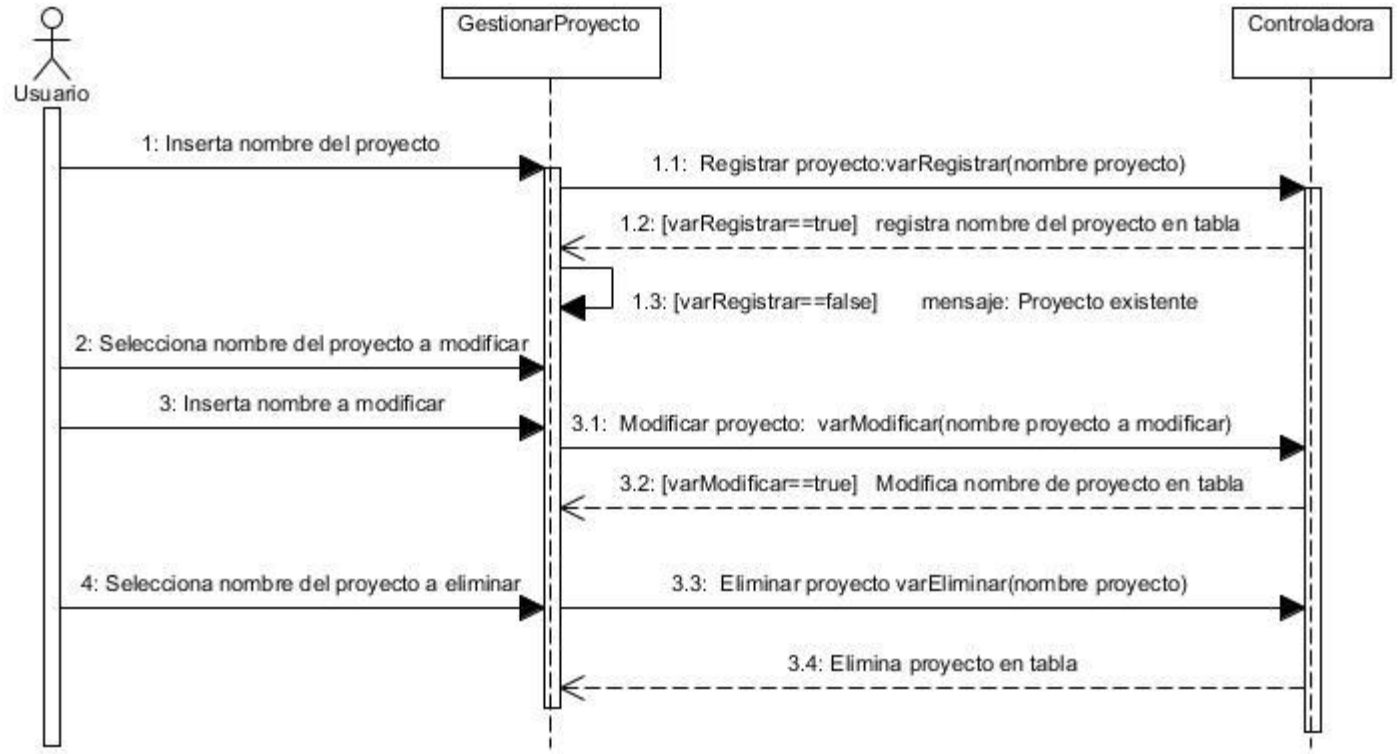

Fig. 11: Diagrama de secuencia del caso de uso Gestionar proyecto.

2.8 Diagrama de Despliegue

El modelo de despliegue es uno de los modelos de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. El modelo de despliegue se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño. (Jacobson, y otros, 2010)

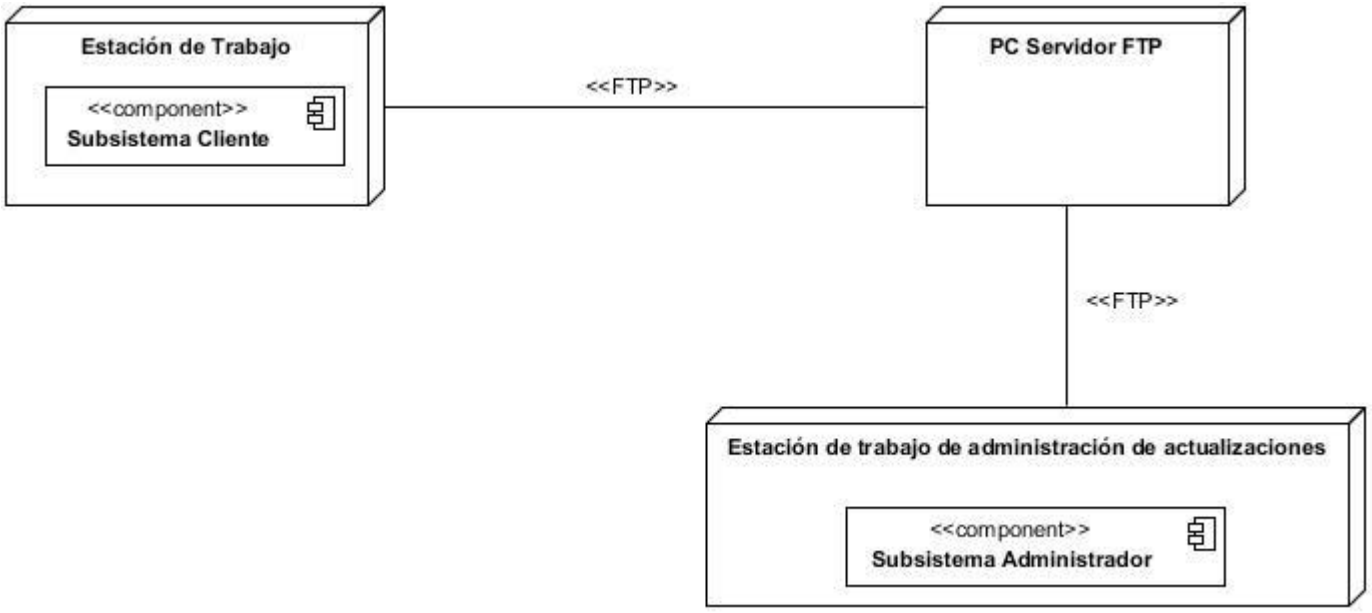

Fig. 12: Diagrama de Despliegue

Descripción de los componentes:

- **Estación de trabajo:** es la computadora local donde se va a encontrar la aplicación cliente que actualizará e instalará todas las actualizaciones.
- **Servidor FTP:** computadora donde se encuentran almacenadas todas las actualizaciones.
- **Estación de trabajo de administración de actualizaciones:** es la computadora donde se encuentra la aplicación administradora que sube todas las actualizaciones al servidor.

2.9 Descripción del sistema propuesto

El sistema propuesto está integrado por dos subsistemas y tres módulos. Los subsistemas son: Subsistema Administrador y Subsistema Cliente. Los módulos son: módulo de seguridad, módulo de administración y módulo de descarga.

Subsistema Administrador

Este subsistema debe ser instalado en el ordenador del futuro administrador del sistema que será el encargado de cargar las actualizaciones nuevas para el servidor; al mismo tiempo que se crea un archivo de texto, que contendrá la dirección donde se guardará de la actualización, sus dependencias y el

nombre. Con el objetivo de evitar que existan demasiadas actualizaciones o que se llene el espacio en disco del servidor, el administrador será capaz de eliminar del servidor las actualizaciones obsoletas.

En este subsistema es donde se crean los proyectos que pertenecen al centro y trabajan con aplicaciones de escritorio del centro GEYSED. Dentro de cada proyecto se contendrán los productos y según el sistema operativo para el que estén hechos, se va a almacenar las diferentes actualizaciones y sus correspondientes dependencias en el servidor. Dichas actualizaciones pueden ser descargadas por los usuarios pertenecientes a sus respectivos proyectos o entidades.

Subsistema Cliente

Este subsistema debe de ser instalado en la entidad cliente donde se despliegue una aplicación de escritorio desarrollado por el centro GEYSED, también puede ser utilizado por los propios desarrolladores dentro de cada proyecto. Una vez instalado, el usuario puede descargar para su ordenador las diferentes actualizaciones que se encuentren almacenadas en el servidor. Si el usuario está interesado en descargar todas las actualizaciones o las dependencias de una actualización dada que pertenezcan a su entidad o proyecto, podrá hacerlo.

Módulo de seguridad

Es donde se crean los usuarios, se valida su autenticación y se gestionan los permisos que tendrán. En dependencia del proyecto o entidad a la que pertenezcan, tendrán acceso a las actualizaciones que necesiten.

Módulo de administración

Es donde se gestionan los proyectos, los productos y las actualizaciones que tributan a los diferentes proyectos que trabajan con aplicaciones de escritorio desarrolladas del centro GEYSED.

Módulo de descarga

Es donde se descargan a partir de las actualizaciones de los diferentes productos según el proyecto, que se tengan almacenadas en el servidor.

Con la utilización del sistema de actualizaciones propuesto, se evita que en los proyectos o en las entidades donde se desplieguen las aplicaciones de escritorio desarrolladas por el centro GEYSED, puedan existir pérdidas de configuraciones almacenadas, que surjan errores de dependencia entre bibliotecas, e incluso, que el equipo de desarrollo tenga que transportarse hasta cada una de los

ordenadores donde se deseen actualizar los archivos de los diferentes productos de un proyecto o entidad.

2.10 Conclusiones parciales

La modelación de las clases del dominio proporcionó el punto de partida para la correcta elaboración del sistema. El levantamiento de los requisitos tanto funcionales como no funcionales proporciona las medidas necesarias con las que debe cumplir el sistema para su correcta implementación. A partir de los artefactos generados, acorde con la metodología de desarrollo RUP, y tomando en cuenta sus especificaciones, es posible realizar la implementación de la propuesta solución que cumpla con las necesidades del centro GEYSED.

CAPÍTULO 3: Implementación y pruebas.

En este capítulo se abarcará todo lo referente al proceso de implementación y prueba del sistema. Se presentará el diagrama de componentes que muestra el ordenamiento y las dependencias entre los componentes de la solución a desarrollar. Además se realizarán las pruebas de *software* las que tienen como objetivo garantizar la calidad del *software* y encontrar errores que no fueron descubiertos con anterioridad.

3.1 Diagrama de Componentes

Un diagrama de componentes se utiliza para representar como se organizan los componentes del *software*. Su objetivo más importante es representar la estructura de alto nivel de la solución a implementar.

El diagrama de componentes que se muestra a continuación está basado en el patrón arquitectónico por capas, donde se representan las dos capas utilizadas en la realización del sistema y los componentes que pertenecen a cada una de las mismas.

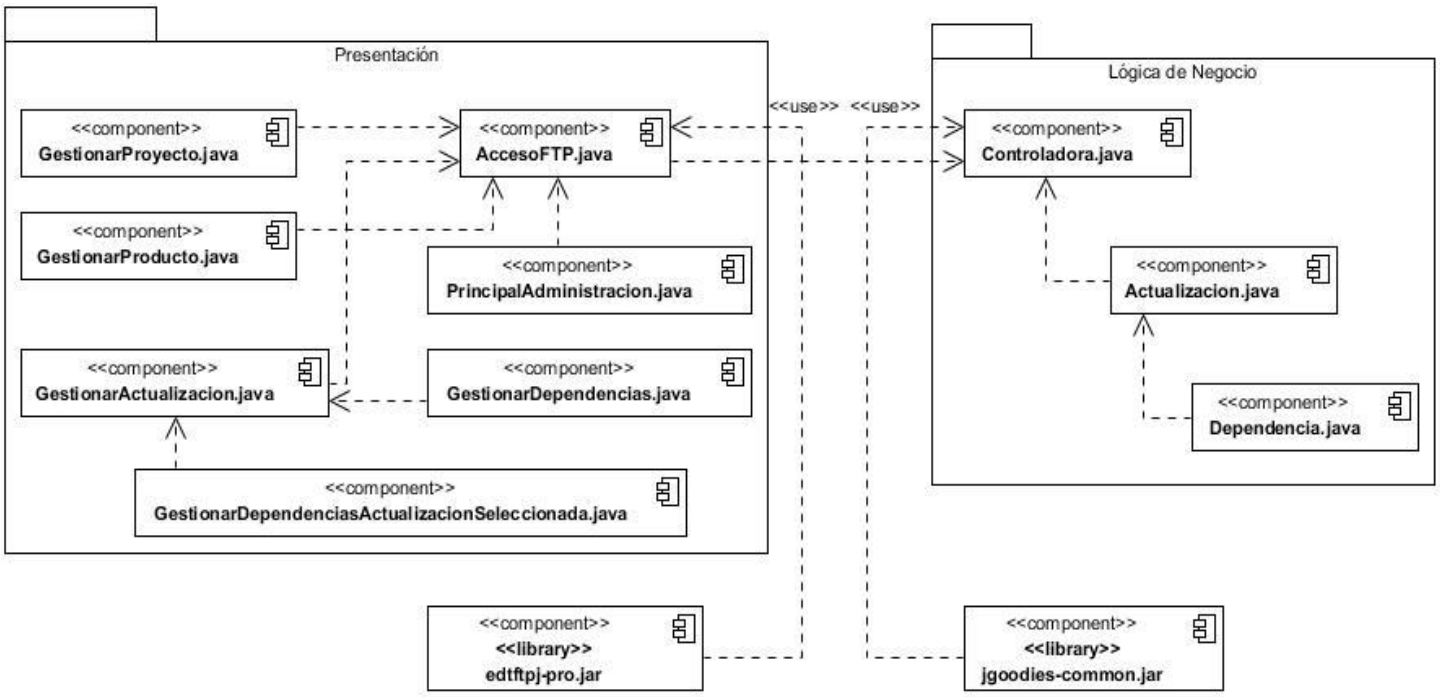

Fig. 13: Diagrama de Componentes de la aplicación administradora.

3.2 Estándar de codificación

Los estándares de codificación son necesarios en los proyectos, ya que permiten a los programadores trabajar de forma organizada y uniforme cuando existen diversos desarrolladores trabajando en el mismo sistema.

A continuación se describen las normas utilizadas por el equipo de desarrollo:

- Los nombres de los métodos y objetos comienzan con minúscula, en caso de tener dos o más palabras a partir de la segunda, el primer carácter de cada una se escribe en mayúscula. Ej: agregarDependencias ().
- Se usan comentarios para describir el objetivo de cada método y de algunas variables. Estos solo intentan describir el objetivo de la función o de la palabra reservada debajo, no describe el funcionamiento interior. Para comentar funciones se usa el doble asterisco entre barras "/**/" y para variables la doble barra "//".
- Solo se declara una instrucción por línea.
- No se declara más de una variable por línea.

3.3 Pruebas de software

El proceso de pruebas se centra en los procesos lógicos internos del *software*, asegurando que todas las sentencias se han comprobado, y en los procesos extremos funcionales; es decir, realizar las pruebas para la detección de errores y asegurar que la entrada definida produce resultados reales de acuerdo con los resultados requeridos. (Pressman, 2011)

Objetivos de las pruebas:

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

3.3.1 Niveles de prueba aplicados al sistema desarrollado

Funcional: Las pruebas Funcionales deben enfocarse en los requisitos funcionales, las pruebas pueden estar basadas directamente en los Casos de Uso (o funciones de negocio), y las reglas del negocio. Las metas de estas pruebas son: (Pressman, 2011)

- Verificar la apropiada aceptación de datos,
- Verificar el procesamiento y recuperación y la implementación adecuada de las reglas del negocio.

3.3.2 Tipo de prueba aplicada al sistema desarrollado

Función: Están enfocadas a validar las funcionalidades del sistema, así como sus métodos, servicios, casos de uso y especificaciones importantes que definen la calidad de su ejecución. (Pressman, 2011)

3.3.3 Método de prueba aplicado al sistema desarrollado

Conocimiento de la función específica para la que fue diseñado el producto (Caja Negra):
Las pruebas de caja negra son diseñadas para validar los requisitos funcionales sin fijarse en el funcionamiento interno de un programa. El método de prueba de caja negra se centra en el ámbito de información de un programa, de forma que se proporcione una cobertura completa de prueba. En este caso se utilizó la técnica partición equivalente del método de prueba caja negra.

La partición equivalente divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores (por ejemplo, proceso incorrecto de todos los datos de carácter) que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. (Pressman, 2011).

3.4 Diseño de prueba realizado para la solución propuesta

3.4.1 Pruebas de caja negra

A continuación se muestran algunos de los casos de pruebas correspondientes a los casos de uso establecidos en el capítulo anterior.

Casos de prueba

Para preparar los casos de pruebas es necesario contar con un número de datos que ayuden a su ejecución y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa, en dependencia de si desea hallar un error o probar una funcionalidad. (Pressman, 2011)

Tabla. 9: Caso de Prueba para el Caso de Uso Gestionar actualización.

Escenario	Descripción	Proyecto	Producto	Nombre	Tipo de Actualización	Cargar Actualización	Dependencias	Respuesta de sistema	Flujo central
EC 1.1 Registrar las actualizaciones correctamente.	Se seleccionan el proyecto y el producto. Se llena el campo del nombre, se selecciona el tipo de actualización y se agregan los archivos. Se presiona el botón "Registrar actualización". Se presiona el botón "Aceptar".	V Video Vigilancia	V Video vigilancia1	V video V	V Directorio	V /Dekstop	NA	Se registra correctamente la actualización que se sube al servidor.	Seleccionar botón "Archivo"/ Botón "Gestionar Actualizaciones"/ Llenar Campos/ Botón "Dependencias" (en caso de que la actualización dependa de otros componentes)/ Botón "Cargar actualización"/ Botón "Registrar Actualización

									".
EC 1.2 Errores al registrar actualizaciones.	Se llenan los campos pero se cometen errores al introducir los valores.	I No selecciona ningún proyecto	I Al no seleccionar proyecto no hay producto a seleccionar	I Campo vacío.	I No se selecciona ningún tipo de actualización.	I No cargar actualización.	NA	No se habilita el botón "Registrar actualización" mientras los campos no estén correctamente llenados.	Seleccionar botón "Archivo"/ Botón "Gestionar Actualizaciones"/ Llenar Campos/ Botón "Dependencias" (en caso de que la actualización dependa de otros componentes)/ Botón "Cargar actualización"/ Botón "Registrar Actualización".
		I No selecciona ningún proyecto.	I Al no seleccionar proyecto no hay producto a seleccionar.	V Plugins.	V Fichero	V /Desktop	NA		

		V Video Vigilancia	V Video vigilancia1	V Plugins.	I No se selecciona ningún tipo de actualización.	V /Desktop	NA		
EC 1.3 Se cancela la operación registrar las actualizaciones.	El usuario decide no registrar una nueva actualización y presiona el botón “Cerrar”.	NA	NA	NA	NA	NA	NA	El sistema cancela la operación de registrar actualización.	Seleccionar botón “Archivo”/ Botón “Gestionar Actualizaciones”/ Llenar Campos/ Botón “Dependencias” (en caso de que la actualización dependa de otros componentes)/ Botón “Cargar actualización”/ Botón “Cerrar”.

Tabla. 10: Caso de Prueba para el Caso de Uso Actualizar Ficheros

Escenario	Descripción	Respuesta de sistema	Flujo central
EC 1.1 Actualizar fichero.	Selecciona la actualización y se pulsa el botón “Actualizar”. Se muestra el mensaje “Desea continuar”. Se pulsa el botón “Aceptar”.	Descarga la actualización con sus dependencias y muestra una ventana con un mensaje: “Se ha descargado la actualización correctamente.”	Seleccionar actualización/ Botón “Actualizar”/ Botón “Aceptar”.
EC 1.1 Se cancela la operación actualizar fichero.	Selecciona la actualización y se pulsa el botón “Actualizar”. Se muestra el mensaje “Desea continuar”. Se pulsa el botón “Cancelar”.	El sistema cancela la operación y se cierra la ventana.	Seleccionar actualización/ Botón “Actualizar”/ Botón “Cancelar”.

Tabla. 11: Caso de Prueba para Caso de Uso Autenticar usuario.

Escenario	Descripción	FTP	Puerto	Usuario	Contraseña	Respuesta de sistema	Flujo central
EC 1.1 Autenticar usuario.	Ejecuta la aplicación y rellena los campos que se muestran en el formulario.	V 10.54.16.203	V 21	V Proyecto_Primicia	V Primicia	Se conecta la aplicación correctamente al servidor.	Llenar Campos/ Botón “Aceptar”.
EC 1.2 Errores al autenticar usuario.	El usuario entra valores erróneos en los campos	I 10.4.7 o Campo vacío.	I Campo vacío.	I Campo vacío.	I Campo vacío.	Muestra un mensaje señalando que existen campos incorrectos y	Llenar Campos/ Botón “Aceptar”.
		I	V	V	V		

		10.4.7 o Campo vacío.	21	Proyecto_Primicia.	Primicia.	que deben ser arreglados.	
		V 10.54.16.203	I Campo vacío.	V Proyecto_Primicia	V Primicia		
EC 1.3 Cancelar la operación autenticar usuario.	El usuario decide no autenticarse en el sistema y presiona el botón "Cancelar".	NA	NA	NA	NA	El sistema cancela la operación de autenticar usuario y regresa a la ventana iniciar.	Llenar Campos/ Botón "Cancelar".

El equipo de desarrollo, ha realizado dos iteraciones al sistema implementado con el objetivo de confirmar su correcto funcionamiento. En la primera iteración se encontraron 4 no conformidades y en la segunda iteración se encontró dos, las cuales se corrigieron correctamente.

Tabla. 12: No conformidades detectadas al aplicar los casos de pruebas al sistema en la primera iteración.

No. NC	Ubicación	Descripción	Impacto	Tipo de NC	Estado
1	Archivo/ Ventana "Gestionar producto".	A la palabra "plataforma" le falta una letra.	M	Ortográfico	Resuelta
2	Archivo/ Ventana "Gestionar actualización".	No funciona el Botón "Dependencias"	A	Funcionalidad	Resuelta.
3	Archivo/ Ventana "Gestionar	El sistema no responde cuando se escoge una plataforma nueva para un	A	Sistema	Resuelta.

	producto”/ Registrar producto.	nuevo producto.			
4	Archivo/ Ventana “Gestionar actualización”.	La palabra “actualización” está escrita sin tilde.	M	Ortográfico	Resuelta.

Tabla. 13: No conformidades detectadas al aplicar los casos de pruebas al sistema en la segunda iteración.

No. NC	Ubicación	Descripción	Impacto	Tipo de NC	Estado
1	Ventana Principal	El sistema no responde cuando se le da clic al botón “Descargar” proyectos.	A	Sistema	Resuelta.
2	Archivo/ Ventana “Gestionar actualización”/ Botón “Dependencia”/ Botón “Eliminar”.	El sistema no se actualiza cuando se elimina una dependencia.	A	Sistema	Resuelta.

3.5 Conclusiones parciales

En el presente capitulo quedaron plasmados todos los artefactos generados en la fase de construcción según la metodología RUP. Se realizó el diagrama de componentes, representando los elementos más importantes que forman parte del sistema. Se le realizaron pruebas de caja negra al sistema para validar su correcto funcionamiento, con el objetivo de obtener no conformidades y errores que puedan influir en su funcionamiento.

Conclusiones generales

Al término de la presente investigación, se concluyó que el sistema para actualizaciones:

- Brinda al usuario la posibilidad de centrar su máxima atención en tareas de mayor importancia sin tener que preocuparse del tiempo que se invierte en la búsqueda de las actualizaciones, ya que se obtienen los servicios de descarga de las mismas.
- Le posibilita a los especialistas del centro GEYSED, desde su puesto de trabajo, poder realizar cambios de versiones o actualizaciones a los componentes una vez instalados.
- Podrá ser utilizada en todos los proyectos productivos.

Recomendaciones

- Utilizar el sistema para todos los proyectos del centro no solamente los que trabajan con aplicaciones de escritorio.
- Establecer una vía de comunicación entre las entidades donde hayan sistemas desplegados y la universidad, para evitar que los especialistas del centro GEYSED tengan que llevar las actualizaciones.

Bibliografía y referencias bibliográficas

- Alegsa, Leandro. 2011. **AlLEGSA.com.ra. [En línea] 11 de Noviembre de 2011. [Citado el: 15 de Diciembre de 2013.] http://www.alegsa.com.ar/Dic/actualizar.php#sthash.F8o5qdkn.dpuf.**
- Dassen, J.H.M. y Stickelman, Chuck. 2012. **The Debian GNU/Linux FAQ. [En línea] 16 de Abril de 2012. [Citado el: 20 de Abril de 2014.] https://www.debian.org.**
- Ezust, Alan y Ezust, Paul. 2006. ***An Introduction to Design Patterns in C++ with Qt 4.*** **s.l. : Prentice Hall, 2006. Print ISBN-10: 0-13-187905-7.**
- Foundation, The Apache Software. 2013. **The Apache Poi Proyect. [En línea] 6 de Septiembre de 2013. [Citado el: 30 de Enero de 2015.] https://poi.apache.org/.**
- Garcia, Gilberto Pedraza. 2008. ***Evolución e Integración de Aplicaciones Legadas: Comenzar de Nuevo o Actualizar?*** **Bogotá: Universidad Piloto de Clombia : s.n., 2008.**
- García, Manuel Sierra. 2013. **aprenderaprogramar.com. [En línea] 2013. http://www.aprenderaprogramar.com.**
- inteco cert. 2009. **Instituno Nacional de Tecnologías de las Comunicaciones. [En línea] 15 de Agosto de 2009. [Citado el: 20 de Mayo de 2014.] http://cert.inteco.es/Proteccion/Actualizaciones_SW/Sistemas_Operativos/.**
- IPS Cuba. 2010. **Inter Press Service En Cuba. [En línea] 2010. http://www.ipscuba.net/.**
- Jacobson, Iva, Booch, Grady y Rumbaugh, James. 2010. ***El Proceso Unificado de Desarrollo de Software.*** **s.l. : Addison Wesley, 2010. I-S-B-N.**
- Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2010. ***El Lenguaje Unificado de Modelado.*** **s.l. : Addison Wesley, 2010. ISBN.**
- Java, Características del lenguaje. 2009. **[En línea] 2009. [Citado el: 25 de Noviembre de 2014.] http://www.mundojava.net/caracteristicas-del-lenguaje.html?Pg=java_inicial_4_1.html.**
- Java, Programación en. 2008. **[En línea] 28 de Agosto de 2008. [Citado el: 29 de Noviembre de 2014.] http://www.lenguajes-de-programacion.com/programacion-java.shtml.**

Bibliografía y referencias bibliográficas

- JTechLabs. 2006. **JTechLabs.com. [En línea] 2006. [Citado el: 5 de Diciembre de 2014.] http://www.jtechlabs.com/components/jdirectorychooser/.**
- Kaspersky Lab. 2011. **Kaspersky Lab Technical Support. [En línea] 19 de Agosto de 2011. [Citado el: 28 de Abril de 2014.] http://support.kaspersky.com/sp/ak8/update?qid=208280730.**
- Larman, Craig. 2003. ***UML y Patrones.* 2003. 2da, Edición.**
- Magazine, ActualPC. 2009. **PCactual. [En línea] 15 de julio de 2009. [Citado el: 20 de Abril de 2014.] http://www.pcactual.com.**
- Maldonado, Daniel. 2007. **El CoDiGo K. Qué son los IDE de Programación. [En línea] 3 de Septiembre de 2007. [Citado el: 5 de diciembre de 2014.] http://www.elcodigok.com.ar/2007/09/que-son-los-ide-de-programacin.html.**
- Martínez, Lic. Sergio G. 2010. **Gestión de Servicios Informáticos. *Administración y Control de Proyectos II.* 2010.**
- Microsoft, Corporation. 2009. **Microsoft Developer Network. [Online] Patterns & Practices Developer Center. Introduction to the Updater Application Block, Marzo 2009. [Cited: 11 29, 2013.] http://msdn.microsoft.com/en-us/library/ff650611.aspx.**
- Mora, Sergio Luján. 2006. ***C++ Paso a Paso.* 2006.**
- Mühlrad, Daniel. 2008. ***Patrones de diseño.* 2008.**
- Oracle Corporation. 2013. **NetBeans. [En línea] 3 de Junio de 2013. [Citado el: 20 de Enero de 2015.] https://netbeans.org/.**
- Pressman, Roger S. 2011. ***Software Engineering, a practitioner's approach.* s.l. : McGraw-Hill, 2011.**
- Qt Project. 2013. **Qt Project. [En línea] 26 de 09 de 2013. [Citado el: 23 de 02 de 2014.] https://qt-project.org/wiki/Category:Tools::QtCreator_Spanish.**
- Riehle, Dirk. 2000. ***Framework Design: A Role Modeling Approach.* s.l. : Swiss Federal Institute of Technology, 2000.**

Bibliografía y referencias bibliográficas

- ServidorFTP. 2011. **ServidorFTP. [En línea] 15 de Julio de 2011. [Citado el: 5 de Febrero de 2015.] http://www.mastermagazine.info.**
- Sommerville, Ian Prentice-Hall. 2002. ***Ingienería de software.* 2002. 6ta. Edición.**
- Swebok. 2004. **Capitulo_2_Requerimientos_de_Software. [aut. libro] Computer Society. *Guide to the Software Engineering Body of.* s.l. : IEEE, 2004.**
- Tabares, Marta Silvia, y otros. 2007. ***UN MÉTODO PARA LA TRAZABILIDAD DE REQUISITOS EN EL PROCESO UNIFICADO DE DESARROLLO.* Medellín (Colombia) : Revista EIA, 2007. ISSN 1794-1237 Número 8.**
- Technologies, Enterprise Distributed. 2012. **Enterprise Distributed Technologies. [En línea] 5 de Marzo de 2012. [Citado el: 6 de Febrero de 2015.] http:// enterprisedt.com/products/edftpj.**
- UCI, Universidad de las Ciencias Informaticas. 2012. **Nova. [En línea] 19 de Octubre de 2012. [Citado el: 15 de Mayo de 2014.] http://www.nova.cu.**
- VPository. 2010. **Visual Paradigm for UML Tutorial english. [aut. libro] Collavorative Visual Modeling Platform. 2010.**
- ZAO, Kaspersky Lab. 2013. **Kaspersky Lab. *Kaspersky Lab.* [En línea] 2013. http://latam.kaspersky.com/productos/productos-para-empresas/administration-kit.**

Anexos

Anexo 1: Entrevista realizada a los jefes de los proyectos del Centro GEYSED que trabajan con aplicaciones de escritorio.

- ¿Cómo funcionan los sistemas de actualizaciones del proyecto?
- ¿Qué problemas presentan los proyectos con la forma de actualización actual?
- ¿Qué ventajas traería la realización de una herramienta para realizar el proceso de actualizaciones de las aplicaciones?
- ¿Es necesario que los usuarios instalen o actualicen en el momento, o se puede descargar la actualización y luego instalarlas?
- ¿Qué tipo de ficheros deben almacenarse en el servidor para luego actualizar?

Anexo 2: Entrevista realizada a los desarrolladores del sistema operativo Nova

- ¿En qué marco de las ciencias informáticas se encuentran las actualizaciones?
- ¿Cómo funciona el sistema de actualización de Nova?
- ¿Cómo se actualiza el repositorio donde se encuentran almacenadas las actualizaciones?