Universidad de las Ciencias Informáticas

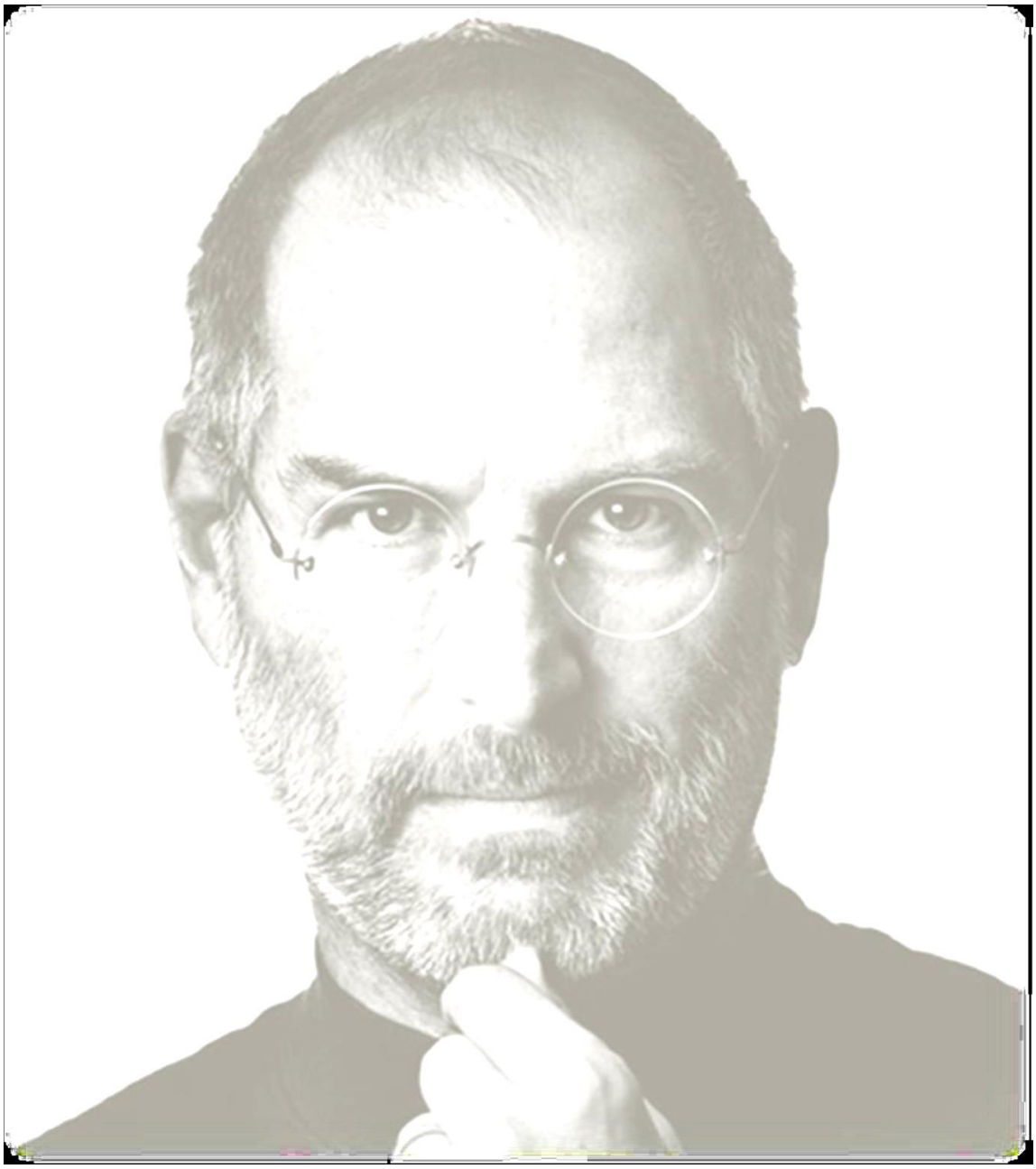
Facultad 2

Módulo para la ejecución automática de scripts en los clientes desde el servidor de GRHS.

Trabajo de Diploma
Presentado para optar por el título de Ingeniero en Ciencias Informáticas

Autor(es):
Alejandro Rodríguez Toledo
Orlando Viera Pérez

Tutor(es):
Ing. Adrian Ezequiel Mena Rodríguez
Ing. Dianet Díaz Oduardo



“Tu tiempo es limitado, de modo que no lo malgastes viviendo la vida de alguien distinto. No quedes atrapado en el dogma, que es vivir como otros piensan que deberías vivir. No dejes que los ruidos de las opiniones de los demás acallen tu propia voz interior. Y lo que es más importante, ten el coraje para hacer lo que te dice tu corazón y tu intuición. Ellos ya saben de algún modo en qué quieres convertirte realmente. Todo lo demás es secundario.”

Steve Jobs

DECLARACIÓN DE AUTORÍA

Por este medio se declara que, Alejandro Rodríguez Toledo y Orlando Viera Pérez, ambos estudiantes de la facultad 2 de la Universidad de las Ciencias Informáticas, son los únicos autores de este trabajo y los mismos autorizan al Centro de Telemática (TLM) en la propia universidad los derechos patrimoniales del mismo, con carácter exclusivo.

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

_____	_____
Alejandro Rodríguez Toledo	Orlando Viera Pérez
Autor	***Autor***
_____	_____
Ing. Adrian Ezequiel Mena Rodríguez	Ing. Dianet Díaz Oduardo
Tutor	***Tutor***

Resumen

El constante aumento de los activos informáticos en las entidades y empresas alrededor del mundo, ha provocado dificultades en mantener el control sobre el hardware y software usado. Para dar respuesta a la necesidad de identificar oportunamente las incidencias que se puedan producir, la Universidad de las Ciencias Informáticas desarrolló el Sistema Gestor de Recursos de Hardware y Software (GRHS); que además de contribuir a realizar el inventario de hardware y software instalado, posee un gran número de opciones de configuración, para detectar el uso no autorizado de los activos. Pero el mismo presenta algunas desventajas ya que no es posible la obtención de información no contenida dentro de las funcionalidades del sistema, que sea necesaria para la entidad y la configuración de parámetros de acuerdo con las políticas de la misma. La presente investigación permitió desarrollar un módulo para GRHS que facilita el trabajo de los administradores en las tareas de configuración y obtención de información de las computadoras. Para su implementación, bajo la guía de la metodología Programación Extrema, se emplearon como principales tecnologías: marco de trabajo Django 1.4 con el lenguaje de programación Python 2.7, el Sistema de Gestión de Base de Datos PostgreSQL 9.2 y Visual Paradigm 5.0 para el modelado. El Módulo, posee un conjunto de características y funcionalidades que contribuyen, al cambio de configuraciones desde GRHS en los computadores de la red y a la agilización de dicho proceso; permitiendo además la ejecución programada de tareas elaboradas en los scripts a ejecutar.

Palabras clave: configuración, GRHS, Programación Extrema, scripts.

Índice

Índice de Tablas

Índice de Ilustraciones

Introducción

Con el decursar del tiempo la humanidad ha renovado su modo de pensar transformándose en un creador de talentos y tecnologías, logrando connotados avances ingenieriles en el campo de la técnica y sus derivados. Pero no fue hasta el siglo XX que creó la computadora, tecnología que utiliza constantemente, modificando su forma de actuar ante los problemas que se le presentan. Hoy en la era de las Tecnologías de la Información y las Comunicaciones (TIC), la información fluye sin fronteras, pues cada día es necesario que grandes volúmenes de datos puedan ser almacenados y gestionados.

Con la gradual Informatización de la Sociedad (IS), la información se ha vuelto un pilar de la humanidad, la manera en que se utiliza y se almacena se ha convertido en parte esencial de los procesos de toda entidad. La informatización no solo ahorra tiempo y esfuerzo en los procesos de las entidades, también con su uso adecuado, puede aportar grandes ganancias a estas, de manera que es de gran importancia contar con software especializado para la gestión de la información en dependencia del negocio y sus procesos. (1)

Actualmente en Cuba existen empresas e instituciones tales como LABIOFAM, la Universidad de las Ciencias Informáticas (UCI), la Universidad de Pinar del Rio, Universidad de Holguín y el grupo de Electrónica para el Turismo[1] que gestionan numerosos recursos de hardware y software, por lo que se hace necesario llevar un control de los mismos. El control de la información del hardware y software de estas entidades se realiza de forma manual, resultando engorroso mantenerla actualizada. Resulta trabajoso además, detectar cualquier alteración no autorizada de los componentes físicos de la computadora. Para dar respuesta a esta situación, la Universidad de las Ciencias Informáticas se dió a la tarea de desarrollar un sistema que permita la gestión de los recursos de hardware y software de una red de computadoras. En coordinación con el Centro de Telemática (TLM) de dicha universidad, deciden crear el Sistema Gestor de Recursos de Hardware y Software (GRHS) que responde a dicha necesidad.

GRHS, es una herramienta informática que permite realizar inventario del hardware y software instalado en una red de computadoras. Actualmente la corrección de las deficiencias detectadas durante el proceso de instalación de GRHS se realiza de forma manual, teniendo

[1] Información suministrada por la asesora de mercadotecnia del Centro Telemática de la UCI.

que dirigirse el administrador a cada cliente para realizar los cambios u orientar y supervisar que cada uno de los usuarios los realice; entre los cambios a realizar puede encontrarse la modificación de la información de los ficheros de configuración del sistema GRHS en el cliente y la correcta configuración de actualizaciones automáticas ya sea del Sistema Operativo u otro software.

La instalación de aplicaciones necesarias en entornos con muchas terminales provoca que el administrador deba ir por cada una de ellas configurando los parámetros necesarios para el correcto funcionamiento de la aplicación. Además en ocasiones puede ser necesario realizar tareas en el cliente a través de Gclient[2] pero GRHS no cuenta con la funcionalidad requerida lo que implica esperar a una actualización del sistema.

El hecho de realizar las tareas anteriores de forma manual trae como consecuencia la pérdida de tiempo en tareas no productivas por parte de los usuarios, y supone una gran carga en esfuerzo para al administrador, desviando su atención de tareas más apremiantes

En vista a las demandas de los directivos a cargo del desarrollo de GRHS para lograr una óptima y adecuada administración, se deben buscar alternativas y estrategias que permitan ir más allá de la administración tradicional haciendo uso de la ciencia y tecnología para obtener nuevos métodos y técnicas para mejorar el proceso de ejecución de tareas.

Considerando la situación anteriormente mencionada se identifica como **problema a resolver:** ¿Cómo ejecutar tareas de forma remota en los clientes de GRHS?

El **objeto de estudio** de la investigación lo constituye: El proceso de ejecución de tareas de forma remota mediante scripts.

Como **objetivo general**, para brindar solución al problema planteado se ha definido: Desarrollar un módulo para la ejecución automática de tareas mediante scripts de forma remota.

El **campo de acción** a partir del objeto de estudio comprende: El proceso de ejecución de scripts de forma remota en los clientes de GRHS.

[2] Aplicación del Sistema GRHS que se encarga de recolectar la información de los ordenadores y enviarla al el Servidor de GRHS.

Para llevar a término el resultado de la investigación se expone como **idea a defender** que con el desarrollo de un módulo para la ejecución automática de scripts en los clientes de GRHS, se contará con una herramienta para la realización de tareas en los clientes de GRHS de forma remota.

Para dar cumplimento al objetivo general se trazan las siguientes **tareas de la investigación:** para dar cumplimento al objetivo principal:

1. Caracterización de los sistemas informáticos que utilicen la ejecución automática de scripts para el desarrollo de conocimientos.
2. Selección de patrones de diseño para realizar el análisis y diseño del módulo.
3. Selección de la metodología y herramientas para el desarrollo de un módulo que permita la ejecución de scripts en los clientes de GRHS.
4. Análisis de las pruebas unitarias y de aceptación a realizar al módulo para detectar errores e inconformidades.

En el desarrollo de la presente investigación se utilizarán un conjunto de **métodos científicos** que servirán de guía y proporcionarán un mejor entendimiento de la investigación. A continuación se explica en detalles el porqué de su selección.

Dentro de los métodos teóricos se encuentran:
Analítico-Sintético: Posibilitará el análisis del proceso de ejecución automática de scripts para determinar con exactitud cómo este deberá funcionar en GRHS.

Los **métodos empíricos** utilizados para obtener información sobre el objeto de estudio son:
Modelación: Se describirá el sistema a realizar a través de las historias de usuario (HU), además se representarán las clases y sus responsabilidades mediante la confección de las tarjetas Clase – Responsabilidad – Colaborador (CRC).

Observación: Se utilizará para investigar los procesos externamente sin tener que llegar a la esencia de los mismos, lo que ayudará al planteamiento del problema a resolver, además de permitir conocer bien el proceso delimitado como objeto de estudio.

El documento está estructurado por cuatro capítulos, el contenido de los cuales se describe a continuación.

Capítulo 1: Fundamentación Teórica de la Investigación, se realiza un estudio de las tendencias actuales de los sistemas que ejecutan scripts. Se detallan las tecnologías, metodologías, herramientas y lenguajes de programación a utilizar en el desarrollo del módulo, definidas por la dirección del proyecto.

Capítulo 2: Características del Módulo, Ejecución y Planificación, se analiza una propuesta de solución del sistema, se describen los requisitos funcionales y las propiedades del módulo así como el entorno de trabajo en el que se desarrollará el módulo, se plantean las Historias de Usuarios, el Plan de iteraciones y su duración. Además se confecciona el Plan de entregas.

Capítulo 3: Análisis y Diseño del módulo.

En este capítulo se define la arquitectura del sistema, así como los patrones de diseño empleados. Además se procede a construir las Tarjetas CRC.

Capítulo 4: Implementación y Pruebas.

Este capítulo aborda la implementación de las HU definidas y tareas de ingeniería en cada iteración para luego hacer una revisión de las mismas mediante las pruebas.

Capítulo 1: Fundamentación Teórica de la Investigación

1.1. Introducción

En el siguiente capítulo se realiza un estudio del arte de los diferentes sistemas de ejecución automática de scripts que existen en la actualidad. Se describen además las características del lenguaje de programación, sistema gestor de base de datos, metodología de desarrollo y otras herramientas definidas por la dirección del centro para dar solución al problema.

1.2. Definiciones

Gestión

*"Hace referencia a la acción y a la consecuencia de administrar o gestionar algo**,** gestionar es llevar a cabo diligencias que hacen posible la realización de una operación comercial o de un anhelo cualquiera."* (2)

Automatización

"La automatización es una tecnología relacionada con la aplicación de sistemas mecánicos, electrónicos y basados en computadora para ejecutar y controlar la producción sustituyendo operadores humanos." (3)

Script

"*Un script es un guión o conjunto de instrucciones. Permiten la automatización de tareas creando pequeñas utilidades. Es muy utilizado para la administración de sistemas UNIX. Son ejecutados por un intérprete de línea de comandos."* (4)

1.3. Sistema Gestión de Recursos de Hardware y Software.

El Sistema de Gestión de Recursos de Hardware y Software (GRHS) está compuesto por varias aplicaciones, un cliente (Gclient) o colector, un servidor (Gserver) y una consola de administración (Gadmin). El cliente posee varios módulos para inventariar software y hardware para los sistemas operativos Windows y la familia GNU/Linux. Tiene un módulo para la detección de las incidencias y otro para ejecutar acciones que se toman ante incidencias. Todos los módulos de inventarios notifican los cambios al módulo de incidencias y este las detecta y comunica al módulo de acciones.

El servidor (Gserver) se encarga de la integración de toda la información de una red de computadoras, así como su configuración y la notificación de alarmas. Está basado en el marco de trabajo Django que posee varias aplicaciones para la gestión de las configuraciones,

inventarios y las acciones. La comunicación entre el cliente y el servidor se realiza mediante los protocolos Hypertext Transfer Protocol[3] (HTTP) y Hypertext Transfer Protocol Secure (HTTPS). Se usa la tecnología REST[4] y el formato JSON[5] para el intercambio de información mediante la aplicación djangorestframework en el servidor. Ante las incidencias puede notificarlas mediante correos electrónicos.

La consola de administración permite gestionar períodos de cambios donde el administrador autorizara un tiempo determinado para que los técnicos o personas responsables del mantenimiento de los ordenadores puedan realizar cambios de hardware o software sin que el sistema los detecte como incidencias. Posibilita además la gestión de reportes donde por determinados criterios de búsqueda los usuarios del sistema pueden obtener información, guardarlas como reportes y posteriormente exportarlas al formato Portable Document Format (PDF).

1.4. Sistemas similares

Actualmente existen disímiles herramientas cuyo principal objetivo es la ejecución automática de scripts en las computadoras de una red, ya sea tanto en el servidor como en los clientes. A continuación se exponen las distintas soluciones estudiadas en el ámbito internacional y nacional que ayudarán a la implementación de dicho módulo.

1.4.1 Ámbito Internacional

Windows PowerShell

"*Windows PowerShell es una interfaz de consola (CLI) con posibilidad de escritura y unión de comandos por medio de instrucciones (scripts en inglés). Es mucho más rica e interactiva que sus predecesores, desde DOS hasta Windows 7. Esta interfaz de consola está diseñada para su uso por parte de administradores de sistemas, con el propósito de automatizar tareas o realizarlas de forma más controlada. PowerShell no sólo permite interactuar con la máquina en que se encuentra sino que interactúa de forma remota con otros computadores. La comunicación remota de Windows PowerShell, que utiliza el protocolo WS-Management,*

[3] HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. desarrollado por el World Wide Web Consortium y la Internet Engineering Task Force

[4] Técnica de arquitectura de software para sistemas hipermedia distribuidos como la World Wide Web.

[5] Acrónimo de JavaScript Object, es un formato ligero para el intercambio de datos, es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

permite ejecutar cualquier comando de Windows PowerShell en uno o muchos equipos remotos. Permite establecer conexiones persistentes, iniciar sesiones interactivas 1:1 y ejecutar scripts en varios equipos." (5)

Saweso

"*Saweso (Sistema de Automatización de Workarounds[6] y Ejecución de Scripts Ocasionales) es una herramienta diseñada controlar la ejecución de comandos y scripts sobre servidores (Debian o Ubuntu) y equipos clientes (Debian o Ubuntu) y que permite, la descarga de los resultados obtenidos, el control del estado de la ejecución y la generación de estadísticas e informes.*

Utiliza cliente SSH y certificado "dsa"[7], para el acceso seguro a los servidores de la plataforma. SSH Tunneling y certificado "rsa"[8], para el acceso a los equipos clientes que se encuentran dentro de una intranet remota dotada de un servidor de seguridad. Shell Scripting, para diseñar el motor del sistema que realizara la copia remota ("scp") y ejecución ("bash") de los scripts necesarios.

Saweso es una herramienta desarrollada por José Antonio Serrano para la plataforma GNU/Linux en el año 2013 como tesis de grado en la Universidad de Oberta Catalunya." (6)

1.4.2 Ámbito Nacional

Remote Execute Scripts and Command (Resc)

"Es un programa que permite ejecutar diferentes comandos y scripts en ordenadores clientes de manera remota. Esta aplicación ha sido desarrollada en Linux con C++[9], Sockets[10], Hilos y Semáforos[11].

---

[6] Método temporal para alcanzar una solución cuando el camino tradicional no funciona. En informática se usa para superar inconvenientes de programación, hardware o comunicación.

7 Digital Signature Algorithm, en español Algoritmo de Firma digital es un estándar del Gobierno Federal de los Estados Unidos de América o FIPS para firmas digitales.

[8] RSA(Rivest, Shamir y Adleman), es un sistema criptográfico de clave pública.

[9] C++ es un lenguaje de programación orientado a objetos que toma la base del lenguaje C y le agrega la capacidad de abstraer tipos como en Smalltalk.

[10] Un **socket** (enchufe), es un método para la comunicación entre un programa del cliente y un programa del servidor en una red. Un socket se define como el punto final en una conexión.

[11] Son una estructura diseñada para sincronizar dos o más threads o procesos, de modo que su ejecución se realice de forma ordenada y sin conflictos entre ellos.

RESC (Remote Execute Scripts and Command) surge como una necesidad para una tesis de grado en la cual era necesario controlar una serie de computadoras por la red y poder ejecutar comandos y scripts de manera remota en cada una de ellas. Posteriormente se extiende la instalación de RESC en su versión 0.1 (inestable) a todos los laboratorios docentes de la antigua facultad 10 de la Universidad de las Ciencias Informáticas (UCI) con la intención de realizar pruebas. Más tarde, tras una serie de pruebas se realizan cambios para mejorar la versión actual y sale la versión 0.2 (estable). RESC es un software que fue realizado en el proyecto Unicornios en la Universidad de las Ciencias Informáticas y su desarrollador es Dayron Pérez Roldan." (7)

1.4.3 Conclusiones del Estado del Arte

El estudio realizado permitió conocer que la herramienta Windows PowerShell no puede integrarse a GRHS y así suplir la necesidad de este sistema de automatizar determinadas tareas. PowerShell posee licencia comercial perteneciente a Microsoft Corporations, compañía radicada en los Estados Unidos, por lo que su utilización violaría la política de migración a software libre en Cuba, que promueve el uso de software diseñado y desarrollado en el país como vía para garantizar la soberanía tecnológica.

Por su parte Saweso es compatible solo con Debian y Ubuntu, por lo cual limita su posible integración con GRHS, sistema que debe ser capaz de funcionar además en los sistemas operativos Nova, Windows 7 y Windows 8. Mientras que RESC es una herramienta que no cuenta con soporte técnico y su uso se reduce a ambientes académicos, por lo que no se recomienda su empleo o integración con aplicaciones que se usarán en ámbitos empresariales, como es el caso de GRHS. El sistema RESC también está construido sobre la plataforma C++, lo cual no permite una integración directa con el GRHS.

Se decide entonces desarrollar un módulo propio para GRHS que permita la ejecución automática de scripts de forma remota. Se utilizaran algunos de los protocolos que emplean los sistemas antes mencionados en la implementación del módulo: Bash (Bourne again shell) para el caso de la plataforma Linux y Batch (procesamiento por lotes) para el caso de la plataforma Windows.

1.5 Metodología de desarrollo de software.

El desarrollo de software no es una tarea fácil. Como solución a este problema ha surgido una alternativa: la Metodología de desarrollo de software. Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible e eficiente,

esto lo logran desarrollando un proceso detallado con un fuerte énfasis en planificar basado por otras disciplinas de la ingeniería.

Fue definido el empleo de la metodología ágil XP para el desarrollo del módulo dado que sus características fundamentales consisten en que posee un equipo de desarrollo pequeño, se necesita estar en constante comunicación con el cliente y se cuenta con poco tiempo para su implementación. Esta metodología está centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software. Plantea que las tareas de desarrollo las realicen de 1 a 2 personas, se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo garantizando una comunicación fluida entre todos los participantes. "*XP brinda simplicidad en las soluciones implementadas y es flexible a la hora de enfrentar los cambios"*. (8)

1.6 Lenguaje de Modelado Business Process Modeling Notation (BPMN).

Fue seleccionado el lenguaje de modelado BPMN para el modelado del negocio del módulo a implementar. *"BPMN es una notación gráfica que describe la lógica de los pasos de un proceso de Negocio. Esta notación ha sido especialmente diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades."* (9) BPMN proporciona un lenguaje común para que las partes involucradas puedan comunicar los procesos de forma clara, completa y eficiente. De esta forma BPMN define la notación y semántica de un Diagrama de Procesos de Negocio (Business Process Diagram, BPD). BPD es un diagrama diseñado para representar gráficamente la secuencia de todas las actividades que ocurren durante un proceso, basado en la técnica de "Flow Chart", incluye además toda la información que se considera necesaria para el análisis. BPD es un diagrama diseñado para ser usado por los analistas, quienes diseñan, controlan y gestionan procesos. "*Dentro de un Diagrama de Procesos de Negocio BPD se utiliza un conjunto de elementos gráficos, agrupados en categorías, que permite el fácil desarrollo de diagramas simples y de fácil comprensión, pero que a su vez manejan la complejidad inherente a los procesos de negocio*". (9)

BPMN brinda las siguientes ventajas:

- *Es un estándar internacional de modelado de procesos aceptado por la comunidad.*
- *Es independiente de cualquier metodología de modelado de procesos.*
- *Crea un puente estandarizado para disminuir la brecha entre los procesos de negocio y la implementación de estos.*

- *Permite modelar los procesos de una manera unificada y estandarizada permitiendo un entendimiento a todas las personas de una organización.* (10)

1.7 Herramienta CASE para el modelado.

Se seleccionó como Herramienta CASE Visual Paradigm. Porque es una herramienta de modelado que utiliza el lenguaje de modelado UML y BPMN, permite la generación de códigos e ingeniería inversa. Con una clase de diseño bien especificada, Visual Paradigm puede generar código hasta en quince lenguajes de programación entre ellos Python. Actualmente cumple con las políticas de migración a Software Libre en Cuba, es una herramienta multiplataforma que se puede utilizar tanto en Linux como en Windows y la universidad cuenta con licencia para su uso. Su interfaz es muy intuitiva, de fácil aprendizaje para los desarrolladores.

Visual Paradigm ofrece:

- *Entorno de creación de diagramas para BPMN.*
- *Uso de un lenguaje estándar a todo el equipo de desarrollo facilitando la comunicación.*
- *Capacidad de ingeniería directa (versión profesional) e inversa.*
- *Un modelo y código que se mantiene sincronizado en todo el ciclo de desarrollo.*
- *Múltiples versiones, para cada necesidad.*
- *La posibilidad de integrarse con el IDE de desarrollo.* (11)

1.8 Lenguajes de programación para el desarrollo.

La elección de un lenguaje de programación que sea capaz de adaptarse a las necesidades del producto que se quiera desarrollar de forma eficiente, alcanzando cada una de las metas que se vayan a trazar durante el desarrollo del sistema, no siempre es sencillo, pues existe la inseguridad en cuanto a si un determinado lenguaje es mejor que otro en lo referente a características y calidad. En la actualidad no existe ningún lenguaje perfecto, debido a que cada uno cumple con un determinado propósito. Indicando que la elección se debe limitar a escoger aquel lenguaje que se asemeje más a las características propias del sistema que se quiera implementar. Se definió Python 2.7 como lenguaje del lado del servidor y HTML5, CSS3 y JavaScript como lenguajes del lado del cliente, ya que el sistema GRHS para el cual será desarrollado el módulo emplea estos lenguajes de programación por lo que su uso ayudaría a hacerlos compatibles una vez se hayan integrado. A continuación una breve descripción de estos lenguajes.

Python v2.7

El uso del lenguaje de programación Python v2.7 permite que el código de la aplicación sea más corto que su equivalente en otros lenguajes de programación y con el tipado dinámico[12] se puede manejar con mayor facilidad el cúmulo de información que se gestiona. Posee extensas bibliotecas estándar y módulos de terceros para prácticamente todas las tareas. Además la organización de los bloques de código hace más sencilla la lectura y comprensión de los mismos. Facilita la realización de pruebas, detecta muchos de los errores de programación que escapan al control de los compiladores y proporciona información muy rica para detectarlos y corregirlos. Posee un rico juego de estructuras de datos que se pueden manipular de modo sencillo. *"Una ventaja fundamental que presenta dicho lenguaje es la gratuidad de su intérprete. Su intérprete tiene versiones para prácticamente cualquier plataforma ya sea sistema operativo Linux o Microsoft Windows"*. (12)

Hypertext Markup Language (HTML)

"*HTML es un lenguaje de composición de documentos y especificación de ligas de hipertexto que define la sintaxis y coloca instrucciones especiales que no muestra el navegador, aunque si le indica cómo desplegar el contenido del documento a este, incluyendo texto, imágenes y otros medios soportados. HTML también le indica cómo hacer un documento interactivo a través de ligas especiales de hipertexto, las cuales conectan diferentes documentos ya sea en su computadora o en otras*". (13) Será usado para la creación de la interfaz de administración del módulo en conjunto con el lenguaje CSS utilizando etiquetas o marcas propias del lenguaje.

JavaScript

JavaScript es un lenguaje de programación interpretado (que no requiere compilación) del lado del cliente ya que el navegador es el que soporta la carga de procesamiento. Gracias a que es un lenguaje muy versátil y potente, tanto para la realización de pequeñas tareas como para la gestión de complejas aplicaciones y a su compatibilidad con la mayoría de los navegadores modernos, será utilizado para controlar la apariencia y manipular los eventos dentro de la ventana del navegador Web, así como para validar datos de entrada en las interfaces de las aplicaciones. (14)

[12] Un lenguaje de programación es dinámicamente tipado si una misma variable puede tomar valores de distinto tipo en distintos momentos. La mayoría de lenguajes de tipado dinámico son lenguajes interpretados, como Python.

Cascading style sheet[13] (CSS)

CSS es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML. CSS es una forma de separar los contenidos y su presentación y es imprescindible para la creación de páginas web complejas. Si el lenguaje HTML se utiliza para marcar los contenidos, es decir, para designar lo que es un párrafo, lo que es un titular o lo que es una lista de elementos, "*el lenguaje CSS se utiliza para definir el aspecto de todos los contenidos, es decir, el color, tamaño y tipo de letra de los párrafos de texto, la separación entre titulares y párrafos, la tabulación con la que se muestran los elementos de una lista, etc*". (13)

Su uso proveerá las siguientes ventajas:

- Control centralizado de la página web con lo que se agiliza la actualización de la misma.
- Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web, con lo que aumenta considerablemente la accesibilidad.
- Una página puede disponer de varias hojas de estilos según el dispositivo que la muestre.
- El documento HTML se vuelve más claro de entender y disminuye considerablemente su tamaño.

1.9 Marcos de trabajo.

Un marco de trabajo o framework no es más que una estructura de soporte definida, mediante la cual un proyecto de software puede ser desarrollado y organizado. Pueden incluir soporte de programas, bibliotecas y lenguajes interpretados entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. Son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional. Los siguientes marcos de trabajo fueron definidos por el centro para el desarrollo de la solución propuesta.

Django 1.4

El crecimiento de Python es cada vez mayor y esto se ha hecho más notorio en los últimos años, con la aparición de herramientas que hacen el trabajo más simple y eficiente con este

[13] Cascading Style Sheets, hojas de estilo en cascada por su traducción al español.

lenguaje de programación. Una de esas herramientas es Django, un framework hecho en Python. Django promueve el desarrollo rápido, se construyen aplicaciones en cuestión de días y con el conocimiento suficiente esos días se pueden reducir a horas. "*Django impulsa el desarrollo de código limpio al promover buenas prácticas de desarrollo web, sigue el principio DRY (conocido también como Una vez y sólo una) para la reducción de código duplicado. Usa una modificación de la arquitectura Modelo-Vista-Controlador (MVC), llamada MTV (Model – Template – View), que sería Modelo-Plantilla-Vista, está forma de trabajar permite que sea pragmático.*" (12)

Ventajas:

- Posee un mapeado objeto-relacional[14].
- Un sistema incorporado de vistas genéricas.
- Soporte de internacionalización, incluyendo traducciones incorporadas de la interfaz de administración.
- Un sistema extensible de plantillas basado en etiquetas, con herencia de plantillas.
- Un sistema middleware[15] para desarrollar características adicionales.
- Documentación incorporada accesible a través de la aplicación administrativa.

Backbone.js

Backbone.js es un marco de trabajo para la programación del lado el cliente, que ofrece una estructura para aplicaciones que hace uso intensivo de JavaScript y que provee de modelos llave / valor enlazables mediante eventos, colecciones con la posibilidad de realizar diferentes utilidades por medio de una API[16], vistas con manejadores de eventos declarativos y conexión a interfaces REST nuevas o ya disponibles anteriormente. (15)

De manera resumida, las facilidades de la librería Backbone son las siguientes:

- Permite la programación atendiendo al paradigma MVC.
- Ayuda a crear estructuras bien definidas para los datos de la aplicación y facilita la creación de eventos cuando los datos cambian.

[14] Técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia.

[15] Software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas.

[16] API del inglés: *Application Programming Interface*, es el conjunto de subrutinas, funciones y que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

- Simplifica el uso de vistas, que te ayudan a renderizar interfaces de usuario en la página.

1.10 Entorno Integrado de Desarrollo (IDE).

Un Entorno Integrado de Desarrollo (IDE de su significado en inglés Integrated Development Environment), es un conjunto de herramientas que ha sido creado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Un IDE provee un marco de trabajo amigable para la mayoría de los lenguajes de programación y en algunos casos puede funcionar como un sistema en tiempo de ejecución en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto.

Sublime Text es un editor de texto sin formato desarrollado en Python, que ofrece una serie de características y facilidades con tal de simplificar el trabajo a aquellos desarrolladores que utilicen un bloc de notas para programar. Este programa tiene tanto una versión de pago como una gratuita, pero la versión gratuita tiene todas las funciones, simplemente cada un número determinado de guardados aparecerá un cuadro de diálogo para que compremos la versión completa. Son muchas las características que hacen de Sublime Text un IDE a considerar por cualquier programador, algunas de ellas son:

- *Atajos de teclado que permiten acceder muy rápidamente a cualquier complemento del programa o realizar cualquier operación.*
- *Resaltado de sintaxis configurable: El marcado de sintaxis es completamente configurable a través de archivos de configuración del usuario.*
- *Minimapa: crea una previa visualización de la estructura del código.*
- *Multi Cursor: permite crear varios cursores con los que se puede escribir texto en diferentes posiciones del archivo simultáneamente.*
- *Interfaz intuitiva y sencilla.*
- *Multiplataforma***.** (16)

1.11 Sistema Gestor de Base de Datos (SGBD).

"*Los sistemas que gestionan datos, así como las transacciones entre estos, constituyen una potente herramienta, la cual está estrechamente vinculada a sistemas de software robustos que necesiten de la persistencia, localización y explotación de un volumen amplio de datos. El propósito general de los gestores de base de datos es el de manejar de manera clara, sencilla*

y ordenada un conjunto de datos que posteriormente se convertirán en información relevante, para un buen manejo de datos." (17)

En la actualidad existe gran cantidad de gestores, cada uno con características distintas, así como ventajas y desventajas a la hora de ser usado. Para la realización de este trabajo la fue determinado usar PostgreSQL ya que es el gestor que utiliza el sistema GRHS, al cual va a estar integrado la solución propuesta.

Postgre SQL.

Es un sistema de gestión de bases de datos objeto-relacional basado en el proyecto Ingres, de la universidad de Berkeley, California. Pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, PostgreSQL incluye características de la orientación a objetos como pueden ser: la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional, además de otras específicas del gestor, como lo son: un mejor soporte para sub-selects, triggers, vistas y procedimientos almacenados. Entre sus características más específicas se encuentran:

- *Implementación del estándar SQL92/SQL99.*
- *Soporte para distintos tipos de datos, como son: datos de tipo fecha, datos sobre redes (MAC, IP), cadenas de bits, etc., así como la creación de tipos de datos propios.*
- *Al igual que con los tipos de datos, PostgreSQL permite la declaración de funciones propias.*
- Soporte en la mayoría de los sistemas operativos más utilizados incluyendo, Linux, varias versiones de UNIX, BeOS y Windows. (18)

1.12 Conclusiones

En este capítulo se realizó un estudio relacionado con los sistemas de ejecución automática de scripts, dando una panorámica general de cómo estos se implementan hoy en día, se pusieron ejemplos de los mismos, lo que tributó a la adquisición de conocimientos e ideas para el desarrollo del sistema. De las soluciones estudiadas se concluyó que ninguna se ajusta a las necesidades del sistema GRHS, por lo que se implementará un módulo propio para este. No obstante serán utilizadas algunas características de los sistemas estudiados para la creación del nuevo módulo. También se definieron y explicaron aspectos relacionados con las herramientas y tecnologías que se utilizarán en la construcción de la aplicación. La metodología de desarrollo a emplear será XP. Para la implementación de las funcionalidades del módulo el lenguaje de programación Python.

Capítulo 2: Características del Módulo, Ejecución y Planificación

2.1. Introducción

En el presente capítulo se realiza un análisis detallado de los aspectos del sistema con el objetivo de comprender las especificidades de lo que permitirá hacer el mismo. Además cumpliendo con la metodología seleccionada (XP) comenzará la fase de exploración donde se presentarán las principales HU y la fase de planificación presentando la estimación del esfuerzo necesario para el desarrollo de las HU.

2.2. Propuesta del módulo

GRHS es un sistema de soluciones informáticas que permite gestionar y auditar activos informáticos (Hardware y Software). En este sistema el administrador debe realizar las configuraciones y corregir lo detectado por las auditorías de forma manual. Se propone la integración al sistema de un módulo de ejecución de scripts al que podrá acceder el administrador. Este módulo estará provisto de una interfaz donde el administrador podrá cargar scripts que serán almacenados en una base de datos junto a algunos datos relacionados como fecha de creación, descripción del script, sistema operativo para el que fue creado el script y el contenido del script que se almacenara en la base de datos en formato string base64 para mayor seguridad y mejor acceso. Además el administrador será capaz de enviar las peticiones de ejecución al cliente siempre que este activo, seleccionando el script que desea enviar. También podrá gestionar las respuestas proporcionadas por el cliente a cada uno de los scripts ejecutados y programar los scripts para ser ejecutados en un momento específico. En la siguiente ilustración se muestra el comportamiento del proceso principal del módulo.

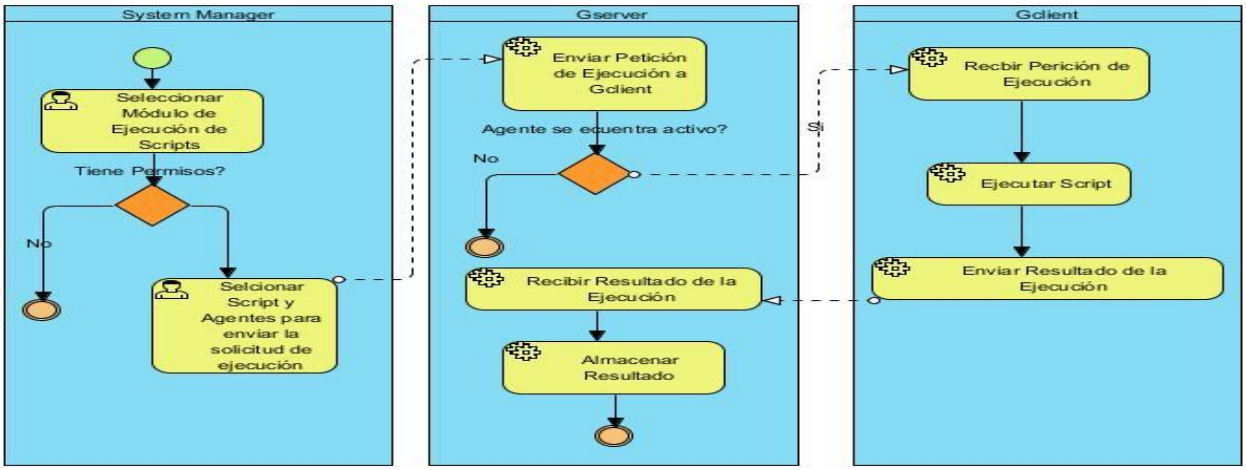


Ilustración 1 Propuesta del sistema.

2.3 Funcionalidades del módulo

El Módulo implementado en Gadmin debe ser capaz de:

1. Gestionar Scripts.
    a. Adicionar Script.
    b. Modificar Script.
    c. Eliminar Script.
    d. Listar Script.
    e. Buscar Script.
2. Mostrar resultados de la ejecución de scripts.
    a. Listar resultados.
    b. Mostrar resultados.
    c. Buscar resultados.
3. Enviar petición de ejecución de scripts.
    a. Enviar petición a cliente específico.
    b. Enviar petición a Subred.
    c. Enviar petición a más de un ordenador.
4. Mostrar estadísticas sobre los scripts almacenados en el sistema.
5. Exportar a hojas de cálculo las respuestas de las ejecuciones.
6. Verificar compatibilidad con el SO.
7. Programar la ejecución de scripts.

El plugin implementado en Gclient debe ser capaz de:

8. Recibir petición del servidor.
9. Ejecutar Script.
10. Enviar resultado de la ejecución.

2.4 Propiedades del módulo

Es necesario que el módulo para la ejecución automática de scripts en los clientes de GRHS cumpla con determinadas cualidades y propiedades; por estas razones se describen a continuación, algunas de las características necesarias para el funcionamiento del producto final.

2.4.1 Usabilidad

El manejo del módulo debe ser sencillo e intuitivo empleando iconos que sean de rápida identificación con su funcionalidad. La navegabilidad no debe ser muy compleja, todas las funcionalidades deben ser rápidamente accesibles por el usuario.

2.4.2 Software

Para el cliente: se usará un sistema operativo:

-Windows XP

-Windows 7

-Windows 8

-Ubuntu 12.04,13.10,14.04,14.10

-Debian 6 y 7

-Nova

Para el servidor: se usará un sistema operativo:

-Ubuntu 12.04,13.10,14.04,14.10

-Debian 6 o 7

-Nova

- El servidor del sistema debe tener instalado Python v2.7.
- Gestor de base de datos: PostgreSQL.

Se recomienda el uso del navegador Mozilla Firefox en su versión 25 o superior para acceder a las interfaces del sistema.

2.4.3 Interfaz de usuario

El módulo deberá contar con las pautas de diseño de la UCI así como con las de GRHS, para su integración con dicho sistema.

2.4.4 Seguridad

Políticas de seguridad por usuarios y roles, el módulo debe contar con un grupo de políticas de accesibilidad a las diferentes funcionalidades del mismo en dependencia del nivel de autorización que presente un usuario determinado. El módulo garantizará la autenticación como primera acción

2.5 Fase de Exploración

"*En esta fase se define el alcance general del proyecto, el cliente define lo que necesita mediante la redacción de historias de usuarios. Los desarrolladores estiman los tiempos de implementación en base a esta información*." (20) Las estimaciones en esta fase son primarias

y pueden variar cuando se analicen más en detalle en cada iteración. Esta fase dura típicamente 2 semanas, y el resultado es una visión general del sistema, y un plazo total estimado.

2.5.1 Historias de Usuario

"*Las HU son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tablas en las cuales el cliente describe brevemente las características que el sistema debe poseer. El tratamiento de las HU es muy dinámico y flexible, en cualquier momento las historias de usuario pueden reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas*." (21) Las HU deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia. Además deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo.

Las HU se clasifican de la siguiente manera:

1. Teniendo en cuenta la Escala Nominal de Prioridad en el Negocio:
   - ✓ **Alta:** se le otorga a las HU que constituyen funcionalidades de vital importancia en el desarrollo del proyecto.
   - ✓ **Media:** se le otorga a las HU que para el cliente constituyen funcionalidades a tener en cuenta sin que tengan una afectación directa sobre el proyecto que se está desarrollando.
   - ✓ **Baja:** se le otorga a las HU que constituyen funcionalidades que sirven de ayuda al control de la estructura y que no tienen nada que ver con el proyecto en desarrollo.
2. Teniendo en cuenta la Escala Nominal de Riesgo en Desarrollo:
   - ✓ **Alta:** se otorga cuando para la implementación de la HU se considera la posible existencia de errores que lleven a la inoperatividad del código.
   - ✓ **Media:** se otorga cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión.
   - ✓ **Baja:** se otorga cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto.

A continuación se muestran algunas de las HU que presentan alta prioridad en el negocio.

Las HU estarán constituidas mediante tablas divididas por las siguientes secciones:

- ➢ **Número:** número de la historia de usuario incremental en el tiempo.

- **Nombre de Historia de Usuario:** el nombre de la historia de usuario para identificarlas mejor entre los desarrolladores y el cliente.
- **Usuario:** involucrados en la ejecución de la HU.
- **Iteración Asignada:** número de la iteración.
- **Prioridad en negocio**: Muy Alta, Alta, Media o Baja.
- **Riesgo en Desarrollo*:*** Alta, Media o Baja.
- **Programador responsable:** involucrados en el desarrollo de la HU.
- **Puntos estimados**: tiempo estimado que se demorará el desarrollo de la HU, debe ser menor que tres semanas.
- **Descripción**: breve descripción de la HU.
- **Observaciones**: señalamiento o advertencia del sistema.
- **Prototipo de interfaz**: prototipo de interfaz si aplica.

Tabla 1 HU #1 Gestionar Script

Historia de Usuario	
Número: 1.	**Usuario:** Administrador.
Modificación de Historia de Usuario #: Ninguna	
Nombre de Historia de Usuario: Gestionar Script.	
Prioridad en negocio: Muy Alta.	**Riesgo de Desarrollo:** Bajo.
Puntos estimados: 1.0	**Iteración asignada:** 1.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Permite gestionar el script en el sistema, adicionando, modificando o eliminando el script creado con los siguientes datos: sistema operativo al que va dirigido, fecha de modificación, breve descripción del script , autor del script, quien lo modifico por última vez.	
Observaciones: El usuario debe encontrarse logueado en el sistema.	
Prototipo de interfaz:	

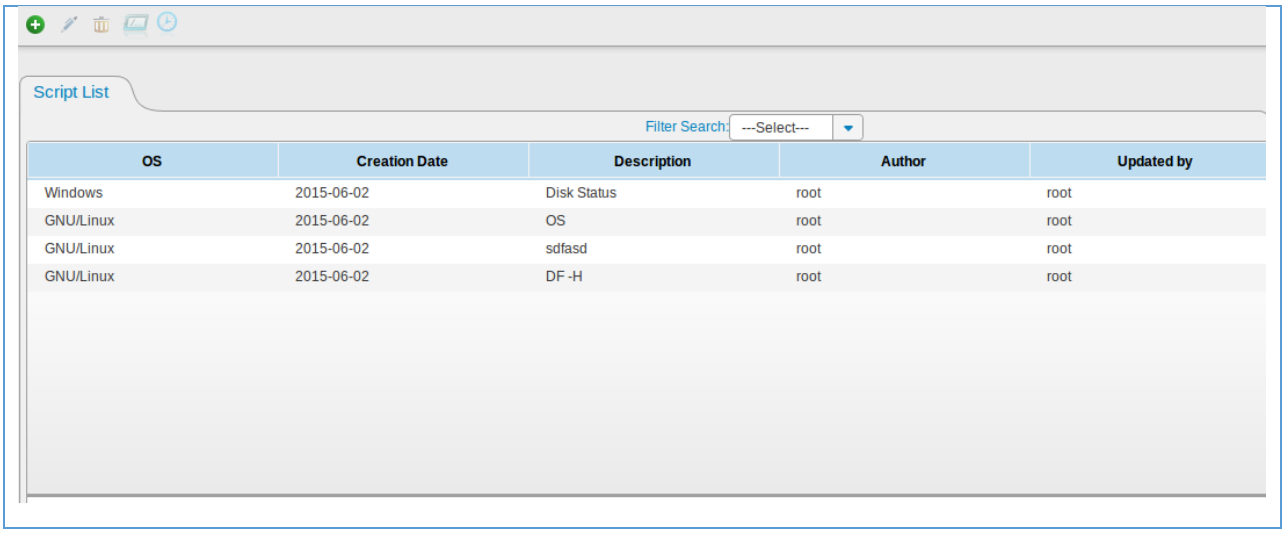


Tabla 2 HU #4 Mostrar Resultados de la ejecución.

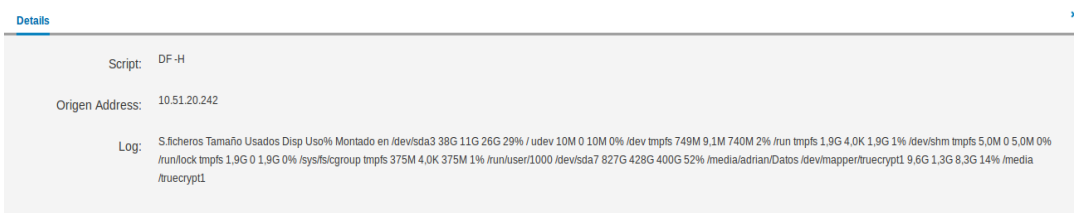
Historia de Usuario	
Número: 4.	**Usuario:** Administrador.
Modificación de Historia de Usuario #: Ninguna	
Nombre de Historia de Usuario: Mostrar resultados de la ejecución.	
Prioridad en negocio: Muy Alta.	**Riesgo de Desarrollo:** Bajo.
Puntos estimados: 0.6	**Iteración asignada:** 1.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Al ejecutarse un script en el cliente el sistema será capaz de mostrar un listado con los siguientes datos: fecha en que fue ejecutado el script, cliente en el que fue ejecutado, dirección ip del cliente donde fue ejecutado y descripción del script que fue ejecutado.	
Observaciones: El usuario debe encontrarse logueado en el sistema	
Prototipo de interfaz: 	

Tabla 3 HU # 6 Enviar petición a cliente específico.

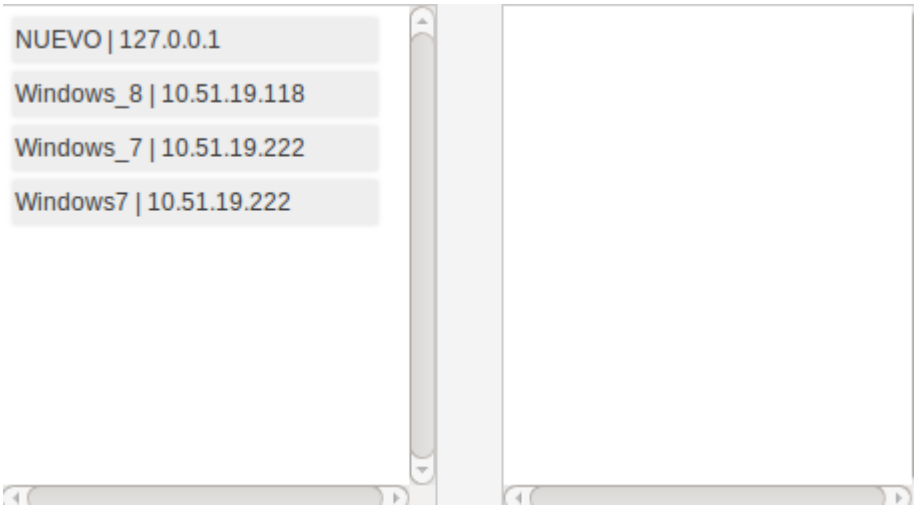
Historia de Usuario	
Número: 6.	**Usuario:** Administrador.
Modificación de Historia de Usuario #: Ninguna	
Nombre de Historia de Usuario: Enviar petición a cliente específico.	
Prioridad en negocio: Alta.	**Riesgo de Desarrollo:** Medio.
Puntos estimados: 0.8.	**Iteración asignada:** 2.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Permite seleccionar del listado de clientes a que cliente se le quiere mandar a ejecutar un script.	
Observaciones: El usuario debe encontrarse logueado en el sistema	
Prototipo de interfaz: 	

Tabla 4 HU # 7 Enviar petición a una Subred

Historia de Usuario	
Número: 7.	**Usuario:** Administrador.
Modificación de Historia de Usuario #: Ninguna	
Nombre de Historia de Usuario: Enviar petición a una Subred.	

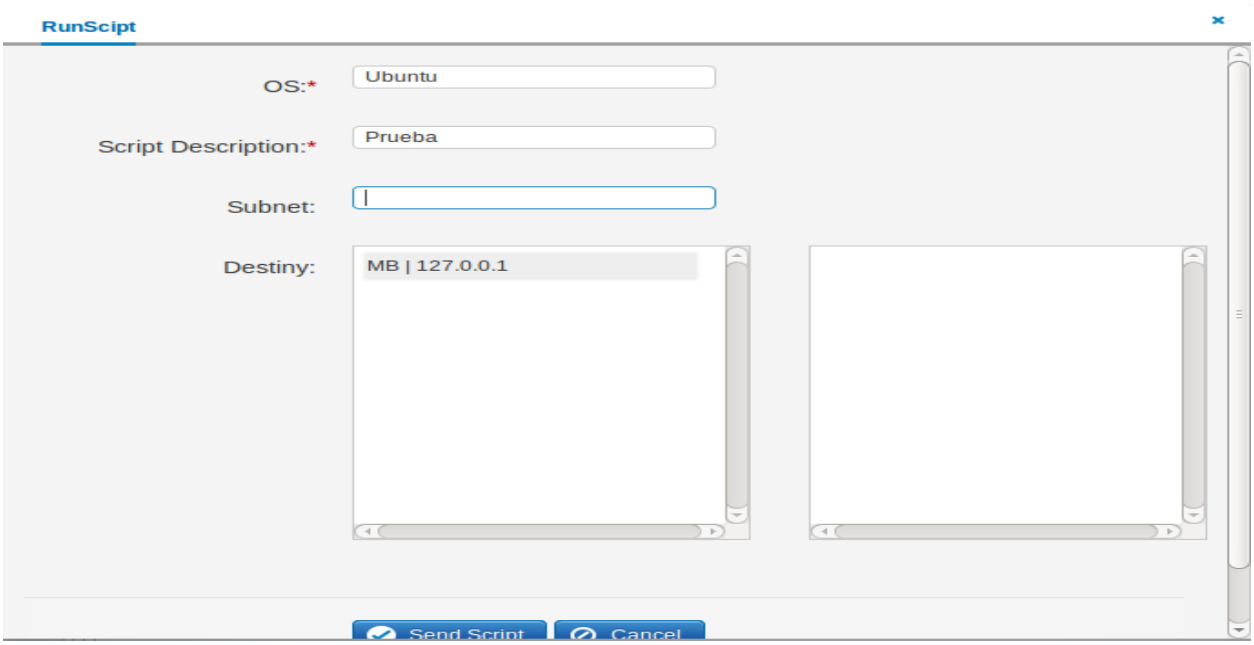
Prioridad en negocio: Alta.	**Riesgo de Desarrollo:** Medio.
Puntos estimados: 0.8.	**Iteración asignada:** 2.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Permite realizar él envió de un script a una subred especificada para su posterior ejecución.	
Observaciones: El usuario debe encontrarse logueado en el sistema	
Prototipo de interfaz: 	

Tabla 5 HU # 8 Enviar petición a más de un cliente

Historia de Usuario	
Número: 8.	**Usuario:** Administrador.
Modificación de Historia de Usuario #: Ninguna	
Nombre de Historia de Usuario: Enviar petición a más de un cliente.	
Prioridad en negocio: Media.	**Riesgo de Desarrollo:** Medio.
Puntos estimados: 0.8.	**Iteración asignada:** 2.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Permite realizar el envío de un script a varios ordenadores seleccionados para ser ejecutados.	

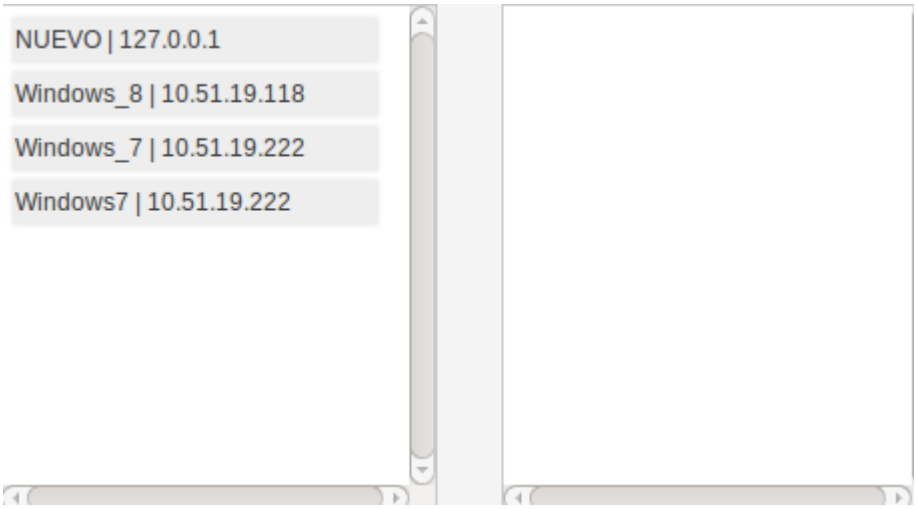
Observaciones: El usuario debe encontrarse logueado en el sistema
Prototipo de interfaz: 

Tabla 6 HU # 11 Recibir petición del servidor.

Historia de Usuario	
Número: 11.	**Usuario:** Administrador.
Modificación de Historia de Usuario #: Ninguna	
Nombre de Historia de Usuario: Recibir petición del servidor.	
Prioridad en negocio: Media.	**Riesgo de Desarrollo:** Medio.
Puntos estimados: 0.6.	**Iteración asignada:** 3.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Permite a Gclient recibir la información referente a una petición de ejecución junto con el script que se desea ejecutar.	
Observaciones:	
Prototipo de interfaz:	

Tabla 7 HU # 12 Ejecutar Script.

Historia de Usuario	
Número: 12.	**Usuario:** Administrador.
Modificación de Historia de Usuario #: Ninguna	
Nombre de Historia de Usuario: Ejecutar Script.	
Prioridad en negocio: Media.	**Riesgo de Desarrollo:** Alto.
Puntos estimados: 1.0.	**Iteración asignada:** 3.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Permite a Gclient a partir de una petición previamente recibida ejecutar un script y capturar el resultado de dicha ejecución.	
Observaciones:	
Prototipo de interfaz:	

2.6 Fase de Planificación

"*La metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores o gerentes. En esta fase el cliente establece la prioridad de cada HU, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Las estimaciones de esfuerzo asociadas a la implementación de las historias se establecen utilizando como medida el punto."* (22) Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

2.6.1 Estimación de esfuerzo por Historias de Usuario

La programación extrema basa sus procesos de planificación en estimaciones temporales de las historias de usuario, las cuáles deben ser realizadas por los desarrolladores durante las diversas reuniones de planificación. El objetivo principal de lo anterior es estimar cuánto tiempo llevará implementar las historias; todas las estimaciones que maneja la programación extrema se cuantifican en semanas de desarrollo ideal, lo cual no es más que la cantidad de trabajo que

se puede realizar durante una semana sin distracciones (llamadas telefónicas, reuniones, enfermedades, comidas, etc.), sin trabajos extras, sin la asignación de otras tareas, sin dependencias, y suponiendo que el desarrollador sabe exactamente lo que hay que hacer.

A continuación se muestra la estimación del esfuerzo por cada HU propuesta para el desarrollo del módulo:

Tabla 4 Estimación de esfuerzo por Historias de Usuario

Historias de Usuario	Puntos de Estimación
1. Gestionar Script	1.0
2. Buscar Script	0.8
3. Listar resultados	0.6
4. Mostrar resultados	0.6
5. Buscar resultados	0.6
6. Enviar petición a cliente específico	0.8
7. Enviar petición a Subred	0.8
8. Enviar petición a más de un cliente	0.8
9. Mostrar estadísticas sobre la ejecución de los scripts	0.6
10. Exportar a hojas de cálculo las respuestas	0.6
11. Recibir petición del servidor	0.6
12. Ejecutar Script	1.0
13. Enviar resultado de la ejecución al servidor	0.6
14. Verificar compatibilidad con el SO	0.6
15. Programar ejecución automática de script	1.6

2.6.2 Plan de Iteraciones

Luego de estimar el esfuerzo dedicado a cada HU, se procede a la planificación de la fase de implementación estableciendo una división de cuatro iteraciones.

Iteración 1.

En esta iteración se lleva a cabo el desarrollo de las HU del número 1 a la número 4, también se. Esto representa que al terminar la iteración se ha implementado un 26.6% de la aplicación.

Iteración 2.

En esta iteración se lleva a cabo el desarrollo de las HU del número 5 a la número 8. Esto representa que al terminar la iteración se ha implementado un 53.2% de la aplicación.

Iteración 3.

En esta iteración se lleva a cabo el desarrollo de las HU del número 9 a la número 12. Esto representa que al terminar la iteración se ha implementado un 79.8 % de la aplicación.

Iteración 3.

En esta iteración se lleva a cabo el desarrollo de las HU del número 13 a la número 15. Esto representa que al terminar la iteración se ha implementado un 100% de la aplicación.

2.6.3 Plan de duración de las iteraciones

El plan de duración de las iteraciones se encarga de mostrar las HU en el orden en que se implementarán en cada iteración así como la duración estimada de las mismas.

Tabla 5 Plan de duración de las iteraciones

Iteración	Orden de las HU a implementar	Duración Total
1	- Gestionar Script. - Buscar Script - Listar resultados - Mostrar resultados	3 Semanas
2	- Buscar resultados - Enviar petición a cliente específico - Enviar petición a Subred - Enviar petición a más de un cliente	3 Semanas
3	- Mostrar estadísticas sobre la ejecución de los scripts - Exportar a hojas de cálculo las respuestas. - Recibir petición del servidor	3 semanas

	- Ejecutar Script	
4	- Verificar compatibilidad con el SO - Enviar resultado de la ejecución al servidor - Programar ejecución automática de script	3 semanas
		Total: 12 Semanas

2.7 Plan de entrega

El plan de entrega marca las fechas de terminación de cada una de las iteraciones que tendrá el módulo, así como los componentes que serán obtenidos en cada una de ellas.

Tabla 6 Plan de entrega

Módulo	Final de la Iteración 1	Final de la Iteración 2	Final de la Iteración 3	Final de la Iteración 4
Módulo de automatización de Script.	2015/03/20	2015/04/10	2015/05/1	2015/05/22

2.8 Conclusiones

Luego de abordar en este capítulo la solución propuesta para la implementación del módulo se puede concluir lo siguiente: La descripción de las Historias de Usuarios y la planeación del tiempo del proyecto serán un aspecto fundamental y de gran ayuda en el desarrollo del módulo, además, se espera que teniendo en cuenta la descripción anterior la implementación sea efectuada de forma eficiente permitiendo un producto final con una buena calidad

Capítulo 3: Análisis y Diseño del Módulo

3.1 Introducción

En este capítulo se identifican y organizan las clases relevantes mediante la creación de las tarjetas Clase – Responsabilidad – Colaborador (CRC). Además se especifica el patrón de arquitectura y los patrones de diseño utilizados en el desarrollo del módulo.

3.2 Patrón arquitectónico

"*En el desarrollo del software los patrones arquitectónicos expresan el esquema de organización estructural fundamental para sistemas de software. Además proveen un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y pautas para la organización de las relaciones entre ellos*". (23)

El Sistema de Gestión de Hardware y Software fue desarrollado utilizando el mismo patrón arquitectónico que implementa el marco de trabajo Django, el Modelo-Template-View (**MTV**). Este patrón está diseñado para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples. El patrón Model Template View (MTV), que se deriva del patrón Modelo Vista Controlador (MVC), sigue una estructura de N-Capas[17]:

- **Model** (Modelo), 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.
- **Template** (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web u otro tipo de documento.
- **View** (Vista), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada: se puede ver como un puente entre los modelos y las plantillas.

La relación existente en el patrón MTV funciona de la siguiente manera: primeramente el navegador manda una solicitud a la View, luego la View interactúa con el Model para obtener datos de la base de datos y llama a la Template, la cual es la encargada de enviar la respuesta a la solicitud del navegador. En la siguiente figura se puede apreciar con mayor detalle su funcionamiento.

[17] Arquitectura de programación donde el objetivo principal es separar los diferentes aspectos del desarrollo, tales como las cuestiones de presentación, lógica de negocio, mecanismos de almacenamiento.

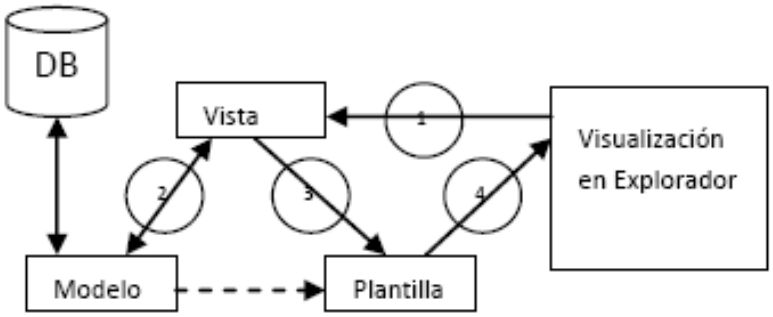


Ilustración 2 Funcionamiento de MTV (23)

Aporta ventajas como:

- *Clara separación entre interfaz, lógica de negocio y de presentación.*
- *Sencillez para crear distintas representaciones de los mismos datos.*
- *Facilidad para la realización de pruebas unitarias de los componentes.*
- *Reutilización de los componentes*. (24)

3.2.1 Aplicación del patrón MTV en el módulo.

Capa Models

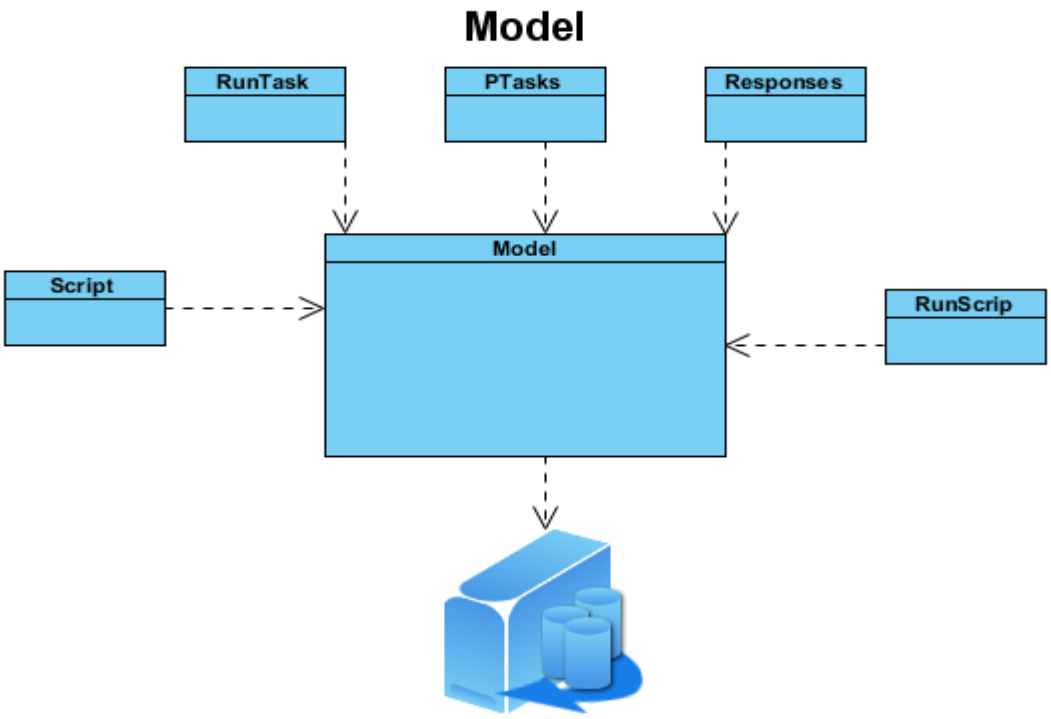


Ilustración 3 Capa Model

Los modelos o models está compuesta por:

Script: clase responsable de registrar, modificar, eliminar, buscar y obtener detalles de los script.

RunTask: clase responsable de ejecutar las tareas programadas.

PTask: clase responsable de registrar, modificar, eliminar, buscar y obtener detalles de las tareas programadas.

RunScrip: clase responsable de ejecutar los script.

Response: clase responsable de registrar, modificar, eliminar, buscar y obtener detalles de los peticiones a las maquinas clientes.

Capa Templates

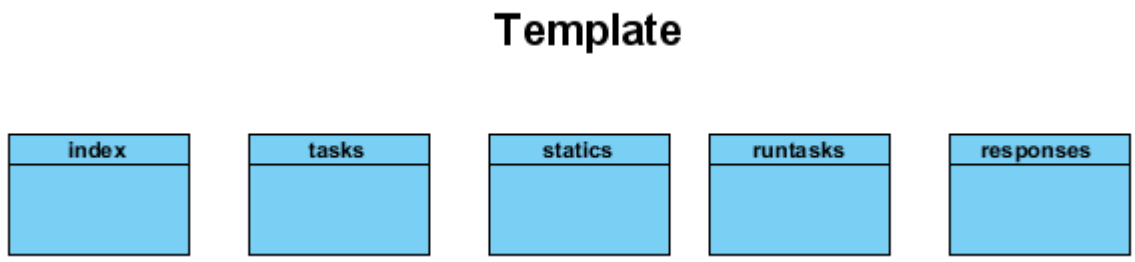


Ilustración 4 Capa Template

Los plantillas o templates están compuesto por los siguientes ficheros: **responses**, **runtasks, tasks, statics** e **index**. Estos archivos definen las plantillas HTML, CSS y JavaScript.

Capa Views

La capa View está compuesta por los ficheros : **ScriptView, RunRestView, TaskRestView y StatisticsRestView** que se encuentran dentro de la carpeta runtasks.

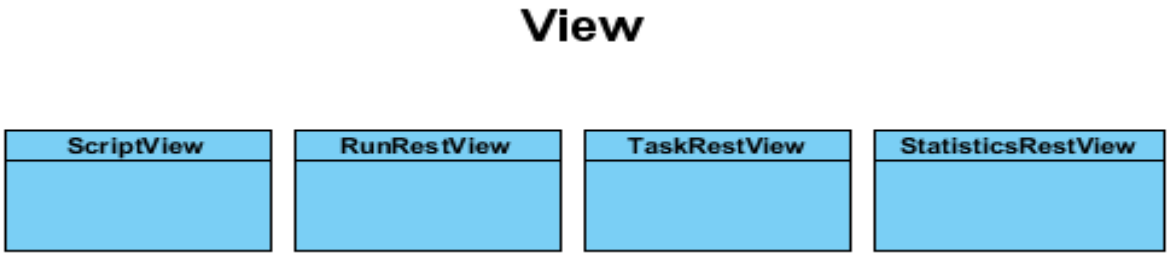


Ilustración 5 Capa View

3.3 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Representan una descripción de las clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. El uso de patrones posibilita estandarizar el modo en que se realiza el diseño y proporciona reusabilidad, extensibilidad y mantenimiento del código. (25) A continuación se muestran los patrones de diseño utilizados en la implementación del módulo.

3.3.1 Patrones General Responsibility Assignment Software Patterns (GRASP)

Los patrones GRASP, describen la asignación de responsabilidades a objetos. Para el desarrollo del módulo fueron utilizados los siguientes patrones de esta clasificación que en su mayoría son implementados de forma nativa por el marco de trabajo Django:

Bajo Acoplamiento: Este patrón se aplica en todas las clases. Su utilización hace posible que una modificación en una clase tenga poca repercusión en las demás. Esto se logra pues el patrón garantiza la existencia de pocas dependencias entre las clases.

Alta Cohesión: Cada elemento del diseño realiza una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Este patrón es usado en todas las clases ya que permite que los datos y responsabilidades de una clase sean coherentes y estén fuertemente ligados a la misma, en un sentido lógico.

Experto: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Un Ejemplo de su uso es en la clase runscript que es la encargada de mandar la petición de ejecución al cliente ya que ella contiene los datos necesarios para realizar esta labor.

Ejemplo de uso del patrón Experto

```
class RunScrip(models.Model):
    control =models.CharField(max_length=20)
    fileid = models.BigIntegerField()
    collectors = ListField()
    subnetv = models.CharField(max_length=20)
    class Meta:
        managed = False

    def save(self, force_insert=False, force_update=False, using=None):
        script= Script.objects.filter(pk=self.fileid)
        operation = self.control
        if self.subnetv != "null":
         slaves = InformationAgent.objects.filter(is_actived=True)
         http_send_script_to_network('send',self.subnetv,script=script[0],queue=slaves,params={'contr

        else:
         slaves = InformationAgent.objects.filter(pk__in=self.collectors)
         http_send_script_to_host('send',script=script[0],queue=slaves,params={'control':self.control
```

Polimorfismo: Cuando se identifican variaciones en un comportamiento, se asigna la clase (interfaz) al comportamiento y se utiliza polimorfismo para implementar los comportamientos alternativos Django utiliza este patrón de forma nativa un ejemplo de su uso son los modelos de Django que heredan de un modelo genérico y solo quedaría reimplementar su comportamiento mediante polimorfismo.

Ejemplo de uso del patrón Polimorfismo

```
class Responses(models.Model):
    origen=models.ForeignKey(InformationAgent,verbose_name=_('Ag
    origen_addres=models.IPAddressField()
    scrip=models.ForeignKey(Script)
    ejecution_log=models.TextField()
    ejecution_date=models.DateTimeField(null=False,auto_now=True
    def __unicode__(self):
        return self.id
```

3.3.2 Patrones The Gang of Four (GoF)

Dentro de los patrones de diseño se encuentran los patrones Gang of Four o Grupo de los Cuatro. Estos patrones definen el comportamiento entre las clases y los objetos. Los patrones de diseño GoF "*se clasifican en 3 grandes categorías basadas en su propósito Creacionales, Estructurales y de Comportamiento"* (26)

En el desarrollo del módulo se utilizaron los siguientes patrones GoF.

Template Method

Propósito: Definir el esqueleto de un algoritmo para una operación, dejando para sus subclases la capacidad de redefinir el funcionamiento de los pasos de este algoritmo, siempre y cuando la estructura del mismo permanezca intacta.

Aplicación: Los formularios creados, se derivan de un formulario genérico con atributos que pueden ser modificados, en dependencia de cada necesidad.

Ejemplo de uso del patrón Template Method

```
var DeleteForm = Form.extend({
    events: {
        'click .accept': function() {
            if (!$('#a-accept').hasClass('ui-state-disabled'))
                this.model.destroy();
        },
        'click .cancel': function () {
            this.trigger('cancel');
        }
    },
    commit: function(options) {
        this.model.destroy();
    },
```

3.4 Tarjetas Clase – Responsabilidad – Colaborador

Las tarjetas CRC son una técnica para la representación de los sistemas, que tienen como principal objetivo identificar y organizar las clases que se necesitan para implementar el sistema en relación que guardan entre ellas. De esta forma se hace más fácil el trabajo del equipo de desarrollo a la hora de analizar y diseñar las clases. (27)

Las tarjetas CRC están formadas por tres secciones:

- Nombre de las clases: se especifica el nombre de la clase que se describe.
- Responsabilidad: son las funciones que contiene implementada la clase
- Colaboradores: representa además clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades

Tabla 7 Tarjeta CRC de la clase Script.

Script	
Responsabilidad:	Colaborador:
Se encarga de gestionar la información de los script en el sistema.	-----

Tabla 8 Tarjeta CRC de la clase Responses

Responses	
Responsabilidad:	Colaborador:
Se encarga de almacenar el resultado de la ejecución de los script.	Script

Tabla 9 Tarjeta CRC de la clase Programed_Task

Programed_Task	
Responsabilidad:	Colaborador:
Se encarga de almacenar los datos de las tareas programadas.	Script

3.5 Diagrama de Clases Persistentes

El lenguaje Python contemplan el desarrollo de software sobre el paradigma de la orientación a objetos, el cual permite crear modelos que guardan una gran correspondencia con el dominio del problema para el que se va a diseñar una solución. El estado de esos objetos, que representa la información sobre la que operan, ha de ser conservada de forma persistente. El diagrama de clases persistentes es la base para la creación del modelo físico de la base de datos del sistema. La siguiente ilustración muestra las clases persistentes del módulo de ejecución automática de scripts.

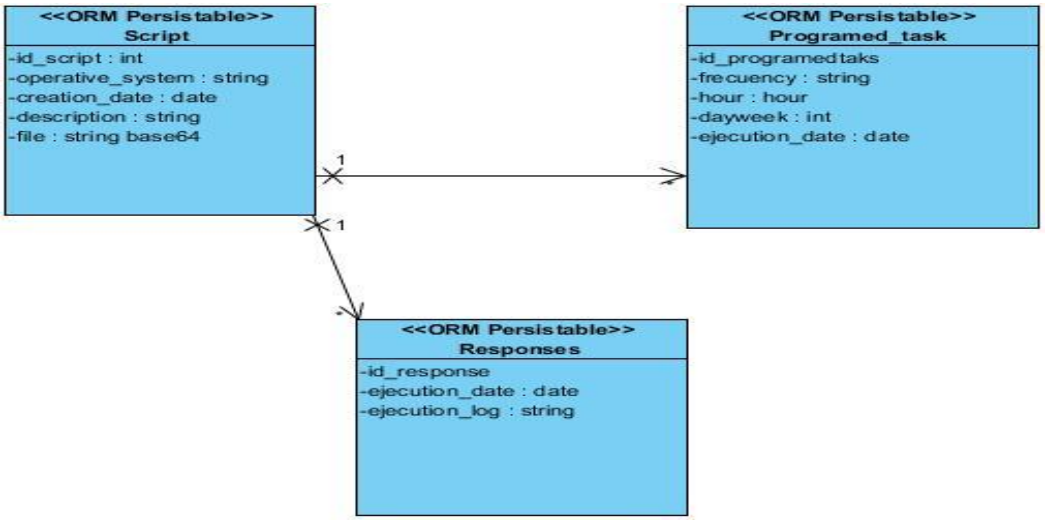


Ilustración 6 Diagrama de clases persistentes

3.6 Modelo de Datos

Para la implementación del módulo se hizo necesario añadir a la base de datos de GRHS las siguientes tablas:

dt_scripts: En esta tabla se almacenan todos los scripts que han sido creados por el administrador en el sistema. En ella se almacena identificador, sistema operativo para el que ha sido creado el script, una breve descripción del contenido del script y su función, el script como archivo codificado a base64 para un mejor tratamiento y la fecha de su última modificación. Además posee relación con las tablas dt_responses y dt_programed_tasks.

dt_responses: En esta tabla quedan registrados los resultados de las ejecuciones de los scripts. En ella se almacena la dirección ip donde fue ejecutado el script y el resultado de esta ejecución y posee una relación con la tabla dt_scripts.

dt_programed_tasks: En esta tabla se guardan los datos de las tareas programadas, de estas se almacena fecha, hora y pc destino para ejecutar la tarea. Esta tabla posee relación con la tabla dt_scripts.

dt_information_agent: Esta tabla es propia del sistema GRHS a la cual el módulo a implementar accederá para obtener los datos de los agentes dirección ip, puerto, si está activo y la plataforma que esta corriendo.

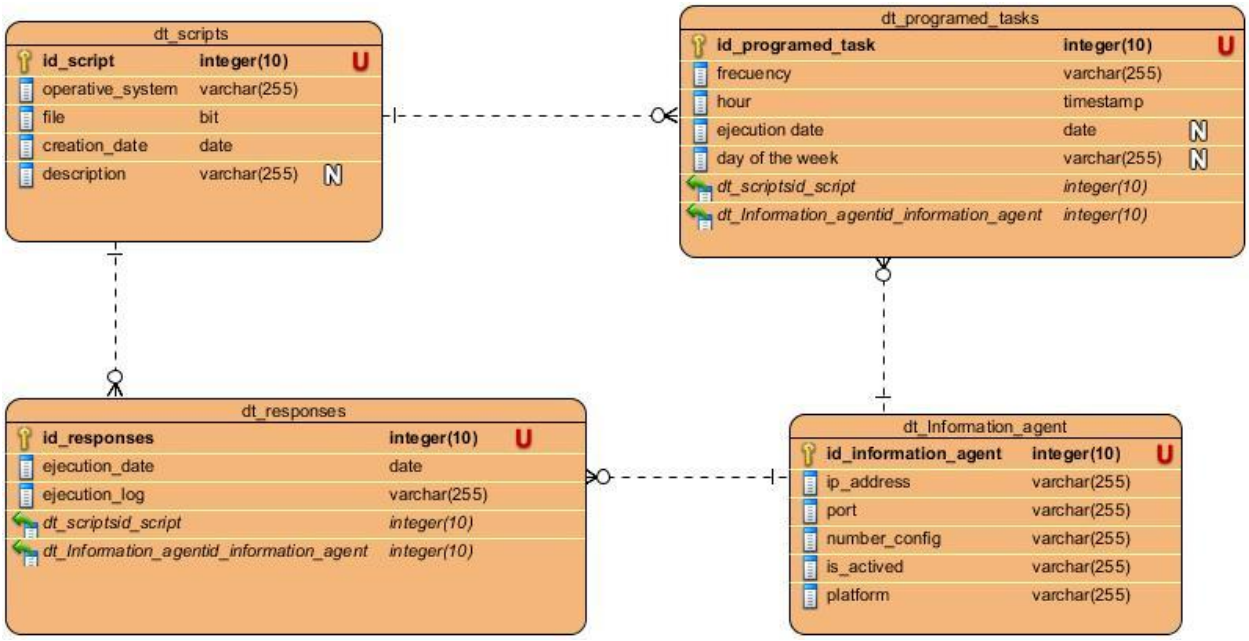


Ilustración 7 Modelo de datos

3.7 Conclusiones

En este capítulo se realizó un análisis sobre el patrón arquitectónico utilizado por el Framework sobre el que se desarrolla el módulo. Fueron seleccionados los patrones de diseño para la solución de problemas de implementación y diseño durante la confección del software. Se confeccionaron las tarjetas CRC para identificar y organizar las clases que son necesarias para implementar el módulo correspondiente y la relación que guardan entre ellas. Se realizó el diseño de las tablas de la BD que serán agregadas a la BD.

Capítulo 4: Implementación y Pruebas

4.1 Introducción

En el presente capítulo se generan las tareas de ingeniería asociadas a cada historia de usuario y se describe la implementación del módulo por iteraciones. Además se realizan las pruebas al sistema a partir de la formulación de los casos de prueba.

4.2 Implementación

En esta fase se implementan las historias de usuario previamente definidas, para ello se crean las tareas de ingenierías especificando las acciones llevadas a cabo por los programadores en cada historia de usuario. Estas tareas son fichas que se confeccionan introduciendo los siguientes datos:

Número de tarea: Representa el identificador de la tarea.

Número de Historia de usuario: Representa la historia de usuario asociada a esta tarea.

Tipo de Tarea: Se especifica si la tarea es de Desarrollo, Corrección, Mejora, etc.

Puntos Estimados: Duración estimada de la tarea.

Fecha Inicio y Fecha Fin: Se especifica el día, mes y año en que comienza y termina la tarea.

Programador Responsable: Persona encargada de su realización.

Descripción: Breve descripción de la tarea.

4.2.1 Iteración 1

A continuación se muestran las HU implementadas en la iteración 1.

Tabla 10 Historias de usuario implementadas en la iteración 1

Historia de Usuario	Funcionalidades	Tiempo Real (días)
Gestionar Script	Adicionar Script.	5
	Modificar Script.	
	Eliminar Script.	
Buscar Script	Listar Script.	4
	Buscar Script.	
Listar resultados	Listar Resultados de las Ejecuciones de los Scripts	3
Mostrar resultados	Mostrar Detalles de la ejecución de un Script	3
Total		**15 días ó 3 Semanas**

Tareas de ingeniería asociadas a las Historias de usuario implementadas en esta iteración.

Tabla 11 Tarea de ingeniería #1 Insertar Script.

Tarea de Ingeniería	
Número de la tarea : 1.	**Historia de Usuario:** 1.
Nombre de la Tarea: Insertar Script.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.4
Fecha Inicio: 2/3/15.	**Fecha fin:** 3/3/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Los scripts son creados y almacenados para su posterior uso mediante el botón insertar script de la vista. Presentan las siguiente propiedades como atributos: -Sistema Operativo para el que fue creado -Descripción -Contenido del Script en base64 -Fecha de Creación	

Tabla 12 Tarea de ingeniería #2 Eliminar Script

Tarea de Ingeniería	
Número de la tarea : 2.	**Historia de Usuario:** 1.
Nombre de la Tarea: Eliminar Script.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.2
Fecha Inicio: 4/3/15.	**Fecha fin:** 4/3/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Del listado de scripts se selecciona uno de ellos y luego se decide a eliminarlo del sistema mediante el uso del botón eliminar de la vista.	

Tabla 13 Tarea de ingeniería #3 Editar Script.

Tarea de Ingeniería	
Número de la tarea : 3.	**Historia de Usuario:** 1.
Nombre de la Tarea: Editar Script.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.4

Fecha Inicio: 5/3/15.	**Fecha fin:** 6/2/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Del listado de scripts se selecciona uno de ellos y se edita modificando los siguientes campos de datos a través del botón modificar de la vista: -Sistema Operativo para el que fue creado -Descripción -Contenido del Script -Fecha de creación	

Tabla 14 Tarea de ingeniería #4 Listar Script.

Tarea de Ingeniería	
Número de la tarea : 4.	**Historia de Usuario:**2.
Nombre de la Tarea: Buscar Script.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.4
Fecha Inicio: 9/3/15.	**Fecha fin:** 10/3/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Muestra un listado con los scripts almacenados en el sistema.	

Tabla 15 Tarea de ingeniería #5 Buscar Script.

Tarea de Ingeniería	
Número de la tarea : 5.	**Historia de Usuario:**2.
Nombre de la Tarea: Buscar Script.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.4
Fecha Inicio: 11/3/15.	**Fecha fin:** 12/3/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Muestra un listado con los scripts almacenados en el sistema y permite filtrar la lista de acuerdo a el Sistema Operativo para el que va dirigido el script y a la fecha de creación.	

Tabla 16 Tarea de ingeniería #6 Mostrar Resultados

Tarea de Ingeniería	
Número de la tarea : 6.	**Historia de Usuario:** 4.
Nombre de la Tarea: Mostrar Resultados de la ejecución de los scripts.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.6
Fecha Inicio: 13/3/15.	**Fecha fin:** 17/3/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Después de recibir los resultado de la ejecución de los scripts el sistema permite mostrar su contenido a partir de la lista de resultados seleccionando el que se desea mostrar.	

Tabla 17 Tarea de ingeniería #7 Listar Resultados

Tarea de Ingeniería	
Número de la tarea : 7.	**Historia de Usuario:** 3.
Nombre de la Tarea: Listar Resultados de la ejecución de los scripts.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.6
Fecha Inicio: 18/3/15.	**Fecha fin:** 20/3/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Una vez almacenados los resultados de ejecución de los scripts, el modulo muestra una lista con todos los resultados con los siguientes datos. -Script Ejecutado(string) -Cliente donde se ejecutó(object) -Fecha de Ejecución (Datetime)	

4.2.2 Iteración 2

A continuación se muestran las HU implementadas en la iteración 2.

Tabla 18 Historias de usuario implementadas en la iteración 2

Historia de Usuario	Funcionalidades	Tiempo Real (días)
Buscar Resultados	Programar ejecución de script	3

Enviar petición a un cliente específico	Enviar petición de ejecución a un cliente	4
Enviar petición a más de un cliente	Enviar petición de ejecución a más de un cliente	4
Enviar petición a Subred	Enviar petición de ejecución a una subred	4
Total		**15 días ó 3 Semanas**

Tareas de ingeniería asociadas a las Historias de usuario implementadas en esta iteración.

Tabla 19 Tarea de ingeniería #8 Buscar Resultados

Tarea de Ingeniería	
Número de la tarea : 8.	**Historia de Usuario:** 5.
Nombre de la Tarea: Buscar Resultados de la ejecución de los scripts.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.6
Fecha Inicio: 23/3/15.	**Fecha fin:** 25/3/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: La lista de resultados puede ser filtrada por los siguientes parámetros de búsqueda: - Cliente donde fue ejecutado - Script ejecutado	

Tabla 20 Tarea de ingeniería #9 Enviar petición a cliente específico

Tarea de Ingeniería	
Número de la tarea : 9.	**Historia de Usuario:** 6.
Nombre de la Tarea: Enviar petición a cliente específico.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.8
Fecha Inicio: 26/3/15.	**Fecha fin:** 31/3/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Se selecciona un script de la lista de scripts y un cliente activo para enviar la petición enviando los siguientes datos: -Script(string base64)	

Tabla 21 Tarea de ingeniería #10 Enviar petición a más de un cliente

Tarea de Ingeniería	
Número de la tarea : 10.	**Historia de Usuario:** 8.
Nombre de la Tarea: Enviar petición a más de un cliente.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.8
Fecha Inicio: 1/4/15.	**Fecha fin:** 6/4/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Se selecciona un script a ejecutar y de una lista de clientes activos se seleccionan los clientes a donde se desea enviar la petición de ejecución. Enviando los siguientes datos: -Script(string base64)	

Tabla 22 Tarea de ingeniería #11 Enviar petición a Subred

Tarea de Ingeniería	
Número de la tarea : 11	**Historia de Usuario:** 7.
Nombre de la Tarea: Enviar petición a Subred.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.8
Fecha Inicio: 7/3/15.	**Fecha fin:** 10/3/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: El administrador seleccionará el script que desea enviar a ejecutar y debe introducir la subred destino para esta ejecución en el formato X.X.X.X/X y el sistema debe enviar esta petición a todos los agentes que se encuentren en esta subred.	

4.2.3 Iteración 3

A continuación se muestran las HU implementadas en la iteración 3.

Tabla 23 Historias de usuario implementadas en la iteración 2

Historia de Usuario	Funcionalidades	Tiempo Real (días)
Mostrar estadísticas sobre la ejecución de los scripts	Mostrar estadísticas sobre la ejecución de los scripts	3
Exportar a hojas de cálculo las respuestas	Exportar a hojas de cálculo las respuestas	3

Recibir petición del servidor	Recibir petición del servidor	3
Ejecutar Script	Ejecutar Script	5
Total		**14 días ó 3 Semanas**

Tareas de ingeniería asociadas a las Historias de usuario implementadas en esta iteración.

Tabla 24 Tarea de ingeniería #12 Mostrar estadísticas sobre la ejecución de script

Tarea de Ingeniería	
Número de la tarea : 12	**Historia de Usuario:** 9.
Nombre de la Tarea: Mostrar estadísticas sobre la ejecución de script.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.6
Fecha Inicio: 13/4/15.	**Fecha fin:** 15/4/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: El servidor mostrará las estadísticas respuestas de los script ejecutados almacenadas en el servidor para su posterior análisis.	

Tabla 25 Tarea de ingeniería #13 Exportar a hojas de cálculo las respuestas

Tarea de Ingeniería	
Número de la tarea : 13	**Historia de Usuario:** 10.
Nombre de la Tarea: Exportar a hojas de cálculo las respuestas.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.6
Fecha Inicio: 16/4/15.	**Fecha fin:** 20/4/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: El servidor exportará a hojas de cálculo las respuestas almacenadas en el servidor que fueron seleccionadas para su posterior análisis.	

Tabla 26 Tarea de ingeniería #14 Recibir petición del servidor

Tarea de Ingeniería	
Número de la tarea : 14.	**Historia de Usuario:** 11.

Nombre de la Tarea: Recibir petición del servidor.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.6
Fecha Inicio: 21/4/15.	**Fecha fin:** 23/4/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: El cliente recibe una petición de ejecución con el script a ejecutar en formato string base64.	

Tabla 27 Tarea de ingeniería #15 Ejecutar Script Recibido

Tarea de Ingeniería	
Número de la tarea : 15.	**Historia de Usuario:** 12.
Nombre de la Tarea: Ejecutar Script.	
Tipo: Desarrollo.	**Puntos Estimados:** 1.0
Fecha Inicio: 24/4/15.	**Fecha fin:** 30/4/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Una vez recibido desde el servidor el script, el cliente lo ejecuta, obteniendo el resultado de la ejecución para su posterior uso y envio.	

4.2.4 Iteración 4

A continuación se muestran las HU implementadas en la iteración 3.

Tabla 28 Historias de usuario implementadas en la iteración 2

Historia de Usuario	Funcionalidades	Tiempo Real (días)
Enviar resultado de la ejecución al servidor	Enviar resultado de la ejecución al servidor	3
Verificar compatibilidad con el SO	Verificar compatibilidad con el SO	3
Programar ejecución automática de Script	Programar ejecución automática de Script	8
Total		**14 días ó 3 Semanas**

Tareas de ingeniería asociadas a las Historias de usuario implementadas en esta iteración.

Tabla 29 Tarea de ingeniería #16 Enviar resultado de la ejecución al servidor

Tarea de Ingeniería	
Número de la tarea : 16	**Historia de Usuario:** 13.
Nombre de la Tarea: Enviar resultado de la ejecución al servidor.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.2
Fecha Inicio: 4/5/15.	**Fecha fin:** 6/5/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Después de obtenido el resultado de la ejecución el cliente lo enviará al servidor para ser almacenado con los siguientes datos. -Script Ejecutado -Cliente donde se ejecutó(object) -Log de Ejecución(string) -Fecha de la Ejecución (datetime)	

Tabla 30 Tarea de ingeniería #17 Verificar compatibilidad con el SO

Tarea de Ingeniería	
Número de la tarea : 17	**Historia de Usuario:** 14.
Nombre de la Tarea: Verificar compatibilidad con el SO.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.3
Fecha Inicio: 7/5/15.	**Fecha fin:** 11/5/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: El administrador del sistema deberá seleccionar el sistema operativo al cual se le enviara el script, de acuerdo a este criterio, solo se podrán enviar los ficheros con las extensiones que pertenecen a cada uno: Sistema Operativo-Windows -- bat Sistema Operativo-GNU/Linux – sh	

Tabla 31 Tarea de ingeniería #18 Programar ejecución automática de script

Tarea de Ingeniería	
Número de la tarea : 18	**Historia de Usuario:** 15.
Nombre de la Tarea: Programar ejecución automática de script.	
Tipo: Desarrollo.	**Puntos Estimados:** 1.6
Fecha Inicio: 12/5/15.	**Fecha fin:** 21/5/15.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: El servidor será capaz de enviar el script automáticamente a las computadoras seleccionadas según un parámetro establecido introducido por el usuario para su posterior ejecución.	

4.3 Pruebas

Las pruebas de software se consideran una actividad en la cual un sistema o uno de sus componentes se ejecutan en circunstancias previamente especificadas. Sus resultados se observan, se registran y se realiza una evaluación. (28) Durante esta etapa serán probados todos los componentes del producto tanto por el cliente como por el equipo de desarrollo.

La metodología XP se basa en probar constantemente el sistema que se está realizando permitiendo aumentar su calidad. Esto posibilita que se disminuya el tiempo transcurrido entre la aparición de un error y su corrección. XP divide las pruebas del sistema en dos grupos:

Pruebas unitarias: Son pruebas de caja blanca encargadas de verificar el código diseñado por los programadores.

Pruebas de aceptación o pruebas funcionales: Tienen como propósito demostrar al cliente el cumplimiento de un requisito del software. (29)

4.3.1 Pruebas Unitarias

De acuerdo con lo que plantea la metodología XP, las pruebas unitarias o pruebas de unidad consisten en comprobaciones (manuales o automatizadas) desarrolladas por los programadores. Las cuales se realizan para verificar que el código correspondiente a un módulo concreto se comporta de manera esperada. Las pruebas unitarias proporcionan beneficios tales como: (30)

- Brindan al programador una inmediata retroalimentación de cómo está realizando su trabajo.

- El programador puede realizar cambios de forma segura respaldada por efectivos casos de prueba.
- Permite saber si una determinada funcionalidad se puede agregar al sistema existente sin alterar el funcionamiento actual del mismo.

En la siguiente ilustración se puede observar un ejemplo de prueba unitaria realizada a uno de los métodos del sistema.

```
from src.apps.runtasks.tasks import scheduled_script_tasks
from django.test import TestCase


class SimpleTest(TestCase):...


class TestScheduled_script_tasks(TestCase):
    def tes_task(self):
        scheduled_script_tasks()

    def test_scheduled_script_tasks(self):
        self.fail()
```

Test: runtasks

Ilustración 8 Prueba unitaria

4.3.1 Pruebas de aceptación

Debido a las características del trabajo realizado, se determina que las pruebas de aceptación son más importantes que las pruebas unitarias ya que el módulo a desarrollar presenta como objetivo principal el trabajo con interfaces de usuarios y se obtendrá una mejor resultado en función de la aceptación del cliente realizando este tipo de pruebas, además marcarán el final de una iteración y el comienzo de la siguiente. Las pruebas de aceptación se elaboran a lo largo de la iteración, en paralelo con el desarrollo del sistema y adaptándose a los cambios que el sistema sufra. Las pruebas de aceptación son consideradas como “pruebas de caja negra”. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Así mismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. (30)

Las pruebas de aceptación poseen gran importancia ya que permiten confirmar que la historia de usuario ha sido implementada correctamente al final de cada iteración. Una historia de

usuario puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. A continuación se muestran los datos que recogen las pruebas de aceptación:

- **Clases Válidas:** se hará la descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las entradas válidas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- **Clases Inválidas:** se hará la descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las posibles entradas inválidas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- **Resultado Esperado:** se hará una breve descripción del resultado que se espera, tanto para entradas válidas como para entradas inválidas.
- **Resultado de la Prueba:** se hará una breve descripción del resultado que se obtiene.
- **Observaciones:** se hará la descripción de cualquier señalamiento o advertencia que sea necesario hacerle a la sección que se está probando.

A continuación se muestran algunas de las pruebas de aceptación realizadas:

Tabla 32 Diseño del caso de prueba: Adicionar Script al sistema

Clases Validas	Clases Invalidas	Resultados Esperados	Resultado de la Prueba	Observaciones
Paso 1: el usuario selecciona el botón adicionar script de la interfaz web. En la ventana de adicionar selecciona un sistema operativo (Windows o Linux), llena la descripción sobre el script y selecciona en la computadora un script en el formato correcto para el sistema operativo seleccionado (extensión .sh para los sistemas Linux y		El sistema adiciona los datos introducidos a la base de datos y se muestra a continuación en el listado de los script.	Satisfactorio.	

.bat para los sistemas Windows). **Ejemplo:** Sistema Operativo: Windows Descripción: Apagar la PC. Script: Mi_Script.bat **Paso 2:** el usuario presiona el botón aceptar.				
	Paso 1: el usuario selecciona el botón adicionar script de la interfaz web. En la ventana de adicionar no selecciona ningún sistema operativo(Windows o Linux), llena la descripción sobre el script y selecciona en la computadora un script en el formato correcto para el sistema operativo seleccionado (extensión .sh para los sistemas Linux y .bat para los sistemas Windows). **Ejemplo:** --Select- Descripción: Apagar la PC. Script:	El sistema muestra un mensaje de error informándole al usuario que los campos están vacíos y deben ser llenados.	Satisfactorio.	

	Mi_Script.bat **Paso 2:** el usuario presiona el botón aceptar.			
	Paso 1: el usuario selecciona el botón adicionar script de la interfaz web. En la ventana de adicionar selecciona un sistema operativo(Windows o Linux), deja en blanco el campo descripción sobre el script y selecciona en la computadora un script en el formato correcto para el sistema operativo seleccionado (extensión .sh para los sistemas Linux y .bat para los sistemas Windows). **Ejemplo:** Sistema Operativo: Windows Descripción: Script: Mi_Script.bat **Paso 2:** presiona el botón insertar.	El sistema muestra un mensaje de error informándole al usuario que existen campos vacíos y deben ser llenados.	Satisfactorio.	
	Paso 1: el usuario selecciona el botón adicionar script de la interfaz web. En la ventana de adicionar selecciona un	El sistema muestra un mensaje de error informándole al usuario que existen campos vacíos y deben	Satisfactorio.	

	sistema operativo (Windows o Linux), llena la descripción sobre el script y no selecciona ningún script. **Ejemplo:** Sistema Operativo: Linux Descripción: Apagar la PC. Script: **Paso 2:** presiona el botón insertar.	ser llenados.		

Tabla 33 Diseño del caso de prueba: Exportar a Excel

Clases Validas	Clases Invalidas	Resultados Esperados	Resultado de la Prueba	Observaciones
Paso 1: el usuario selecciona el botón Exportar a Excel de la interfaz.		Se permite descargar la información de la lista de los responses a Excel.	Satisfactorio.	Existen elementos en la lista de responses.
	Paso 1: el usuario selecciona el botón Exportar a Excel de la interfaz pero no existen elementos almacenados.	El sistema muestra un mensaje de error informándole al usuario que no existen elementos en la lista.	Satisfactorio.	No existen elementos previos en la lista de responses.

Al terminar las pruebas de aceptación se obtuvo un total de 5 no conformidades en la primera iteración de las cuales todas fueron resueltas fueron resueltas. En la segunda iteración se obtuvieron 3 no conformidades donde todas fueron resueltas; durante las dos iteraciones no quedó ninguna no conformidad pendiente. La ilustración 9 muestra los resultados obtenidos durante las pruebas de aceptación.

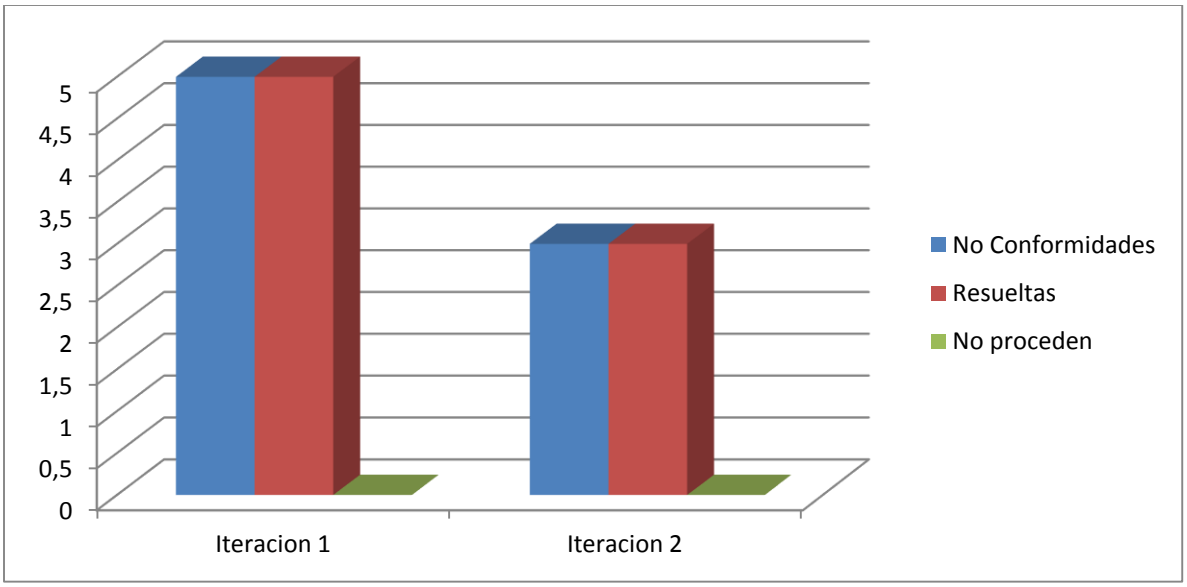


Ilustración 9 Diagrama de No Conformidades

4.3.2 Pruebas de integración

Consiste en combinar y probar diferentes componentes juntos verificando que la integración de estos se realice correctamente. Para poder detectar y aislar los errores del módulo se realiza la integración incremental, la cual puede ser de dos tipos: la integración descendente la cual consiste en integrar los módulos en dependencia de la jerarquía de módulos; y la ascendente que consiste en integrar los módulos de los niveles más bajos. La prueba de integración que se utilizó en el proyecto GRHS fue la descendente, integrando Gserver como módulo principal con los demás módulos del sistema. A continuación se muestra la tabla de prueba de integración entre Gserver y Tasks para la funcionalidad “Enviar script a ejecutar y almacenar respuesta recibida del cliente”.

Tabla 34 Prueba de Integración

Módulo Actual	**Gserver**
Módulo Integrado	Tasks
Funcionalidad	Enviar script a ejecutar y almacenar respuesta recibida del cliente.
Condiciones de Ejecución	Tener instalado el sistema de gestor de recursos de hardware y software GRHS en la pc.
Escenario de Prueba	Enviar script a ejecutar y almacenar respuesta recibida del cliente.
Resultado Previsto	Almacenar en la BD la respuesta enviada por el cliente.

Resultado Real	Se almacenó correctamente la respuesta.

4.4 Conclusiones

En el capítulo concluido se redactaron las tareas de ingeniería correspondientes a las HU que se definieron en el capítulo 2 para especificar las acciones llevadas a cabo por los programadores en cada historia de usuario. Además bajo la guía de la metodología XP se realizaron las pruebas unitarias y de aceptación. La ejecución de pruebas al módulo de ejecución de scripts permitió detectar las deficiencias presentes, subsanarlas en el menor tiempo posible y ofrecer una aplicación con mayor calidad, seguridad y usabilidad.

Conclusiones Generales

Con la realización del presente trabajo de diploma se desarrolló un módulo para GRHS que facilita la obtención de información y la ejecución de tareas de forma remota en los ordenadores de la red. Es por ello que se puede concluir afirmando que:

- El estudio realizado de los sistemas existentes relacionados permitió comprender la necesidad de desarrollar una aplicación capaz de ejecutar scripts de manera remota en los ordenadores de una red.
- La realización de las historias de usuarios facilitó una descripción detallada de las funcionalidades del sistema.
- El diseño de la solución permitió obtener un modelo que representa las clases que conformarán al sistema y con la elaboración de las tarjetas CRC se evidencian las relaciones entre ellas, de manera que se facilita la implementación de las funcionalidades de la aplicación.
- A partir de esta implementación se obtuvo un módulo capaz de satisfacer las necesidades del cliente.
- La ejecución de pruebas a la solución permitió asegurar la el correcto funcionamiento del módulo, de manera que se cumplan las expectativas del cliente.
- La implementación del módulo significa un aporte importante para el proyecto de GRHS del Centro de Desarrollo de Telemática de la UCI.

Recomendaciones

Debido a los resultados de la investigación efectuada y a la experiencia adquirida durante la realización de este trabajo, y con el propósito de asegurar la posterior ampliación, modificación y mejora del módulo, se exponen a continuación algunas recomendaciones:

- Adicionar al módulo la capacidad de ejecutar script creados en otros lenguajes de programación.
- Agregar la funcionalidad de verificar la correcta implementación de los scripts a enviar a los clientes.
- Adicionar al módulo la funcionalidad de escribir el código de los scripts a través de una interfaz web.

Referencias Bibliográfica

1. **Instituto de Investigaciones Jurídicas de la UNAM.** Biblioteca Juridica Virtual. [En línea] Instituto de Investigaciones Jurídicas de la UNAM. [Citado el: 14 de Marzo de 2015.] biblio.juridicas.unam.mx/libros/4/1941/4.pdf.

2. **WordPress.** Definición.de. [En línea] WordPress, septiembre de 2014. [Citado el: 24 de noviembre de 2014.] http://definicion.de/gestion/.

3. **Instituto Tecnológico y de Estudios Superiores de Monterrey.** Sifunpro. [En línea] 2014. www.sifunpro.tripod.com/automatizacion. z6.

4. **Buscador Hispano.** Pergaminovirtual. [En línea] 2007. www.pergaminovirtual.es.

5. **Corporation, Microsoft.** www.microsoft.com. [En línea] 2008. www.microsoft.com.

6. **Serrano, José Antonio.** Sistema de Automatización de Workarounds y Ejecución de Script Ocasionales. *Saweso, Sistema de Automatización de Workarounds y Ejecución de Script Ocasionales.* Catalunya : s.n., 2013.

7. **Roldan, Dayron Pérez.** Remote Execute Script and Commnad. *Sistema de Clonación y Distribución de Imágenes de Sistemas Operativos.* La Habana : s.n., 2008.

8. **Universidad de Malaga.** Eumet. [En línea] 2000. www.eumet.net.

9. **BPMN.** bpmn. [En línea] 2000. www.bpmn.org.

10. **White, R.** BPMN. [En línea] 2010. www,bpmn.org/Document.

11. **Visual Parading.** visual parading. [En línea] 2010. www.visual-parading.com-features.

12. **Moore, Dana, Budd, Raymond y Wright, Willian.** *Professional Python Frameworks.* s.l. : Wiley Publishing, Inc., 2007.

13. **Hogan, Brian P.** *HTML5 and CSS3.* s.l. : Pragmatic Programmers, LLC., 2010.

14. **Desarrollo Web. [En línea] [Citado el: 25 de noviembre de 2014.] www.desarrolloweb.com/articulos/25.php.**

15. Backbone. Backbone.js. [En línea] 2015. http://backbonejs.org/.

16. Newsroom. Una docena de.. [En línea] 2014. www.unadocenade.com.

17. Desarrollo Web. [En línea] 2013. www.desarrolloweb.com.

18. The PostgreSQL Global Development Group. Postgres SQL. [En línea] 2014. www.postgresql.org-docs-9.3.

19. Joskowicz, José. *Reglas y prácticas en eXtreme Programming.* España : s.n, 2008.

20. Jeffries, R., Anderson, A. y Hendrickson, C. *Extreme Programming Installed.* 2001.

21. Letelier, Partricio. Cyta. [En línea] Universidad Politécnica de Valencia (UPV). [Citado el: 19 de noviembre de 2014.] www.cyta.com.ar/ta0502/b_v5n2a1.

22. Allen, Robert y Garlan, David. *A formal approach to software architectures.* s.l. : Proceedings of the IFIP Congress , 1993.

23. Forcier, Jeff. *Python web Development whit Django.* s.l. : Django, 2009.

24. Wordpres. Wordpres. [En línea] exequielc.wordpress.com/2007/08/20/arquitectura-modeloplantillavista..

25. Larman, Craig. *UML y Patrones. Una introducción al análisis y el diseño orientado a objetos y al procceso unificado.* 2009.

26. Benneth , Christiansson, y otros. *GoF Design Patterns.* 2008.

27. Casas, Sandra I. y Reinaga., Héctor H. *Un enfoque basado en las tarjetas CRC.* Argentina : s.n., 2009.

28. [En línea] [Citado el: 28 de Abril de 2015.] http://equipomaquila.blogspot.com/2010/05/definiciones-de-pruebas-de-software.html..

29. Linkedin. [En línea] [Citado el: 28 de abril de 2015.] http://www.linkedin.com/groups/Qu%C3%A9-es-Prueba-Aceptaci%C3%B3n-3636186.S.48805747..

30. Gutierrez, Javier J, Escalona, M. J. y Mejías, J., M. Sistemas de Programación Extrema. [En línea] [Citado el: 28 de abril de 2015.] http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf..

31. Gich, J. *Teoría General de Sistemas.* México : IBIDEM, 1981.

Bibliografía

1. **Instituto de Investigaciones Jurídicas de la UNAM.** Biblioteca Juridica Virtual. [En línea] Instituto de Investigaciones Jurídicas de la UNAM. [Citado el: 14 de Marzo de 2015.] biblio.juridicas.unam.mx/libros/4/1941/4.pdf.

2. **WordPress.** Definición.de. [En línea] WordPress, septiembre de 2014. [Citado el: 24 de noviembre de 2014.] http://definicion.de/gestion/.

3. **Instituto Tecnológico y de Estudios Superiores de Monterrey.** Sifunpro. [En línea] 2014. www.sifunpro.tripod.com/automatizacion. z6.

4. **Buscador Hispano.** Pergaminovirtual. [En línea] 2007. www.pergaminovirtual.es.

5. **Corporation, Microsoft.** www.microsoft.com. [En línea] 2008. www.microsoft.com.

6. **Serrano, José Antonio.** Sistema de Automatización de Workarounds y Ejecución de Script Ocasionales. *Saweso, Sistema de Automatización de Workarounds y Ejecución de Script Ocasionales.* Catalunya : s.n., 2013.

7. **Roldan, Dayron Pérez.** Remote Execute Script and Commnad. *Sistema de Clonación y Distribución de Imágenes de Sistemas Operativos.* La Habana : s.n., 2008.

8. **Universidad de Malaga.** Eumet. [En línea] 2000. www.eumet.net.

9. **BPMN.** bpmn. [En línea] 2000. www.bpmn.org.

10. **White, R.** BPMN. [En línea] 2010. www,bpmn.org/Document.

11. **Visual Parading.** visual parading. [En línea] 2010. www.visual-parading.com-features.

12. **Moore, Dana, Budd, Raymond y Wright, Willian.** *Professional Python Frameworks.* s.l. : Wiley Publishing, Inc., 2007.

13. **Hogan, Brian P.** *HTML5 and CSS3.* s.l. : Pragmatic Programmers, LLC., 2010.

14. **Desarrollo Web. [En línea] [Citado el: 25 de noviembre de 2014.] www.desarrolloweb.com/articulos/25.php.**

15. Backbone. Backbone.js. [En línea] 2015. http://backbonejs.org/.

16. Newsroom. Una docena de.. [En línea] 2014. www.unadocenade.com.

17. Desarrollo Web. [En línea] 2013. www.desarrolloweb.com.

18. The PostgreSQL Global Development Group. Postgres SQL. [En línea] 2014. www.postgresql.org-docs-9.3.

19. Joskowicz, José. ***Reglas y prácticas en eXtreme Programming.*** **España : s.n, 2008.**

20. Jeffries, R., Anderson, A. y Hendrickson, C. ***Extreme Programming Installed.*** **2001.**

21. Letelier, Partricio. Cyta. [En línea] Universidad Politécnica de Valencia (UPV). [Citado el: 19 de noviembre de 2014.] www.cyta.com.ar/ta0502/b_v5n2a1.

22. Allen, Robert y Garlan, David. ***A formal approach to software architectures.*** **s.l. : Proceedings of the IFIP Congress , 1993.**

23. Forcier, Jeff. ***Python web Development whit Django.*** **s.l. : Django, 2009.**

24. Wordpres. Wordpres. [En línea] exequielc.wordpress.com/2007/08/20/arquitectura-modeloplantillavista..

25. Larman, Craig. ***UML y Patrones. Una introducción al análisis y el diseño orientado a objetos y al procceso unificado.*** **2009.**

26. Benneth , Christiansson, y otros. ***GoF Design Patterns.*** **2008.**

27. Casas, Sandra I. y Reinaga., Héctor H. ***Un enfoque basado en las tarjetas CRC.*** **Argentina : s.n., 2009.**

28. [En línea] [Citado el: 28 de Abril de 2015.] http://equipomaquila.blogspot.com/2010/05/definiciones-de-pruebas-de-software.html..

29. Linkedin. [En línea] [Citado el: 28 de abril de 2015.] http://www.linkedin.com/groups/Qu%C3%A9-es-Prueba-Aceptaci%C3%B3n-3636186.S.48805747..

30. Gutierrez, Javier J, Escalona, M. J. y Mejías, J., M. Sistemas de Programación Extrema. [En línea] [Citado el: 28 de abril de 2015.] http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf..

31. Gich, J. ***Teoría General de Sistemas.*** **México : IBIDEM, 1981.**

32. Pressman, Roger S. ***Ingenieria de Software.*** **s.l. : Mc Graw Hill, 2002.**

33. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. ***El Proceso Unificado de Desarrollo de Software.*** **Madrid : Addison Wesley, 2000.**

34. Quintero, Héctor Ramón Peñaranda. ***La informática como mecanismo de gestión de la información.*** **Maracaibo, Venezuela : Monografias.com S.A, 2014.**

35. GitHub. [En línea] 2015. www.github.com.

36. Lurtz, Mark. ***Learning.Python.4th.Edition.*** **s.l. : Oreilly, 2009.**

37. Anders, Michel. ***Python.3.Web.Development.Beginners.*** **s.l. : Packt publishing, 2010.**

38. Ziade, Tarek. ***Expert-Python-Programming.*** **s.l. : Packt, 2008.**

39. Maestros de la Web. [En línea] 2015. www.maestrosdelweb.com/guias/#guias-django.

40. STEPHEN A. WHITE, PHD DEREK MIERS. ***Guía de Referencia y Modelado BPMN.*** **US : Future Strategies Inc.,Book Division, 2009.**

41. Holovaty, Adrian y Kaplan-Moss, Jacob. ***The Definitive Guide to Django: Web Development Done Right.*** **s.l. : Apress, 2010.**

42. Python.org. [En línea] http://docs.python.org/3.1/library/2to3.html..

43. Sommerville, Ian. ***Ingenieria del Software.*** **s.l. : Pearson Educación S.A,, 2005.**

44. Alvarez de Zayas, Carlos. ***Metodología de la Investigación Científica.*** **Santiago de Cuba : Centro de Estudios de Educación Superior ¨Manuel F. Gran. 1995.**

45. Escribano, Gerardo Fernández. ***Introducción a Extreme Programming.*** **2002.**

46. Larman, Craig. ***UML y Patrones. Una introducción al análisis y el diseño orientado a objetos y al proceso unificado.*** **2009.**

47. James, Sugrue. ***Beginning Backbone.js.*** **s.l. : Apress, 2010.**

48. Leonard, Richardson. ***RESTful Web Services.*** **s.l. : O´Reilly, 2001.**

49. DesarrolloWeb.com. DesarrolloWeb.com. [En línea] www.desarrolloweb.com/manuales/manual-backbonejs.html.

50. Dana Moore, Raymond Budd,William Wright. ***Professional Python® Frameworks Web 2.0 Programming with Django.*** **s.l. : Wiley.**

51. RUÍZ, JOSÉ MARÍA. ***REFORZANDO DJANGO.*** **2000.**

52. hackipedia.org. [En línea] 2015. http://hackipedia.org/Languages,%20computer/Batch%20file,%20MS-DOS/MS-DOS%20Batch%20Script%20Guide,%20SCRIPT%20LANGUAGE,%20SYNTAX%20AND%20EXAMPLES%20by%20JASE%20T.%20WOLFE.pdf.

53. Python.org. [En línea] 2015. https://www.python.org/.

54. Kilby, Mark. ***Extreme Programing an agile methodology.*** **s.l. : CONVERGYS, 2002.**

55. Beck, Kent. ***Second Edition, Kent Beck 2005 - Extreme Programming (XP.*** **2005.**

56. Gorakavi, Pavan Kumar. ***Build your Project using Extreme Programming.*** **s.l. : asapm.org, 2009.**

57. Schooenderwoert, Nancy Van. ***Extreme Programming – An Effective Framework For Knowledge.*** **2004.**

58. Adrian Holovaty, Jeremy Dunck. The Django Book. [En línea] 2007. http://www.djangobook.com.

59. Keith-Magee, Russell. The web framework for perfectionists with deadlines. [En línea] https://www.djangoproject.com/.

60. tldp.org. [En línea] 2015. http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html.

61. Backbone.org. Backbone.org. [En línea] [Citado el: 1 de Junio de 2015.] www.backbone.org.

Tabla 35 HU#2 Buscar Script

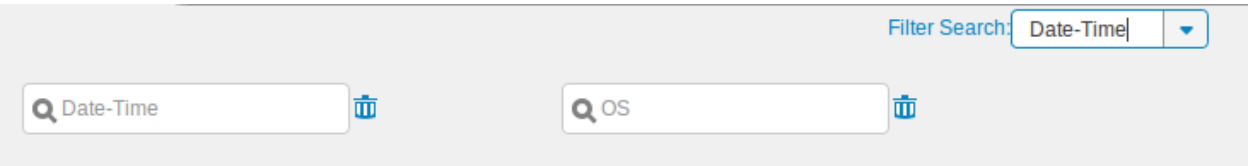
Historia de Usuario	
Número: 2.	**Usuario:** Administrador.
Modificación de Historia de Usuario #: Ninguna	
Nombre de Historia de Usuario: Buscar Script.	
Prioridad en negocio: Muy Alta.	**Riesgo de Desarrollo:** Bajo.
Puntos estimados: 0.8	**Iteración asignada:** 1.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción Permite realizar una búsqueda de un script en el sistema según los parámetros Descripción, Sistema Operativo, Fecha y Autor.	
Observaciones: El usuario debe encontrarse logueado en el sistema.	
Prototipo de interfaz: 	

Tabla 36 HU #3 Listar Resultados de la ejecución

Historia de Usuario	
Número: 3.	**Usuario:** Administrador.
Modificación de Historia de Usuario #: Ninguna	
Nombre de Historia de Usuario: Listar Resultados de la ejecución.	
Prioridad en negocio: Muy Alta.	**Riesgo de Desarrollo:** Bajo.
Puntos estimados: 0.6	**Iteración asignada:** 1.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción Permite mostrar todos los resultados de las ejecuciones de script del sistema.	
Observaciones: El usuario debe encontrarse logueado en el sistema.	

Prototipo de interfaz:

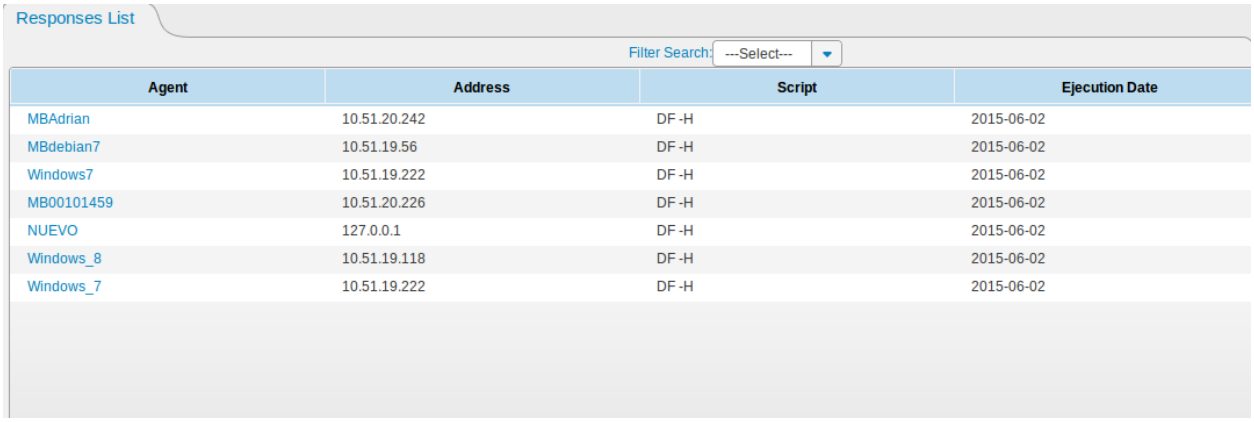


Tabla 37 HU #5 Buscar resultados de la ejecución

Historia de Usuario	
Número: 5.	**Usuario:** Administrador.
Modificación de Historia de Usuario #: Ninguna	
Nombre de Historia de Usuario: Buscar resultados de la ejecución.	
Prioridad en negocio: Alta.	**Riesgo de Desarrollo:** Bajo.
Puntos estimados: 0.6	**Iteración asignada:** 2.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción Permite realizar una búsqueda de las respuestas en el sistema según los parámetros Descripción del Script, Sistema Operativo, Fecha , Dirección Origen	
Observaciones: El usuario debe encontrarse logueado en el sistema.	
Prototipo de interfaz:	

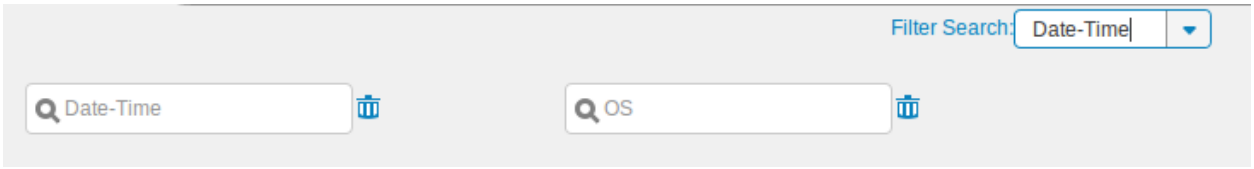


Tabla 38 HU #9 Mostrar estadísticas sobre los scripts almacenados

Historia de Usuario

Número: 9.	**Usuario:** Administrador.
Modificación de Historia de Usuario #: Ninguna	
Nombre de Historia de Usuario: Mostrar estadísticas sobre los scripts almacenados.	
Prioridad en negocio: Media.	**Riesgo de Desarrollo:** Bajo.
Puntos estimados: 0.6	**Iteración asignada:** 3.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción Permite mostrar una tabla con algunas estadísticas de los scripts almacenados	
Observaciones: El usuario debe encontrarse logueado en el sistema.	
Prototipo de interfaz:	

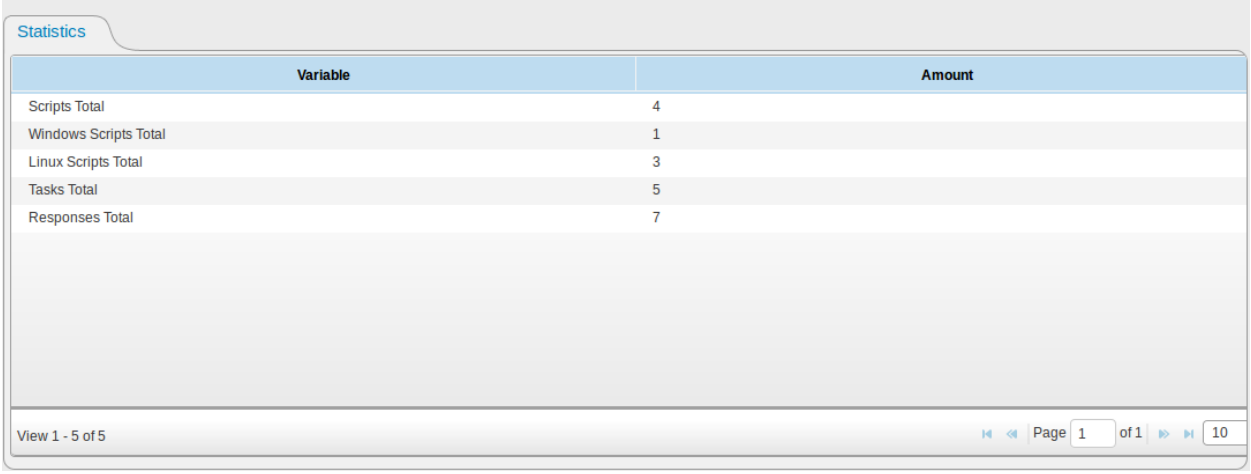


Tabla 39 HU #10 Exportar a hojas de cálculo las respuestas de las ejecuciones de los scripts

Historia de Usuario	
Número: 10.	**Usuario:** Administrador.
Modificación de Historia de Usuario #: Ninguna	
Nombre de Historia de Usuario: Exportar a hojas de cálculo las respuestas de las ejecuciones de los scripts.	
Prioridad en negocio: Media.	**Riesgo de Desarrollo:** Bajo.
Puntos estimados: 0.6	**Iteración asignada:** 3.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción Permite exportar a una hoja de cálculo todas las respuestas almacenadas en el	

sistemas.
Observaciones: El usuario debe encontrarse logueado en el sistema.
Prototipo de interfaz: 

Tabla 40 HU 13 Enviar resultado de la ejecución

Historia de Usuario	
Número: 13.	**Usuario:** Administrador.
Modificación de Historia de Usuario #: Ninguna	
Nombre de Historia de Usuario: Enviar resultado de la ejecución.	
Prioridad en negocio: Baja.	**Riesgo de Desarrollo:** Bajo.
Puntos estimados: 0.6	**Iteración asignada:** 4.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción Envía el resultado de la ejecución del script al servidor para su almacenamiento, guardando los datos Origen, Script y Fecha de Ejecución.	
Observaciones: El usuario debe encontrarse logueado en el sistema.	
Prototipo de interfaz:	

Tabla 41 HU #15 Programar ejecución automática de scripts

Historia de Usuario	
Número: 15.	**Usuario:** Administrador.
Modificación de Historia de Usuario #: Ninguna	
Nombre de Historia de Usuario: Programar ejecución automática de scripts	
Prioridad en negocio: Baja.	**Riesgo de Desarrollo:** Bajo.
Puntos estimados: 1.6	**Iteración asignada:** 4.

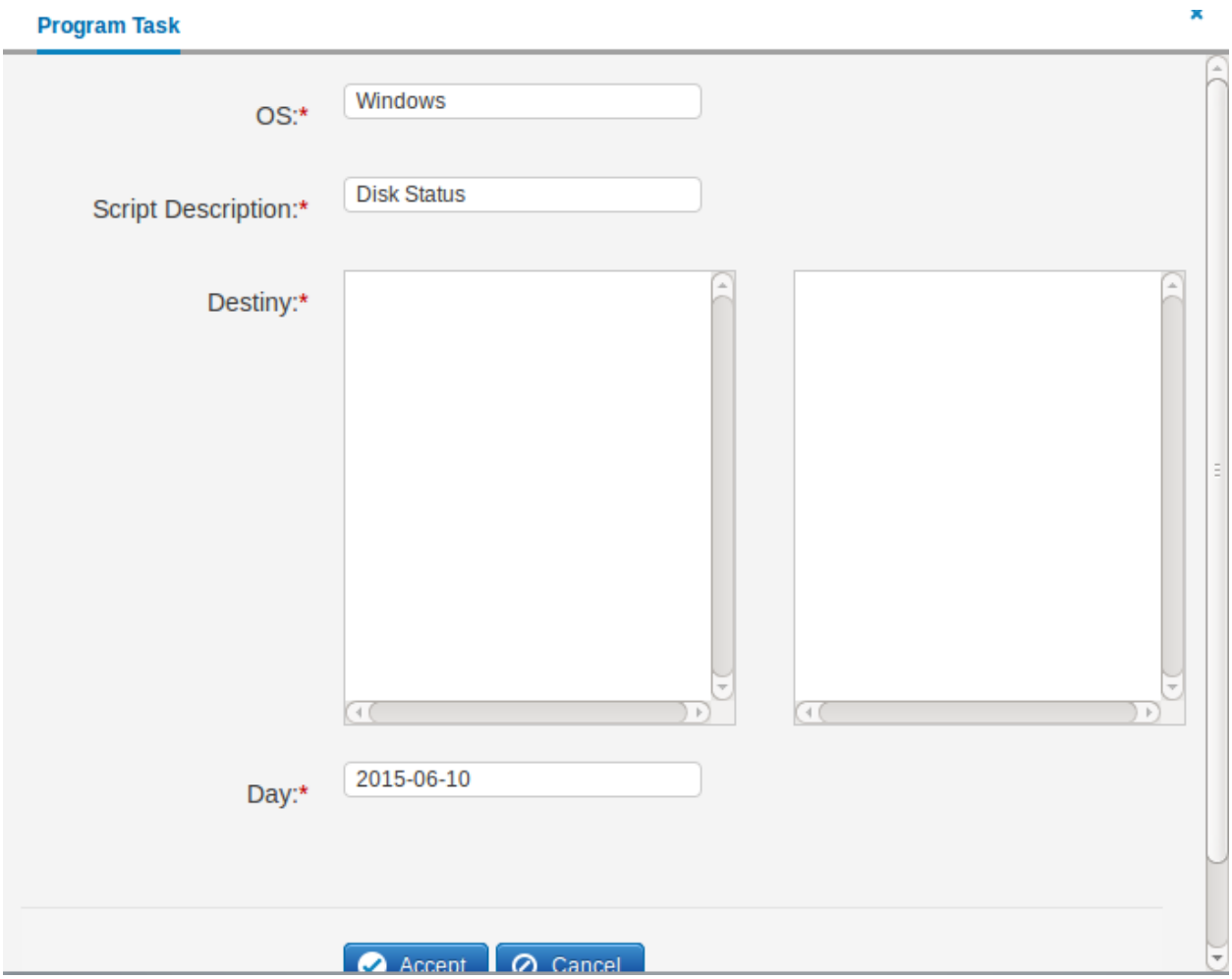
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.
Descripción Permite programar la ejecución de un script dado una fecha determinada .
Observaciones: El usuario debe encontrarse logueado en el sistema.
Prototipo de interfaz: 

Tabla 42 HU #14 Verificar compatibilidad con el Sistema Operativo

Historia de Usuario	
Número: 14.	**Usuario:** Administrador.
Modificación de Historia de Usuario #: Ninguna	
Nombre de Historia de Usuario: Verificar compatibilidad con el Sistema Operativo	

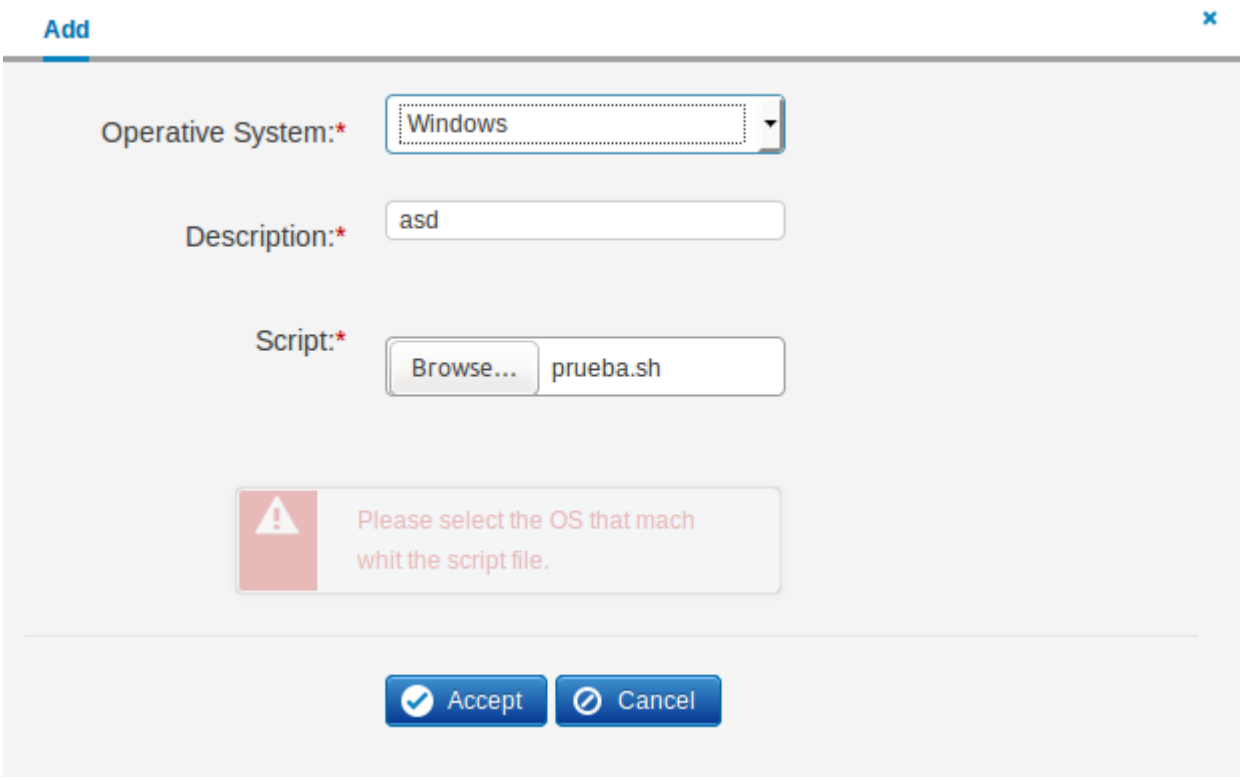
Prioridad en negocio: Baja.	**Riesgo de Desarrollo:** Bajo.
Puntos estimados: 0.6	**Iteración asignada:** 4.
Programador(es) responsable(s): Alejandro Rodríguez Toledo, Orlando Viera Pérez.	
Descripción: Permite verificar que el script enviado sea ejecutado correctamente en dependencia con el sistema operativo instalado.	
Observaciones: El usuario debe haber enviado el script hacia los terminales para ser ejecutados.	
Prototipo de interfaz: 	

Tabla 42 Diseño del caso de prueba: Enviar Script a un colector

Clases Validas	Clases Invalidas	Resultados Esperados	Resultado de la Prueba	Observaciones
Paso 1: el usuario selecciona del interfaz un colector a donde será enviado,		El sistema envía la petición al colector en cuestión.	Satisfactorio.	El modelo del Script debe haber sido insertado previamente. Se ha de

además se cargará el script para ser enviado al igual que el Sistema Operativo. **Ejemplo:** Sistema Operativo: Windows Colector: Mi_Pc|10.0.0.1 Script: Mi_Script.bat **Paso 2:** presiona el botón enviar.				seleccionar en la interfaz de listar para poder enviar.
	Paso 1: el usuario deja en blanco de la interfaz el colector a donde será enviado, además se cargará el script para ser enviado al igual que el Sistema Operativo. **Ejemplo:** Sistema Operativo: Windows Colector: Script: Mi_Script.bat **Paso 2:** presiona el botón enviar.	El sistema muestra un mensaje de error informándole al usuario que los campos están vacíos y deben ser llenados.	Satisfactorio.	

Tabla 43 Diseño del caso de prueba: Tarea Programada.

Clases Validas	Clases Invalidas	Resultados Esperados	Resultado de la Prueba	Observaciones
Paso 1: el usuario selecciona del interfaz un colector a donde será enviado, además se cargará el script para ser enviado al igual que el Sistema Operativo y el día en que se va ejecutar. **Ejemplo:** Sistema Operativo: Windows Colector: Mi_Pc|10.0.0.1 Script: Mi_Script.bat Día: 2015-9-6 **Paso 2:** presiona el botón enviar.		El sistema guarda la petición en la base de datos para luego ser ejecutada.	Satisfactorio.	El modelo del Script debe haber sido insertado previamente. Se ha de seleccionar en la interfaz de listar para poder enviar.
	Paso 1: el usuario deja en blanco de la interfaz el colector a donde será enviado, además se cargará el script para ser enviado al igual que el Sistema Operativo. **Ejemplo:** Sistema	El sistema muestra un mensaje de error informándole al usuario que los campos están vacíos y deben ser llenados.	Satisfactorio.	

	Operativo: Windows Colector: Script: Mi_Script.bat Día: 2015-9-6 **Paso 2:** presiona el botón enviar.			
	Paso 1: el usuario deja en blanco de la interfaz el campo día. El campo del colector a donde será enviado fue llenado, además se cargará el script para ser enviado al igual que el Sistema Operativo. **Ejemplo:** Sistema Operativo: Windows Colector: Script: Mi_Script.bat Día: **Paso 2:** presiona el botón enviar.	El sistema muestra un mensaje de error informándole al usuario que los campos están vacíos y deben ser llenados.	Satisfactorio.	