

Universidad de las Ciencias Informáticas

Facultad 2



Título: Gupdater, sistema de actualización v2.0 para GRHS

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

Autores

Roberto Rafael Ramírez Martínez

Yosján Pérez Cuello

Tutores

Ing. Jenny De la Rosa Pasteur

Ing. Jorge Armando Túñez González

La Habana, Cuba

Junio, 2015

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos al Centro de Telemática de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Roberto Rafael Ramírez Martínez

Firma del autor

Yosján Pérez Cuello

Firma del autor

Ing. Jenny De la Rosa Pasteur

Firma del tutor

Ing. Jorge Armando Túñez González

Firma del tutor

DATOS DE CONTACTO

Datos del autor

Roberto Rafael Ramírez Martínez

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Correo electrónico: rramirez@estudiantes.uci.cu

Datos del autor

Yosján Pérez Cuello

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Correo electrónico: ypcuello@estudiantes.uci.cu

Datos del tutor

Ing. Jenny De la Rosa Pasteur

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Correo electrónico: jdelarosa@uci.cu

Datos del tutor

Ing. Jorge Armando Túnnez González

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Correo electrónico: janunez@uci.cu

RESUMEN

En la presente investigación se abordan los aspectos relacionados a un sistema de actualización automática (Gupdater) para el cliente (Gclient) de la solución Gestor de Recursos de Hardware y Software (GRHS), desarrollada por el Centro de Telemática (TLM) de la Facultad 2 de la Universidad de las Ciencias Informáticas (UCI), para el control de inventario a los medios informáticos (computadoras). Se describe la necesidad existente de un actualizador automático para Gclient. Se presenta la descripción de las funcionalidades y los artefactos asociados a la metodología de desarrollo adoptada para dar solución a la problemática planteada por el cliente. La propuesta de solución realizará la actualización automática de Gclient y entre sus beneficios se encuentra la recuperación del cliente ante fallos en el proceso de actualización, así como permitir actualizar una parte específica de Gclient. El actualizador se integrará a la administración (Gadmin) de GRHS para la configuración de las actualizaciones y la visualización de reportes, además utilizará el servidor (Gserver) existente para manejar el flujo de comunicación y gestionar las actualizaciones.

Palabras claves: actualización de software, actualización automática, actualizador.

DEDICATORIA

A mis padres, hermano, familia y amigos.

Roberto Rafael

A la familia y los mejores amigos, por el apoyo y la confianza recibida en estos años.

Yosján

ÍNDICE

INTRODUCCIÓN	1
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.....	5
Introducción.....	5
1.1 Conceptos	5
Software	5
Optimización de software.....	5
Actualización de software	5
Actualizador de software	5
1.2 Estudio de soluciones existentes.....	5
Windows Update.....	6
LiveUpdate.....	7
Actualizador SIGHO.....	8
Appupdater	8
Gupdater 1.0.....	9
1.3 Metodología de desarrollo	10
Extreme programming (XP)	10
1.4 Lenguajes de programación	12
Python	12
JavaScript.....	13
1.5 Herramientas y tecnologías utilizadas	13
Visual Paradigm-UML	13
Sublime Text.....	14
Xilema Base Web	14
Django	14
jQuery	15
Backbone.....	16
PostgreSQL	16
Conclusiones parciales.....	17
CAPÍTULO 2: EXPLORACIÓN Y PLANIFICACIÓN	18

Introducción.....	18
2.1 Propuesta de solución	18
2.2 Funcionalidades	19
2.3 Características no funcionales.....	20
Software	20
Hardware	21
2.4 Diagrama de despliegue.....	22
2.5 Fase de exploración	22
Involucrados en el sistema.....	23
2.6 Historias de usuario (HU)	23
2.7 Estimación de esfuerzo	24
2.8 Plan de iteraciones	25
2.9 Plan de entregas	26
Conclusiones parciales.....	27
CAPÍTULO 3: DISEÑO	28
Introducción.....	28
3.1 Arquitecturas y patrón arquitectónico.....	28
Arquitectura Cliente/Servidor	28
Arquitectura basada en componentes.....	29
Model Template View (MTV).....	30
3.2 Patrones de diseño.....	31
Patrones GRASP	31
Estándares de nomenclatura y codificación	32
3.3 Modelo de datos.....	32
3.4 Tarjetas Clase Responsabilidad Colaborador (CRC)	33
Conclusiones parciales.....	35
CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS	36
Introducción.....	36
4.1 Tareas de ingeniería.....	36
4.2 Proceso de pruebas	37

Tipos de pruebas	38
Métodos de pruebas	38
Estrategia de pruebas	38
Pruebas unitarias	39
Resultados de las pruebas unitarias	39
Pruebas de aceptación	40
Resultados de las pruebas de aceptación.....	41
Resultados de las pruebas de integración.....	42
Conclusiones parciales.....	43
CONCLUSIONES GENERALES.....	44
RECOMENDACIONES	45
REFERENCIAS BIBLIOGRÁFICAS	46
BIBLIOGRAFÍA.....	48
ANEXOS.....	52
Anexo 1: Historias de usuario.....	52
Anexo 2: Tareas de ingeniería.....	57
Anexo 3: Pruebas unitarias.....	62
Anexo 4: Pruebas de aceptación	66

INTRODUCCIÓN

El desarrollo informático, en su constante cambio, requiere que las soluciones desarrolladas sean cada vez más eficientes y cuenten con la seguridad y el soporte adecuado para garantizar un buen funcionamiento. En la actualidad los sistemas informáticos requieren de una disponibilidad continua y que los servicios brindados puedan ser consumidos de la manera más rápida y segura por los usuarios finales (1). En este sentido, preservar las mejores características de los sistemas y asegurar su actualización teniendo en cuenta el soporte a nuevas tecnologías y configuraciones es una tarea de vital importancia para los desarrolladores que aspiran a contar con un producto de calidad y confiabilidad.

Las actualizaciones al software, con el fin de conseguir altos índices de rendimiento, se presentan como una de las soluciones más eficientes en cuanto al soporte brindado a las aplicaciones desplegadas (2). Realizar estas actualizaciones puede resultar una tarea de difícil cumplimiento cuando no se cuenta con los recursos necesarios, factor que compromete la disponibilidad de los sistemas y los resultados obtenidos en la utilización de los mismos.

Con el objetivo de facilitar las actualizaciones se han generado a lo largo de la historia de la industria del software numerosas soluciones denominadas actualizadores de software. A nivel mundial dichas soluciones han garantizado la eficiencia en los sistemas informáticos de numerosas compañías y han permitido la corrección de vulnerabilidades que ponen en riesgo la privacidad de los usuarios, entre otros factores (3). Cuba no ha estado exenta de la necesidad de contar con sistemas que permitan la actualización automática de las soluciones desplegadas. Para obtener tales resultados el país se ha apoyado en varias entidades productoras de software. Una de estas entidades es la Universidad de las Ciencias Informáticas (UCI), que pertenece al Ministerio de Educación Superior (MES) y tiene la misión de graduar profesionales del sector informático que contribuyan a la informatización del país. También en la UCI se han desarrollado actualizadores automáticos. Uno de ellos es el desarrollado por el Centro de Telemática (TLM) para la actualización del cliente de la solución Gestor de Recursos de Hardware y Software (GRHS). Esta solución se encarga de contabilizar los recursos informáticos (computadoras) y tener un control de todas sus partes y piezas, así como del software instalado en ellos. GRHS soluciona algunos problemas surgidos de la intervención humana en el proceso de inventario, estos problemas son: demoras en el registro de reportes; imperfección en los inventarios por errores humanos en el almacenamiento y procesamiento de la información y la utilización de recursos humanos para esta tarea que maximiza la carga de trabajo y el empleo de tiempo. Para solucionar estas deficiencias GRHS cuenta con un agente (Gclient) instalado en las máquinas de la entidad, que envía la información de hardware y software del inventario realizado a un

servidor (Gserver) donde se procesa y almacena. A esta información se puede acceder mediante una aplicación web (Gadmin) que permite la visualización de los datos, realización de reportes e informes, entre otras funcionalidades. GRHS permite mejorar la gestión de los recursos, ofrece mayor facilidad en la obtención de informes tecnológicos, evita la duplicación y pérdida de la información, entre otros beneficios (4).

GRHS cuenta con el actualizador Gupdater, que se encuentra en la versión 1.0 y es el encargado de realizar las actualizaciones automáticas de Gclient, pero durante este proceso:

- Gclient no logra recuperarse de los fallos ocurridos durante su actualización.
- Los administradores no tienen conocimiento de los errores en caso de fracaso de la actualización.
- Los administradores no conocen si la actualización fue satisfactoria o no.
- No se tiene conocimiento de los resultados de las actualizaciones realizadas.

A partir de la situación problemática expuesta se identifican los siguientes componentes del diseño teórico de la investigación:

Problema a resolver

¿Cómo contribuir a perfeccionar la actualización de Gclient en el sistema GRHS?

Objeto de estudio

Proceso de actualización en aplicaciones informáticas.

Objetivo general

Desarrollar una aplicación informática para perfeccionar la actualización de Gclient en el sistema GRHS.

Objetivos específicos

- Elaborar el marco teórico de la investigación.
- Definir las herramientas informáticas y la metodología de desarrollo de software a utilizar en la implementación del sistema.
- Definir las funcionalidades que debe cumplir el sistema.
- Implementar un sistema informático para perfeccionar la actualización de Gclient que cumpla con los requerimientos definidos.
- Validar la solución propuesta.

Campo de acción

El sistema de actualización de Gclient.

Idea a defender

Con el desarrollo de una aplicación que perfeccione el proceso de actualización de Gclient en el sistema GRHS se automatizará el proceso de actualización de los clientes remotos minimizando el uso de recursos humanos, ganando en tiempo y planificación. Además se garantizaría la disponibilidad del cliente remoto.

Tareas de investigación

1. Determinar los elementos del diseño teórico para garantizar el éxito de la investigación.
2. Analizar los sistemas de actualización de software existentes para determinar los patrones, técnicas actuales y las buenas prácticas.
3. Analizar el actualizador actual para determinar sus deficiencias.
4. Determinar las funcionalidades que debe ofrecer el sistema para cumplir con las necesidades del cliente.
5. Realizar pruebas a cada una de las funcionalidades para garantizar que se cumplan con las necesidades del cliente.

Métodos de investigación científica

El método científico se hace concreto en las diversas etapas o pasos que se deben dar para solucionar un problema. Esos pasos son las técnicas o procesos. Los objetos de investigación determinan el tipo de método que se va a emplear. Los métodos de investigación científica se dividen en dos: empíricos y teóricos, pero, en realidad, en el proceso de investigación, estos métodos nunca están separados. Unos y otros emplean técnicas específicas, lo mismo que técnicas comunes a ambos (5).

Métodos teóricos

Analítico-Sintético: Se analizan las teorías presentadas en las bibliografías consultadas y se sintetizan los elementos más importantes de los sistemas de actualización en aplicaciones informáticas.

Modelación: Se utiliza para diseñar modelos y diagramas que describan la realidad con el objetivo de entender el flujo de operaciones en los sistemas de actualizaciones automáticas.

Métodos empíricos

Entrevista: Se realizan varias entrevistas al cliente para obtener información sobre las principales ventajas y desventajas del actualizador Gupdater 1.0 y sobre las buenas prácticas que implementan otros actualizadores. Sobre esa base se determinan las funcionalidades que debe tener el nuevo sistema de actualización del cliente GRHS.

Observación: Se evalúa el funcionamiento de soluciones similares aplicadas al proceso de actualización en sistemas informáticos, determinando estándares y buenas prácticas aplicables al sistema de actualización Gupdater 2.0.

Estructura del documento

El presente documento se compone de cuatro capítulos, estructurados de la siguiente forma:

Capítulo 1. Fundamentación teórica: En este capítulo se define el marco teórico conceptual constituyendo el punto de partida de la investigación. Se realiza el estado del arte de los sistemas actualizadores automáticos y se abordan las tecnologías y herramientas utilizadas en la implementación de la solución, así como la metodología de desarrollo a seguir.

Capítulo 2. Exploración y planificación: En el capítulo se identifican las funcionalidades que debe tener la solución, se detallan las historias de usuario y se realiza la estimación del esfuerzo. Se presenta la propuesta de solución al problema de investigación y se crea el plan de iteraciones inherente a la metodología de desarrollo seleccionada.

Capítulo 3. Diseño: Se elaboran los artefactos propuestos por la Metodología XP para la fase de diseño. Se mencionan los patrones de diseño utilizados, se detallan las arquitecturas de software implementadas en cada sub-sistema, el modelo arquitectónico por el que se rige la solución y el modelo de datos empleado para crear la infraestructura de base de datos necesaria.

Capítulo 4. Implementación y pruebas: Se realiza la confección de las tareas de ingeniería como artefacto que guía el proceso de desarrollo de software. Se presentan los resultados de las pruebas unitarias, de integración y de aceptación que se realizan a la solución y el resultado obtenido de las mismas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se describen los conceptos fundamentales relacionados con el problema a resolver. Se abordan las principales tecnologías empleadas y se selecciona la metodología de desarrollo de software a utilizar. Además, se presenta una descripción de los lenguajes de programación empleados en la solución. Este capítulo abarca el estudio de algunos actualizadores existentes a nivel mundial.

1.1 Conceptos

Software

El término inglés original define el concepto por oposición a hardware: blando-duro, en referencia a la intangibilidad de los programas y corporeidad de la máquina. Software es un término genérico que designa al conjunto de programas de distinto tipo (sistema operativo y aplicaciones diversas) que hacen posible operar con el ordenador (6).

Optimización de software

La optimización de software es una rama de la Ingeniería de Software que trata de convertir programas existentes en otros programas que realicen las mismas tareas en menos tiempo, con menos requerimientos de memoria, o en general empleando los recursos de forma óptima (7).

Actualización de software

Cambiar o alterar una aplicación por una versión más actual de la misma. Esta actualización puede ser por un parche, un service-pack o una actualización completa del programa (8).

Actualizador de software

Un actualizador de software automatiza los pasos implicados en el proceso de actualización del software (9).

1.2 Estudio de soluciones existentes

Para caracterizar a los actualizadores de software o actualizadores automáticos se definieron cuatro grupos fundamentales: 1. Herramientas de despliegue por paquetes, 2. Productos actualizadores genéricos, 3. Productos actualizadores del proveedor y los 4. Productos actualizadores en caliente (10).

A pesar de la existencia de cuatro tipos fundamentales de actualizadores, en (9) se proporciona un diagrama general del proceso de actualización de software que debe servir de referencia para estos cuatro tipos de actualizadores.

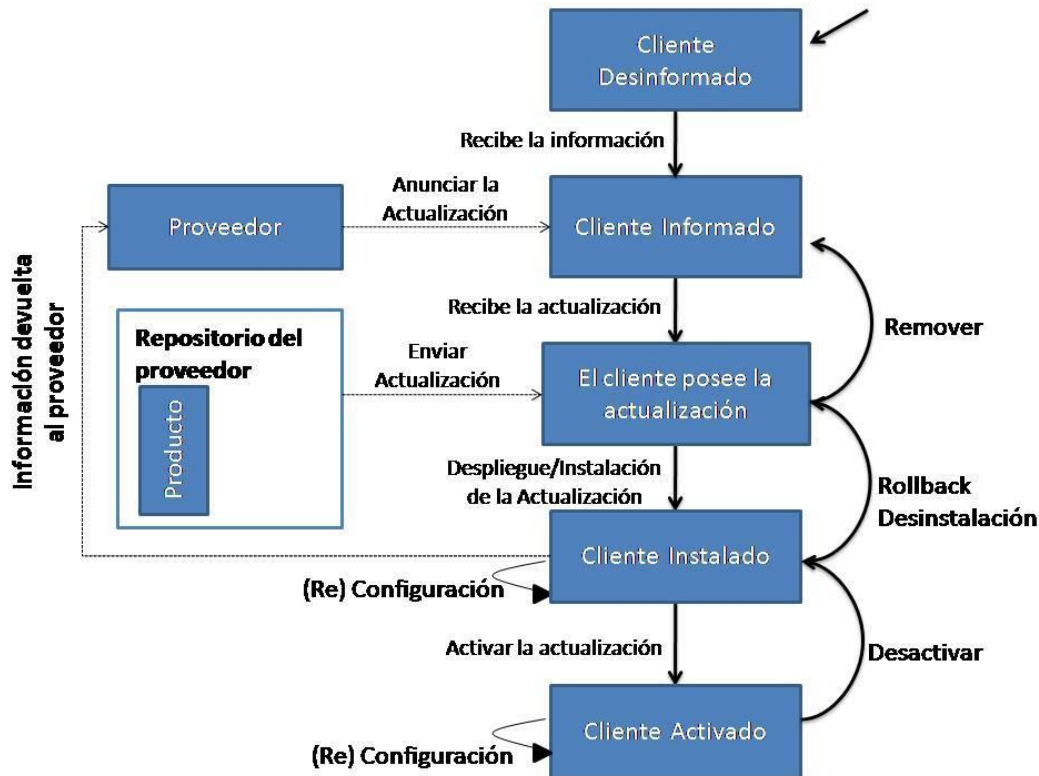


Figura 1 Modelo general del proceso de actualización de software (8)

Como muestra la Figura 1, la interacción entre el proveedor y el cliente es la clave de los sistemas de actualizaciones automáticas. El modo en que se realiza dicha interacción puede ser a decisión de los propios desarrolladores. Un aspecto importante en este modelo general de actualización es el llamado Rollback, esta es una de las funciones fundamentales para la recuperación del sistema ante fallos en la ejecución de las actualizaciones. Sin embargo no todos los sistemas de actualización cuentan con esta función incorporada y el modo de implementarla también varía en dependencia de la programación realizada para la funcionalidad.

Teniendo en cuenta lo antes expuesto, a continuación se mencionan y describen brevemente algunos de los sistemas de actualización de software desarrollados en las últimas décadas. Todos ellos aplican su propio mecanismo de actualización y no todos pertenecen al mismo grupo de actualizadores.

Windows Update

A partir de Windows 98, Microsoft incluyó el módulo Windows Update, que contactaba al sitio oficial con un ActiveX que permitía ver la información del sistema y descargar las actualizaciones adecuadas. A partir de Windows Me, Microsoft decidió hacer más fácil la búsqueda de actualizaciones, incluyendo *Actualización automática*, en su término en inglés (Automatic update). A partir de la versión Windows XP, Microsoft desarrolló Microsoft Update, herramienta

que facilitaría más las actualizaciones ya que éste no sólo buscaba actualizaciones para Windows, sino que de forma automática busca actualizaciones para el paquete Microsoft Office y las instala (11).

Ventajas:

- Windows Update se instala junto con el sistema operativo Windows.
- Chequea constantemente en busca de nuevas actualizaciones disponibles y las instala.
- Busca nuevas versiones de programas que pueden mejorar el entorno de trabajo y notifica su existencia.

Desventajas:

- Es propiamente para el Sistema Operativo Windows.
- Se centra en determinar las vulnerabilidades del sistema operativo e instalar las actualizaciones que las corrijan, sin tener en cuenta las actualizaciones a otros programas que no pertenezcan a la gama de productos Microsoft Office.

Este sistema se descarta como solución al problema a resolver por ser software propietario de la empresa Microsoft y porque no es configurable para entornos de actualización fuera de la gama de productos Microsoft Office y el sistema operativo Windows.

LiveUpdate

Creado en 1996, ha mantenido sus funciones hasta nuestros días. Con el clic de un botón, LiveUpdate lanza un asistente que guía al usuario a través del proceso de actualización de software. En primer lugar, LiveUpdate detecta la disponibilidad de una conexión a Internet o módem y automáticamente se conecta a un servidor de Symantec. Una vez que LiveUpdate se conecta al sitio de Symantec, el sistema descargará la información de archivos correspondientes a ese producto, eliminando la necesidad de los usuarios de desplazarse a través de la información que no se aplica a su producto. Si un usuario utiliza LiveUpdate para Norton AntiVirus, LiveUpdate tendrá que localizar y descargar toda nueva información relacionada con las definiciones de virus o cualquier otra información de Norton AntiVirus. Una vez que el usuario confirma que quieren promulgar la actualización, LiveUpdate entonces completa la operación. Para aumentar aún más la productividad, LiveUpdate funciona en segundo plano para que los usuarios puedan seguir trabajando mientras el programa se está actualizando (12).

Ventajas:

- Soporta una conexión directa desde el servidor de la red local, sin necesidad de conectar individualmente cada ordenador, por lo que se ahorra tiempo y ancho de banda.
- La descarga de este programa es totalmente gratis para cualquier usuario interesado.

Desventajas:

- Solo actualiza productos de la compañía Symantec.
- No dispone de mecanismo que verifique la integridad de lo descargado.
- No está disponible para múltiples plataformas.
- Aunque es distribuido gratuitamente es un producto privativo.

El sistema no es utilizado como solución al problema planteado por ser propiedad de la empresa Symantec, por actualizar solo productos de dicha empresa y por no contar con versiones disponibles para múltiples plataformas.

Actualizador SIGHO

Fue diseñado para realizar la actualización del Sistema de Información para la Gerencia Hospitalaria (SIGHO). Funciona como una herramienta que facilita el proceso de actualización del SIGHO, el cual guía al usuario mediante una interfaz que le permitirá realizar la sustitución de los antiguos archivos del sistema (13).

Ventajas:

- Posee un mecanismo de autenticación que evita que personal ajeno realice modificaciones que puedan poner en riesgo el funcionamiento del sistema.

Desventajas:

- Solo facilita la actualización para el SIGHO.
- El paquete de actualización debe ser copiado manualmente en el cliente donde se encuentra instalado el SIGHO.

El sistema se descarta como solución porque no presenta funcionalidades de actualización automática desde un servidor y el mismo solo funciona para el SIGHO.

Appupdater

Es otra de las herramientas de instalación y actualización de programas que se maneja en línea de comandos. No hay diferencia con los comandos apt-get o yum de Linux, que son la vía tradicional de instalación de aplicaciones en este sistema operativo. Appupdater actualiza periódicamente la base de datos con información sobre cientos de aplicaciones para que el usuario la consulte e instale. Además de ser multiplataforma, incluye soporte para dispositivos USB y se pueden sugerir nuevas aplicaciones que se incluirán en la lista de aplicaciones soportadas. Crear un repositorio personalizado si se desea es otra de las posibilidades que brinda Appupdater (14).

Ventajas:

- Appupdater es multiplataforma con soporte para Windows/UNIX/Linux/MAC OS X.
- Realiza actualizaciones automáticas diariamente sin la intervención de un usuario.
- Se descarga totalmente gratis para cualquier usuario interesado.

Desventajas:

- Appupdater descarga una lista de todas las aplicaciones, lo que instala no es enviado ni mostrado, por lo que no se tiene conocimiento de los ficheros o registros actualizados.
- Durante la instalación del programa se descarga directamente desde el sitio web, no es posible hacer un seguimiento de lo que se está instalando y si falla la conexión ocurre un error.
- Presenta problemas de compatibilidad con versiones en español de programas en Windows u otras versiones.

Se descarta esta solución porque Appupdater durante la actualización descarga directamente desde el sitio web y no se puede hacer un seguimiento de las aplicaciones que instala o saber si ocurre un error en este proceso. Si se incluye una actualización en la lista de aplicaciones la misma sólo se podrá instalar para el sistema operativo Windows.

Gupdater 1.0

Gupdater 1.0 es un software desarrollado con Python 2.7, es un actualizador automático desarrollado para actualizar el cliente GRHS (Gclient). Se instala en cada cliente donde está ejecutándose Gclient y es el encargado de actualizarlo automáticamente.

Ventajas:

- Almacena en registro de logs las trazas generadas en el proceso de ejecución.
- Detecta automáticamente la última versión disponible del software instalado.
- Realiza actualizaciones automáticas puesto que no interviene un usuario en el proceso de actualización.
- Es multiplataforma con soporte para Windows y Linux.
- Comprueba la integridad de la descarga mediante la verificación del md5.

Desventajas:

- No realiza una recuperación de Gclient ante los fallos ocurridos durante el proceso de actualización.
- No realiza notificaciones a Gserver de los cambios realizados (sean o no satisfactorios).
- La actualización debe realizarse por reemplazo total del cliente.

Luego de haber analizado las principales características presentes en los sistemas mencionados anteriormente, se identifican un conjunto de aspectos que se pueden interpretar como debilidades, excesos, carencias o características propias de un negocio en específico. Entre los aspectos que aportan valor al desarrollo de la solución propuesta se encuentran:

- Una actualización es concebida por algunos actualizadores como una nueva versión del software, cuando puede tratarse del remplazo de uno de los componentes del mismo, un cambio de configuración o la corrección de un error.
- Entre los propósitos de un actualizador automático está solucionar problemas de seguridad a las aplicaciones que actualizan, pero ellos por si solos no garantizan su seguridad. Tanto la conexión a los repositorios compartidos por un recurso local como el uso propio de los actualizadores en la mayoría de los casos, no requieren de autenticación. Esto hace que los procesos de actualización no sean controlados y evita en ocasiones tener que generar reportes de interés, como un historial de versiones donde se relacione el personal que ejecutó el proceso en distintos momentos.

Atendiendo a la existencia de la solución Gupdater 1.0 se plantea la necesidad de evaluar la factibilidad de dar soporte a la misma o desarrollar una nueva versión. El equipo de desarrollo, en conversación con el cliente acuerda desarrollar una nueva versión Gupdater 2.0 que se integre al sistema GRHS, utilizando la arquitectura basada en componentes empleada en Gclient. Además se pretende centralizar la información y Gupdater 1.0 utiliza un servidor web independiente de la solución GRHS. También serán añadidas nuevas funcionalidades. Se decide reutilizar código ya existente que facilitará y agilizará el proceso de desarrollo así como parte de la lógica del negocio establecida por Gupdater 1.0.

1.3 Metodología de desarrollo

Extreme programming (XP)

La programación extrema o Extreme Programming (XP) es un enfoque de la ingeniería de software formulado por Kent Beck, autor del primer libro sobre la materia, Extreme programming Explained: Embrace Change (1999). Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. XP tiene éxito porque hace hincapié en la satisfacción del cliente. Faculta a los desarrolladores para responder con seguridad a las necesidades cambiantes de los clientes. XP enfatiza el trabajo en equipo. Los gerentes, clientes y desarrolladores son socios iguales en un equipo de

colaboración. Implementa un entorno sencillo, pero eficaz que permite a los equipos ser altamente productivos. XP se basa en cinco principios fundamentales: la comunicación, la sencillez, la retroalimentación, el respeto y el valor.

Prácticas básicas de la programación extrema

- **Equipo completo:** Forman parte del equipo todas las personas que tienen algo que ver con el proyecto, incluido el cliente y el responsable del proyecto.
- **Planificación:** Se hacen las historias de usuario y se planifica en qué orden se van a hacer y las mini-versiones. La planificación se revisa continuamente.
- **Test del cliente:** El cliente, con la ayuda de los desarrolladores, propone sus propias pruebas para validar las mini-versiones.
- **Versiones pequeñas:** Las mini-versiones deben ser lo suficientemente pequeñas como para poder hacer una cada pocas semanas. Deben ser versiones que ofrezcan algo útil al usuario final y no fragmentos de código que no tengan funcionalidad por sí solos.
- **Diseño simple:** Hacer siempre lo mínimo imprescindible de la forma más sencilla posible. Mantener siempre sencillo el código.
- **Pareja de programadores:** Los programadores trabajan por parejas (dos delante del mismo ordenador) y se intercambian las parejas con frecuencia (un cambio diario).
- **Desarrollo guiado por las pruebas automáticas:** Se deben realizar programas de prueba automática y deben ejecutarse con mucha frecuencia. Cuantas más pruebas se hagan, mejor.
- **Integración continua:** Debe tenerse siempre un ejecutable del proyecto que funcione y en cuanto se tenga una nueva funcionalidad, debe recompilarse y probarse. Es un error mantener una versión congelada dos meses mientras se hacen mejoras y luego integrarlas todas de golpe. Cuando falle algo, no se sabe qué es lo que falla.
- **El código es de todos:** Cualquiera puede y debe conocer cualquier parte del código. Para eso se hacen las pruebas automáticas.
- **Normas de codificación:** Debe haber un estilo común de codificación (no importa cuál), de forma que parezca que ha sido realizado por una única persona.
- **Metáforas:** Hay que buscar unas frases o nombres que definan cómo funcionan las distintas partes del programa, de forma que sólo con los nombres se pueda entender qué es lo que hace cada parte del programa. Ayuda a que todos los programadores y el cliente sepan de qué se trata y que no existan malentendidos.

- **Ritmo sostenible:** Se debe trabajar a un ritmo que se pueda mantener indefinidamente. Esto quiere decir que no debe haber días muertos en que no se sabe qué hacer y que no se deben hacer un exceso de horas otros días. Al tener claro semana a semana lo que debe hacerse, hay que trabajar duro en ello para conseguir el objetivo cercano de terminar una historia de usuario o mini-versión (15).

Se decidió utilizar XP debido a que se adapta en gran medida tanto al tipo de proyecto a desarrollar como a las condiciones de trabajo. A continuación se exponen varias de las razones que llevaron al uso de esta metodología:

- El proyecto es pequeño.
- Los requisitos del sistema cambian frecuentemente.
- El cliente forma parte del equipo de desarrollo. Mediante la aplicación de XP se puede lograr una retroalimentación mayor y lograr un producto que satisfaga sus necesidades.
- El riesgo de desarrollo es elevado debido al corto tiempo de entrega planteado y a los continuos cambios de requerimientos. XP está diseñada para mitigar los riesgos en proyectos con estas características.
- Poca disponibilidad de personal. El sistema debe ser realizado por dos personas solamente, no siendo posible la existencia de muchos roles ni la especialización en un rol específico por parte de los miembros.
- Uno de los principios básicos de XP es la programación en equipos pequeños (2 a 12 personas) con pocos roles, pudiendo los miembros del equipo intercambiar responsabilidades en un momento determinado.

1.4 Lenguajes de programación

Python

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses *Monty Python*. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos. Un lenguaje interpretado o de script es aquel que se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora (lenguajes compilados). Python tiene, no obstante, muchas de las características de los lenguajes compilados, por lo que se podría decir que es semi-interpretado. En Python, como en Java y muchos otros lenguajes, el

código fuente se traduce a un pseudo código máquina intermedio llamado bytecode la primera vez que se ejecuta, generando archivos `.pyc` o `.pyo` (bytecode optimizado), que son los que se ejecutarán en sucesivas ocasiones (16).

Python es seleccionado como lenguaje para el desarrollo porque la infraestructura de GRHS se basa en este lenguaje y Gupdater 2.0 será un sub-sistema más de dicha infraestructura; forma parte de la solicitud del cliente la utilización de este lenguaje para la implementación; los desarrolladores podrán abstraerse de la plataforma en la que va a operar la solución por la portabilidad que ofrece el desarrollo con este lenguaje.

JavaScript

JavaScript es un lenguaje de scripting multiplataforma y orientado a objetos. Es un lenguaje pequeño y liviano. Dentro de un ambiente de host, JavaScript puede conectarse a los objetos de su ambiente y proporcionar control programático sobre ellos.

JavaScript contiene una librería estándar de objetos, tales como Array, Date, y Math, y un conjunto central de elementos del lenguaje, tales como operadores, estructuras de control, y sentencias.

JavaScript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con JavaScript se pueden crear diferentes efectos e interactuar con nuestros usuarios (17).

1.5 Herramientas y tecnologías utilizadas

Visual Paradigm-UML

Visual Paradigm-UML es una herramienta diseñada para usuarios como ingenieros de software, analistas de sistemas, arquitectos de sistemas y otros que estén interesados en el diseño de software orientado a objetos. Con VP-UML se pueden crear los diferentes diagramas de UML con solo realizar la operación de arrastrar y soltar (18).

A continuación se muestran las razones de elección:

- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista) y Linux.
- Permite modelar diagramas de base de datos.
- Se pueden realizar diagramas de despliegue del producto, donde se plantea la distribución física que debe tener la solución.

Sublime Text

Sublime Text es un editor de texto y editor de código fuente que está escrito en C++ y Python para los plugins. Desarrollado originalmente como una extensión de Vim, con el tiempo fue creando una identidad propia. Se distribuye de forma gratuita, sin embargo no es software libre o de código abierto, se puede obtener una licencia para su uso ilimitado, pero no disponer de ésta no genera ninguna limitación más allá de una alerta cada cierto tiempo (19).

A continuación se muestran las razones de elección:

- Mini mapa: consiste en una pre-visualización de la estructura del código, es muy útil para desplazarse por el archivo cuando se conoce bien la estructura de éste.
- Soporte nativo para varios lenguajes: Soporta de forma nativa 43 lenguajes de programación y texto plano y entre ellos se encuentra Python.
- Búsqueda dinámica: Se puede hacer búsqueda de expresiones regulares o por archivos, proyectos, directorios, una conjunción de ellos o todo a la vez.
- Auto completado y marcado de llaves: Se puede ir a la llave que cierra o abre un bloque de una forma sencilla.
- Paleta de comandos: Un intérprete de Python diseñado solo para el programa con el cual se puede realizar infinidad de tareas.
- Coloreado y envoltura de sintaxis: Si se escribe en un lenguaje de programación o marcado, resalta las expresiones propias de la sintaxis de ese lenguaje para facilitar su lectura.
- Pestañas: Se pueden abrir varios documentos y organizarlos en pestañas.
- Resaltado de paréntesis: Cuando el usuario coloca el cursor en un paréntesis, corchete o llave, resalta ésta y el paréntesis, corchete o llave de cierre o apertura correspondiente (19).

Xilema Base Web

Xilema Base Web es un marco de trabajo desarrollado en el Centro de TLM que está constituido por Django como framework base, librerías de JavaScript como son jQuery y Backbone, además implementa las pautas de diseño de la Universidad (20).

Django

Es un framework para desarrollo web con Python como lenguaje base. Este framework nació en el otoño boreal de 2003, cuando los desarrolladores del diario *Lawrence Journal-World*, Adrian Holovaty y Simon Willison, comenzaron a usar Python para crear sus aplicaciones. El equipo de *The World Online*, responsable de la producción y mantenimiento de varios sitios locales de noticias, prosperaba en un entorno de desarrollo dictado por las fechas límites del periodismo.

Para los sitios -- incluidos LJWorld.com, Lawrence.com y KUsports.com -- los periodistas (y los directivos) exigían que se agregaran nuevas características y que aplicaciones enteras se crearan a una velocidad vertiginosa, a menudo con sólo días u horas de preaviso. Es así que Adrian y Simon desarrollaron por necesidad un framework de desarrollo web que les ahorrara tiempo -- era la única forma en que podían crear aplicaciones que se pudieran mantener en tan poco tiempo. En el verano boreal de 2005, luego de haber desarrollado este framework hasta el punto en que estaba haciendo funcionar la mayoría de los sitios World Online, el equipo de World Online, que ahora incluía a Jacob Kaplan-Moss, decidió liberar el framework como software de código abierto. Lo liberaron en julio de 2005 y lo llamaron Django, por el guitarrista de jazz Django Reinhardt (21).

Una característica común en los framework de desarrollo web es la implementación del patrón Modelo Vista Controlador (MVC) que permite una optimización, organización y mejor distribución de las funciones de cada clase, objetos y funciones en la aplicación. En Django se aplica la misma filosofía pero con un cambio de nombre, pues este patrón pasa a llamarse MTV, que es el acrónimo de Modelo Plantilla Vista (del inglés, Model Template View). En semejanza con MVC, en el MTV cada uno de estos componentes sería: Model (modelo de datos, el mismo para MVC), Template (plantillas de datos, las vistas en MVC) y View (vistas de datos, los controladores en MVC) (21).

A continuación se muestran las principales características del framework Django utilizadas en la implementación del Módulo de Administración y el Servidor de Actualizaciones de Gupdater 2.0:

- Cuenta con un potente gestor de plantillas HTML que permiten reutilizar la estructura establecida en una plantilla base, modificando solo los bloques necesarios en las plantillas hijas.
- Django emplea el Mapeo Objeto-Relacional (ORM) para crear una base de datos orientada a objetos virtuales.
- Permite la conexión con el sistema gestor de base de datos PostgreSQL, que es de código abierto y libre de licencias comerciales.
- Cuenta con soporte para *URLs amigables*, gestionadas por el URLConfig, que sirve de intermediario entre la petición del navegador web y las vistas (controladores en Django).
- Permite la creación de múltiples aplicaciones dentro de un proyecto, lo que posibilita la inserción de nuevos módulos sin realizar cambios en los existentes.

jQuery

jQuery es un framework JavaScript, que sirve como base para la programación avanzada de aplicaciones, aporta una serie de funciones o códigos para realizar tareas habituales. Este

framework JavaScript, ofrece una infraestructura con mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Con jQuery se obtienen mejoras en la creación de interfaces de usuario, efectos dinámicos y aplicaciones que hacen uso de AJAX de un modo más eficiente y fácil de implementar (22).

Backbone

Backbone.js permite construir aplicaciones usando JavaScript siguiendo el patrón MVC (Modelo-Vista-Controlador). Muchas empresas han usado este framework para construir sus sitios web y aplicaciones. El objetivo de Backbone.js ha sido desde sus inicios probar y definir un conjunto de estructuras de datos (Models y Collections) junto al manejo de la interfaz por medio de Vistas y URLs que fueran útiles cuando se construyan aplicaciones JavaScript (23).

PostgreSQL

PostgreSQL es un sistema gestor de base de datos. Sus características técnicas lo hacen uno de los más potentes y robustos del mercado. Su desarrollo comenzó hace más de 16 años, y durante este tiempo la estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema (24).

Entre sus principales ventajas destacan:

- Ampliamente popular e ideal para tecnologías web.
- Fácil de administrar.
- Su sintaxis SQL es estándar y fácil de aprender.
- Es multiplataforma.

Gupdater 2.0 utiliza las tecnologías antes mencionadas por su integración al sistema GRHS. La utilización de las mismas garantiza la ejecución favorable de la solución en la infraestructura GRHS y por tanto se evita la incompatibilidad tecnológica.

Conclusiones parciales

En este capítulo se presentaron los conceptos fundamentales relacionados con el objeto de estudio de la investigación, tales como: conceptos, estudio de soluciones existentes, metodología de desarrollo de software, herramientas y tecnologías. A pesar de que los análisis hechos arrojaron que las soluciones existentes para el desarrollo del producto no brindan una solución completa, fueron de alto valor para la solución concebida. El perfil tecnológico queda estructurado de la siguiente manera: metodología de desarrollo: XP, Python y JavaScript como lenguajes de programación, Xilema Base Web como marco de trabajo y PostgreSQL como sistema gestor de base de datos.

CAPÍTULO 2: EXPLORACIÓN Y PLANIFICACIÓN

Introducción

Para facilitar la comprensión del sistema este capítulo tiene como objetivo detallar los aspectos fundamentales del mismo, tales como las funcionalidades, las historias de usuario y la estimación del esfuerzo. Se presenta la propuesta de solución al problema de investigación y se crea el plan de iteraciones inherente a la metodología de desarrollo seleccionada.

2.1 Propuesta de solución

Como propuesta de solución se plantea el desarrollo de Gupdater 2.0 compuesto por: un **Ciente de Actualización**, un **Módulo de Administración** en Gadmin y un **Servidor de Actualizaciones** integrado a Gserver. El Cliente de Actualización se instalará con el cliente GRHS en cada puesto de trabajo y será el encargado de la actualización de dicho cliente. El Módulo de Administración posibilitará la inserción y modificación de las actualizaciones así como la visualización de reportes. Estas actualizaciones son procesadas por el Servidor de Actualizaciones y puestas a disposición del Cliente de Actualización.

Al concluir el proceso de actualización el Cliente de Actualización envía reportes a Gserver. Estos reportes incluyen el estado final del proceso de actualización (éxito, recuperación o fracaso), la dirección IP del cliente actualizado, el Sistema Operativo de la PC, la versión de actualización con la que funciona el cliente después de realizado el proceso, la fecha en que se realizó la actualización y el error ocurrido en el proceso en caso de que el estado final no sea exitoso.

Para suplir una de las deficiencias de Gupdater 1.0, la solución propuesta incluye las actualizaciones de tipo parcial. Este tipo de actualización permite cambiar solo los componentes (plugins) que han sido modificados para el cliente. Es posible eliminar la base de datos del cliente para que se genere con las nuevas modificaciones realizadas por algún componente. Es posible limpiar el registro de logs evitando el cúmulo de información innecesaria que satura el fichero en cuestión. También se puede sustituir el fichero de configuración del cliente, permitiendo añadir y reasignar valores de configuración. Estas nuevas funcionalidades permiten que sea más eficiente el proceso de actualización, evitando la copia innecesaria de un cliente completo, mediante el reemplazo total, cuando sólo se requieren pequeños cambios.

Como se ha dicho la solución propuesta tiene interacción en todos los sub-sistemas de GRHS y se considera un sub-sistema más en el negocio GRHS. La Figura 2 muestra la representación de la propuesta de solución y su integración a GRHS.

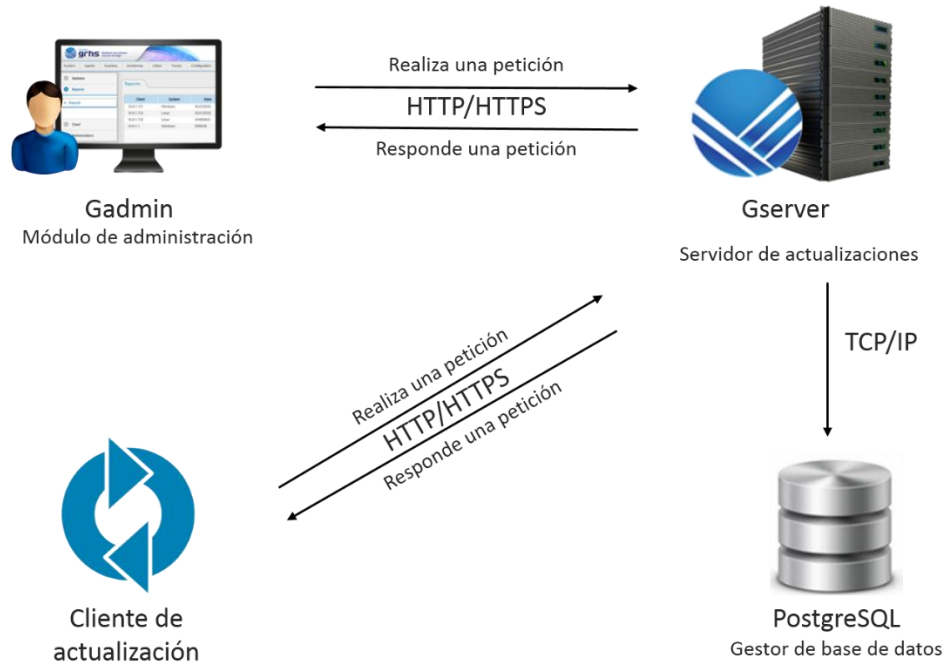


Figura 2 Representación de la propuesta de solución

2.2 Funcionalidades

Las funcionalidades son capacidades o condiciones que el sistema debe cumplir, describen la funcionalidad o los servicios que se espera que éste provea. Estos dependen del tipo de software, del sistema que se desarrolle y de los posibles usuarios del software. Para el desarrollo de este sistema se identificaron las siguientes funcionalidades:

- 1: Verificar actualizaciones disponibles.
- 2: Descargar actualizaciones.
- 3: Comprobar integridad.
- 4: Descomprimir actualizaciones.
- 5: Realizar copia de seguridad.
- 6: Ejecutar actualización.
- 7: Recuperar el sistema ante fallos de actualización.
- 8: Notificar estado de actualización a Gserver.
- 9: Configurar actualizaciones en Gadmin.
- 10: Mostrar reporte de actualizaciones en Gadmin.
- 11: Filtrar reportes por estado de actualización.

2.3 Características no funcionales

Las características no funcionales del producto se determinan para tener en cuenta restricciones, requerimientos técnicos o características que debe tener el sistema. A continuación se muestran estas características reflejadas en la solución propuesta.

Usabilidad

Se necesitará una preparación previa para operar con el sistema. Se requiere un nivel básico de conocimientos de computación, el manejo de la aplicación es sencillo.

Portabilidad

El software debe operar en los Sistemas Operativos Windows XP, Windows 7, Windows 8 y los basados en Linux: Ubuntu, (12.04 – 14.04), Nova 2013 y Debian (6 - 7).

Seguridad

- Confidencialidad: Accederán al Módulo de Administración los usuarios autenticados en el sistema con los permisos correspondientes.
- Integridad: La información de las actualizaciones sólo será modificada por los usuarios autorizados.
- Disponibilidad: El acceso al Módulo de Administración se realizará en el momento requerido por los usuarios autorizados.

Software

Requerimientos de software (Servidor)

Servidores	Especificaciones
Servidor web	Servidor web: Apache 2.0 / Nginx 1.x
Servidor de base de datos	Gestor de base de datos: PostgreSQL

Requerimientos de software (Cliente)

Cliente	Especificaciones
PC cliente	Sistema Operativo: Windows XP, Windows 7, Windows 8 y los basados en Linux: Ubuntu, (12.04 – 14.04), Nova 2013 y Debian (6 - 7)

	Navegador web: Mozilla Firefox v31.4 o superior
--	---

Hardware

Requerimientos de hardware (Servidor)

Servidores	Especificaciones
Servidor web	Procesador: 3.00 GHZ RAM: 1GB Disco duro: 10 GB o superior UPS: 1 Tarjeta de red: 1
Servidor de base de datos	Procesador: 3.00 GHZ RAM: 1GB Disco duro: 4 GB para 2000 clientes UPS: 1 Tarjeta de Red: 1

Requerimientos de hardware (Cliente)

Cliente	Especificaciones
PC cliente	Procesador: 1.00 GHZ o superior RAM: 128 MB o superior Tarjeta de red: 1

Interfaz de usuario

Para la interfaz gráfica se utilizará el estándar de diseño gráfico establecido por la UCI para la marca Xilema.

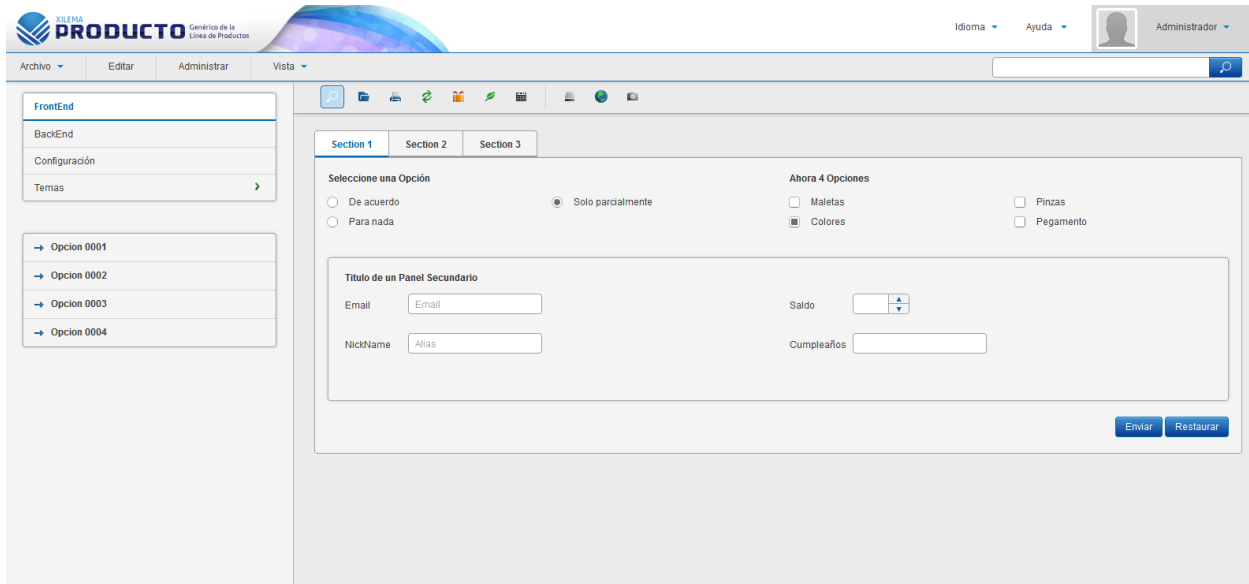


Figura 3 Prototipo de la interfaz de usuario utilizada

2.4 Diagrama de despliegue

Un diagrama de despliegue puede ser usado para modelar la arquitectura física y (en el caso de las redes o sistemas distribuidos) la topología del sistema de software desarrollando. Se describen los dispositivos de hardware (conocidos como nodos), los componentes de software que se ejecutan en ellos (conocidos como artefactos), y los enlaces de comunicación entre los distintos nodos y artefactos (25). Aunque no es requisito de la Metodología XP la creación del diagrama de despliegue, el mismo fue generado para facilitar la comprensión de la distribución de los componentes que integran la solución propuesta. A continuación se muestra el diagrama de despliegue realizado.



Figura 4 Diagrama de despliegue de la propuesta de solución

2.5 Fase de exploración

Primera fase definida por la Metodología XP, se define el alcance real del sistema permitiendo una familiarización con las herramientas, tecnologías y procesos. Se crean las historias de usuario que definen las necesidades del cliente y los programadores estiman el tiempo de desarrollo.

Involucrados en el sistema

Se definen como involucrados en el sistema todos aquellos que realizan una función o interactúan con él de una forma u otra.

2.6 Historias de usuario (HU)

Las historias de usuario son un conjunto de fichas escritas por el cliente que indican las funciones que debe realizar el sistema, constituyendo el mecanismo base de captura de requerimientos de la programación extrema. Cada HU incluye una breve descripción, en torno a tres sentencias de texto escritas por el cliente en su terminología, es importante procurar no incluir sintaxis técnica, de modo que se centren en las necesidades y no en la especificación del aspecto de las interfaces de usuario ni en la implementación, como base de datos o algoritmos específicos (15).

Las HU se representan a modo de tabla (por lo general mediante tarjetas impresas) y los campos de las mismas se describen a continuación:

Número: Es un número que identifica (ID) la HU, en un grupo de HU es único e incremental para cada una de las que le suceden.

Nombre historia de usuario: Es un nombre que por lo general describe en pocas palabras la funcionalidad. Con leer este nombre el cliente y el desarrollador deben hacerse una idea de lo que representa en la aplicación.

Modificación de historia de usuario número: Es un número que indica la cantidad de veces que ha sido modificada la HU, de no haber sido modificada se coloca la palabra *ninguna*.

Usuario: Establece el usuario que interactuará en la ejecución de la funcionalidad. Puede ser una persona o una máquina (sistema).

Programador responsable: Representa el nombre del desarrollador encargado de programar la funcionalidad en cuestión.

Prioridad del negocio: Se clasifica en alta, media o baja. Representa la importancia que el cliente le concede a esta tarea del software.

Riesgo en desarrollo: Especifica el nivel de dificultad que se puede encontrar en la implementación de la funcionalidad.

- **Muy alto:** Cuando en la implementación de las HU pueden surgir errores que lleven a la inoperatividad del sistema.
- **Alto:** Cuando en la implementación de las HU pueden surgir errores que son significativos, pero una parte del sistema ya es funcional.
- **Medio:** Cuando en la implementación de las HU pueden existir errores que retrasen la entrega del producto.

- **Bajo:** Cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto.

Iteración asignada: Representa el número de la iteración en la que se desarrollará la HU.

Puntos estimados: Es una estimación de las semanas que demorará desarrollar la HU.

Descripción: Detalla en pocas palabras las acciones que realiza el usuario y la respuesta que el sistema da a dichas acciones.

Observaciones: Informaciones de interés, como glosarios, detalles del usuario, entre otros.

Prototipo de interfaz: Es la imagen que muestra cómo debe lucir la funcionalidad cuando esté implementada. La imagen puede representar un formulario u otro componente del software que indique el modo en que el usuario podrá identificarla.

A continuación se presenta una HU correspondiente a la solución propuesta. El resto de las HU se presentan en el [Anexo 1](#) de este documento.

Tabla 1 HU Verificar actualizaciones disponibles

Historia de usuario	
Número: HU-1	Nombre de la historia de usuario: Verificar actualizaciones disponibles
Modificación de la historia de usuario número: Ninguna	Usuario: Sistema
Programador responsable: Yosján Pérez Cuello Roberto Rafael Ramírez Martínez	Iteración asignada: Primera
	Puntos estimados: 1
Prioridad en el negocio: Muy alta	Riesgo en desarrollo: Alto
Descripción: Se verifica si en el servidor existen actualizaciones disponibles para el cliente.	
Observaciones: El Servidor de Actualizaciones debe estar operando correctamente.	
Prototipo de interfaz: No procede	

2.7 Estimación de esfuerzo

Cada HU lleva consigo una cuota de esfuerzo por parte de los desarrolladores. Los puntos estimados representan las semanas laborables, una semana está compuesta por los días y horarios tradicionales en que se desempeña la organización que emplea a los desarrolladores. Después de calcular el esfuerzo que requiere cada HU se calcula el total de esfuerzo requerido,

que indica el tiempo en semanas que puede durar el desarrollo de la aplicación. Los puntos estimados se calculan a partir del estudio de HU similares de las que ya se tiene una estimación y de previo acuerdo con los desarrolladores.

Tabla 2 Estimación de esfuerzo

Historia de usuario	Puntos estimados
Verificar actualizaciones disponibles	1
Descargar actualizaciones	1
Comprobar integridad	1
Descomprimir actualizaciones	1
Realizar copia de seguridad	1
Ejecutar actualización	1
Recuperar el sistema ante fallos de actualización	1
Notificar estado de actualización a Gserver	1
Configurar actualizaciones en Gadmin	1
Mostrar reporte de actualizaciones en Gadmin	1
Filtrar reportes por estado de actualización	1
Total	11

2.8 Plan de iteraciones

El plan de iteraciones comprende el organigrama trazado por el equipo de desarrollo para dar cumplimiento a la implementación de las HU. Se tiene en cuenta para determinar cada iteración la importancia que tiene cada HU para el negocio, así como su complejidad, cantidad de miembros del equipo de desarrollo y otros factores. El objetivo es contar con la lógica organizativa suficiente para alcanzar el éxito en la etapa de desarrollo.

Atendiendo a la complejidad de la HU, importancia para el negocio, el equipo de desarrollo y a las dependencias que algunas HU tienen de otras, el plan de iteraciones se concibe con un total de cuatro iteraciones. Las mismas se describen a continuación:

Iteración 1: Se implementarán aquellas HU cuya complejidad es **muy alta** para el negocio.

Iteración 2: Se implementarán aquellas HU con una complejidad **alta** para el negocio.

Iteración 3: Se implementarán aquellas HU con una complejidad **media** para el negocio.

Iteración 4: Se implementarán aquellas HU que tienen una complejidad **baja** para el negocio.

Tabla 3 Plan de iteraciones

Iteración	Orden de las HU en la iteración	Duración de la iteración (semanas)
Iteración 1	Verificar actualizaciones disponibles	3
	Descargar actualizaciones	
	Comprobar integridad	
Iteración 2	Descomprimir actualizaciones	3
	Realizar copia de seguridad	
	Ejecutar actualización	
Iteración 3	Recuperar el sistema ante fallos de actualización	3
	Notificar estado de actualización a Gserver	
	Configurar actualizaciones en Gadmin	
Iteración 4	Mostrar reporte de actualizaciones en Gadmin	2
	Filtrar reportes por estado de actualización	

2.9 Plan de entregas

El plan de entregas es el acuerdo firmado entre el cliente y el equipo de desarrollo para la entrega de los resultados obtenidos durante la programación de la solución. Se realiza teniendo en cuenta la duración de las iteraciones y es de mutuo acuerdo entre las partes. Para la realización de la solución propuesta las partes se guían por el siguiente plan de iteraciones. Aunque se decide

realizar entregas al cliente al final de la tercera iteración y una vez concluida la cuarta se hace entrega del producto terminado.

Tabla 4 Plan de iteraciones

Artefacto	Final de la 1ra iteración	Final de la 2da iteración	Final de la 3ra iteración	Final de la 4ta iteración
Gupdater, sistema de actualización v2.0 para GRHS	06 de abril 2015	27 de abril 2015	18 de mayo 2015	01 de junio 2015

Conclusiones parciales

Después de investigados los elementos referentes al análisis de la aplicación a desarrollar se concluye que la versión de actualización Gupdater 1.0 ya existente para Gclient no cumplía con las necesidades del cliente. Este escenario propició una rápida identificación de los procesos a automatizar, lográndose identificar de esta manera 11 historias de usuario. Una vez estimado el tiempo de duración de cada historia de usuario, se obtuvieron mediante los cálculos de fórmulas propuestas en la metodología XP, la cantidad de iteraciones necesarias para llevar a cabo la codificación de las mismas, así como el promedio de duración de las iteraciones, teniendo en cuenta la prioridad de las funcionalidades expuestas por el cliente. De esta manera se llegaron a obtener 4 iteraciones, las cuales fueron organizadas en 2 momentos de liberación del producto, cada uno con funcionalidades completas y listas para trabajar en entornos reales. De esta manera el escenario queda listo para comenzar a realizar el diseño de la aplicación.

CAPÍTULO 3: DISEÑO

Introducción

En el presente capítulo se generan los artefactos propuestos por la Metodología XP para la fase de diseño. Se mencionan los patrones de diseño y las arquitecturas de software utilizadas en cada sub-sistema, el modelo arquitectónico por el que se rige la solución y el modelo de datos empleado para crear la infraestructura de base de datos necesaria.

3.1 Arquitecturas y patrón arquitectónico

Arquitectura Cliente/Servidor

La arquitectura Cliente/Servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requerimientos a uno o más servidores centrales. En este tipo de arquitectura el cliente puede ser desde uno hasta muchos consumidores del servicio que brinda un servidor distribuido en una misma computadora física o no (26).

La arquitectura Cliente/Servidor ha obtenido notable importancia en el ámbito de la programación web, producto a los beneficios que la misma brinda para que muchos usuarios accedan a un mismo recurso, sin importar la ubicación geográfica. No obstante el planteamiento anterior, diversas son las aplicaciones de escritorio y de script que utilizan esta arquitectura. Y en los entornos empresariales alcanza notable importancia para la gestión de recursos compartidos.

En este tipo de arquitectura el servidor se caracteriza por estar operativo todo el tiempo, a la escucha de las peticiones que realizan los diferentes clientes y procesando dichas peticiones para enviar la respuesta correspondiente.

Es el cliente, por lo general, el que inicia el proceso de comunicación entre los dos sistemas, esto en dependencia de sus necesidades de actualización, información u otras. La Figura 5 muestra un esquema que simula este tipo de arquitectura.

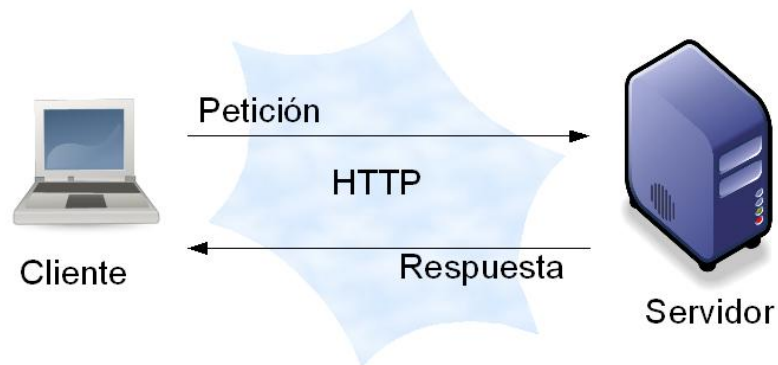


Figura 5 Modelo de Arquitectura Cliente/Servidor

La solución GRHS se basa en esta arquitectura para su funcionamiento, Gserver es el encargado de manejar toda la comunicación entre los demás sub-sistemas que integran GRHS. Así, aunque indiscutiblemente el cliente y el actualizador del mismo tienen que tener relación, las notificaciones realizadas por este último van dirigidas a Gserver para posteriormente ser mostradas en el Módulo de Administración. Gadmin es un componente propio de Gserver y se puede considerar la interfaz gráfica del mismo.

Arquitectura basada en componentes

La arquitectura basada en componentes se emplea en el sub-sistema de actualizaciones en el cliente. Esta arquitectura se rige por el estándar de un bajo acoplamiento en la implementación, lo que permite realizar modificaciones en las clases sin que esto implique modificar el resto (27). Para la implementación del sub-sistema de actualización del cliente, en Gupdater 2.0 se utiliza esta arquitectura. El Cliente de Actualización se compone de varios plugins (considerados componentes) que interactúan entre sí para facilitar el funcionamiento de la aplicación. Estos plugins pueden ser modificados sin tener que modificar los restantes. Los plugins en el cliente-base se ubican en tres niveles principales:

- Plugins del core, que ocupan la mayor relevancia en la aplicación y se encargan de las tareas principales que son requeridas por varios plugins del segundo nivel.
- Plugins de aplicaciones, que son los plugins encargados de realizar tareas específicas dentro del funcionamiento, dependen de los plugins del core y son los que contienen a los plugins del tercer nivel.
- Sub-plugins, que son los plugins que utilizan los plugins de aplicaciones para delegar responsabilidades.

La Figura 6 muestra los tres niveles antes mencionados y en cada uno los plugins que son empleados en el Cliente de Actualización. Entre los plugins del core se encuentran *logger_manager* para la gestión de los registros de logs y *request_manager* para la comunicación con el servidor. Como plugin de aplicación se encuentra *update_manager* que es el controlador principal de la funcionalidad. Entre los subplugins están *partialupdate* para la realización de actualizaciones parciales, *fullupdate* para realizar un reemplazo total del cliente y *recover* para realizar la recuperación del cliente ante fallos ocurridos en el proceso de actualización.

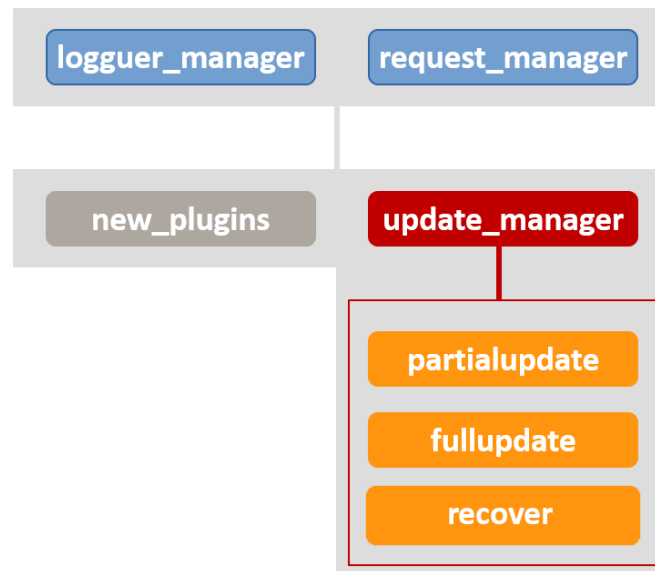


Figura 6 Arquitectura basada en componentes en el Cliente de Actualización

Model Template View (MTV)

Django sigue el patrón MVC tan al pie de la letra que puede ser llamado un framework MVC. Someramente, la M, V y C se separan en Django de la siguiente manera:

M, la porción de acceso a la base de datos, es manejada por la capa de la base de datos de Django.

V, la porción que selecciona qué datos mostrar y cómo mostrarlos, es manejada por la vista y las plantillas.

C, la porción que delega a la vista dependiendo de la entrada del usuario, es manejada por el framework mismo siguiendo el URLconf y llamando a la función apropiada de Python para la URL obtenida.

Debido a que la "C" es manejada por el mismo framework y la parte más importante se produce en los modelos, las plantillas y las vistas, Django es conocido como un Framework MTV. En el patrón de diseño MTV:

M significa Modelo (en inglés Model), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.

T significa Plantilla (en inglés Template), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: cómo algunas cosas son mostradas sobre una página web u otro tipo de documento.

V significa Vista (en inglés View), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada (28).

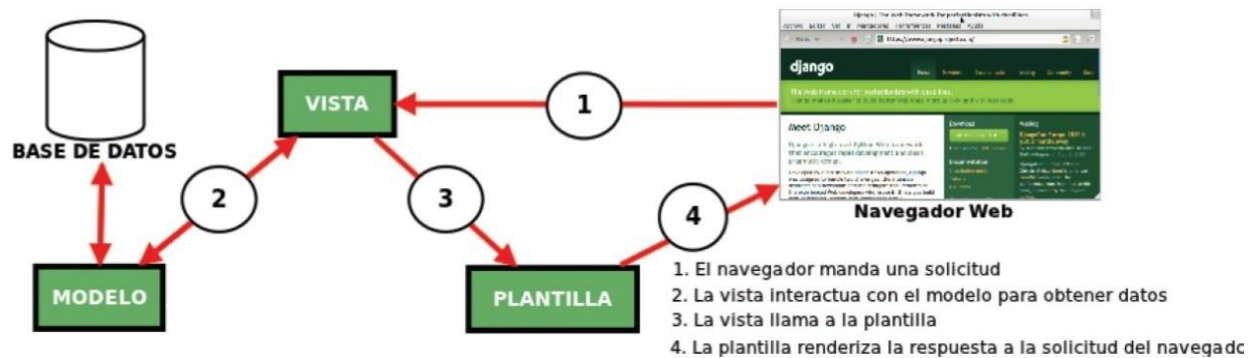


Figura 7 Patrón Model Template View (MTV) (29)

3.2 Patrones de diseño

Los patrones de diseño son un conjunto de prácticas de óptimo diseño que se utilizan para abordar problemas recurrentes en la programación orientada a objetos. Los patrones de diseño que se especifican a continuación son los aplicados en la solución propuesta. Aunque no son de utilización obligatoria, seguir estos patrones garantiza mayor robustez en una aplicación informática y que el mantenimiento o ampliación de la misma pueda realizarse más fácilmente (30).

Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. GRASP es un acrónimo que significa General Responsibility Assignment Software Patterns. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos (31).

Experto

El patrón experto define que la responsabilidad de crear un objeto debe asignarse a la clase que mayor información tenga para la creación del mismo. Esto permite mayor cohesión y el encapsulamiento de la información.

Creador

Define quién debe crear o instanciar nuevos objetos, teniendo en cuenta que se cuente con toda la información necesaria para hacerlo.

Alta cohesión

Plantea que la información que almacena una clase debe estar lo más relacionada posible con la clase, evitando la implementación de funcionalidades que pueden ser heredadas de otras clases especializadas.

Bajo acoplamiento

Consiste en reducir las dependencias entre clases, o reducir la mezcla de las mismas con el propósito de evitar que al modificar una clase sea necesario la modificación del resto de las clases.

Estándares de nomenclatura y codificación

Los estándares de nomenclatura y codificación surgen por la necesidad de homogenizar el código fuente facilitando la lectura y entendimiento del mismo. Los programadores deben acogerse a un conjunto de pautas antes determinadas para lograr hablar todos en el mismo idioma y que quede reflejado en el código fuente. Algunos de los elementos tenidos en cuenta por estos estándares son: indentación, tabuladores o espacios, tamaño máximo de líneas, líneas en blanco, codificación de caracteres, espacios en blanco en expresiones y sentencias (32).

Para la implementación del Módulo de Administración se siguen los estándares de interfaz definidos por Xilema Base Web que se acoge a los estándares de la Universidad. Así como se sigue el estándar definido por Guido van Rossum Creador de Python para la escritura de código. El estándar es PEP (Python Enhancement Proposals) 0008.

3.3 Modelo de datos

El modelo de datos tiene gran importancia en el ciclo de desarrollo de software, y de manera particular para la fase de implementación, pues define formalmente las estructuras permitidas y las restricciones que se aplican con el fin de representar los datos del dominio de la aplicación. Está compuesto por objetos: entidades que existen y se manipulan; y atributos: características básicas de dichos objetos y relaciones: forma en que se enlazan los objetos entre sí (33).

A continuación se muestra el modelo de datos diseñado para el Servidor de Actualizaciones.

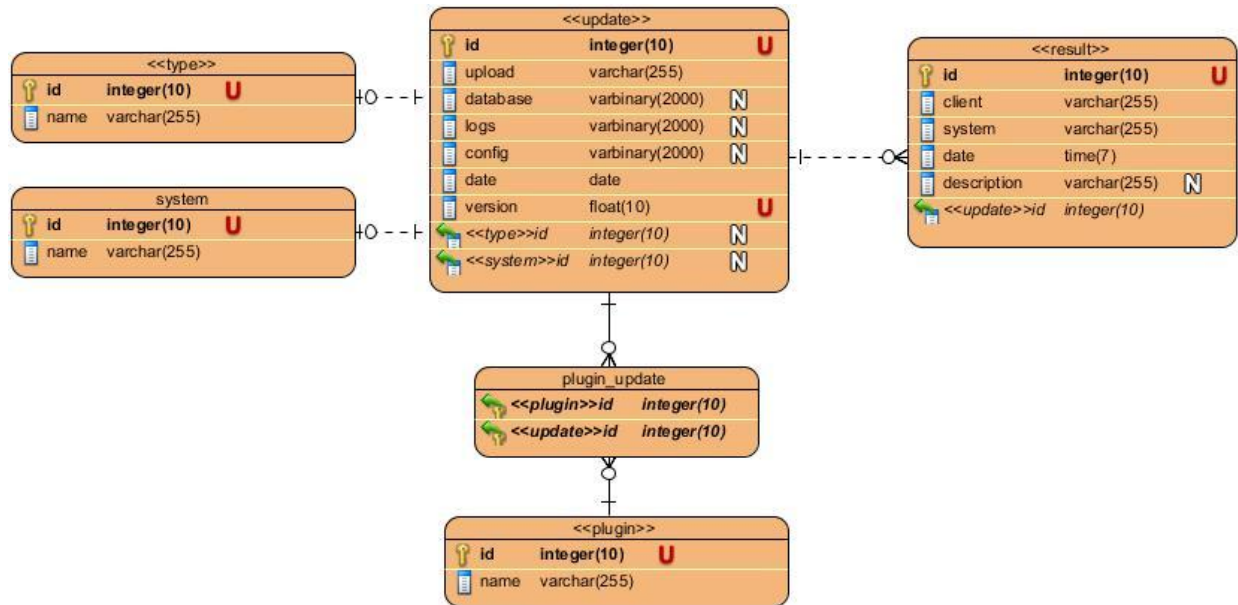


Figura 8 Modelo de datos

La descripción de los componentes de este modelo se muestra en la siguiente tabla, donde se especifica la función principal de cada modelo.

Tabla 5 Descripción del modelo de datos

Modelo	Descripción
<<update>>	Almacena la información correspondiente a las actualizaciones. Establece una relación con las entidades <i>type</i> , <i>plugin</i> y <i>system</i> .
<<file>>	Nomenclador, representa los plugins que serán modificados en una actualización de tipo parcial.
<<type>>	Nomenclador, representa el tipo de actualización (total, parcial) que se va a realizar.
<<result>>	Almacena la información de los reportes de actualización realizados por el cliente.

3.4 Tarjetas Clase Responsabilidad Colaborador (CRC)

Las tarjetas CRC describen las clases que actúan en la aplicación. Posibilitan la comprensión del proceso de una manera orientada a objetos permitiendo así reducir el modo procedural de pensar de los desarrolladores. Cada tarjeta almacena el nombre de la clase, la responsabilidad que esta

tiene dentro del sistema y los colaboradores, que son las clases con las que se relaciona durante su ejecución.

A continuación se muestra el resultado de crear las tarjetas CRC de las clases presentes en la aplicación desarrollada.

Tabla 6 Tarjeta CRC Updater

Updater	
Responsabilidad	Colaborador
Controladora principal del negocio. Se encarga de realizar el proceso de actualización del cliente. Realiza los procesos fundamentales y delega responsabilidades a los sub plugins para realizar distintas tareas. Comprueba la existencia de nuevas actualizaciones en el servidor; realiza la copia de seguridad del cliente; descarga la actualización desde el servidor; comprueba el md5 de la descarga; descomprime la actualización; verifica el tipo de actualización para encargarse de ejecutar el sub plugin correspondiente; realiza el reporte de estado al servidor; ejecuta el sub plugin de recuperación del cliente.	Request LoggingManager

Tabla 7 Tarjeta CRC Request

Request	
Responsabilidad	Colaborador
Clase encargada de la comunicación con el servidor. Se utiliza para hacer consultas utilizando el protocolo HTTP/HTTPS al Servidor de Actualizaciones para la obtención de información sobre actualizaciones disponibles y en la realización de reportes. Su método principal es request que se encarga	

<p>de la lógica de la comunicación realizando las peticiones. Esta clase se corresponde con un plugin del core de la arquitectura cliente base y no fue implementada por los autores.</p>	
---	--

Tabla 8 Tarjeta CRC LoggingManager

LoggingManager	
Responsabilidad	Colaborador
<p>Se encarga del registro en ficheros de logs. Es utilizada frecuentemente en el sistema para almacenar información sobre el flujo de ejecución y los errores que se han producido durante el mismo. Esta clase se corresponde con un plugin del core de la arquitectura cliente base y no fue implementada por los autores.</p>	

Conclusiones parciales

Después de analizados los elementos referentes al diseño de la solución a desarrollar se concluye que la construcción de un sistema informático debe estar soportado por un buen diseño arquitectónico, dado que, realizar actualizaciones y perfeccionamientos al mismo representaría una pérdida de tiempo, pues la modificación a realizar abarcaría desde la implementación más compleja hasta las funcionalidades más simples. Por ello la presente investigación emplea en el diseño de la solución patrones arquitectónicos bien establecidos, los cuales fueron seleccionados en dependencia de su función y de las posibilidades de desarrollo que ofrecen.

Para describir las responsabilidades de cada clase presente en la propuesta de solución se emplearon las tarjetas CRC, su sencillez garantiza una rápida actualización en caso de cambios en la concepción del diseño. Se presentó el diagrama de modelo de la base de datos utilizada para tener una idea de cómo fue elaborada la persistencia de la información manejada por la aplicación. Concluyen las actividades relacionadas con el diseño se da paso a la implementación de los diferentes artefactos generados.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS

Introducción

En este capítulo se confeccionan las tareas de ingeniería como artefacto que guía el proceso de desarrollo de software. También se presentan los resultados de las pruebas unitarias y de aceptación que se realizaron a la aplicación y el resultado obtenido de las mismas.

4.1 Tareas de ingeniería

Las tareas de ingeniería son aquellas actividades que deben realizarse para que se cumpla una HU. La siguiente tabla muestra la relación de cada HU con las tareas de ingeniería que se desarrollaron para implementar dicha HU.

Tabla 9 Relación HU - Tareas de ingeniería

Historia de usuario	Tareas de ingeniería
Verificar actualizaciones disponibles	Implementar verificar nuevas actualizaciones
Descargar actualizaciones	Implementar descargar actualizaciones
Comprobar integridad	Implementar comprobar integridad
Descomprimir actualizaciones	Implementar descomprimir actualizaciones
Realizar copia de seguridad	Implementar realizar copia de seguridad
Ejecutar actualización	Implementar ejecutar actualización
Recuperar el sistema ante fallos de actualización	Implementar recuperar el sistema ante fallos de actualización
Notificar estado de actualización a Gserver	Implementar notificar estado de actualización a Gserver
Configurar actualizaciones en Gadmin	Implementar configurar actualizaciones en Gadmin
Mostrar reporte de actualizaciones en Gadmin	Implementar Mostrar reporte de actualizaciones en Gadmin

Filtrar reportes por estado de actualización	Implementar filtrar reportes por estado de actualización
---	--

A continuación se muestra la tabla correspondiente a una tarea de ingeniería. El resto de las tareas de ingeniería se encuentran en el [Anexo 2](#) de este documento.

Tabla 10 Tarea de ingeniería 1 Implementar verificar nuevas actualizaciones

Tarea de ingeniería	
Número de la tarea: 1	Número de la historia de usuario: 1
Nombre de la tarea: Implementar verificar nuevas actualizaciones	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 16/03/2015	Fecha fin: 20/03/2015
Programador responsable: Yosján Pérez Cuello	
Descripción: El Cliente de Actualización realiza una petición HTTP al Servidor de Actualizaciones enviando la versión de la actualización más reciente instalada en el cliente y el Sistema Operativo donde se está ejecutando. El Cliente de Actualización recibe la respuesta del servidor en formato JSON y la procesa para determinar la existencia o no de una nueva versión disponible. De existir una nueva versión disponible se crea un arreglo que contiene la información de dicha actualización, este arreglo será utilizado por el resto de los métodos de la aplicación para validar información.	

4.2 Proceso de pruebas

Niveles de prueba de XP

- Las pruebas **unitarias** se realizan para comprobar la correcta implementación de las funcionalidades. La Metodología XP propone realizar pruebas frecuentemente para corregir a tiempo los errores y reducir la posibilidad de que ocurran otros. Las pruebas unitarias son diseñadas por los propios desarrolladores (34).
- Las pruebas de **aceptación** son diseñadas por el cliente, estas se enfocan en el cumplimiento de lo establecido en el acuerdo. El cliente comprueba que las funcionalidades solicitadas se han realizado satisfactoriamente. Para realizar una prueba de aceptación se confecciona una tabla con las casillas Clases válidas, Clases inválidas, Resultado esperado, Resultado de la prueba y Observaciones.
- Las Prueba de **integración** son diseñadas para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados para ejecutar

un caso de uso. Se prueba un paquete o un conjunto de paquetes del modelo de implementación. Estas pruebas descubren errores o incompletitud en las especificaciones de las interfaces de los paquetes. Esta prueba debe ser responsabilidad de desarrolladores.

Tipos de pruebas

- Las pruebas de tipo **función** centran su atención en la validación de las funciones, métodos y servicios.
- Las pruebas de tipo **integridad** están enfocadas en la valoración de la robustez (resistencia a fallos).
- Las pruebas de tipo **estructura** están enfocadas en la valoración de la adherencia a su diseño y formación. Este tipo de prueba se realiza a las aplicaciones web asegurando que todos los enlaces estén conectados, el contenido deseado es mostrado y no hay contenido huérfano.
- Las pruebas de tipo **configuración** están enfocadas a asegurar que funciona en diferentes configuraciones de hardware y software. Esta prueba es implementada también como prueba de rendimiento del sistema.

Métodos de pruebas

- La prueba de **caja negra** se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.
- En la prueba de la **caja blanca** del software se comprueban los caminos lógicos del mismo proponiendo casos de prueba donde se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado.

Estrategia de pruebas

Se realizarán pruebas en dependencia de la funcionalidad en cuestión siguiendo los tres niveles que define la Metodología XP. Por lo que se realizaran pruebas de unidad de forma automática, aceptación e integración de forma manual. Sobre las que se aplicarán pruebas de tipos función, integridad, estructura y configuración según corresponda bajo los métodos de caja blanca y caja negra. Se analizarán los resultados obtenidos.

Pruebas unitarias

A continuación se muestra una de las pruebas unitarias realizadas. Las demás pruebas unitarias se encuentran en el [Anexo 3](#) de este documento. Las pruebas unitarias mostradas representan el resultado final de la evaluación.

Tabla 11 Prueba unitaria 1, Verificar actualizaciones disponibles

Prueba de: Unidad	
Nombre de la prueba: Verificar actualizaciones disponibles	
Estado: Satisfactoria	Última ejecución: 21/03/2015
Ejecutado por: Yosján Pérez Cuello	Verificado por: Yosján Pérez Cuello
Descripción: La aplicación realiza una petición HTTP/HTTPS al Servidor de Actualizaciones enviando la última versión de actualización instalada y el Sistema Operativo donde se ejecuta. Procesa la información de retorno para comprobar si existen nuevas actualizaciones disponibles. Si hay nuevas actualizaciones genera un arreglo de información con los detalles de la nueva actualización.	
Entrada: Se envía en la consulta la versión de la última actualización realizada de manera exitosa.	
Criterio de aceptación: La funcionalidad se ejecuta satisfactoriamente. Grado de aceptación 100%.	
Resultado: La funcionalidad se encuentra lista para ser integrada al sistema.	

Resultados de las pruebas unitarias

Durante el proceso de pruebas de nivel unitario se detectaron seis no conformidades que fueron satisfactoriamente corregidas. Estas no conformidades fueron:

- No se realiza comprobación del estado (activo o inactivo) del servicio *Gclient* en el sistema operativo Windows.
- El servicio *Gclient* no es detenido en el sistema operativo Windows.
- No se detiene la ejecución de la actualización si el servicio *Gclient* no puede ser detenido.
- La comprobación de integridad de descarga mediante el método md5 arroja resultados diferentes en los sistemas operativos Windows y Linux.
- El sistema no soporta conexión segura mediante el protocolo de comunicación HTTPs.
- El sistema elimina la copia de seguridad sin comprobar que el servicio *Gclient* esté ejecutándose satisfactoriamente.

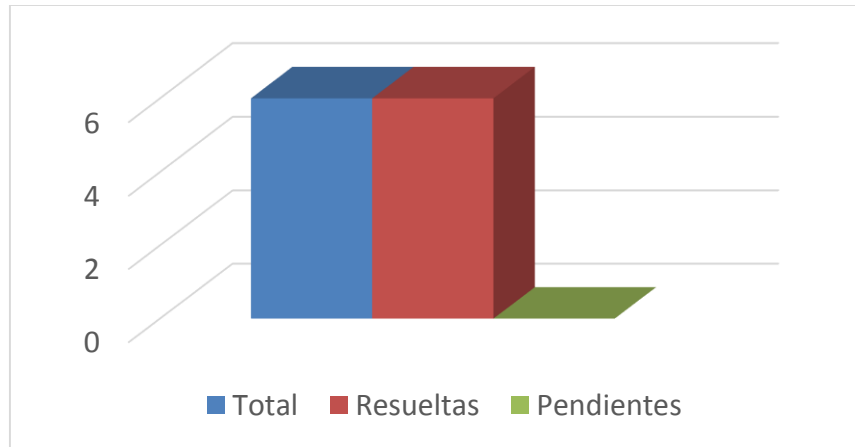


Gráfico 1 Resultados de las pruebas unitarias

La ejecución automática de las pruebas unitarias devino en el resultado mostrado en la Figura 9. Las pruebas unitarias automatizadas fueron ejecutadas utilizando el framework PyUnit. El mismo se encuentra integrado a Django en la solución GRHS.

```

...
-----
Ran 3 tests in 0.004s

OK
Destroying test database for alias 'default'...
    
```

Figura 9 Resultado de la ejecución de las pruebas unitarias automáticas

Pruebas de aceptación

Para realizar las pruebas de aceptación se crearon los casos de pruebas siguientes. A continuación se muestra una prueba de aceptación realizada. El resto de las pruebas de aceptación se encuentran en el [Anexo 4](#).

Tabla 12 Prueba de aceptación 1, HU Verificar actualizaciones disponibles

Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba	Observaciones
Se ejecuta la funcionalidad mientras el servidor GRHS se encuentra		Recibe la información de la última actualización disponible en el servidor. Si no	Satisfactorio.	

operando correctamente.		hay versiones posteriores informa del suceso y termina el proceso. Si hay versiones posteriores crea un arreglo de información que contenga los detalles de la actualización y ejecuta otras funcionalidades.		
	Se ejecuta la funcionalidad mientras el servidor GRHS no se encuentra operando.	Se reintenta realizar la solicitud de información al servidor según se especifica en el fichero de configuración. Al término de todos los intentos se detiene la ejecución del proceso.	Satisfactorio.	

Resultados de las pruebas de aceptación

Durante el proceso de pruebas de nivel de aceptación se detectaron cinco no conformidades que fueron satisfactoriamente corregidas. Estas no conformidades fueron:

- No se puede definir el Sistema Operativo para el que estará disponible la actualización.
- No se restringe el formato del fichero de actualización (comprimido) que puede ser insertado.

- La versión de actualización registrada sólo admite números enteros.
- La comprobación de la última versión registrada en el servidor se realiza por el ID de la actualización en lugar de la fecha de publicación.
- Los reportes realizados por el Cliente de Actualización no pueden ser filtrados correctamente.

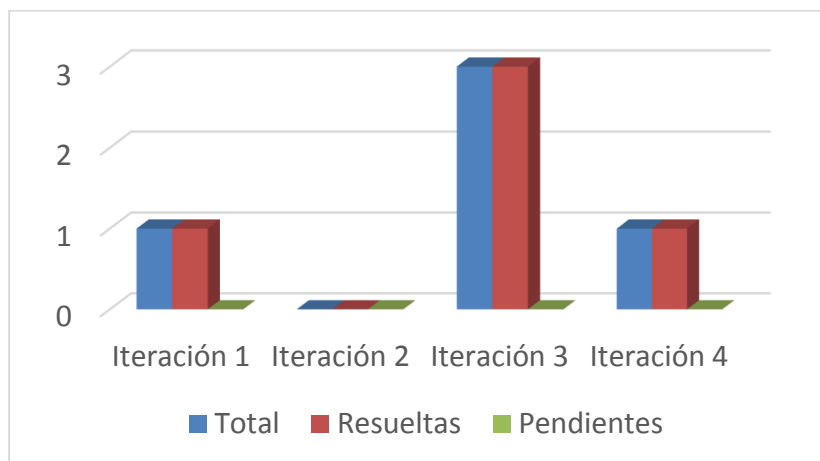


Gráfico 2 Resultados de las pruebas de aceptación

Resultados de las pruebas de integración

Durante el proceso de pruebas del nivel de integración se detectaron dos no conformidades que fueron satisfactoriamente corregidas. Estas no conformidades fueron:

- Incompatibilidad para establecer conexiones por protocolo HTTPS utilizando el plugin *request_manager*.
- Incompatibilidad en el almacenamiento del fichero de actualización utilizando el componente *File* proporcionado por GRHS.

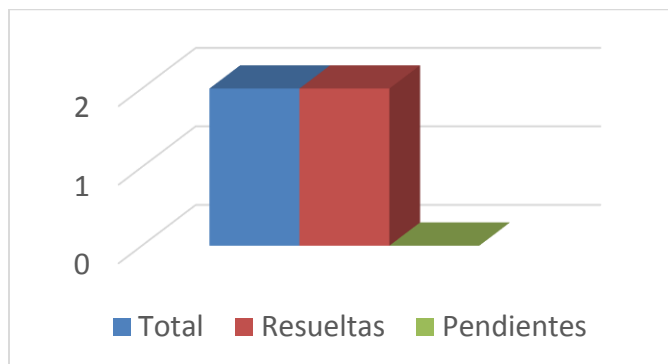
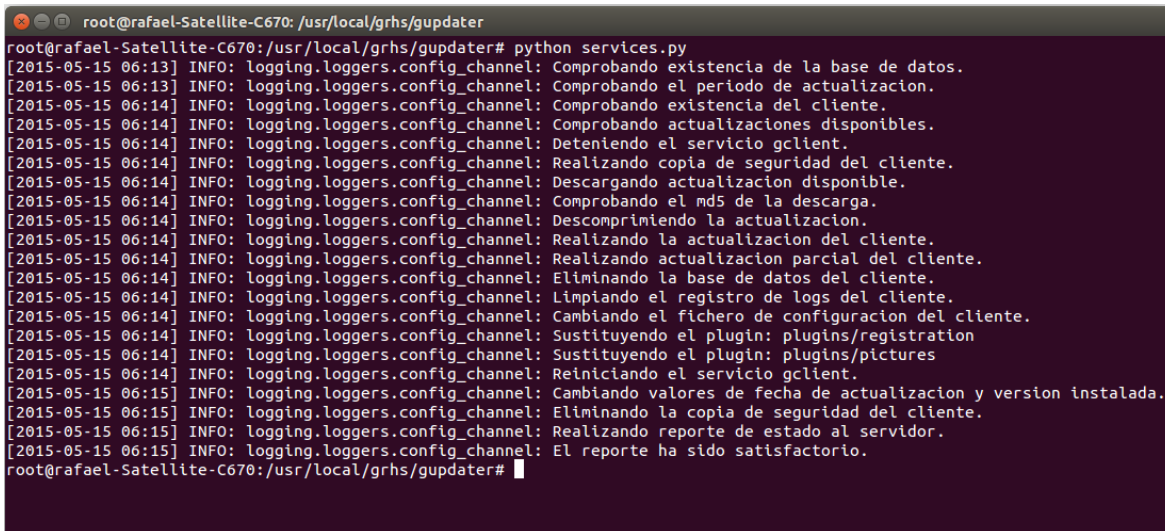


Gráfico 3 Resultados de las pruebas de integridad

Las pruebas mostradas representan la evaluación del estado final de las funcionalidades. Después de haber realizado todas estas pruebas y corregido los errores, el actualizador opera correctamente. Prueba de ello lo constituye la Figura 10 que muestra el resultado de ejecutar el Cliente de Actualización de Gupdater 2.0, cuando una actualización de tipo parcial está disponible en el Servidor de Actualizaciones.



```
root@rafael-Satellite-C670: /usr/local/grhs/gupdater
root@rafael-Satellite-C670: /usr/local/grhs/gupdater# python services.py
[2015-05-15 06:13] INFO: logging.loggers.config_channel: Comprobando existencia de la base de datos.
[2015-05-15 06:13] INFO: logging.loggers.config_channel: Comprobando el periodo de actualizacion.
[2015-05-15 06:14] INFO: logging.loggers.config_channel: Comprobando existencia del cliente.
[2015-05-15 06:14] INFO: logging.loggers.config_channel: Comprobando actualizaciones disponibles.
[2015-05-15 06:14] INFO: logging.loggers.config_channel: Deteniendo el servicio gclient.
[2015-05-15 06:14] INFO: logging.loggers.config_channel: Realizando copia de seguridad del cliente.
[2015-05-15 06:14] INFO: logging.loggers.config_channel: Descargando actualizacion disponible.
[2015-05-15 06:14] INFO: logging.loggers.config_channel: Comprobando el md5 de la descarga.
[2015-05-15 06:14] INFO: logging.loggers.config_channel: Descomprimiendo la actualizacion.
[2015-05-15 06:14] INFO: logging.loggers.config_channel: Realizando la actualizacion del cliente.
[2015-05-15 06:14] INFO: logging.loggers.config_channel: Realizando actualizacion parcial del cliente.
[2015-05-15 06:14] INFO: logging.loggers.config_channel: Eliminando la base de datos del cliente.
[2015-05-15 06:14] INFO: logging.loggers.config_channel: Limpiando el registro de logs del cliente.
[2015-05-15 06:14] INFO: logging.loggers.config_channel: Cambiando el fichero de configuracion del cliente.
[2015-05-15 06:14] INFO: logging.loggers.config_channel: Sustituyendo el plugin: plugins/registration
[2015-05-15 06:14] INFO: logging.loggers.config_channel: Sustituyendo el plugin: plugins/pictures
[2015-05-15 06:14] INFO: logging.loggers.config_channel: Reiniciando el servicio gclient.
[2015-05-15 06:15] INFO: logging.loggers.config_channel: Cambiando valores de fecha de actualizacion y version instalada.
[2015-05-15 06:15] INFO: logging.loggers.config_channel: Eliminando la copia de seguridad del cliente.
[2015-05-15 06:15] INFO: logging.loggers.config_channel: Realizando reporte de estado al servidor.
[2015-05-15 06:15] INFO: logging.loggers.config_channel: El reporte ha sido satisfactorio.
root@rafael-Satellite-C670: /usr/local/grhs/gupdater#
```

Figura 10 Registro en consola de la ejecución del Cliente de Actualización de Gupdater 2.0

Conclusiones parciales

Con la conclusión de este capítulo se obtiene la terminación de la solución. Fueron implementadas las tareas de ingeniería para cada historia de usuario. Fueron realizadas las pruebas a la aplicación obteniendo la satisfacción por parte del cliente en las pruebas de aceptación y un alto índice de eficiencia en las pruebas unitarias y de integración realizadas.

CONCLUSIONES GENERALES

Se determinaron los elementos del diseño teórico de la investigación que constituyen el punto de partida y guían del proceso para garantizar el éxito de la investigación.

Se realizó un estudio de sistemas actualizadores de software existentes a nivel mundial, donde se analizaron sus características así como ventajas y desventajas para obtener una mejor comprensión y determinar qué características se podían aplicar a la aplicación desarrollada.

El uso de patrones de diseño y arquitectónicos brindó como resultado un diseño sólido y flexible para la codificación de la aplicación.

Se desarrolló la propuesta de solución basada en las necesidades del cliente y el estudio realizado sobre los sistemas actualizadores de software similares y las ventajas de Gupdater 1.0.

Se desarrolló el Módulo de Administración en Gadmin para configurar las actualizaciones y tener acceso a los reportes de estado generados por el Cliente de Actualización.

Se implementó el modo de actualización parcial del cliente, lo que garantiza que sea reemplazado sólo lo necesario posibilitando solucionar desde un error simple hasta significativo que pueda comprometer el correcto funcionamiento de Gclient.

Se implementó la funcionalidad de rollback en el actualizador, lo que posibilita la recuperación del cliente a una versión estable ante errores ocurridos en el proceso de actualización.

Se implementó la funcionalidad de notificación al servidor, que garantiza mayor información sobre el estado de la actualización y conocer las causas de problemas surgidos en el proceso.

De manera general, se desarrolló la versión 2.0 de Gupdater, permitiendo mayor funcionalidad y posibilidades de éxito en el proceso de actualización de Gclient, concluyendo que se ha cumplido satisfactoriamente con los objetivos propuestos en el presente trabajo.

RECOMENDACIONES

Teniendo en cuenta los resultados obtenidos en la implementación de la solución se recomienda:

- Implementar la funcionalidad que permita realizar cambios de configuración en el propio actualizador.
- Implementar la funcionalidad que permita notificar a los administradores, por correo electrónico, de los clientes que no han logrado recuperarse ante fallos de actualización.

REFERENCIAS BIBLIOGRÁFICAS

1. **Docsetools.** Docsetools. [En línea] [Citado el: 23 de marzo de 2015.] http://docsetools.com/articulos-de-todos-los-temas/article_25832.html.
2. **Cestero, José M.** [En línea] 16 de junio de 2009. [Citado el: 12 de noviembre de 2014.] <http://www.pymesyautonomos.com/tecnologia/el-soporte-tecnico-es-la-clave>.
3. **Diario Clarín.** Diario Clarín. [En línea] 14 de octubre de 2010. [Citado el: 18 de noviembre de 2014.] http://www.clarin.com/internet/Microsoft-mayor-parche-seguridad-historia_0_352764974.html.
4. **Centro de Telemática.** Wiki GRHS. [En línea] [Citado el: 28 de mayo de 2015.] <http://10.128.50.236/grhs-doc/index.php/GRHS>.
5. **Universidad Nacional de Colombia.** [En línea] [Citado el: 14 de marzo de 2015.] http://www.virtual.unal.edu.co/cursos/IDEA/2007219/lecciones/cap_4/sub8.html.
6. **La Web del Programador.** Diccionario La Web del Programador. [En línea] [Citado el: 15 de 03 de 2015.] <http://www.lawebdelprogramador.com/diccionario/buscar.php?opc=1&charSearch=software>.
7. —. Diccionario La Web del Programador. [En línea] [Citado el: 19 de noviembre de 2014.] <http://www.lawebdelprogramador.com/diccionario/buscar.php?opc=1&charSearch=Optimizaci%C3%B3n+de+software>.
8. **Instituto de Tecnología de Massachusetts.** s.l. : MIT, 2004.
9. **Jasen, Remy y G, Ballintijn.** *A process framework and typology for software product updaters.* s.l. : IEEE, 2005. 0-7695-2304-8.
10. **Carzaniga, Antonio.** *A Characterization Framework for Software Deployment Technologies.* s.l. : University of Colorado,, 1998.
11. **Microsoft.** Microsoft Windows. *Windows Update.* [En línea] [Citado el: 26 de enero de 2015.] <http://windows.microsoft.com/es-419/windows7/products/features/windows-update>.
12. **Symantec.** Symantec. [En línea] [Citado el: 12 de noviembre de 2014.] http://www.symantec.com/about/news/release/article.jsp?prid=19960618_01.
13. **Sigho.** Sigho. [En línea] [Citado el: 21 de noviembre de 2014.] <http://sigho.ses-gro.gob.mx/wp-content/uploads/2009/10/Actualizador-SIGHO-v3.3.0.pdf>.
14. **Nabber.** Nabber. [En línea] [Citado el: 12 de noviembre de 2014.] <http://www.nabber.org/projects/appupdater/>.
15. **Q, Jose M Bautista.** Ingeniería de Software. [En línea] [Citado el: 27 de mayo de 2015.] http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.
16. **Duque, Raúl González.** Mundo geek. *Python para todos.* [En línea] [Citado el: 15 de noviembre de 2014.] <http://mundogeek.net/tutorial-python/>.
17. **Valdés, Damián Pérez.** Maestros del Web. [En línea] 03 de julio de 2007. [Citado el: 26 de mayo de 2015.] <http://www.maestrosdelweb.com/que-es-javascript/>.

18. **Softwaresea.** Softwaresea. [En línea] [Citado el: 21 de marzo de 2015.] <http://es.softwaresea.com/download-Visual-Paradigm-for-UML-10404767.htm>.
19. **Falasco, Roxana.** Roxana Falasco. [En línea] 23 de julio de 2012. [Citado el: 21 de marzo de 2015.] <http://falasco.org/guia-definitiva-sublime-text-2>.
20. **Centro de Telemática.** Wiki GRHS. *Plugin para Xilema Base Web*. [En línea] [Citado el: 12 de mayo de 2015.] http://10.128.50.236/grhs-doc/index.php/Plugin_para_Xilema_Base_Web.
21. **Holovaty, Adrian y Kaplan-Moss, Jacob.** *El libro de Django*. s.l. : Jeremy Dunck,, 2008.
22. **Alvarez, Miguel Angel.** Desarrollo web. [En línea] [Citado el: 18 de mayo de 2015.] <http://www.desarrolloweb.com/manuales/manual-jquery.html>.
23. **Rodríguez, Txema.** Gen beta. [En línea] 23 de mayo de 2013. [Citado el: 18 de mayo de 2015.] <http://www.genbetadev.com/desarrollo-web/backbone-js-el-framework-para-construir-aplicaciones-usando-javascript-siguiendo-el-patron-mvc-alcanza-la-version-1-0>.
24. **PostgreSQL.** [En línea] 02 de octubre de 2010. [Citado el: 28 de mayo de 2015.] http://www.postgresql.org.es/sobre_postgresql.
25. **TechnologyUK.** TechnologyUK. [En línea] [Citado el: 06 de junio de 2015.] www.technologyuk.net/computing/uml/deployment_diagrams.shtml.
26. **Nuñez, Gabriel, Cardeso, Fabio y Camacho, Erika.** *Arquitecturas de Software. Guía de estudio*. 2004.
27. **Gil, José A. San.** Scribd. [En línea] [Citado el: 12 de abril de 2015.] <http://es.scribd.com/doc/14704374/Arquitectura-Basada-en-Componentes#scribd>.
28. **Libros Web.** Libros Web. [En línea] [Citado el: 13 de mayo de 2015.] https://librosweb.es/libro/django_1_0/capitulo_5/el_patron_de_diseno_mtv.html.
29. **Montero, Sergio Infante.** Maestros del web. [En línea] 30 de abril de 2012. [Citado el: 12 de abril de 2015.] <http://www.maestrosdelweb.com/curso-django-entendiendo-como-trabaja-django/>.
30. **Kioskea.** [En línea] [Citado el: 18 de marzo de 2015.] <http://es.kioskea.net/contents/224-patrones-de-diseno>.
31. **Grosso, Andrés.** Prácticas de Software. [En línea] 21 de mayo de 2011. [Citado el: 12 de marzo de 2015.] <http://www.practicadesoftware.com.ar/2011/03/patrones>.
32. **Python Software Foundation.** Python. *PEP 0008 -- Style Guide for Python Code*. [En línea] 01 de agosto de 2013. [Citado el: 29 de mayo de 2015.] <https://www.python.org/dev/peps/pep-0008/>.
33. **Pressman, Roger S.** *Ingeniería del Software. Un enfoque práctico. Quinta edición*. s.l. : Mc Graw Hill.
34. **Universidad de Alicante.** Servicio de Informática - Universidad de Alicante. [En línea] [Citado el: 21 de abril de 2015.] <http://si.ua.es/es/documentacion/c-sharp/documentos/pruebas/07pruebasunitarias.pdf>.

BIBLIOGRAFÍA

1. **Microsoft.** Microsoft Windows. *Windows Update*. [Online] [Cited: enero 26, 2015.]
<http://windows.microsoft.com/es-419/windows7/products/features/windows-update>.
2. **Lindoro, M.C. María Lourdes Armenta.** Dirección General de Información en Salud (DGIS). [Online] noviembre 10, 2010. [Cited: octubre 20, 2014.]
http://www.dgis.salud.gob.mx/descargas/pdf/11_SinaloaSIGHO_LourdesArmenta.pdf.
3. **Codeplex.** Codeplex. *Appupdater*. [Online] [Cited: octubre 15, 2014.]
<https://appupdater.codeplex.com/>.
4. **Valdéz, José Luis Cendejas.** Ecumed. [Online] [Cited: noviembre 15, 2014.]
<http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm>.
5. **Pastrana, Ophelia.** Intelligence to Business. [Online] octubre 2014. [Cited: enero 23, 2015.]
<http://www.i2btech.com/blog-i2b/tech-deployment/5-beneficios-de-aplicar-metodologias-agiles-en-el-desarrollo-de-software/>.
6. **Python Org.** Wiki Python. [Online] [Cited: enero 27, 2015.]
<https://wiki.python.org/moin/SpanishLanguage>.
7. —. About Python. [Online] [Cited: febrero 02, 2015.] <https://www.python.org/about/>.
8. **Eguiluz, Javier.** Librosweb. *Introducción a JavaScript*. [Online] [Cited: febrero 02, 2015.]
<http://librosweb.es/libro/javascript/>.
9. **Mozilla Foundation.** Mozilla Developer Network. [Online] 04 12, 2015. [Cited: mayo 03, 2015.] <https://developer.mozilla.org/es/docs/Web/JavaScript>.
10. **Sublime Text.** Sublime Text 2 Documentation. [Online] julio 08, 2013. [Cited: febrero 24, 2015.] <https://www.sublimetext.com/docs/2/>.
11. **Django.** Django Project. *Getting started with Django*. [Online] [Cited: enero 06, 2015.]
<https://www.djangoproject.com/start/>.
12. **jQuery Foundation.** [Online] [Cited: marzo 01, 2015.] <http://api.jquery.com/>.
13. **Murphey, Rebecca.** *Fundamentos de jQuery*. [Online] agosto 2013. [Cited: marzo 03, 2015.] <http://librojquery.com/>.
14. **Backbone.** Backbone Documentation. [Online] [Cited: marzo 12, 2015.]
<http://backbonejs.org/>.
15. **Ubuntu Foundation.** Ubuntu Documentation. [Online] abril 26, 2015. [Cited: mayo 12, 2015.] <https://help.ubuntu.com/community/PostgreSQL>.

16. **Universidad de Sevilla.** Departamento de Lenguajes y Sistemas Informáticos. *Introducción a la Arquitectura del Software.* [Online] 2013. [Cited: marzo 02, 2015.]
<https://www.lsi.us.es/docencia/get.php?id=6496>.
17. **Castro, Juana.** Prezi. [Online] octubre 26, 2013. [Cited: febrero 12, 2015.]
<https://prezi.com/jaknoft7mzmt/1-arquitecturas-cliente-servidor/>.
18. **Ruiz, Julián.** Prezi. [Online] julio 02, 2014. [Cited: febrero 04, 2015.]
<https://prezi.com/nnm5yq0k9uc/proceso-de-desarrollo-de-software-basado-en-componentes/>.
19. **Django.** Django Project. [Online] [Cited: marzo 12, 2015.]
<https://docs.djangoproject.com/en/1.8/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names>.
20. **Jasen, Remy and G, Ballintijn.** *A process framework and typology for software product updaters.* s.l. : IEEE, 2005. 0-7695-2304-8.
21. **Symantec.** Symantec. [Online] [Cited: noviembre 12, 2014.]
http://www.symantec.com/about/news/release/article.jsp?prid=19960618_01.
22. **Microsoft Computers.** Microsoft Update Product Team. [Online] [Cited: noviembre 12, 2014.] <http://blogs.technet.com/b/mu>.
23. **Nabber.** Nabber. [Online] [Cited: noviembre 12, 2014.]
<http://www.nabber.org/projects/appupdater/>.
24. **Duque, Raúl González.** Mundo geek. *Python para todos.* [Online] [Cited: noviembre 15, 2014.] <http://mundogeek.net/tutorial-python/>.
25. **Pressman, Roger S.** *Ingeniería del Software. Un enfoque práctico. Quinta edición.* s.l. : Mc Graw Hill.
26. **Holovaty, Adrian and Kaplan-Moss, Jacob.** *El libro de Django.* s.l. : Jeremy Dunck,, 2008.
27. **Carzaniga, Antonio.** *A Characterization Framework for Software Deployment Technologies.* s.l. : University of Colorado,, 1998.
28. **Instituto de Tecnología de Massachusetts.** s.l. : MIT, 2004.
29. **Grosso, Andrés.** *Prácticas de Software.* [Online] mayo 21, 2011. [Cited: marzo 12, 2015.]
<http://www.practicadesoftware.com.ar/2011/03/patrones>.
30. **Universidad Nacional de Colombia.** [Online] [Cited: marzo 14, 2015.]
http://www.virtual.unal.edu.co/cursos/IDEA/2007219/lecciones/cap_4/sub8.html.
31. **Libros Web.** Libros Web. [Online] [Cited: mayo 13, 2015.]
https://librosweb.es/libro/django_1_0/capitulo_5/el_patron_de_diseno_mtv.html.

32. **Kioskea.** [Online] [Cited: marzo 18, 2015.] <http://es.kioskea.net/contents/224-patrones-de-diseno>.
33. **Cestero, José M.** [Online] junio 16, 2009. [Cited: noviembre 12, 2014.] <http://www.pymesyautonomos.com/tecnologia/el-soporte-tecnico-es-la-clave>.
34. **Sigho.** Sigho. [Online] [Cited: noviembre 21, 2014.] <http://sigho.ses-gro.gob.mx/wp-content/uploads/2009/10/Actualizador-SIGHO-v3.3.0.pdf>.
35. **Softwaresea.** Softwaresea. [Online] [Cited: marzo 21, 2015.] <http://es.softwaresea.com/download-Visual-Paradigm-for-UML-10404767.htm>.
36. **Falasco, Roxana.** Roxana Falasco. [Online] julio 23, 2012. [Cited: marzo 21, 2015.] <http://falasco.org/guia-definitiva-sublime-text-2>.
37. **Gil, José A. San.** Scribd. [Online] [Cited: abril 12, 2015.] <http://es.scribd.com/doc/14704374/Arquitectura-Basada-en-Componentes#scribd>.
38. **Montero, Sergio Infante.** Maestros del web. [Online] abril 30, 2012. [Cited: abril 12, 2015.] <http://www.maestrosdelweb.com/curso-django-entendiendo-como-trabaja-django/>.
39. **Universidad de Alicante.** Servicio de Informática - Universidad de Alicante. [Online] [Cited: abril 21, 2015.] <http://si.ua.es/es/documentacion/c-sharp/documentos/pruebas/07pruebasunitarias.pdf>.
40. **Docsetools.** Docsetools. [Online] [Cited: marzo 23, 2015.] http://docsetools.com/articulos-de-todos-los-temas/article_25832.html.
41. **La Web del Programador.** Diccionario La Web del Programador. [Online] [Cited: noviembre 19, 2014.] <http://www.lawebdelprogramador.com/diccionario/buscar.php?opc=1&charSearch=Optimizaci%C3%B3n+de+software>.
42. —. Diccionario La Web del Programador. [Online] [Cited: 03 15, 2015.] <http://www.lawebdelprogramador.com/diccionario/buscar.php?opc=1&charSearch=software>.
43. **Centro de Telemática.** Wiki GRHS. [Online] [Cited: mayo 28, 2015.] <http://10.128.50.236/grhs-doc/index.php/GRHS>.
44. **Valdés, Damián Pérez.** Maestros del Web. [Online] julio 03, 2007. [Cited: mayo 26, 2015.] <http://www.maestrosdelweb.com/que-es-javascript/>.
45. **Centro de Telemática.** Wiki GRHS. *Plugin para Xilema Base Web.* [Online] [Cited: mayo 12, 2015.] http://10.128.50.236/grhs-doc/index.php/Plugin_para_Xilema_Base_Web.
46. **Alvarez, Miguel Angel.** Desarrollo web. [Online] [Cited: mayo 18, 2015.] <http://www.desarrolloweb.com/manuales/manual-jquery.html>.

47. **Rodríguez, Txema.** Gen beta. [Online] mayo 23, 2013. [Cited: mayo 18, 2015.]
<http://www.genbetadev.com/desarrollo-web/backbone-js-el-framework-para-construir-aplicaciones-usando-javascript-siguiendo-el-patron-mvc-alcanza-la-version-1-0>.
48. **Python Software Foundation.** Python. *PEP 0008 -- Style Guide for Python Code*. [Online] agosto 01, 2013. [Cited: mayo 29, 2015.] <https://www.python.org/dev/peps/pep-0008/>.
49. **TechnologyUK.** TechnologyUK. [Online] [Cited: junio 06, 2015.]
www.technologyuk.net/computing/uml/deployment_diagrams.shtml.
50. **PostgreSQL.** [Online] octubre 02, 2010. [Cited: mayo 28, 2015.]
http://www.postgresql.org.es/sobre_postgresql.
51. **Diario Clarín.** Diario Clarín. [Online] octubre 14, 2010. [Cited: noviembre 18, 2014.]
http://www.clarin.com/internet/Microsoft-mayor-parche-seguridad-historia_0_352764974.html.
52. **Q, Jose M Bautista.** Ingeniería de Software. [Online] [Cited: mayo 27, 2015.]
http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.
53. **Nuñez, Gabriel, Cardeso, Fabio and Camacho, Erika.** *Arquitecturas de Software. Guía de estudio*. 2004.
54. **Canós, José H, Letelier, Patricio and Penadés, Ma Carmen.** *Metodologías Ágiles en el Desarrollo de Software*. s.l. : DSIC -Universidad Politécnica de Valencia.

ANEXOS

Anexo 1: Historias de usuario

Tabla 13 HU Descargar actualizaciones

Historia de usuario	
Número: HU-2	Nombre de la historia de usuario: Descargar actualizaciones
Modificación de la historia de usuario número: Ninguna	Usuario: Sistema
Programador responsable: Yosján Pérez Cuello Roberto Rafael Ramírez Martínez	Iteración asignada: Primera
	Puntos estimados: 1
Prioridad en el negocio: Muy alta	Riesgo en desarrollo: Muy alto
Descripción: Se descarga la actualización desde el Servidor de Actualizaciones.	
Observaciones: El Servidor de Actualizaciones debe estar operando correctamente. La descarga se realiza al directorio de ficheros comprimidos de la solución.	
Prototipo de interfaz: No procede	

Tabla 14 HU Comprobar integridad

Historia de usuario	
Número: HU-3	Nombre de la historia de usuario: Comprobar integridad
Modificación de la historia de usuario número: Ninguna	Usuario: Sistema
Programador responsable: Yosján Pérez Cuello Roberto Rafael Ramírez Martínez	Iteración asignada: Primera
	Puntos estimados: 1
Prioridad en el negocio: Muy alta	Riesgo en desarrollo: Muy alto
Descripción: Se realiza la comprobación de integridad de lo descargado.	
Observaciones:	
Prototipo de interfaz: No procede	

Tabla 15 HU Descomprimir actualizaciones

Historia de usuario	
Número: HU-4	Nombre de la historia de usuario: Descomprimir actualizaciones
Modificación de la historia de usuario número: Ninguna	Usuario: Sistema
Programador responsable: Yosján Pérez Cuello Roberto Rafael Ramírez Martínez	Iteración asignada: Segunda
	Puntos estimados: 1
Prioridad en el negocio: Alta	Riesgo en desarrollo: Muy alto
Descripción: Se realiza la descompresión de la actualización descargada.	
Observaciones: La descompresión se realiza en el directorio de descomprimidos (downloads/unpack) de la solución.	
Prototipo de interfaz: No procede	

Tabla 16 HU Realizar copia de seguridad

Historia de usuario	
Número: HU-5	Nombre de la historia de usuario: Realizar copia de seguridad
Modificación de la historia de usuario número: Ninguna	Usuario: Sistema
Programador responsable: Yosján Pérez Cuello Roberto Rafael Ramírez Martínez	Iteración asignada: Segunda
	Puntos estimados: 1
Prioridad en el negocio: Alta	Riesgo en desarrollo: Alto
Descripción: Se realiza una copia de seguridad del cliente antes de actualizarlo.	
Observaciones: La copia de seguridad se realiza en el directorio <i>backup</i> de la solución.	
Prototipo de interfaz: No procede	

Tabla 17 HU Ejecutar actualización

Historia de usuario	
----------------------------	--

Número: HU-6	Nombre de la historia de usuario: Ejecutar actualización
Modificación de la historia de usuario número: Ninguna	Usuario: Sistema
Programador responsable: Yosján Pérez Cuello Roberto Rafael Ramírez Martínez	Iteración asignada: Segunda
	Puntos estimados: 1
Prioridad en el negocio: Alta	Riesgo en desarrollo: Muy alto
Descripción: Se realiza la actualización del cliente, para ello se detiene el servicio del mismo y se procede a ejecutar las modificaciones. Posteriormente se reinicia el servicio del cliente.	
Observaciones: Los cambios se realizan según el tipo de actualización (total/parcial).	
Prototipo de interfaz: No procede	

Tabla 18 HU Recuperar el sistema ante fallos de actualización

Historia de usuario	
Número: HU-7	Nombre de la historia de usuario: Recuperar el sistema ante fallos de actualización
Modificación de la historia de usuario número: Ninguna	Usuario: Sistema
Programador responsable: Yosján Pérez Cuello Roberto Rafael Ramírez Martínez	Iteración asignada: T
	Puntos estimados: Tercera
Prioridad en el negocio: Media	Riesgo en desarrollo: Alto
Descripción: Se realiza la recuperación del cliente ante fallos ocurridos en el proceso de actualización. Consiste en pasar el cliente a la versión de salva realizada si se producen errores durante el proceso de actualización.	
Observaciones:	
Prototipo de interfaz: No procede	

Tabla 19 HU Notificar estado de actualización a Gserver

Historia de usuario	
----------------------------	--

Número: HU-8	Nombre de la historia de usuario: Notificar estado de actualización a Gserver
Modificación de la historia de usuario número: Ninguna	Usuario: Sistema
Programador responsable: Yosján Pérez Cuello Roberto Rafael Ramírez Martínez	Iteración asignada: Tercera
	Puntos estimados: 1
Prioridad en el negocio: Media	Riesgo en desarrollo: Medio
Descripción: Se envía una notificación al servidor sobre el estado de la actualización. El estado indica si la actualización fue exitosa (si se actualizó correctamente el cliente), con advertencias (si el cliente tuvo que ser recuperado) o fue fallida (si no se pudo recuperar el cliente).	
Observaciones:	
Prototipo de interfaz: No procede	

Tabla 20 HU Configurar actualizaciones en Gadmin

Historia de usuario	
Número: HU-9	Nombre de la historia de usuario: Configurar actualizaciones en Gadmin
Modificación de la historia de usuario número: Ninguna	Usuario: Administrador
Programador responsable: Yosján Pérez Cuello Roberto Rafael Ramírez Martínez	Iteración asignada: Tercera
	Puntos estimados: 1
Prioridad en el negocio: Media	Riesgo en desarrollo: Medio
Descripción: Se configura en el Módulo de Administración las actualizaciones para el cliente. Las actualizaciones pueden ser de tipo total (se reemplaza completamente el cliente) o de tipo parcial (se sustituyen los plugins modificados, se limpia el registro de logs, se elimina la base de datos o se cambia la configuración del cliente).	
Observaciones: El usuario debe estar autenticado en el sistema y tener los privilegios correspondientes a la gestión de actualizaciones.	
Prototipo de interfaz:	

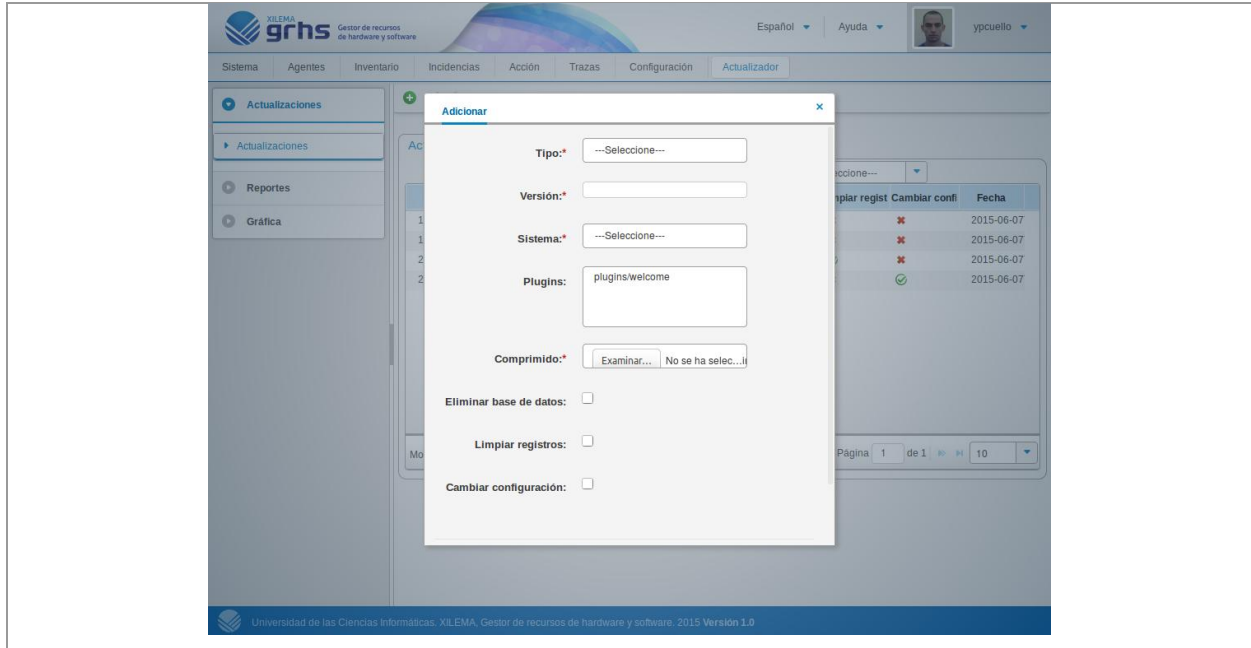


Tabla 21 HU Mostrar reporte de actualizaciones en Gadmin

Historia de usuario	
Número: HU-10	Nombre de la historia de usuario: Mostrar reporte de actualizaciones en Gadmin
Modificación de la historia de usuario número: Ninguna	Usuario: Administrador
Programador responsable: Yosján Pérez Cuello Roberto Rafael Ramírez Martínez	Iteración asignada: Cuarta
	Puntos estimados: 1
Prioridad en el negocio: Baja	Riesgo en desarrollo: Bajo
Descripción: En Gadmin se muestra el listado de reportes realizado por el actualizador instalado en cada cliente.	
Observaciones: El usuario debe estar autenticado en el sistema y tener los privilegios correspondientes a la gestión de reportes.	
Prototipo de interfaz:	

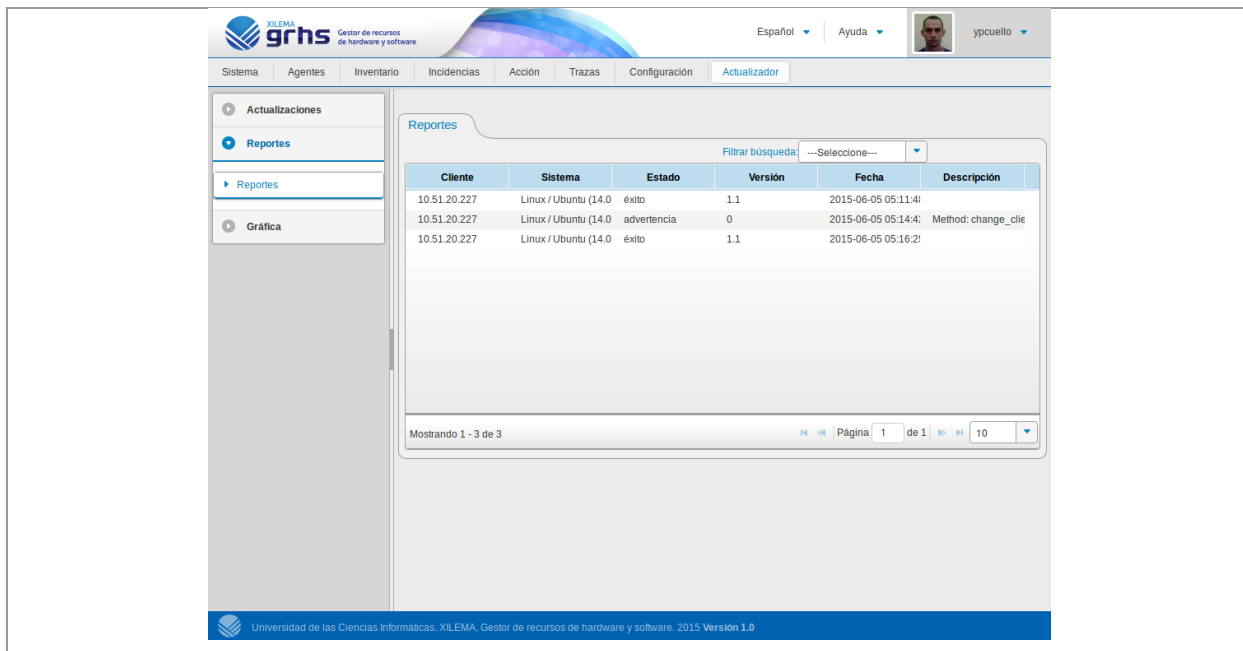


Tabla 22 HU Filtrar reportes por estado de actualización

Historia de usuario	
Número: HU-11	Nombre de la historia de usuario: Filtrar reportes por estado de actualización
Modificación de la historia de usuario número: Ninguna	Usuario: Administrador
Programador responsable: Yosján Pérez Cuello Roberto Rafael Ramírez Martínez	Iteración asignada: Cuarta
	Puntos estimados: 1
Prioridad en el negocio: Baja	Riesgo en desarrollo: Bajo
Descripción: Se realiza un filtrado de los reportes por los parámetros IP del cliente, estado del reporte, fecha del reporte y versión de actualización ejecutada.	
Observaciones: El usuario debe estar autenticado en el sistema y tener los privilegios correspondientes a la gestión de reportes.	
Prototipo de interfaz:	

Anexo 2: Tareas de ingeniería

Tabla 24 Tarea de ingeniería 2 Implementar descargar actualizaciones

Tarea de ingeniería

Número de la tarea: 2	Número de la historia de usuario: 2
Nombre de la tarea: Implementar descargar actualizaciones	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 23/03/2015	Fecha fin: 27/03/2015
Programador responsable: Yosján Pérez Cuello	
Descripción: El Cliente de Actualización realiza una petición HTTP/HTTPS al Servidor de Actualizaciones solicitando la última actualización disponible. La descarga se realiza en el directorio de ficheros comprimidos.	

Tabla 25 Tarea de ingeniería 4 Implementar comprobar integridad

Tarea de ingeniería	
Número de la tarea: 3	Número de la historia de usuario: 3
Nombre de la tarea: Implementar comprobar integridad	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 30/03/2015	Fecha fin: 03/04/2015
Programador responsable: Yosján Pérez Cuello	
Descripción: Se realiza la comprobación del md5 del fichero descargado y se compara con el valor recibido del servidor (almacenado en el arreglo creado en la Tarea de ingeniería 1), de ser iguales los valores se continúa el proceso de actualización. Caso contrario se detiene el proceso de actualización.	

Tabla 2623 Tarea de ingeniería 6 Implementar descomprimir actualizaciones

Tarea de ingeniería	
Número de la tarea: 4	Número de la historia de usuario: 4
Nombre de la tarea: Implementar descomprimir actualizaciones	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 06/04/2015	Fecha fin: 10/04/2015
Programador responsable: Roberto Rafael Ramírez Martínez	
Descripción: Se realiza la descompresión del fichero descargado en el directorio de descomprimidos (downloads/unpack) del sistema.	

Tabla 27 Tarea de ingeniería 8 Implementar realizar copia de seguridad

Tarea de ingeniería

Número de la tarea: 5	Número de la historia de usuario: 5
Nombre de la tarea: Implementar realizar copia de seguridad	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 13/04/2015	Fecha fin: 17/04/2015
Programador responsable: Roberto Rafael Ramírez Martínez	
Descripción: Se realiza una copia de seguridad (backup) íntegra del cliente GRHS en el directorio backup.	

Tabla 28 Tarea de Ingeniería 10 Implementar ejecutar actualización

Tarea de ingeniería	
Número de la tarea: 6	Número de la historia de usuario: 6
Nombre de la tarea: Implementar ejecutar actualización	
Tipo de tarea: Desarrollo	Puntos estimados: 2
Fecha inicio: 20/04/2015	Fecha fin: 24/04/2015
Programador responsable: Roberto Rafael Ramírez Martínez	
<p>Descripción: Se verifica, a partir del arreglo de información creado en la Tarea de ingeniería 1, el tipo de actualización a ejecutar.</p> <p>Si el tipo de actualización es total se procede a realizar el reemplazo total del cliente GRHS, para ello se elimina el directorio donde se encuentra el mismo y se procede a copiar en su lugar el directorio con igual nombre contenido en el descomprimido de la actualización.</p> <p>Si el tipo de actualización es parcial se procede a realizar las siguientes verificaciones:</p> <ul style="list-style-type: none"> - Si la actualización incluye eliminar la base de datos se elimina la base de datos del cliente GRHS. - Si la actualización incluye limpiar el registro de logs se elimina la información contenida en el fichero de logs del cliente GRHS. - Si la actualización incluye cambiar el fichero de configuración se realiza el parseo de la información contenida en un fichero de configuración que se incluye en la descarga y se cambian los valores del fichero de configuración del cliente GRHS por los nuevos valores. - Si existen plugins para modificar se procede a eliminar dichos plugins del cliente GRHS y se copian en su lugar los plugins contenidos en la descarga. 	

Tabla 24 Tarea de ingeniería 12 Implementar recuperar el sistema ante fallos de actualización

Tarea de ingeniería	
Número de la tarea: 7	Número de la historia de usuario: 7
Nombre de la tarea: Implementar recuperar el sistema ante fallos de actualización	
Tipo de tarea: Desarrollo	Puntos estimados: 2
Fecha inicio: 27/04/2015	Fecha fin: 01/05/2015
Programador responsable: Yosján Pérez Cuello	
Descripción: Se procede a realizar la recuperación del cliente. De producirse errores en el proceso de ejecutar las actualizaciones este método es llamado. Se realiza la eliminación del directorio del cliente GRHS y este es sustituido por los ficheros existentes en la copia de seguridad (backup). Se realiza un reporte al servidor indicando es estado de advertencia pues no se realizó una actualización satisfactoria.	

Tabla 25 Tarea de ingeniería 14 Implementar notificar estado de actualización a Gserver

Tarea de ingeniería	
Número de la tarea: 8	Número de la historia de usuario: 8
Nombre de la tarea: Implementar notificar estado de actualización a Gserver	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 04/05/2015	Fecha fin: 08/05/2015
Programador responsable: Yosján Pérez Cuello	
Descripción: El método es invocado en varios estados del proceso. Se realiza una consulta HTTP/HTTPS al Servidor de Actualizaciones enviando la información del estado de actualización. Si la actualización fue exitosa envía al servidor el estado satisfactorio. En caso de recuperación exitosa del cliente GRHS se envía el estado de advertencia, en caso de fallo en la recuperación del cliente se envía el estado de error. En todos los casos se envía la dirección IP del cliente de actualizaciones, la versión que se está ejecutando en el cliente después de realizar el proceso de actualización, y en caso de que no sea exitosa se envía como descripción del error el método donde se produjo y la causa del error.	

Tabla 26 Tarea de ingeniería 16 Implementar configurar actualizaciones en Gadmin

Tarea de ingeniería	
Número de la tarea: 9	Número de la historia de usuario: 9
Nombre de la tarea: Implementar configurar actualizaciones en Gadmin	

Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 11/05/2015	Fecha fin: 15/05/2015
Programador responsable: Yosján Pérez Cuello	
Descripción: El usuario selecciona la opción <i>Actualizador</i> en el menú del panel de administración de GRHS. Posteriormente selecciona la opción <i>Actualizaciones</i> y se muestra el listado de actualizaciones disponibles. El usuario selecciona la opción <i>Adicionar (ícono de adición)</i> y se muestra el formulario correspondiente a una actualización. El usuario selecciona el tipo de actualización (parcial/total) y el fichero comprimido. En caso de ser actualización de tipo parcial el cliente selecciona los plugins que serán modificados, si será eliminada la base de datos, si será limpiado el registro de logs y si serán cambiados los parámetros de configuración. El usuario selecciona la opción <i>Cancelar</i> y se termina el proceso o selecciona la opción <i>Aceptar</i> y se registra una nueva actualización.	

Tabla 27 Tarea de ingeniería 17 Implementar mostrar reporte de actualizaciones en Gadmin

Tarea de ingeniería	
Número de la tarea: 10	Número de la historia de usuario: 10
Nombre de la tarea: Implementar mostrar reporte de actualizaciones en Gadmin	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 18/05/2015	Fecha fin: 22/05/2015
Programador responsable: Yosján Pérez Cuello	
Descripción: El usuario selecciona la opción <i>Actualizador</i> en el menú del panel de administración de GRHS. Posteriormente selecciona la opción <i>Reportes</i> y se muestra el listado de reportes realizados por los clientes de actualizaciones. Los reportes pueden ser filtrados por cada atributo.	

Tabla 28 Tarea de ingeniería 18 Implementar filtrar reportes por estado de actualización

Tarea de ingeniería	
Número de la tarea: 11	Número de la historia de usuario: 11
Nombre de la tarea: Implementar filtrar reportes por estado de actualización	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 25/05/2015	Fecha fin: 29/05/2015
Programador responsable: Roberto Rafael Ramírez Martínez	

Descripción: El usuario selecciona la opción *Actualizador* en el menú del panel de administración de GRHS. El usuario selecciona la opción *Reportes* y se muestra el listado de reportes realizados. El usuario selecciona la opción *Filtrar búsqueda* y especifica los parámetros por los que desea filtrar los reportes. Se muestra el listado de reportes que cumplen con los valores especificados.

Anexo 3: Pruebas unitarias

Tabla 29 Prueba unitaria 2, Descargar actualizaciones

Prueba de: Unidad	
Nombre de la prueba: Descargar actualizaciones	
Estado: Satisfactoria	Última ejecución: 27/03/2015
Ejecutado por: Yosján Pérez Cuello	Verificado por: Yosján Pérez Cuello
Descripción: La aplicación realiza una petición HTTP al Servidor de Actualizaciones y descarga el fichero que es devuelto por el servidor en el directorio de comprimidos.	
Entrada: Se envía en la consulta la versión de la actualización a descargar.	
Criterio de aceptación: La funcionalidad se ejecuta satisfactoriamente. Grado de aceptación 100%.	
Resultado: La funcionalidad se encuentra lista para ser integrada al sistema.	

Tabla 30 Prueba unitaria 3, Comprobar integridad

Prueba de: Unidad	
Nombre de la prueba: Comprobar integridad	
Estado: Satisfactoria	Última ejecución: 03/04/2015
Ejecutado por: Yosján Pérez Cuello	Verificado por: Yosján Pérez Cuello
Descripción: Se realiza la comprobación del md5 del fichero descargado. Se comprueba el resultado con la información obtenida del servidor. De ser diferentes culmina la ejecución.	
Entrada: Ubicación del fichero descargado; valor del md5 devuelto por el servidor.	
Criterio de aceptación: La funcionalidad se ejecuta satisfactoriamente. Grado de aceptación 100%.	
Resultado: La funcionalidad se encuentra lista para ser integrada al sistema.	

Tabla 31 Prueba unitaria 4, Descomprimir actualizaciones

Prueba de: Unidad	
Nombre de la prueba: Descomprimir actualizaciones	
Estado: Satisfactoria	Última ejecución: 10/04/2015
Ejecutado por: Yosján Pérez Cuello	Verificado por: Yosján Pérez Cuello
Descripción: Se realiza la descompresión del fichero descargado en el directorio de descomprimidos (downloads/unpack).	
Entrada: Ubicación del fichero descargado.	
Criterio de aceptación: La funcionalidad se ejecuta satisfactoriamente. Grado de aceptación 100%.	
Resultado: La funcionalidad se encuentra lista para ser integrada al sistema.	

Tabla 32 Prueba unitaria 5, Realizar copia de seguridad

Prueba de: Unidad	
Nombre de la prueba: Realizar copia de seguridad	
Estado: Satisfactoria	Última ejecución: 17/04/2015
Ejecutado por: Roberto Rafael Ramírez Martínez	Verificado por: Roberto Rafael Ramírez Martínez
Descripción: Se realiza la copia de seguridad del cliente en el directorio backup de la aplicación.	
Entrada: Ubicación del cliente.	
Criterio de aceptación: La funcionalidad se ejecuta satisfactoriamente. Grado de aceptación 100%.	
Resultado: La funcionalidad se encuentra lista para ser integrada al sistema.	

Tabla 33 Prueba unitaria 6, Ejecutar actualización

Prueba de: Unidad	
Nombre de la prueba: Ejecutar actualización	
Estado: Satisfactoria	Última ejecución: 24/04/2015
Ejecutado por: Roberto Rafael Ramírez Martínez	Verificado por: Roberto Rafael Ramírez Martínez
Descripción: Se comprueba el tipo de actualización a realizar. (Si es total) se realiza el reemplazo total del cliente. (Si es parcial), se limpia el registro de logs del cliente y/o se elimina	

la base de datos del cliente y/o se cambia la configuración del cliente y/o se sustituyen los plugins especificados en la actualización.
Entrada: Ubicación del cliente; tipo de actualización; información de la actualización.
Criterio de aceptación: La funcionalidad se ejecuta satisfactoriamente. Grado de aceptación 100%.
Resultado: La funcionalidad se encuentra lista para ser integrada al sistema.

Tabla 34 Prueba unitaria 7, Recuperar el sistema ante fallos de actualización

Prueba de: Unidad	
Nombre de la prueba: Recuperar el sistema ante fallos de actualización	
Estado: Satisfactoria	Última ejecución: 01/05/2015
Ejecutado por: Yosján Pérez Cuello	Verificado por: Yosján Pérez Cuello
Descripción: De producirse errores en la actualización se realiza reemplazo total del cliente por la copia de seguridad realizada en el directorio backup. Se realiza el cambio de estado de actualización a <i>advertencia</i> . Se reinicia el servicio cliente. Si el servicio no se reinicia se realiza el cambio de estado de actualización a <i>error</i> . Se detiene la ejecución del proceso.	
Entrada: Ubicación del cliente.	
Criterio de aceptación: La funcionalidad se ejecuta satisfactoriamente. Grado de aceptación 100%.	
Resultado: La funcionalidad se encuentra lista para ser integrada al sistema.	

Tabla 35 Prueba unitaria 8, Notificar estado de actualización a Gserver

Prueba de: Unidad	
Nombre de la prueba: Notificar estado de actualización a Gserver	
Estado: Satisfactoria	Última ejecución: 08/05/2015
Ejecutado por: Roberto Rafael Ramírez Martínez	Verificado por: Roberto Rafael Ramírez Martínez
Descripción: De realiza una consulta HTTP al Servidor de Actualizaciones informando del estado de la actualización.	
Entrada: Estado de actualización; versión que se ejecuta en el cliente después del proceso de actualización; descripción del error de haberse producido alguno; dirección IP del cliente.	
Criterio de aceptación: La funcionalidad se ejecuta satisfactoriamente. Grado de aceptación 100%.	

Resultado: La funcionalidad se encuentra lista para ser integrada al sistema.

Tabla 36 Prueba unitaria 9, Configurar actualizaciones en Gadmin

Prueba de: Unidad	
Nombre de la prueba: Configurar actualizaciones en Gadmin	
Estado: Satisfactoria	Última ejecución: 15/05/2015
Ejecutado por: Yosján Pérez Cuello	Verificado por: Yosján Pérez Cuello
Descripción: El usuario selecciona la opción <i>Añadir actualización</i> . El usuario agrega los datos de la actualización: tipo, versión, sistema, plugins, modificar configuración, limpiar registros, eliminar base de datos, fichero comprimido. El usuario selecciona la opción <i>Aceptar</i> y se registra la nueva actualización.	
Entrada: Tipo de actualización; plugins a modificar; modificar configuración; limpiar registros; eliminar base de datos; fichero comprimido.	
Criterio de aceptación: La funcionalidad se ejecuta satisfactoriamente. Grado de aceptación 100%.	
Resultado: La funcionalidad se encuentra lista para ser integrada al sistema.	

Tabla 37 Prueba unitaria 10, Mostrar reporte de actualizaciones en Gadmin

Prueba de: Unidad	
Nombre de la prueba: Mostrar reporte de actualizaciones en Gadmin	
Estado: Satisfactoria	Última ejecución: 22/05/2015
Ejecutado por: Roberto Rafael Ramírez Martínez	Verificado por: Roberto Rafael Ramírez Martínez
Descripción: El usuario selecciona la opción <i>Reportes</i> . Se muestra el listado de reportes realizado por el cliente de actualizaciones.	
Entrada: No se reciben datos.	
Criterio de aceptación: La funcionalidad se ejecuta satisfactoriamente. Grado de aceptación 100%.	
Resultado: La funcionalidad se encuentra lista para ser integrada al sistema.	

Tabla 38 Prueba unitaria 11, Filtrar reportes por estado de actualización

Prueba de: Unidad	
Nombre de la prueba: Filtrar reportes por estado de actualización	

Estado: Satisfactoria	Última ejecución: 29/05/2015
Ejecutado por: Roberto Rafael Ramírez Martínez	Verificado por: Roberto Rafael Ramírez Martínez
Descripción: El usuario selecciona la opción <i>Filtrar búsqueda</i> . Se muestra el listado de los reportes realizados por el cliente de actualizaciones que cumple con el criterio seleccionado.	
Entrada: Dirección IP del cliente, sistema, número de la versión, fecha, estado.	
Criterio de aceptación: La funcionalidad se ejecuta satisfactoriamente. Grado de aceptación 100%.	
Resultado: La funcionalidad se encuentra lista para ser integrada al sistema.	

Anexo 4: Pruebas de aceptación

Tabla 39 Prueba de aceptación 2, HU Descargar actualizaciones

Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba	Observaciones
Se realiza la solicitud de descarga al servidor mediante HTTP cuando este se encuentra operando correctamente.		Descarga el archivo que devuelve la consulta HTTP en el directorio de ficheros comprimidos de la aplicación y ejecuta otras funcionalidades.	Satisfactorio.	
	Se realiza la solicitud de descarga al servidor mediante HTTP cuando este no se encuentra operando.	Se reintentará realizar la descarga según se especifica en el fichero de configuración. Al término de todos los intentos se detiene la ejecución del proceso.	Satisfactorio.	

Tabla 40 Prueba de aceptación 3, HU Comprobar integridad

Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba	Observaciones
Se realiza la comprobación del md5 del fichero descargado y la clave es semejante a la enviada por el servidor.		Comprueba el md5 del fichero comprimido que se ha descargado y de ser correcto ejecuta otras funcionalidades.	Satisfactorio.	
	Se realiza la comprobación del md5 del fichero descargado y la clave es diferente a la enviada por el servidor.	Notifica de errores en la descarga y detiene la ejecución del proceso.	Satisfactorio.	

Tabla 41 Prueba de aceptación 4, HU Descomprimir actualizaciones

Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba	Observaciones
Se realiza la descompresión del fichero descargado cuando este existe en el directorio destino.		Descomprime el fichero en el directorio de descomprimidos (downloads/unpack) y ejecuta otras funcionalidades.	Satisfactorio.	

	Se realiza la descompresión del fichero descargado cuando este no está presente en el directorio destino.	Informa de la inexistencia del fichero y detiene la ejecución del proceso.	Satisfactorio.	
--	---	--	----------------	--

Tabla 42 Prueba de aceptación 5, HU Realizar copia de seguridad

Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba	Observaciones
Se realiza la copia de seguridad del cliente cuando el directorio del mismo existe en el sistema de directorios del sistema operativo.		Realiza una copia total del directorio del cliente hacia el directorio de salva de la aplicación (directorio backup) y ejecuta nuevas funcionalidades.	Satisfactorio.	
	Se realiza la copia de seguridad del cliente cuando el directorio del mismo es inexistente en el sistema de directorios del sistema operativo.	Informa de la inexistencia del directorio del cliente y detiene la ejecución del proceso.	Satisfactorio.	

Tabla 43 Prueba de aceptación 6, HU Ejecutar actualización total

Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba	Observaciones
Se realiza la actualización del cliente cuando el tipo de actualización es total y existe la nueva copia del cliente en la descarga.		Realiza el reemplazo total del cliente por la copia que se encuentra en el directorio de descarga y ejecuta otras funcionalidades.	Satisfactorio.	
	Se realiza la actualización del cliente cuando el tipo de actualización es total y no existe la nueva copia del cliente en la descarga.	Informa de la inexistencia del directorio del cliente y detiene la ejecución del proceso.	Satisfactorio.	

Tabla 44 Prueba de aceptación 7, HU Ejecutar actualización parcial

Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba	Observaciones
Se realiza la actualización del cliente cuando el tipo de actualización es parcial y existen todas las configuraciones.		Según la configuración específica se realiza el cambio asociado a la misma (eliminar base de datos, limpiar fichero de registros, cambiar configuraciones,	Satisfactorio.	

		sustituir plugins) y se ejecutan otras funcionalidades.		
	Se realiza la actualización del cliente cuando el tipo de actualización es parcial y no existen todas las configuraciones.	Reintenta realizar la actualización según se especifica en el fichero de configuración. Al término de todos los intentos informa de los errores relacionados con cada configuración específica y detiene la ejecución del proceso.	Satisfactorio.	

Tabla 45 Prueba de aceptación 8, HU Recuperar el sistema ante fallos de actualización

Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba	Observaciones
Se realiza la recuperación del cliente y el servicio del mismo es reiniciado.		Realiza la sustitución total del cliente por la copia de seguridad que está disponible en el directorio backup y ejecuta el envío de reportes especificando el estado a <i>advertencia</i> .	Satisfactorio.	
	Se realiza la recuperación del cliente y el	Realiza la sustitución total del cliente por la copia	Satisfactorio.	

	servicio del mismo no puede ser reiniciado.	de seguridad que está disponible en el directorio backup y ejecuta el envío de reportes especificando el estado a <i>error</i> .		
--	---	--	--	--

Tabla 46 Prueba de aceptación 9, HU Notificar estado de actualización a Gserver

Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba	Observaciones
Se envía el estado de la actualización al servidor GRHS cuando este se encuentra operando.		Realiza una consulta HTTP/HTTPS al servidor enviando los detalles de la actualización realizada o los motivos de error surgidos en el proceso. Detiene la ejecución del proceso.	Satisfactorio.	
	Se envía el estado de la actualización al servidor GRHS cuando este se encuentra operando.	Notifica del error en la conexión y detiene la ejecución del proceso.	Satisfactorio.	

Tabla 47 Prueba de aceptación 10, HU Configurar actualizaciones en Gadmin

Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba	Observaciones
-----------------------	-------------------------	---------------------------	-------------------------------	----------------------

El usuario selecciona la opción añadir actualización (ícono de adición) y se muestra el formulario de adición, el usuario completa el formulario y selecciona la opción <i>Aceptar</i> .		Se registra una nueva actualización en el Servidor de Actualizaciones.	Satisfactorio.	
--	--	--	----------------	--

Tabla 48 Prueba de aceptación 11, HU Mostrar reporte de actualizaciones en Gadmin

Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba	Observaciones
El usuario selecciona la opción <i>Reportes</i> .		Se muestra el listado de reportes realizados por el Cliente de Actualización al servidor.	Satisfactorio.	

Tabla 49 Prueba de aceptación 12, HU Filtrar reportes por estado de actualización

Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba	Observaciones
El usuario selecciona la opción <i>Filtrar búsqueda</i> .		Se muestran los reportes que coinciden con el criterio de búsqueda especificado.	Satisfactorio.	