Universidad de las Ciencias Informáticas

Facultad 2

MÓDULO PARA MONITOREAR EL USO DE SOFTWARE DE LAS COMPUTADORAS EN LA RED CON EL SISTEMA GESTIÓN DE RECURSOS DE HARDWARE Y SOFTWARE

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores:

Caridad Vinent Puig

Yoel Cornell Milián

Tutores:

Ing. Vivian Daylin Alemán Estrada

Ing. Alexander Rodríguez Bonet

La Habana, junio del 2015

"Podemos ver poco del futuro, pero lo suficiente para darnos cuenta de que queda mucho por hacer"

Alan Turing

DECLARACIÓN DE AUTORÍA

Declaro que somos los únicos autores de este trabajo y autorizamos a la Facultad 2 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Caridad Vinent Puig

_____
(Autor)

Yoel Cornell Milián

_____
(Autor)

Alexander Rodríguez Bonet

_____
(Tutor)

Vivian Daylin Alemán Estrada

_____
(Tutor)

DATOS DE CONTACTO

Datos del Autor:
Caridad Vinent Puig
Universidad de las Ciencias Informáticas, La Habana, Cuba.
Email: cvinent@estudiantes.uci.cu

Datos del Autor:
Yoel Cornell Milián
Universidad de las Ciencias Informáticas, La Habana, Cuba.
Email: ycornell@estudiantes.uci.cu

Datos de la Tutor:
Ing. Vivian Daylin Alemán Estrada, graduado de Ingeniero en Ciencias Informáticas. Pertenece al Centro TLM, Dpto. Desarrollo de Aplicaciones.
Universidad de las Ciencias Informáticas, La Habana, Cuba.
Email: vdaleman@uci.cu

Datos del Tutor:
Ing. Alexander Rodríguez Bonet, graduado de Ingeniero en Ciencias Informáticas .Pertenece al Centro TLM, Dpto. Desarrollo de Aplicaciones.
Universidad de las Ciencias Informáticas, La Habana, Cuba.
Email: abonet@uci.cu

DEDICATORIA

Dedico este trabajo a mi familia, amigos, especialmente a mi abue por haberme querido tanto, educado, por estar presente en muchos momentos difíciles y felices de mi vida, a ti abue, te quiero mucho y le doy gracias a DIOS porque hoy estés aquí conmigo.

A mi mamá, por darme siempre su apoyo, por preocuparse por mí y ser una gran madre luchadora, a ti mami gracias por todo, eres lo mejor para mí.

A mi papá, que a pesar de no poder estar aquí, sé que está muy orgulloso de mí por seguir sus pasos para lograr ser una gran profesional. Gracias papi por ser un excelente padre y ejemplo a seguir.

Caridad

Dedico este trabajo especialmente a mi madre por haberme guiado, acompañado y apoyado en todo momento, a mi tía por siempre brindarme su gran cariño, a mi familia y amigos.

Yoel

AGRADECIMIENTOS

Quiero agradecer con profundo amor a mi mamá , Lázaro, mi hermanito Lachy, a mi tío Allen, a mis abuela Carmen ,mis 2 papá Minervino y Dupo, a Marta, Dunia, Lázara, mi tía Rixy, y a toda los que de cierta manera me ayudaron a lograr esta meta en mi vida.

Agradezco eternamente a mi compañero de tesis que desde el inicio confió en mí para llevar a cabo este trabajo de diploma.

A mis tutores, por ser nuestros guías en la realización de este trabajo, por su ayuda incondicional y brindarnos apoyo en todo momento.

A todos mis compañeros de aula, de proyectos, de fiestas, a todos gracias por su apoyo y su amistad, especialmente mi hermanito Reydel, por brindarme su cariño, ayuda, y ser mi compañero de mesa durante los 5 años, a Yoel por quererme tanto y estar siempre ahí para lo que lo necesitara, también eres como un hermano para mí, a mis dos mejores amigas de la Universidad mi enanita Ive, sabes que te quiero mucho mi peque y mi negri Yeny, eres como una hermana para mí.

Al Pelú, el Blade, Osmar, Luisi, Rodri,el Yera, Rolo, el Yasser, Ide, Exxon, Muschet, Erisel, Lisi, Jorge, Melo, Sol, Zure, Claudia, Liyanis, Yoa, Clara, Félix, Jose, Yissel, Diana, Daniel, Victor, Karel, , Dariel, el Pabli, Yorx.

A George gracias por formar parte de mi vida y por todo el apoyo que me has brindado durante mucho tiempo, te quiero mi amor. A todos los que me ayudaron a vencer muchos obstáculos, gracias, Eduardo, César, Argota, Raúl, Alejandro, Yasmany, Franklin, Ronal, Ciro. A todos mis profesores durante los cinco años de la carrera.

A todos, incluso los que me faltaron por mencionar muchas gracias.

Caridad

RESUMEN

Demoré mucho en escribir estos agradecimientos, no quería que se quedara nadie al cual le debo mis más humildes gracias por estar ahí.

Sin dudas los primeros son para mi mamá por todo el apoyo y la comprensión que siempre ha tenido con su único hijo, a Gerardo por siempre soportar mis malcriadeces y ayudarme en todo lo que siempre le consultaba, a mi tía, mi abuela, mi abuelo, mi madrina Barbi y el cascarrabias de Oreste.

Un agradecimiento muy grande a Cesar por ser mi guía en todos estos años de carrera, así mismo a su mamá Álida que me aconsejó y cuidó como una madre, a Yasmany por cargar conmigo en otros de estos tantos momentos, especialmente a Franklin por dotarme de paciencia, ayudarme y hacer por mí cuando estaba ocupado en la tesis, por enseñarme que siempre hay algo nuevo que aprender, por hacerme mejor persona y más fuerte, a Mayté por sus preocupación y sacarme de muchos apuros, a Reydel por estar en todo momento para mí, a Alejandro que ha sido mi bastón, mi ayuda y mi amigo más importante de todos los tiempos. A mi hermanita por siempre Mariam.

Unas líneas en exclusivas para mi compañera de tesis y gran amiga Caridad, las palabras no alcanzarían para describir esta amistad incondicional.

A mis tutores, por ser nuestros guías en la realización de este trabajo, por su ayuda incondicional y brindarnos apoyo en todo momento. Darnos el permiso de molestarlos, casi dejarlos sin comer en ocasiones y hacer que sus horas de sueño fuesen más cortas. Muchas gracias.

A mis compañeros de aula y amigos de la universidad, a mis profes de la carrera especialmente a Aliennis, Antonio, Gendor, Leonardo, Leonid, Bárbara y Yasser.

A todos, incluso los que no mencioné muchas gracias…

Yoel

RESUMEN

RESUMEN:

Mantener un seguimiento y control detallado de los software en una computadora es un aspecto importante para todo tipo de entidad, por lo que es necesario contar con una herramienta capaz de realizar estas acciones. En la actualidad existen varios sistemas que se encargan de monitorear el uso de las aplicaciones en las computadoras, siendo los procesos de cada software una fuente importante para llevar a cabo dicho monitoreo.

Con el fin de satisfacer las necesidades del cliente respecto al tema se ha realizado un estudio de las aplicaciones que realizan el inventario y monitoreo de software en las computadoras, además de analizar y justificar las herramientas y metodologías de desarrollo de software para el diseño e implementación de una solución al problema.

El desarrollo de la investigación está dirigido a la implementación de un módulo para monitorear los software de las computadoras en la red con el sistema Gestión de Recursos de Hardware y Software, capaz de listar procesos pertenecientes a las aplicaciones más utilizadas por los usuarios, así como su tiempo de uso, realizándose la configuración de los procesos del software a monitorear desde el servidor, logrando así una aplicación que cumpla con las expectativas del cliente.

Palabras Clave: módulo, procesos del software a monitorear, Gestión de Recursos de Hardware y Software

ÍNDICE GENERAL

INTRODUCCIÓN 1
CAPITULO 1: FUNDAMENTOS TEÓRICOS NECESARIOS PARA LA IMPLEMENTACIÓN DE UN MÓDULO PARA MONITOREAR EL USO DE SOFTWARE DE LAS COMPUTADORAS EN LA RED CON EL SISTEMA GESTIÓN DE RECURSOS DE HARDWARE Y SOFTWARE. 5
1.1. Introducción 5
1.2. Conceptos fundamentales 5
1.3 Análisis de sistemas similares 6
1.3.1 ManicTime 6
1.3.2 Activity Monitor 6
1.3.3 VEO Ultimate 7
1.3.4 Open Computer and Software Inventory (OCS Inventory) 8
1.5 Sistema de Gestión de Recursos de Hardware y Software (GRHS) 8
1.6 Metodología de Desarrollo 9
1.6.1 Metodología de Desarrollo seleccionada Extreme Programming (XP) 9
1.7 Herramientas y tecnologías 10
1.7.1 Lenguaje de programación 10
1.7.2 Framework de desarrollo 11
1.7.3 Entorno de Desarrollo Integrado (IDE) 11
1.7.4 Lenguaje de Modelado BPMN 12
1.7.5 Lenguaje Unificado de Modelado (UML) 12
1.7.6 Herramienta CASE 12
1.7.7 Sistema Gestor de Base de Datos 13
1.8 Conclusiones del Capítulo 1 14
CAPITULO 2: MÓDULO PARA MONITOREAR EL USO DE SOFTWARE DE LAS COMPUTADORAS EN LA RED CON EL SISTEMA GESTIÓN DE RECURSOS DE HARDWARE Y SOFTWARE. 16
2.1 Introducción 16
2.2 Propuesta de solución 16
2.2.1 Estructura de los plugins para GRHS 17
2.3 Modelado del proceso de negocio del inventario de software 18
2.4 Funcionalidades del sistema 19
2.5 Propiedades del producto 19

2.6 Exploración 20
2.6.1 Historia de usuario 20
2.7 Planificación 22
2.7.1 Estimación de esfuerzo por HU 22
2.7.2 Plan de iteraciones 23
2.7.3 Plan de duración de las iteraciones 24
2.7.4 Plan de entrega 25
2.8 Conclusiones del Capítulo 2 27
CAPÍTULO 3: DISEÑO, IMPLEMENTACIÓN Y PRUEBAS DEL MÓDULO PARA MONITOREAR EL USO DE SOFTWARE DE LAS COMPUTADORAS EN LA RED CON EL SISTEMA GESTIÓN DE RECURSOS DE HARDWARE Y SOFTWARE. 28
3.1 Introducción 28
3.2 Arquitectura 28
3.3 Arquitectura Cliente/Servidor 28
3.4 Patrón Arquitectónico 29
3.5 Patrones de Diseño 32
3.5.1 Patrones GRAPS 32
3.6 Tarjeta CRC (Clase, Responsabilidad y Colaboración) 33
3.7 Diagrama de Paquetes 34
3.8 Modelo Físico de la Base de Datos 35
3.9 Tareas de Ingeniería 37
3.10 Pruebas 38
3.10.1 Pruebas unitarias 39
3.10.2 Pruebas de aceptación 41
3.10.3 Registro de no conformidades 43
3.11 Conclusiones del Capítulo 3 45
CONCLUSIONES GENERALES 47
RECOMENDACIONES 48
REFERENCIAS BIBLIOGRÁFICAS 49
BIBLIOGRAFÍA 53
ANEXOS 57
Anexo I. Modelo de entrevista aplicada a especialistas 57
Anexo II. Historias de usuario. 58

Iteración 1.....58
Iteración 2.....58
Iteración 3.....59
Iteración 4.....62
Iteración 5.....64
Iteración 6.....66

ÍNDICE DE TABLAS

Tabla 1 : Muestra de una Historia de Usuario 20
Tabla 2: HU#2 Realizar monitoreo de los procesos de las aplicaciones activas....22
Tabla 3: Estimación de esfuerzo por HU....23
Tabla 4: Plan de duración de las iteraciones....25
Tabla 5: Plan de entregas....26
Tabla 6 : Tarjeta CRC: Process 33
Tabla 7 : Tarjeta CRC: TProcess 33
Tabla 8: Tarjeta CRC: ProcessCreateManyView....33
Tabla 9: Tarjeta CRC: ProcessResumeView....34
Tabla 10: Tarjeta CRC: Processes....34
Tabla 11: Tarjeta CRC: AppProcess 34
Tabla 12: Realizar monitoreo de los procesos de las aplicaciones activas para Windows 38
Tabla 13: Realizar monitoreo de los procesos de las aplicaciones activas para Linux 38
Tabla 14: Prueba # 5 Adicionar procesos a monitorear....42
Tabla 15: Registro de no conformidades....44
Tabla 16: HU #1 Realizar inventario de los procesos de las aplicaciones activas 58
Tabla 17: HU#3 Enviar inventario de los procesos de las aplicaciones activas 58
Tabla 18: HU#4 Calcular el tiempo que están activos los procesos 59
Tabla 19: HU#5 Almacenar inventario de los procesos de las aplicaciones activas 59
Tabla 20: HU#6 Adicionar nombre de procesos y aplicaciones....59
Tabla 21: HU#7 Listar nombre de procesos y aplicaciones....60
Tabla 22: HU#8 Modificar nombre de procesos y aplicaciones 61
Tabla 23: HU#9 Eliminar nombre de procesos y aplicaciones....61
Tabla 24: HU#10 Adicionar los procesos a monitorear....62
Tabla 25: HU#11 Listar los procesos a monitorear....63
Tabla 26: HU#12 Modificar los procesos a monitorear 63
Tabla 27: HU#13 Eliminar los procesos a monitorear....64
Tabla 28: HU# 14 Mostrar datos de los procesos de las aplicaciones activas por computadoras clientes 64
Tabla 29: HU#15 Mostrar estadísticas generales de los procesos de las aplicaciones activas .65
Tabla 30: HU#16 Exportar a excel las estadísticas generales de los procesos de las aplicaciones

activas 66
Tabla 31: HU#17 Ordenar datos de los procesos.....66
Tabla 32: HU#18 Filtrar por datos de los procesos 67

ÍNDICE DE FIGURAS

Figura 1. Flujo de ejecución del sistema 17
Figura 2. Estructura de los plugins a desarrollar 17
Figura 3. Proceso de negocio del inventario y monitoreo de software 18
Figura 4. Arquitectura Cliente/Servidor 29
Figura 5. Patrón de Arquitectura MTV 30
Figura 6. Capa Modelo 30
Figura 7. Capa Vista 31
Figura 8. Capa Plantilla 31
Figura 9. Diagrama de Paquetes 35
Figura 10. Diagrama de clases persistentes 36
Figura 11. Modelo Entidad Relación 37
Figura 12. Pruebas unitarias 39
Figura 13. Pruebas unitarias 40
Figura 14. Pruebas unitarias 40
Figura 15. Pruebas unitarias 41
Figura 16. Resultados de las pruebas 44

INTRODUCCIÓN

El creciente avance de las Tecnologías de la Información y las Comunicaciones (TICs) como resultado del desarrollo científico-técnico, ha impulsado progresivamente el desarrollo del software y hardware en el mundo. El software como herramienta visual que permite interactuar con el hardware de un equipo, engloba desde pequeñas aplicaciones para llevar a cabo tareas muy específicas, hasta sistemas operativos con capacidad para realizar miles de funciones. El mismo se ha convertido en un elemento clave en la evolución de productos informáticos, que tiene como fin satisfacer las necesidades de la sociedad en general (1).

Actualmente en la industria del software, el desarrollo del mismo constituye una importante ventaja competitiva donde los sistemas resultantes se enfrentan a grandes demandas. Con el beneficio obtenido por estos se ha logrado en las empresas un gran impulso del aprendizaje, la innovación, mejor administración de los recursos, obtención de información con mayor eficiencia, así como descenso de los costos de las operaciones. A pesar de que es muy difícil predecir hasta qué punto puede influir en el futuro las tecnologías informáticas, se hace necesario contribuir al desarrollo de aplicaciones para solventar los problemas que se presentan en la vida cotidiana; donde el número de aplicaciones y servicios que soportan las computadoras hace que cada vez sea más complejo su funcionamiento (2) . Por tal razón son desarrolladas aplicaciones encargadas de monitorear el uso que le brindan los usuarios a los programas en las computadoras.

El monitoreo constante de los medios de hardware y software resulta una tarea vital para la administración o control interno, que tiene como fin analizar su estado, necesidades de actualización, violaciones de normas establecidas y contribuir al buen funcionamiento en una entidad. Las herramientas de monitoreo facilitan de cierta manera el trabajo a los administradores. Directamente permite conocer el uso de los recursos en el tiempo, el estado de las aplicaciones o servicios, entre otros elementos. Indirectamente permite proyectar la adquisición de nuevos equipos o partes, prevenir futuros problemas, incluso solucionar problemas actuales (3), en relación con lo anterior se considera de manera general que en las computadoras, el monitoreo permite conocer si todos los componentes de hardware y software están disponibles y trabajando de acuerdo a lo esperado.

En el Centro de Telemática (TLM) de la Universidad de las Ciencias Informáticas (UCI) se

desarrolló el sistema Gestión de Recursos de Hardware y Software (GRHS). Este sistema posee una estructura cliente-servidor que automatiza los procesos de obtención de información de hardware y software en redes de computadoras, y dispone de un sistema de gestión de alertas. Además permite monitorear todo el equipamiento de un ordenador y gestiona adecuadamente el inventario de software a través de la red.

Este sistema permite obtener del inventario de software que realiza, datos del sistema operativo, el antivirus, programas instalados, dominio y usuarios registrados en las computadoras, pero no permite controlar el tiempo de uso de las aplicaciones, así como el uso en porciento de la memoria RAM (Random Access Memory), además no existe ninguna funcionalidad para la configuración de los procesos de software que se quieren monitorear y la configuración del estado de inactividad (tiempo establecido para conocer los software de poco uso) y sobreactividad (tiempo establecido para conocer los software muy usados) de los mismos. Además se instalan en las computadoras software que luego no se utilizan, ocupando espacio y capacidad de rendimiento. También se emplean software de ocio descontroladamente en horario de trabajo.

Conocer esta información sería muy útil para cualquier entidad ya que permitiría tomar decisiones respecto al hardware que necesita el ordenador donde se ejecuten dichos software, y controlar que no se violen las normas establecidas para el uso de las TIC en cada área, con el objetivo de mejorar la forma en cómo los trabajadores desempeñan su trabajo.

Teniendo en cuenta lo expuesto anteriormente se plantea el siguiente **problema a resolver:** ¿Cómo monitorear los software de las computadoras en la red?
En correspondencia con el problema planteado, se define como **objeto de estudio** de la presente investigación: proceso de monitoreo de los software instalados de las computadoras en la red.
Con el propósito de solucionar el problema planteado se define como **objetivo general** de la investigación: Desarrollar un módulo que permita monitorear los software instalados de las computadoras en la red para el sistema GRHS.
El **campo de acción** se enmarca en el monitoreo de software instalados de las computadoras en la red con el sistema GRHS.
De acuerdo con el problema planteado, la **idea a defender** es la siguiente: Si se desarrolla un módulo para el sistema GRHS que permita monitorear los software instalados de las computadoras en la red, se incrementa el control de las aplicaciones en el centro de Telemática.

Para dar cumplimiento al objetivo planteado se desglosan las siguientes **tareas de investigación:**

- ✓ Definición de los principales conceptos asociados al monitoreo de los procesos de los software para una mejor comprensión de los mismos.
- ✓ Estudio del estado del arte sobre los sistemas de monitoreo de los software en las computadoras para identificar las necesidades del sistema.
- ✓ Justificación de metodología, herramientas y tecnologías a utilizar para el desarrollo del módulo.
- ✓ Identificación de los procesos de negocio relacionados con el módulo para monitorear el uso de los software en las computadoras en la red con el sistema GRHS para un correcto diseño de los mismos.
- ✓ Desarrollo de las funcionalidades del módulo para el monitoreo del uso de los software en la red.
- ✓ Definición de los tipos de pruebas para comprobar el funcionamiento del módulo.

Para el desarrollo de esta investigación y el logro de su objetivo, se utilizaron los siguientes métodos teóricos y empíricos:

Métodos Teóricos:

- ✓ **Histórico-Lógico:** Utilizado para conocer con mayor profundidad los antecedentes y los sistemas encargados de monitorear los software en las computadoras, facilitando de forma general conocer el funcionamiento y desarrollo de los mismos.
- ✓ **Análisis-Síntesis:** Este método se empleó para el análisis de las bibliografías consultadas sobre las aplicaciones para el monitoreo de los software en las computadoras, en la justificación de las herramientas y tecnologías a utilizar para el desarrollo de la aplicación, además de la integración de las ideas fundamentales de la información obtenida.
- ✓ **Modelación:** Se empleó para la creación de modelos, facilitando la definición de los componentes del módulo, así como sus relaciones.

Métodos Empíricos:

- ✓ **Observación:** Se utilizó para observar el funcionamiento del sistema, así como los resultados obtenidos de las pruebas realizadas.
- ✓ **Entrevista:** Este método se empleó para determinar las necesidades del cliente. Para ello se realizó una serie de entrevistas con el cliente involucrado con el sistema GRHS y en

base a éstas se trabajó para satisfacer sus necesidades.

Los capítulos que conforman el informe de investigación tienen la siguiente estructura:

Capítulo 1: Fundamentos teóricos necesarios para la implementación de un módulo para monitorear el uso de software de las computadoras en la red con el sistema Gestión de Recursos de Hardware y Software:

En este capítulo se realizó la investigación y el estudio del estado del arte, analizando los sistemas similares encargados del monitoreo de los software en las computadoras. Además se realizó una justificación de la metodología, herramientas y tecnologías que se utilizaron para el desarrollo del módulo.

Capítulo 2: Módulo para monitorear el uso de software de las computadoras en la red con el sistema Gestión de Recursos de Hardware y Software:

En este capítulo se abordan las diferentes características que presenta el sistema, siguiendo los aspectos que plantea la metodología XP en sus fases de Exploración y Planificación, obteniendo así los artefactos necesarios para la implementación.

Capítulo 3: Diseño, implementación y pruebas del módulo para monitorear el uso de software de las computadoras en la red con el sistema Gestión de Recursos de Hardware y Software:

En este capítulo se definió la arquitectura del sistema y los patrones arquitectónicos y de diseño empleados. Se organizan las clases fundamentales del sistema utilizando las tarjetas Clase-Responsabilidad-Creador (CRC). Posteriormente se realizan las tareas de ingeniería en correspondencia con las historias de usuario (HU) desarrolladas anteriormente. Por último se realizan las pruebas para verificar el cumplimiento de todas las funcionalidades.

CAPITULO 1: FUNDAMENTOS TEÓRICOS NECESARIOS PARA LA IMPLEMENTACIÓN DE UN MÓDULO PARA MONITOREAR EL USO DE SOFTWARE DE LAS COMPUTADORAS EN LA RED CON EL SISTEMA GESTIÓN DE RECURSOS DE HARDWARE Y SOFTWARE.

1.1. Introducción

En el presente capítulo se resaltan los principales conceptos relacionados con el tema que se desarrolla en este trabajo. Se hace un estudio de sistemas similares existentes en el mundo que se encargan de monitorear los software que utilizan los usuarios en las computadoras. Además se profundiza sobre la metodología, herramientas y tecnologías a utilizar, así como las características de los mismos para el desarrollo del módulo propuesto.

1.2. Conceptos fundamentales

Software: Es un conjunto de programas, instrucciones y reglas informáticas que permiten ejecutar distintas tareas en una computadora (4). Cada programa que se está ejecutando en un momento dado se suele llamar proceso.

Proceso: El contexto de un programa que está en ejecución es lo que se llama un proceso. Este contexto puede ser más procesos hijos que se hayan generado del principal (proceso padre), los recursos del sistema que esté consumiendo, sus atributos de seguridad (tales como su propietario y permisos de archivos así como roles), entre otros (5) , o sea un software al estar en ejecución tendrá un proceso padre que genera más procesos hijos, y cada orden que se ejecute en un ordenador iniciará un proceso nuevo. Los procesos de los software serán la principal fuente para obtener determinada información de los software en las computadoras.

Monitoreo: Es la observación mediante aparatos especiales el curso de uno o varios parámetros fisiológicos o de otra naturaleza para detectar posibles anomalías (6). El monitoreo es una forma de evaluación, que permite determinar en una computadora qué software o componentes están funcionando y cuáles no, logrando de esta forma hacer ajustes para su mejor desempeño.

Monitoreo de procesos: El monitoreo de procesos, a rasgos generales, consiste en la observación del curso de uno o más parámetros para detectar eventuales anomalías en el comportamiento de los procesos, además de realizar acciones que permitan obtener información de los procesos (5). Algunos de los parámetros son el identificador, la fecha, el usuario que ejecuta el proceso, el comando que lanzó el proceso, el consumo de los recursos del sistema

como la memoria (memoria virtual y memoria residente en el sistema), porciento de uso del procesador, entre otros.

1.3 Análisis de sistemas similares

En la actualidad existen varias aplicaciones que permiten realizar el monitoreo del uso de los software en las computadoras. A continuación se reflejan algunas características importantes de estas herramientas necesarias para la solución a obtener:

1.3.1 ManicTime

ManicTime es un programa que permite registrar automáticamente qué aplicaciones se están usando en un ordenador, a qué hora y durante cuánto tiempo, así como los períodos de actividad e inactividad. Manic Time registra en varias líneas de tiempo gráficas lo que se hace en el computador, en todo momento, en una línea de tiempo registra la aplicación que está activa en cada momento, y en otra apunta el nombre del documento que se tenga abierto o la página web que se esté visitando. Además permite desplegar un historial detallado con todas las acciones recientes y su duración. Seleccionando uno o más elementos de este historial, se muestra el espacio que ocupan en la línea de tiempo mediante un tono sombreado. Otra opción que brinda este programa es exportar la base de datos recopilada por el programa a un archivo, y así asegurar de no perder los datos (7).

A pesar de que Manic Time es una alternativa para llevar un registro detallado de qué se hace realmente cuando se está frente a un ordenador, no se considera como solución al problema investigativo debido a que solo funciona en sistemas operativos Windows XP y Vista y es un software propietario, puesto que no se puede acceder a su código fuente, dificultando de esta forma la integración a la solución.

1.3.2 Activity Monitor

Activity Monitor es una herramienta privativa que se adapta a las necesidades del cliente ajustando funcionalidades y precios. El software permite supervisar cualquier red de área local, facilitándole una información detallada sobre lo que hacen los usuarios de la red. El programa se compone de dos partes: servidor y cliente. La parte de servidor de Activity Monitor puede instalarse en cualquier ordenador de la red de área local. El programa espía remoto (Agente) es la parte cliente del programa que se instala en todos los ordenadores de la red que desee controlar (8). Dicha herramienta brinda una serie de ventajas y funcionalidades (referentes al

monitoreo de software):

- ✓ Aumenta la productividad de sus empleados inmediatamente.
- ✓ Módulos inteligentes para análisis y generación de informes.
- ✓ Funciona con todas las plataformas modernas de Windows.
- ✓ Base de datos central de registros que sirve para guardar datos de todos los ordenadores en la ubicación única.
- ✓ La parte supervisora del programa que está instalada en el ordenador controlado funciona en un modo inadvertido para los usuarios. Los usuarios no pueden cerrar el Agente de Activity Monitor en el administrador de tareas sin tener derechos de administrador.
- ✓ Permite ver qué aplicaciones se están ejecutando.
- ✓ Registra cuando se inicia la aplicación, se detuvo, y el tiempo que se utiliza realmente.

A pesar de todas las ventajas que posee Activity Monitor, el conjunto de funcionalidades dedicadas a la obtención de información y monitoreo de software no se consideró como solución al problema investigativo ya que, aun presentando soluciones necesarias, no se puede instalar en otra plataforma que no sean versiones de Windows y es un software propietario.

1.3.3 VEO Ultimate

VEO Ultimate es una aplicación para administrar los equipos en una red de computadoras con protocolo TCP / IP. Permite a los gerentes y directores de empresas, aprovechar al máximo sus recursos de cómputo. Entre sus principales funcionalidades se tienen:

- ✓ Medir la productividad de los usuarios de las computadoras.
- ✓ Obtener inventarios de software.
- ✓ Obtiene un listado de las ventanas abiertas.
- ✓ Explora los archivos y restringe el uso de aplicaciones en determinados horarios.
- ✓ Despliega mensajes cuando se realice algo indebido así como cerrar las aplicaciones que en ese momento están en ejecución.

Reúne varias características y tecnologías que facilitan el trabajo a los administradores de red además del beneficio que tiene en el trabajo en las empresas los cuales se muestran a continuación:

- ✓ Planear actualizaciones de sistemas operativos u otros programas, así como estrategias de cambios de equipos en base a sus características.

- ✓ Controlar accesos a programas que no agregan valor a la empresa, con las tareas programadas se puede restringir el acceso a aplicaciones (panel de control, juegos, chat, entre otras) (9) .

A pesar de todas las ventajas que brinda esta herramienta, el conjunto de funcionalidades dedicadas a la obtención de información y monitoreo de software no se consideró como solución debido a que, obtiene solamente el inventario de ordenadores con sistema operativo Windows, aspecto a valorar por los administradores de red a la hora de la selección de una herramienta de este tipo, además es un software propietario y no obtiene el tiempo de uso de las aplicaciones, funcionalidad necesaria para el módulo propuesto.

1.3.4 Open Computer and Software Inventory (OCS Inventory)

OSC Inventory es un sistema para mantener el inventario de máquinas de forma fácil y automatizada. Es un programa sencillo pero muy útil, permite tener un inventario centralizado de software y hardware (10) . Dispone en forma rápida y completa de los datos de los equipos y su relación con los usuarios. Es un sistema multiplataforma que en un servidor recopila la información que le envía un software agente instalado en cada uno de los dispositivos de la subred. Algunas de las características presentes en este sistema son:

- ✓ El agente debe ser instalado y configurado en cada servidor o computadora a ser inventariada.
- ✓ La interfaz web muestra información referente al software: sistema operativo, aplicaciones instaladas.
- ✓ Soporte para múltiples sistemas operativos, incluyendo Microsoft Windows y Linux (11).

La herramienta es ideal para trabajar con computadoras en red, aun así no brindan la totalidad de las funcionalidades que necesita el cliente, como: la obtención del tiempo de uso de las aplicaciones. Además es un sistema desarrollado con tecnologías y herramientas diferentes a las empleadas en el módulo propuesto, por lo cual se hace complejo integrarlo a la solución.

1.5 Sistema de Gestión de Recursos de Hardware y Software (GRHS)

GRHS es un sistema que permite la captura, análisis y consulta de la información de los activos informáticos en una red. Es una plataforma cliente-servidor, que posee un diseño modular, y utiliza el protocolo HTTP y HTTPS para la comunicación entre estos. Posee, además versiones para Windows y GNU/Linux. Es una herramienta desarrollada con tecnologías libres. Aporta una

gran cantidad de información sobre el inventario de los recursos de hardware y software de una red ya que integra varias bibliotecas para la obtención de la misma. Permite la ejecución de acciones posibilitando tomar el control de los ordenadores en caso de incidencia. Esta herramienta es configurable en cuanto a qué es una incidencia, cuándo ocurre y qué acciones tomar en determinada situación. Permite agregarle funcionalidades al sistema en todas sus aplicaciones sin modificar las existentes (12).

Actualmente permite obtener los datos del sistema operativo, el antivirus, programas instalados, dominio y las aplicaciones instaladas en las computadoras, pero no permite monitorear el uso de estas aplicaciones.

1.6 Metodología de Desarrollo

"La Ingeniería de Software es el establecimiento y uso de principios sólidos de ingeniería, orientados a obtener software económico que sea fiable y trabaje de manera eficiente en máquinas reales" (13).

Una metodología define una representación que permite facilitar la manipulación de modelos, y la comunicación e intercambio de información entre todas las partes involucradas en la construcción de un sistema (14). Las metodologías se pueden clasificar como ágiles y tradicionales. Las tradicionales son aquellas que se centran fundamentalmente en el control del proceso, además son las más efectivas para proyectos de gran tamaño. La ingeniería de software ágil representa una opción razonable a la ingeniería convencional para ciertas clases de software y ciertos tipos de proyectos. Ha demostrado su utilidad al entregar sistemas exitosos con rapidez.

1.6.1 Metodología de Desarrollo seleccionada Extreme Programming (XP)

XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes (15).

En términos generales XP es una metodología adecuada en proyectos medianos y pequeños, donde los equipos de desarrollo no pasan de 10 programadores. Una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. El proyecto se inicia

con la fase de exploración donde el cliente redacta las historias de usuario (HU) y los desarrolladores calculan los puntos estimados de cada HU. El plan de entrega a seguir es concebido durante la planificación, con la participación del cliente y los desarrolladores que estiman el esfuerzo total y la velocidad del equipo. A continuación tiene lugar una serie de iteraciones que no concluyen hasta obtener una primera entrega del sistema. En la fase de producción se completa y actualiza el plan de entrega, el cliente y los desarrolladores realizan pruebas continuas al sistema (16) .

Basado en estas características y teniendo en cuenta que el equipo de desarrollo está integrado por 2 personas en su totalidad, que el cliente participa con el equipo de desarrollo constantemente, que los requisitos están sujetos a cambios y que el proyecto es considerado pequeño y de corto plazo, se hizo decisiva la utilización de XP para el desarrollo de la solución.

1.7 Herramientas y tecnologías

En el proceso de desarrollo de software es fundamental la correcta elección de las tecnologías y herramientas que mejor se adapten y mayores facilidades brinden para dar solución al problema de la investigación. Para la realización de esta investigación fueron analizadas la metodología de desarrollo, las herramientas y tecnologías a utilizar, así como sus principales características, que se ajustan a la solución a desarrollar.

1.7.1 Lenguaje de programación

Los lenguajes de programación permiten a los desarrolladores crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador, para que este pueda comunicarse con los dispositivos de hardware y software existentes.

La elección de un lenguaje de programación que sea capaz de adaptarse a las necesidades del producto que se quiera desarrollar de forma eficiente, no siempre es sencillo, pues existe la duda en cuanto a, si un determinado lenguaje es mejor que otro en lo referente a características y calidad. Todo esto indica que la elección se debe limitar a escoger aquel lenguaje que se asemeje a las características propias del sistema que se quiera desarrollar.

1.7.1.1 Python 2.7

Python es un lenguaje de programación que soporta múltiples paradigmas, incluyendo programación orientada a objetos, programación imperativa y funcional. El código Python define objetos incorporados como listas enlazadas, tuplas, tablas hash y enteros de longitud arbitraria

(17).

Teniendo en cuenta que la Línea de Desarrollo del Centro TLM define las tecnologías con que se desarrollarán sus productos, exceptuando los casos en que el cliente las determine, se decidió utilizar para la solución propuesta en conjunto con el framework Django a Python en su versión 2.7 debido al soporte que brinda, su integración con otros lenguajes y herramientas, y por la razón que los plugins a desarrollar son para un sistema hecho en Python. Además de que permite la integración de numerosas librerías que facilitarán la implementación de algunas funcionalidades del módulo, es la versión utilizada en el desarrollo del sistema GRHS.

1.7.2 Framework de desarrollo

Un framework es un esquema para el desarrollo y/o la implementación de una aplicación, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto. Está orientado a la reutilización de componentes permitiendo así facilidad tanto en el desarrollo como mantenimiento de aplicaciones. Teniendo en cuenta las ventajas de su uso se determinó como framework a emplear para la implementación del módulo a Django 1.4.

1.7.2.1 Django 1.4

Django es un entorno de desarrollo web de código abierto escrito en Python que permite construir aplicaciones web más rápido y con menos código. Usa una modificación de la arquitectura Modelo-Vista-Controlador (MVC), llamada MTV (Model – Template – View), que sería Modelo-Plantilla-Vista, esta forma de trabajar permite que sea pragmático (18). Teniendo en cuenta las ventajas de su uso y que es utilizado en el sistema GRHS, el framework se utiliza para el desarrollo de la interfaz web en su versión 1.4.

1.7.3 Entorno de Desarrollo Integrado (IDE)

Un IDE es un programa que le brinda facilidades al programador, tales como: un editor de código, un compilador, un depurador, y opcionalmente un constructor de interfaz gráfica.

1.7.3.1 PyCharm 3.4

PyCharm es un entorno de desarrollo integrado, multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica, y soporte

para el desarrollo web con Django. Dentro de todas las ventajas de PyCharm se encuentran: autocompletado, resaltador de sintaxis, herramienta de análisis y refactorización, la integración con lenguajes de plantillas como Django Template. Además soporta entornos virtuales e intérpretes de Python 2.x, 3.x, PyPy, Iron Python y Jython (19). Se emplea este IDE ya que el módulo que se implementa utiliza el lenguaje Python y como framework a Django, además de los beneficios que brinda resaltados anteriormente.

1.7.4 Lenguaje de Modelado BPMN

Business Process Modeling Notation (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. BPMN permite la comprensión de los procesos de negocio de la empresa, para mejorarlos proporciona un lenguaje común para que las partes involucradas puedan comunicar los procesos de forma clara, completa y eficiente (20). Esta notación gráfica se utiliza en el trabajo para el modelado del proceso de negocio, creando un vínculo entre las etapas de diseño e implementación.

1.7.5 Lenguaje Unificado de Modelado (UML)

El Lenguaje Unificado de Modelado (UML), en inglés Unified Modeling Language, es el lenguaje de modelado que permite especificar, visualizar y construir los artefactos de los sistemas informáticos. Está destinado a los sistemas de modelado que utilizan conceptos orientados a objetos. Lo fundamental de una herramienta UML es la capacidad de diagramación, y los diferentes tipos de diagramas que soporta la herramienta (21).

UML como lenguaje de modelado proporcionando técnicas y especificaciones para la creación de diagramas que sirven de guías en el proceso de desarrollo del software, por lo que se decide utilizarlo para la realización de los diferentes diagramas necesarios en el proceso de desarrollo del módulo: diagrama de paquetes y diagrama de clases persistentes.

1.7.6 Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering por sus siglas en inglés) son empleadas con el fin de automatizar los aspectos claves de todo el proceso de desarrollo de software de un sistema (22). Para el desarrollo de la propuesta de solución se emplea el Visual Paradigm en su versión 8.0; es una herramienta profesional empleada para el modelado de los diferentes diagramas utilizados en el desarrollo del módulo, ejemplo: modelado de proceso del

negocio, diagrama de paquetes, tarjetas CRC (Clase-Responsabilidad-Colaboración), modelo físico de la base de datos y diagrama de clases persistentes.

1.7.7 Sistema Gestor de Base de Datos

Un sistema gestor de base de datos (SGBD) es un conjunto de programas que administran y gestionan la información contenida en una base de datos. Ayuda a realizar las siguientes acciones:

- ✓ Definición de los datos.
- ✓ Mantenimiento de la integridad de los datos dentro de la base de datos.
- ✓ Control de la seguridad y privacidad de los datos.
- ✓ Manipulación de los datos (23).

1.7.7.1 PostgreSQL 9.2

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (24). Algunas de las características más importantes a destacar son:

- ✓ Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes, cadenas de bits. También permite la creación de tipos propios.
- ✓ Permite la declaración de funciones propias, así como la definición de disparadores.
- ✓ Incluye herencia entre tablas, por lo que a este gestor de base de datos se le incluye entre los gestores objeto-relacionales.
- ✓ Multiplataforma (25).

Para la realización de este trabajo se determinó usar PostgreSQL debido a que es el gestor de base de datos que utiliza el sistema GRHS para la gestión de los datos en el servidor, al cual va a estar integrado la solución propuesta.

1.7.7.2 SQLite

SQLite es un SGBD de dominio público. Una de las principales características por las que se destaca SQLite es por su completo soporte para tablas e índices en un único archivo por bases de datos, soporte transaccional, rapidez y su completa portabilidad. Algunas de sus principales

ventajas son:

- ✓ Es un proyecto de dominio público.
- ✓ Las bases de datos se guardan en forma de ficheros, por lo que es posible trasladar sin problemas una base de datos(o fichero) a cualquier dispositivo que tenga instalado SQLite.
- ✓ Es multiplataforma.

SQLite dispone de una completa interfaz orientada a objetos, con distintas funciones que facilitan la manipulación de datos (26). Para la realización de este trabajo se determinó usar SQLite debido a que es el gestor de base de datos que utiliza el sistema GRHS para la gestión de los datos en cada cliente.

1.7.7.3 PgAdminIII

PgAdminIII es una aplicación de diseño y manejo de base de datos para su uso con PostgreSQL. Este software fue diseñado para responder a las necesidades de todos los usuarios, desde la escritura de simples consultas SQL a la elaboración de base de datos complejas. La interfaz gráfica es compatible con todas las características de PostgreSQL y facilita la administración. La aplicación se puede utilizar para manejar PostgreSQL 7.3 y superiores, y está disponible para varios sistemas operativos (27).

1.8 Conclusiones del Capítulo 1

En este capítulo se analizaron los conceptos y definiciones relacionados con el estudio del estado del arte sobre sistemas existentes relacionados con el monitoreo de los software en las computadoras en una red de datos. Se logró obtener las características, ventajas y desventajas de estos sistemas, sirviendo de apoyo para el desarrollo del módulo que permitirá la supervisión mediante la red de los programas que utilizan los usuarios, el tiempo que le dedican a estos, entre otros aspectos, llegando a la conclusión de que ninguno ofrece solución a la totalidad de los objetivos propuestos.

Se ha logrado definir las herramientas y tecnologías necesarias a utilizar en el desarrollo del módulo, definiendo a XP como metodología de desarrollo, para el modelado de los procesos del negocio se utilizará el Visual Paradigm en su versión 8.0 y como lenguajes de modelado UML y BPMN. Para la implementación se empleará el lenguaje de programación Python en su versión 2.7; como framework de desarrollo Django 1.4 y el IDE de desarrollo PyCharm 3.4. Además se

hará uso del gestor de base de datos PostgreSQL 9.2 por el lado del servidor, SQLite del lado del cliente, y el PgAdminIII como gestor de PostgreSQL.

CAPITULO 2: MÓDULO PARA MONITOREAR EL USO DE SOFTWARE DE LAS COMPUTADORAS EN LA RED CON EL SISTEMA GESTIÓN DE RECURSOS DE HARDWARE Y SOFTWARE.

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. Se realiza el modelado del proceso de negocio para mejor entendimiento del sistema. Además, se describen las funcionalidades a tener presentes para el desarrollo del módulo.

Se siguen los aspectos que plantea la metodología ágil XP en sus fases de Exploración y Planificación, donde se confeccionan las historias de usuarios (HU), encargadas de la especificación de los requerimientos del módulo a desarrollar, realizándose la estimación del esfuerzo para cada una de ellas. Seguidamente se realiza el plan de iteraciones y de entrega definiéndose la duración de cada iteración y fechas de entrega del producto al cliente.

2.2 Propuesta de solución

El módulo que se propone en la presente investigación estará formado por plugins que serán integrados al sistema GRHS. Su objetivo es, una vez instalado en las computadoras clientes, contribuir a la gestión de inventario que éste realiza para obtener información de los procesos de cada software que se utiliza en las áreas de trabajo. Para desarrollar la propuesta de solución se implementarán plugins para los sistemas operativos (SO) Windows y GNU/Linux.

El flujo de ejecución del módulo inicia en las computadoras clientes cuando se obtiene la configuración de los procesos a monitorear del servidor. Se capturan todos los procesos que están en ejecución, los cuales son filtrados según la configuración obtenida, para posteriormente enviar el inventario de los procesos de software al servidor. Seguidamente se inicia un monitoreo constante para detectar la ejecución o cierre de algún proceso configurado, en caso de detectarse algún cambio se realiza nuevamente el inventario y se envía al servidor. El mismo obtiene la información de cada cliente y la registra o actualiza en la base de datos teniendo en cuenta el cálculo del tiempo de uso de los procesos, para luego ser mostrados en el subsistema GADMIN, quien se encarga de presentarle la información al usuario. A continuación se muestra una imagen con el flujo de ejecución del funcionamiento del módulo para un mejor entendimiento de la propuesta de solución:

Figura 1. Flujo de ejecución del sistema

2.2.1 Estructura de los plugins para GRHS

Los plugins para el sistema GRHS están compuestos por una serie de ficheros y directorios que le brindan su estructura y función, así como la comunicación con el sistema GRHS. A continuación se muestra una imagen de elaboración personal, que define la estructura de la propuesta de solución:

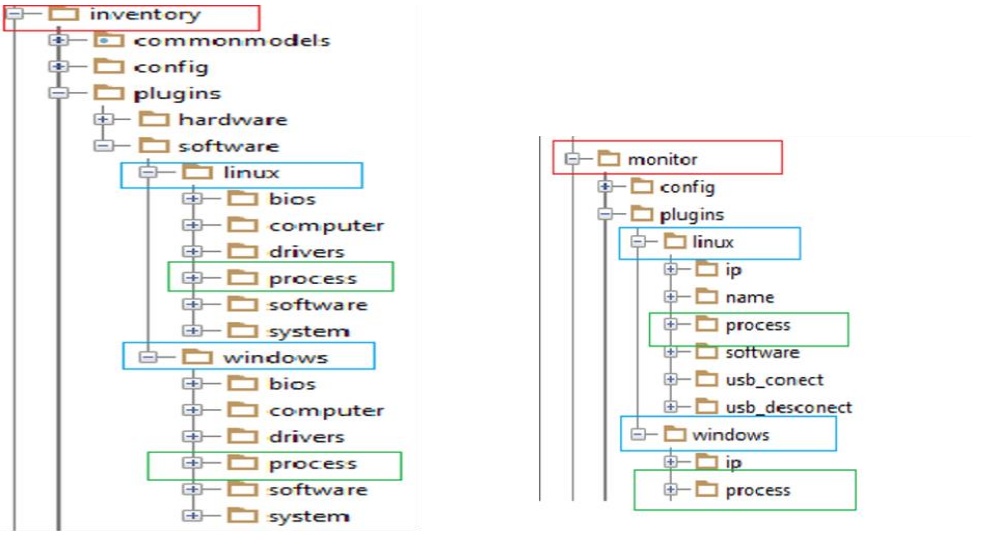


Figura 2. Estructura de los plugins a desarrollar

En la imagen se resalta en color rojo la posición que deben tener los plugins a desarrollar dentro del sistema GRHS, en azul la ubicación dentro de cada uno de los sistemas operativos que serán soportados y en verde, los plugins a desarrollar, que estarán constituido por dos ficheros .py, donde en uno se implementa el modelo y en el otro las funcionalidades que permiten obtener la información en cada computadora cliente.

2.3 Modelado del proceso de negocio del inventario de software

En la siguiente imagen se presenta un modelo detallado del negocio utilizando BPMN, para proveer una notación estándar que sea fácilmente legible y entendible por parte de todos los involucrados e interesados del negocio:

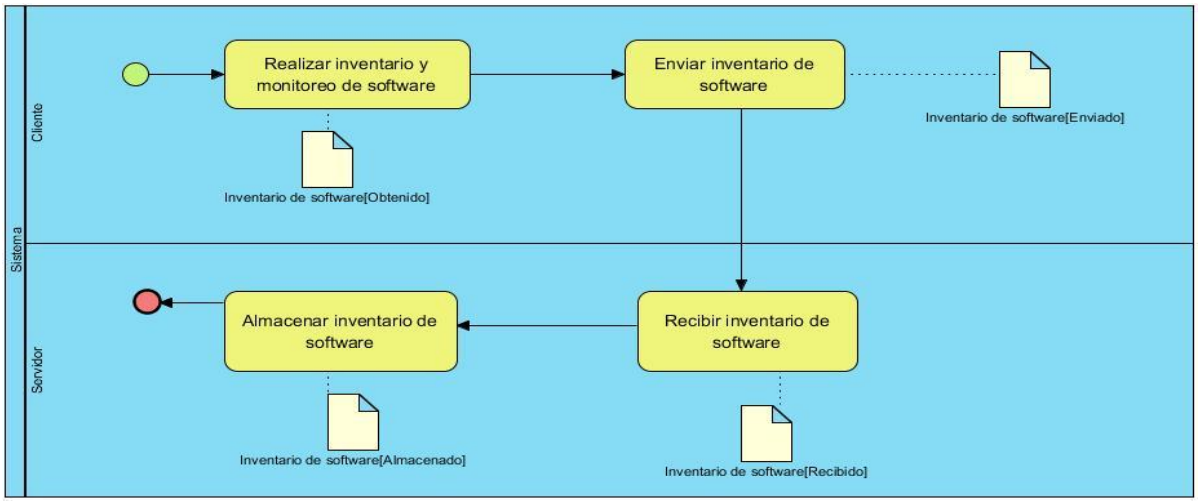


Figura 3. Proceso de negocio del inventario y monitoreo de software

Realizar inventario y monitoreo de software: Consiste en la recolección de información referente al sistema operativo en el que corre la aplicación, programas instalados, antivirus, usuarios conectados, dominio de la computadora, información del bios, además de monitorear los programas, para conocer información de aquellos que se instalan o desinstalan.

Enviar inventario de software: El cliente establece una conexión con el servidor para enviarle la información del inventario realizado.

Recibir inventario de software: El servidor recibe un inventario con la información del software obtenida de los ordenadores.

Almacenar inventario de software: En el servidor de base de datos se almacenan los inventarios enviados por cada cliente, pudiendo el administrador a través de una consola acceder a todos estos datos.

2.4 Funcionalidades del sistema

- ✓ Realizar inventario de los procesos de las aplicaciones activas.
- ✓ Realizar monitoreo de los procesos de las aplicaciones activas.
- ✓ Enviar inventario de los proceso de las aplicaciones activas.
- ✓ Calcular el tiempo que están activos los procesos.
- ✓ Almacenar inventario de los proceso de las aplicaciones activas.
- ✓ Gestionar nombre de procesos y aplicaciones.
    1. Adicionar nombre de procesos y aplicaciones.
    2. Listar nombre de procesos y aplicaciones.
    3. Modificar nombre de procesos y aplicaciones.
    4. Eliminar nombre de procesos y aplicaciones.
- ✓ Gestionar los procesos a monitorear.
    1. Adicionar los procesos a monitorear.
    2. Listar los procesos a monitorear.
    3. Modificar los procesos a monitorear.
    4. Eliminar los procesos a monitorear.
- ✓ Mostrar datos de los procesos de las aplicaciones activas por computadoras clientes.
- ✓ Mostrar estadísticas generales de los procesos de las aplicaciones activas.
- ✓ Exportar a excel las estadísticas generales de los procesos de las aplicaciones activas.
- ✓ Ordenar datos de los procesos.
- ✓ Filtrar por datos de los procesos.

2.5 Propiedades del producto

Portabilidad

El sistema debe ser compatible con los Sistemas Operativos: Windows y GNU/Linux.

Interfaz

El módulo propuesto poseerá una interfaz que cumpla con las pautas de diseño elaboradas por la UCI dirigida a las personas que se relacionen con el sistema.

Soporte

Se realizará un período de prueba a la aplicación para observar su funcionamiento con el objetivo de detectar vulnerabilidades y corregirlas.

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 o 7

Nova

- ✓ Las computadoras cliente del sistema y ubicadas en el dominio de la organización deben tener instaladas Python v2.7.
- ✓ 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.
- ✓ Es necesario el Navegador Mozilla Firefox para acceder a las interfaces del sistema.

2.6 Exploración

La exploración es la etapa del proceso de desarrollo de software para comenzar la construcción de un producto. Es aquí donde los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto, así como la familiarización de los equipos de trabajo con las tecnologías y herramientas seleccionadas para la construcción del módulo.

2.6.1 Historia de usuario

Las historias de usuarios (HU) son la técnica utilizada en XP para representar los requisitos del software en pocas líneas, en los que el cliente detalla una actividad que realizará el sistema de forma sencilla y clara (28).Son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Permiten responder rápidamente a los requisitos cambiantes. Las HU son representadas mediante tablas divididas por diferentes secciones, a continuación un ejemplo:

Tabla 1 : Muestra de una Historia de Usuario

Historia de usuario

Número: (Número de la HU Incremental en el tiempo)	**Usuario:** (Se citan las personas encargadas de interactuar con la HU)
Nombre de Historia de Usuario: (El nombre de la HU sería para identificarlas mejor entre los desarrolladores y el cliente)	
Prioridad en negocio: (Alta / Media / Baja)	**Riesgo en Desarrollo:** (Alta / Media / Baja)
Puntos estimados: (El tiempo estimado en semanas que se demorará el desarrollo de la HU)	**Iteración Asignada:** (Número de la iteración)
Programador(es) responsable(s): (Se citan los desarrolladores responsables de la implementación de la HU)	
Descripción: (Breve descripción de la HU)	
Observaciones: (Señalamiento o advertencia del sistema)	
Prototipo de interfaz: (Prototipo de interfaz si aplica)	

Se define por el cliente, la prioridad en el negocio y por el equipo, el riesgo en el desarrollo de cada HU.

La prioridad en el negocio:

- ✓ **Alta:** Se le otorga a las HU que resultan funcionalidades fundamentales en el desarrollo del sistema, a las que el cliente define como principales para el control integral del sistema.
- ✓ **Media:** Se le otorga a las HU que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.
- ✓ **Baja:** Se le otorga a las HU que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el sistema en desarrollo (29).

El riesgo en su desarrollo:

- ✓ **Alto:** Cuando en la implementación de las HU se considera la posible existencia de errores que lleven la inoperatividad del código.
- ✓ **Medio:** Cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión.

- ✓ **Bajo:** Cuando pueden aparecer errores que serán tratados con relativa facilidad sin que aporten problemas mayores para el desarrollo del proyecto (29).

A continuación se expone una muestra de las HU que presentan alta prioridad en el negocio. Las restantes se encuentran en el Anexo II.

Tabla 2: HU#2 Realizar monitoreo de los procesos de las aplicaciones activas

Historia de usuario	
Número: 2	**Usuario:** Sistema
Nombre de Historia de Usuario: Realizar monitoreo de los procesos de las aplicaciones activas	
Prioridad en negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos estimados: 1	**Iteración Asignada:** 1
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: Monitorea los procesos de las aplicaciones activas, y en caso de detectar algún cambio (ejecución o cierre de una aplicación), ejecutar el inventario de los procesos de las aplicaciones activas.	
Observaciones:	
Prototipo de interfaz: No aplica	

2.7 Planificación

En la fase de planificación de la metodología XP, el cliente establece la prioridad de cada historia de usuario y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas (30). Este se expresa utilizando el método de estimación por punto. Un punto se considera como una semana ideal de trabajo, planificada para cinco días. Esta estimación incluye todo el esfuerzo asociado a la implementación de las HU, basada en desarrollos anteriores de los programadores, tiempo disponible y referencia de otros proyectos.

2.7.1 Estimación de esfuerzo por HU

A continuación se muestra la estimación del esfuerzo por cada historia de usuario propuesta para el desarrollo de la aplicación:

Tabla 3: Estimación de esfuerzo por HU

Historias de Usuario	Puntos de estimación
1- Realizar inventario de los procesos de las aplicaciones activas.	1
2- Realizar monitoreo de los procesos de las aplicaciones activas.	1
3- Enviar inventario de los proceso de las aplicaciones activas.	1
4- Calcular el tiempo que están activos los procesos.	2
5- Almacenar inventario de los proceso de las aplicaciones activas.	1
6- Adicionar nombre de procesos y aplicaciones.	4/5
7- Listar nombre de procesos y aplicaciones.	4/5
8- Modificar nombre de procesos y aplicaciones.	4/5
9- Eliminar nombre de procesos y aplicaciones.	3/5
10- Adicionar los procesos a monitorear.	4/5
11- Listar los procesos a monitorear.	4/5
12- Modificar los procesos a monitorear.	4/5
13- Eliminar los procesos a monitorear.	3/5
14-Mostrar datos de los procesos de las aplicaciones activas por computadoras clientes.	2
15- Mostrar estadísticas generales de los procesos de las aplicaciones activas.	4/5
16- Exportar a excel las estadísticas generales de los procesos de las aplicaciones activas.	1/5
17- Ordenar datos de los procesos.	2/5
18- Filtrar por datos de los procesos.	2/5

2.7.2 Plan de iteraciones

Una vez identificadas las HU y estimado el esfuerzo dedicado a la realización de cada una de estas, se procede a realizar el plan de iteraciones. Para una mejor organización en el desarrollo del trabajo, el equipo de desarrollo dividió la implementación en 6 iteraciones teniendo en cuenta la prioridad asignada a cada historia de usuario en correspondencia con las solicitudes del cliente y que la duración de las iteraciones es de aproximadamente 3 semanas, de manera que el tiempo de desarrollo de las mismas esté balanceado y se conforme un producto que aún sin terminar pueda brindar importantes funcionalidades del sistema.

2.7.2.1 Iteración 1

En esta primera iteración se llevará a cabo la implementación de las HU del número 1 hasta el 3, que corresponden a algunas HU de prioridad alta, funcionalidades que son indispensables para satisfacer las necesidades del cliente. Al terminar la iteración esto representará un 16,6% de la implementación del módulo.

2.7.2.2 Iteración 2

En la segunda iteración se llevará a cabo la implementación de otras HU de prioridad alta, comenzando desde el número 4 hasta el 5, para ir incorporando las principales funcionalidades al sistema. Al terminar la iteración esto representará un 27,7% de la implementación de módulo.

2.7.2.3 Iteración 3

En esta última iteración se llevará a cabo la implementación de las HU del número 6 hasta el 9, que corresponden a algunas HU de prioridad media. Al terminar la iteración esto representará un 49,9% de la implementación del módulo.

2.7.2.4 Iteración 4

En esta última iteración se llevará a cabo la implementación de otras HU de prioridad media, comenzando desde el número 10 hasta el 13. Al terminar la iteración esto representará un 72,1% de la implementación del módulo.

2.7.2.5 Iteración 5

En esta última iteración se llevará a cabo la implementación de HU de prioridad media y alta, comenzando desde el número 14 hasta el 15. Al terminar la iteración esto representará un 88,7% de la implementación del módulo.

2.7.2.6 Iteración 6

En esta última iteración se llevará a cabo la implementación de otras HU de prioridad media, comenzando desde el número 16 hasta el 18. Al terminar la iteración esto representará un 100% de la implementación del módulo.

2.7.3 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 4: Plan de duración de las iteraciones

Iteración	Orden de las Historias de Usuario a implementar	Duración total
1	✓ Realizar inventario de los procesos de las aplicaciones activas. ✓ Realizar monitoreo de los procesos de las aplicaciones activas. ✓ Enviar inventario de los proceso de las aplicaciones activas.	3 semanas
2	✓ Calcular el tiempo que están activos los procesos. ✓ Almacenar inventario de los proceso de las aplicaciones activas.	3 semanas
3	✓ Adicionar nombre de procesos y aplicaciones. ✓ Listar nombre de procesos y aplicaciones. ✓ Modificar nombre de procesos y aplicaciones. ✓ Eliminar nombre de procesos y aplicaciones.	3 semanas
4	✓ Adicionar los procesos a monitorear. ✓ Listar los procesos a monitorear. ✓ Modificar los procesos a monitorear. ✓ Eliminar los procesos a monitorear.	3 semanas
5	✓ Mostrar datos de los procesos de las aplicaciones activas por computadoras clientes. ✓ Mostrar estadísticas generales de los procesos de las aplicaciones activas.	3 semanas
6	✓ Exportar a excel las estadísticas generales de los procesos de las aplicaciones activas. ✓ Ordenar datos de los procesos. ✓ Filtrar por datos de los procesos.	1 semanas

2.7.4 Plan de entrega

El plan de entrega es un documento que especifica que HU serán implementas en cada entrega del sistema y sus prioridades, de modo que también se permita conocer con exactitud que HU serán implementadas en la próxima entrega. El mismo es elaborado entre el cliente y el equipo de desarrollo, con la idea de realizar entregas frecuentes para obtener una mayor retroalimentación (31). En la siguiente tabla se muestra el plan de entregas para el módulo propuesto.

Tabla 5: Plan de entregas

Herramienta		Módulo para monitorear el uso de software de las computadoras en la red con el sistema GRHS		
Historias de Usuario a Implementar en la Entrega				
	Número de Historia	**Título**	**Prioridad**	**Fecha de entrega**
Entrega Número: 1	**1**	Realizar inventario de los procesos de las aplicaciones activas.	Alta	20/2/2015
	2	Realizar monitoreo de los procesos de las aplicaciones activas.	Alta	
	3	Enviar inventario de los proceso de las aplicaciones activas.	Alta	
Entrega Número: 2	**4**	Calcular el tiempo que están activos los procesos.	Alta	13/3/2015
	5	Almacenar inventario de los proceso de las aplicaciones activas.	Alta	
Entrega Número: 3	**6**	Adicionar nombre de procesos y aplicaciones.	Media	3/4/2015
	7	Listar nombre de procesos y aplicaciones	Media	
	8	Modificar nombre de procesos y aplicaciones.	Media	
	9	Eliminar nombre de procesos y aplicaciones.	Media	
Entrega	**10**	Adicionar los procesos a monitorear.	Media	24/4/2015

Número: 4	**11**	Listar los procesos a monitorear.	Media	
	12	Modificar los procesos a monitorear.	Media	
	13	Eliminar los procesos a monitorear.	Media	
Entrega Número: 5	**14**	Mostrar datos de los procesos de las aplicaciones activas por computadoras clientes.	Alta	4/5/2015
	15	Mostrar estadísticas generales de los procesos de las aplicaciones activas.	Media	
Entrega Número: 6	**16**	Exportar a excel las estadísticas generales de los procesos de las aplicaciones activas.	Media	21/5/2015
	17	Ordenar datos de los procesos.	Media	
	18	Filtrar por datos de los procesos.	Media	

2.8 Conclusiones del Capítulo 2

En este capítulo se describió la propuesta de solución, se identificaron los requerimientos que debe cumplir el sistema y se realizó el modelado del proceso de negocio para una mejor comprensión del módulo a desarrollar. Además se obtuvo los artefactos necesarios para la posterior implementación, tales como: HU, el esfuerzo estimado por HU, el plan de iteraciones y el plan de entregas, los cuales fueron detallados de forma clara.

CAPÍTULO 3: DISEÑO, IMPLEMENTACIÓN Y PRUEBAS DEL MÓDULO PARA MONITOREAR EL USO DE SOFTWARE DE LAS COMPUTADORAS EN LA RED CON EL SISTEMA GESTIÓN DE RECURSOS DE HARDWARE Y SOFTWARE.

3.1 Introducción

En este capítulo se describen las fases de implementación y pruebas, propias de la metodología XP. Se define la arquitectura del sistema, así como los patrones arquitectónicos y de diseño utilizados en el mismo. Además se definen las tarjetas CRC (Clase-Responsabilidad-Colaborador), para identificar y organizar las clases orientadas a objetos y las tareas de ingeniería derivadas por cada historia de usuario. Posteriormente se ejecutan las pruebas para evaluar la calidad del sistema.

3.2 Arquitectura

"La Arquitectura de Software es la forma en la que se organizan los componentes de un sistema, interactúan y se relacionan entre sí y con el contexto, aplicando normas y principios de diseño y calidad, que fortalezcan y fomenten la usabilidad a la vez que dejan preparado el sistema, para su propia evolución" (32). La arquitectura empleada para el desarrollo del módulo es la arquitectura Cliente/Servidor, debido a que esta ha sido la utilizada para el desarrollo del sistema GRHS, y los plugins a implementar estarán integrados a dicho sistema.

3.3 Arquitectura Cliente/Servidor

Esta arquitectura es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. El cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio) (Ver Figura 4). La arquitectura Cliente/Servidor permite al usuario final el acceso a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo, a través de la organización, en múltiples plataformas (33).

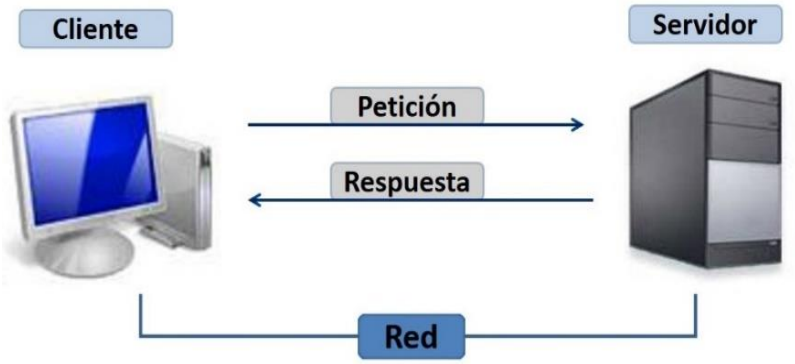


Figura 4. Arquitectura Cliente/Servidor

Una de las principales ventajas que proporciona esta arquitectura es la posibilidad de agregar o quitar clientes, sin afectar el funcionamiento de la red y sin la necesidad de realizar mayores modificaciones.

3.4 Patrón Arquitectónico

Los patrones arquitectónicos definen la estructura de un sistema de software, con el objetivo de facilitar la tarea del diseño para el sistema a desarrollar. Para el desarrollo del módulo el patrón a seguir es el Modelo-Plantilla-Vista (MTV según sus siglas en inglés).

3.4.1 Modelo Plantilla Vista

El framework Django, una de las tecnologías empleadas en el módulo a desarrollar, usa una modificación de la arquitectura Modelo-Vista-Controlador (MVC), llamada Modelo-Plantilla-Vista (MTV) (34). En este patrón el modelo es el que contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos y las relaciones entre estos, donde las clases pertenecientes a esta capa se encuentran en el archivo models.py. La vista es quien decide que funcionalidad del modelo a utilizar y la información que se va a mostrar, ubicada en el archivo views.py. En tanto la plantilla decide cómo se mostrará la información, donde las plantillas pertenecientes a esta capa se encuentran en la carpeta templates. En la siguiente figura de elaboración personal se muestra su funcionamiento:

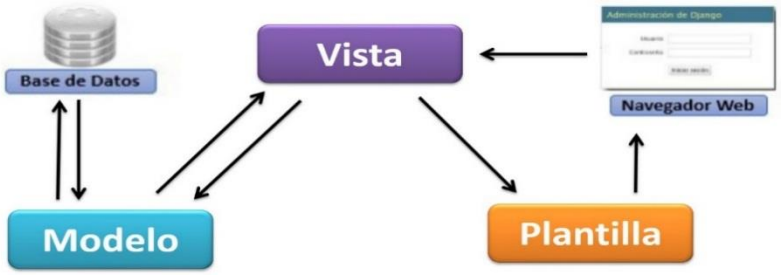


Figura 5. Patrón de Arquitectura MTV

Capa Modelo

La capa modelo define los datos almacenados, que se encuentran en forma de clases de Python, donde cada tipo de dato que debe ser almacenado se encuentra en una variable con ciertos parámetros. A continuación se representan las clases de dicha capa.

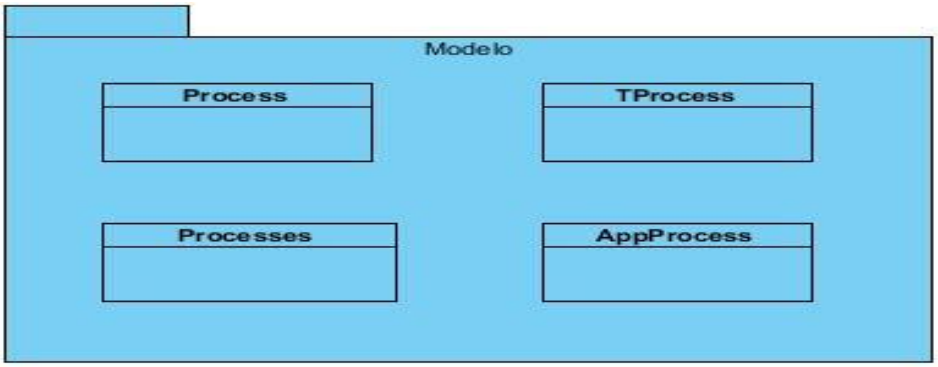


Figura 6. Capa Modelo

Process: clase responsable de registrar, modificar, eliminar, buscar y obtener datos de los procesos de cada aplicación a monitorear.

TProcess: clase responsable de registrar, modificar, eliminar, buscar y obtener temporalmente datos de los procesos de cada aplicación a monitorear.

Processes: clase responsable de registrar, modificar, eliminar, buscar y obtener la configuración de los procesos de cada aplicación a monitorear.

AppProcess: clase responsable de registrar, modificar, eliminar, buscar y obtener nombre de la aplicación y su proceso.

Capa Vista

La vista se presenta en forma de funciones en Python, su propósito es determinar qué datos serán visualizados. Lo más importante con respecto a la vista es que no tiene nada que ver con el estilo de presentación de los datos, sólo se encarga de los datos. La siguiente figura muestra la representación de las clases de la capa vista.

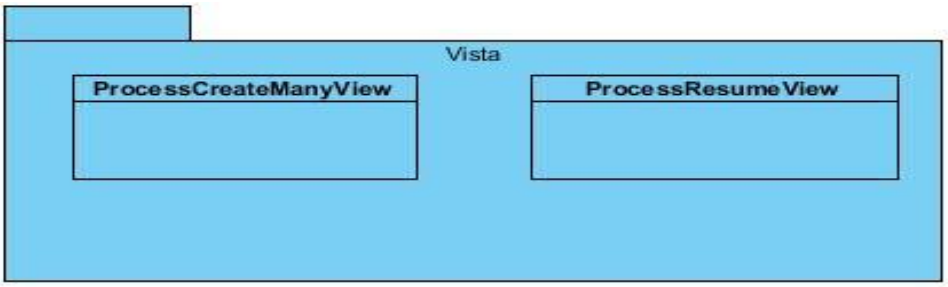


Figura 7. Capa Vista

En este paquete se representan las clases y métodos necesarios para que cada vez que se envíe una transferencia de datos del cliente al servidor con respecto a los procesos, se validen los datos, se calcule el tiempo de uso de los procesos de las aplicaciones activas y se almacenen los datos a mostrar.

Capa Plantilla

La capa plantilla recibe lo datos de la vista y luego los organiza para la presentación en el navegador web. A continuación se muestran cada uno de los archivos pertenecientes esta capa.

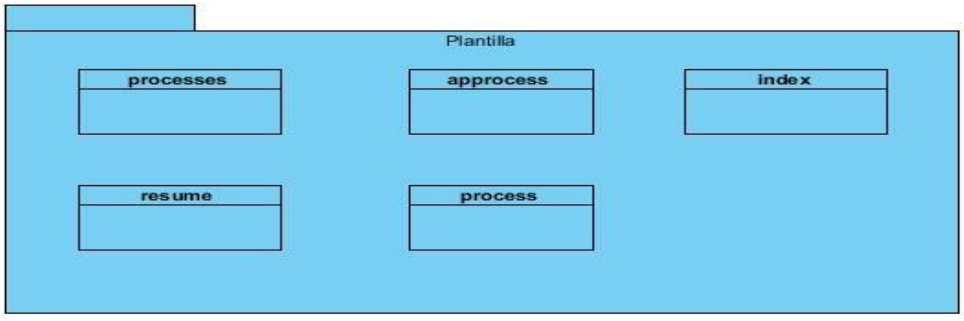


Figura 8. Capa Plantilla

Cada uno de estos archivos define las plantillas HTML, estilos CSS y lenguaje JavaScript empleados.

3.5 Patrones de Diseño

Los patrones de diseño constituyen el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Brindan una solución probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares (35).

3.5.1 Patrones GRAPS

GRAPS es un acrónimo que significa General Responsability Asignment Sotware Patterns (patrones de asignación de responsabilidades), representa parejas de problema solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades.

Para el diseño de la solución se emplearon los siguientes patrones GRASP:

- ✓ **Experto**: Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo (36). Este patrón se utiliza en las clases modelos de Django las cuales son expertas en la información de la base de datos.
- ✓ **Creador**: Las clases que tienen la responsabilidad de crear una nueva instancia de alguna clase que contiene toda la información necesaria para llevar a cabo sus tareas (36). Se utiliza en una de las clases desarrolladas, ProcessResumeView donde se instancian las clases modelos para almacenar información en la base de datos.
- ✓ **Bajo acoplamiento**: Asigna la responsabilidad de que cada clase se comunica con un número relativamente pequeño de clases (36). Un ejemplo de este patrón está en la clase Process, la cual hereda solamente de la clase Software.
- ✓ **Alta Cohesión**: Se basa en asignar una responsabilidad de modo que la cohesión siga siendo alta. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que realicen un trabajo enorme. El patrón propone el diseño de clases con responsabilidades en su área funcional y que colabore

con las otras para llevar a cabo una tarea (36), por ejemplo las clases Process y Processes se le asignan responsabilidades con el objetivo que trabajen en la misma área de aplicación y que no tengan mucha complejidad.

3.6 Tarjeta CRC (Clase, Responsabilidad y Colaboración)

Las tarjetas CRC propias de la metodología XP, es una técnica de diseño orientada a objetos que sirven para diseñar el sistema en conjunto entre todo el equipo de desarrollo, de esta forma pretende facilitar el análisis y discusión de las mismas con el objetivo de que el diseño sea lo más simple posible verificando las especificaciones del sistema. Las tarjetas se dividen en tres secciones: nombre de la clase, responsabilidades y sus colaboradores. Una responsabilidad es cualquier cosa que la clase sabe o hace. Los colaboradores son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades (37). A continuación se describen algunas de las tarjetas CRC diseñadas para la implementación:

Tabla 6 : Tarjeta CRC: Process

Clase: Process	
Responsabilidad	**Colaboración**
Describe el modelo de los procesos, definiendo las características que se van a mostrar y almacenar.	Software

Tabla 7 : Tarjeta CRC: TProcess

Clase: TProcess	
Responsabilidad	**Colaboración**
Describe el modelo de los procesos.	Software

Tabla 8: Tarjeta CRC: ProcessCreateManyView

Clase: ProcessCreateManyView	
Responsabilidad	**Colaboración**

Recibe datos de los procesos de las computadoras clientes, los crea y/o actualiza.	CreateManyView View Process

Tabla 9: Tarjeta CRC: ProcessResumeView

Clase: ProcessResumeView	
Responsabilidad	**Colaboración**
Calcula el tiempo de uso de los procesos, recursos del sistema y clasificación del estado de los procesos (inactivo, uso normal o sobre activo).	RetrieveAPIView Process Processes

Tabla 10: Tarjeta CRC: Processes

Clase: Processes	
Responsabilidad	**Colaboración**
Describe el modelo de la configuración de los procesos a monitorear.	AgentConfig

Tabla 11: Tarjeta CRC: AppProcess

Clase: AppProcess	
Responsabilidad	**Colaboración**
Describe el modelo de la configuración de los nombres de las aplicaciones y los nombres procesos correspondientes a estas aplicaciones.	ServerConfig

3.7 Diagrama de Paquetes

Los diagramas de paquetes se usan con el objetivo de tener una mejor organización e interpretación del sistema. Estos muestran cómo están agrupados los componentes del sistema, así como sus dependencias, donde cada uno realiza una tarea específica. En la siguiente figura se muestra el diagrama correspondiente al módulo que se propone:

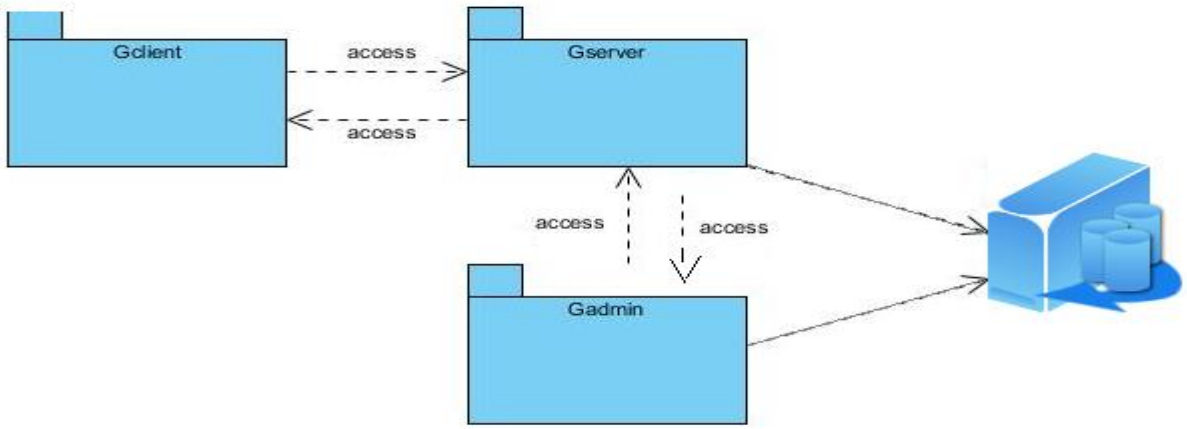


Figura 9. Diagrama de Paquetes

Gclient: En este paquete están incluidas las clases y funcionalidades necesarias que permiten el acceso a los datos, realizar el inventario y monitoreo de los procesos de las aplicaciones activas, así como envió del inventario al servidor.

Gserver: En este paquete están incluidas las funcionalidades necesarias que permiten almacenar el inventario de los procesos de las aplicaciones activas y enviar al cliente las configuraciones.

Gadmin: En este paquete están incluidas las funcionalidades necesarias que permiten mostrar y gestionar los procesos de las aplicaciones activas.

3.8 Modelo Físico de la Base de Datos

Un modelo de datos describe el tipo de datos con los que se trabaja y la forma en que se relacionan estos datos, refleja las condiciones que deben cumplir los datos para mostrar correctamente la realidad deseada y finalmente operaciones de manipulación de los datos que no es más que el agregar, borrar, modificar y recuperar de la Base de Datos. Para lograr un buen diseño de la Base de datos se hizo necesario definir el diagrama de clases persistentes y el modelo entidad relación. La figura 10 muestra el diagrama de clases persistentes y la 11 el modelo entidad relación (diagrama físico). Ambas figuras muestran solo las principales tablas utilizadas en la solución debido a la cantidad de tablas existentes en el sistema GRHS:

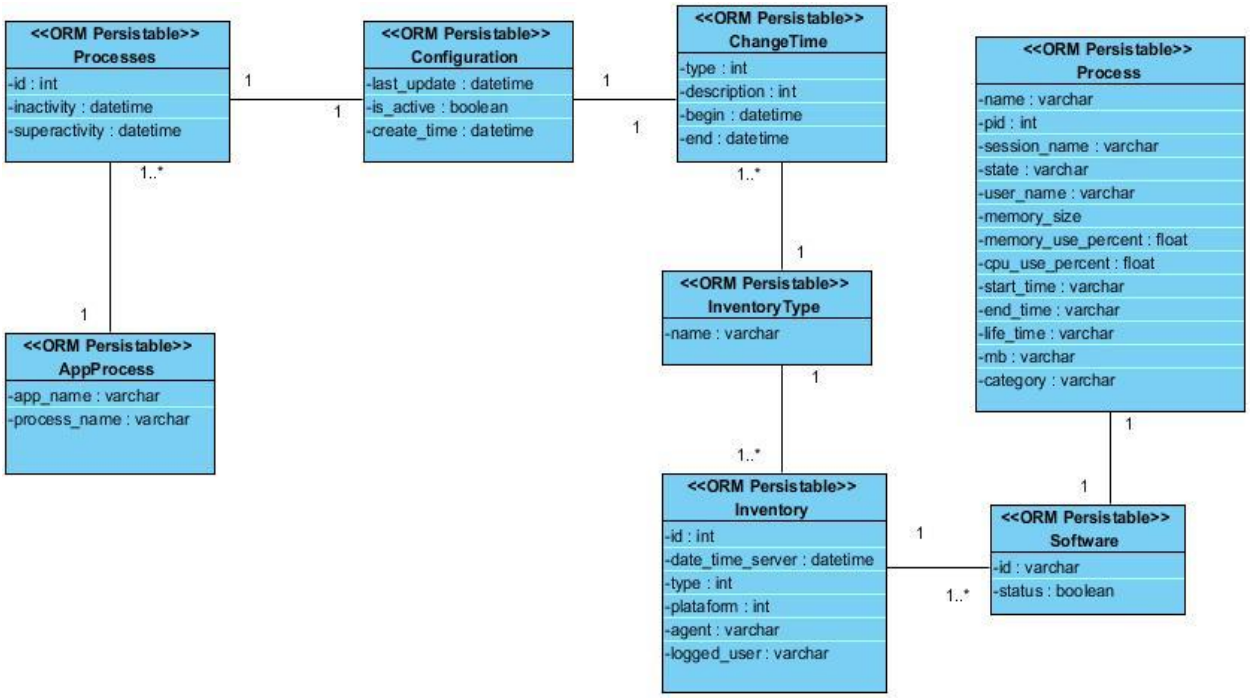


Figura 10. Diagrama de clases persistentes

En la figura 11 se resaltan en color azul las 3 tablas que son agregadas al modelo de datos existentes en el sistema GRHS para el correcto funcionamiento de la solución.

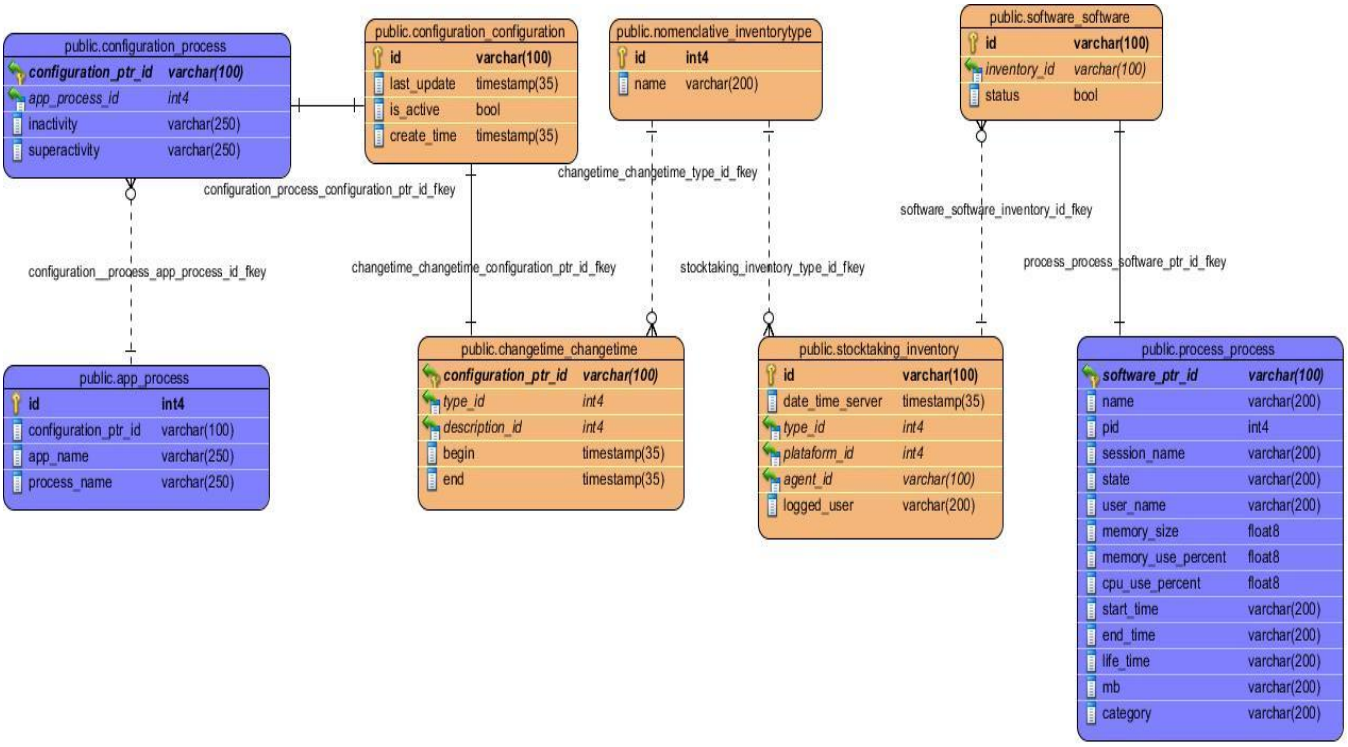


Figura 11. Modelo Entidad Relación

La siguiente descripción corresponde a las tablas de la solución propuesta:

La tabla app_process almacena los nombres de software instalados en las computadoras y su correspondiente proceso.

La tabla process_process almacena información perteneciente a los procesos de las aplicaciones activas.

La tabla configuration_process almacena la información de cada proceso a monitorear.

3.9 Tareas de Ingeniería

Las tareas de la ingeniería son escritas por el equipo de desarrollo a partir de las HU elaboradas por el cliente, brindando un detalle más profundo para realizar una implementación de las mismas y estimando un tiempo más cercano a la realidad para cada una de ellas. A continuación se muestra una de las tareas de ingeniería correspondiente a la HU representada anteriormente, el resto de las tareas de ingeniería definidas por el equipo de desarrollo se encuentran en el Anexo III .

Tabla 12: Realizar monitoreo de los procesos de las aplicaciones activas para Windows

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 2
Nombre Tarea: Realizar monitoreo de los procesos de las aplicaciones activas para Windows	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2/5
Fecha Inicio: 9-2-2015	**Fecha Fin:** 10-2-2015
Programador(es) Responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: Se monitorean los procesos de las aplicaciones activas cada 5 segundos, y en caso de detectar algún cambio (ejecución o cierre de una aplicación), ejecutar el inventario de los procesos de las aplicaciones activas. Se realiza en cada computadora con sistema operativo Windows.	

Tabla 13: Realizar monitoreo de los procesos de las aplicaciones activas para Linux

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 2
Nombre Tarea: Realizar monitoreo de los procesos de las aplicaciones activas para Linux	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3/5
Fecha Inicio: 11-2-2015	**Fecha Fin:** 13-2-2015
Programador(es) Responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: Se monitorean los procesos de las aplicaciones activas cada 5 segundos, y en caso de detectar algún cambio (ejecución o cierre de una aplicación), ejecutar el inventario de los procesos de las aplicaciones activas. Se realiza en cada computadora con sistema operativo Linux.	

3.10 Pruebas

Uno de los pilares de la metodología XP es el proceso de pruebas. Las pruebas dan la oportunidad de saber si lo implementado es lo que en realidad se tenía en mente, son las que indican que el trabajo funciona realmente. XP propone la constante realización de pruebas,

donde los desarrolladores dirigen el trabajo en la búsqueda de errores, someten sistemáticamente las funcionalidades del sistema lo que permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados.

XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñadas por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final (38).

3.10.1 Pruebas unitarias

El objetivo fundamental de las pruebas unitarias es asegurar el correcto funcionamiento de las interfaces, o flujos de datos entre componentes (39). Las pruebas unitarias fueron desarrolladas constantemente cada vez que se terminaba de implementar alguna funcionalidad, probándola directamente en el entorno real. Para lo cual se empleó la librería de Python unittest y se puso a prueba todas las funcionalidades, comprobando así el correcto funcionamiento del sistema. A continuación se muestra el resultado de las pruebas unitarias realizadas:

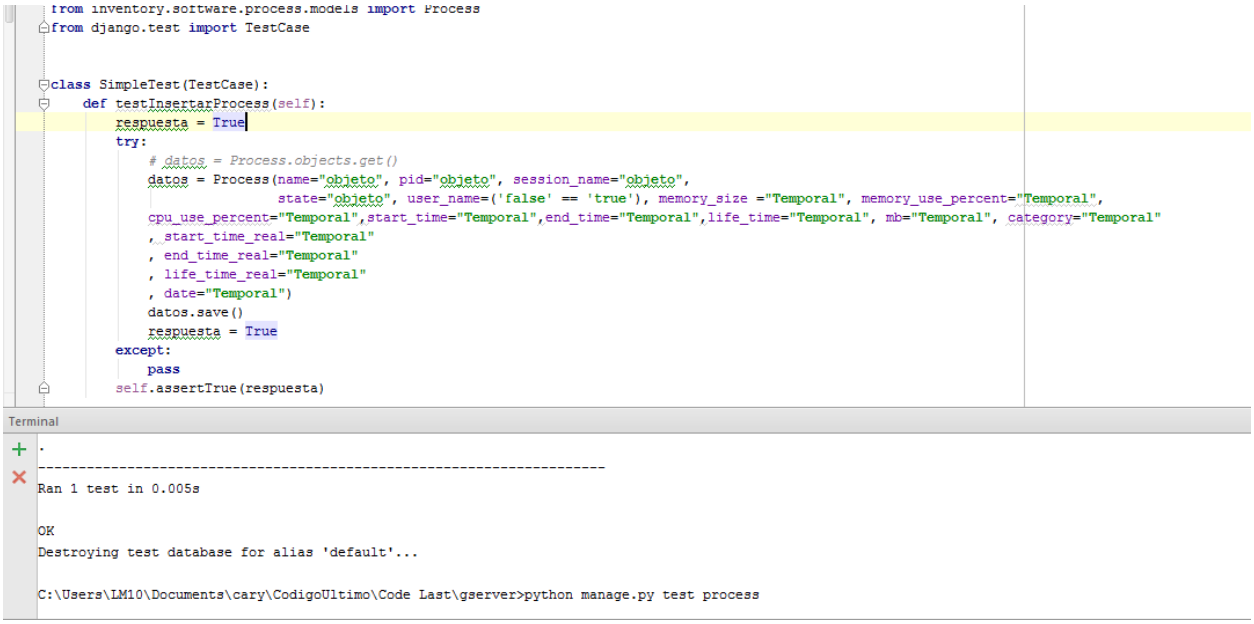


Figura 12. Pruebas unitarias

```python
import unittest
from django.test import TestCase
from src.apps.configuration.serializers import ProcessSerializer

class Test_Process(unittest.TestCase):

    def test_serializer(self):
        data={'id':'process_host','names':'name_process','inactivity':10,'superactivity':10.2}
        serializer= ProcessSerializer(data=data)
        self.assertTrue(serializer.is_valid())
```

```
root@yoel-Satellite-L655: /home/yoel/Escritorio/Code/gserver
[<Permission_Menu: Events | Items List >]
[<Permission_Menu: Monitoring Types | Can add nomenclators >]
[<Permission_Menu: Monitoring Types | Can delete nomenclators >]
[<Permission_Menu: Monitoring Types | Can change nomenclators >]
[<Permission_Menu: Monitoring Types | Items List >]
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
Destroying test database for alias 'default'...
root@yoel-Satellite-L655:/home/yoel/Escritorio/Code/gserver#
```

Figura 13. Pruebas unitarias

```python
    def testInsertarProcesses(self):
        respuesta = True
        try:
            datos = Processes(names="objeto", inactivity="objeto", superactivity="objeto")
            datos.save()
            respuesta = True
        except:
            pass
        self.assertTrue(respuesta)
```

```
Terminal
----------------------------------------------------------------------
Ran 2 tests in 0.042s

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

C:\Users\LM10\Documents\cary\CodigoUltimo\Code Last\gserver>
```

Figura 14. Pruebas unitarias

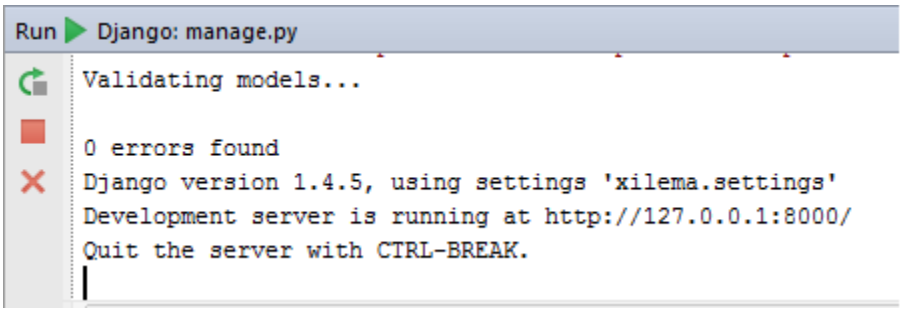


Figura 15. Pruebas unitarias

3.10.2 Pruebas de aceptación

Las pruebas de aceptación o test de aceptación permiten comprobar que el software cumple con un requisito de negocio. Una funcionalidad escrita con el lenguaje del cliente pero que puede ser ejecutada por la máquina. Estas pruebas son creadas a partir de las historias de usuario. Las mismas son el punto de partida del desarrollo en cada iteración (40). Las pruebas de aceptación permiten además, comprobar que la funcionalidad que se está probando sea la esperada por el cliente. Este tipo de pruebas funcionan como una caja negra pues cada una de ellas representa una salida esperada del sistema, donde es responsabilidad del cliente verificar la corrección de las pruebas y tomar decisiones acerca de las mismas.

La representación de las pruebas de aceptación correspondientes a cada una de las funcionalidades del Módulo para monitorear el uso de los software se representarán mediante tablas, las cuales se encuentran divididas en las siguientes secciones:

- ✓ **Clases Válidas:** describe cada uno de los pasos seguidos durante el desarrollo de la prueba, se tiene en cuenta cada una de las entradas válidas que hace el usuario con el objetivo de verificar si se obtiene el resultado esperado.
- ✓ **Clases Inválidas**: describe cada uno de los pasos seguidos durante el desarrollo de la prueba, se tiene en cuenta cada una de las posibles entradas inválidas que hace el usuario con el objetivo de verificar si se obtiene el resultado esperado y cómo responde el sistema.

- ✓ **Resultado Esperado**: se describe el resultado que se espera ya sea para entradas válidas o inválidas.
- ✓ **Resultado de la Prueba**: se describe el resultado que se obtiene.
- ✓ **Observaciones**: algún señalamiento o advertencia que sea necesario hacerle a la sección que se está probando.

A continuación se muestra un caso de prueba correspondiente a una de las funcionalidades del sistema, el resto se encuentran en los anexos del presente documento (Ver Anexo IV).

Tabla 14: Prueba # 5 Adicionar procesos a monitorear

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El usuario accede a la interfaz de Configuración. En dicha interfaz después de seleccionar Proceso, escoger la opción de Adicionar y seguidamente seleccionar nombre de proceso a monitorear, e introduce el tiempo para la inactividad y sobreactividad de los mismos.		El sistema verifica que la entrada es válida y adiciona los datos de un nuevo proceso configurado.	Satisfactorio.	
	El usuario accede a la interfaz de Configuración. En dicha interfaz después de	El sistema señala un error en el campo vacío, mostrando el siguiente cartel :	Satisfactorio.	

	seleccionar Proceso, escoger la opción de Adicionar y presiona el botón Aceptar aún con campos vacíos.	“Por favor inserte un valor para este campo”		
	El usuario accede a la interfaz de Configuración. En dicha interfaz después de seleccionar Proceso, escoger la opción de Adicionar e inserta en los campos inactividad y sobreactividad un tiempo con un formato distinto al siguiente: 00:00:00, luego presiona el botón Aceptar.	El sistema señala un error en esos campos, mostrando el siguiente cartel : “Debe tener el formato 00:00:00 ”	Satisfactorio.	

3.10.3 Registro de no conformidades

Uno de los detalles a no pasar por alto en el momento de realizar las pruebas es identificar las no conformidades, las cuales se traducen en los errores encontrados y funcionalidades no deseadas por el cliente, detectadas en cada iteración de pruebas. Al final de cada iteración se le muestra al cliente una versión funcional del software de forma que pueda detectar aquellas no conformidades que serán corregidas al inicio de la siguiente iteración.

La siguiente figura representa un gráfico de barras con los resultados de las pruebas desarrolladas en las 6 iteraciones al sistema, asociadas a las no conformidades detectadas en cada una de ellas:

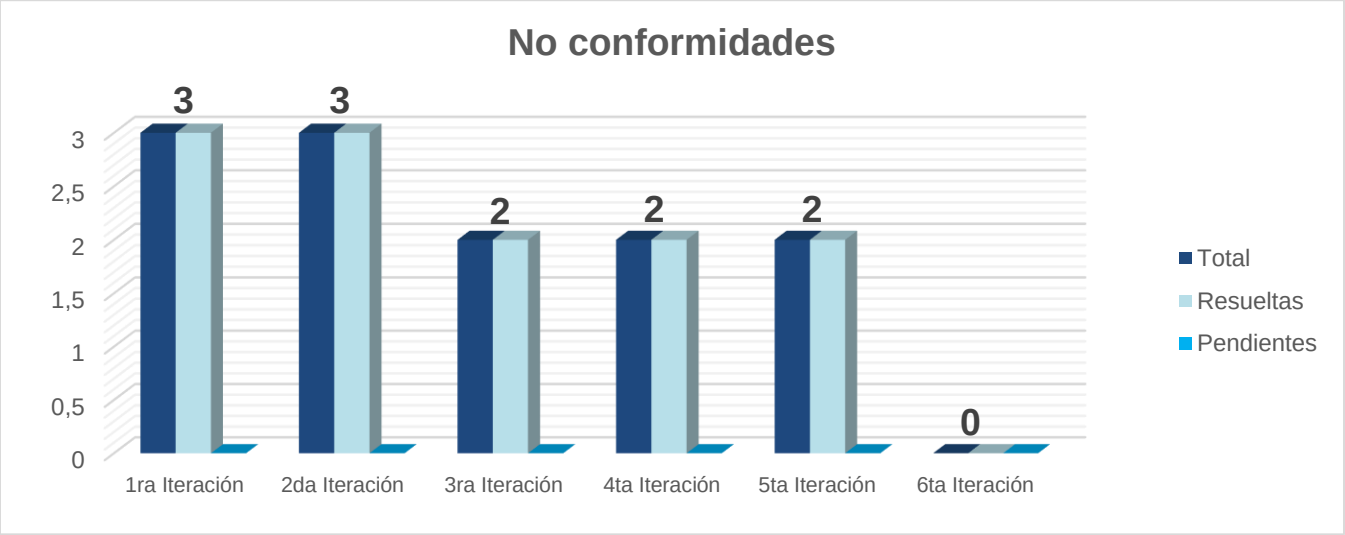


Figura 16. Resultados de las pruebas

A continuación se listan cada una de las no conformidades encontradas en cada iteración:

Tabla 15: Registro de no conformidades

No conformidades	
Iteración	**Descripción**
1	✓ Existen dependencias que no son identificadas por el sistema. ✓ Las computadoras clientes no reciben del servidor la configuración de los procesos a monitorear. ✓ Las computadoras clientes obtienen procesos que no están en la configuración.

2	✓ El servidor no recibe todos los procesos enviados por cada computadora cliente. ✓ No se reconoce el formato de la fecha de los procesos enviados al servidor. ✓ Cálculo incorrecto del tiempo de uso de los procesos.
3	✓ Permite introducir letras en campos numéricos. ✓ Permite insertar una hora no válida en los campos de inactividad y sobreactividad.
4	✓ Los datos de la configuración (nombre de aplicación-nombre de proceso) no son visibles para la selección en la configuración de los procesos a monitorear. ✓ Errores de falta de ortografía en los mensajes.
5	✓ No se muestra el tiempo de uso de algunos procesos. ✓ En la lista de los procesos de las aplicaciones activas que se muestran aparecen procesos repetidos.
6	No se encuentran no conformidades en el sistema.

Como parte de la metodología XP, las no conformidades encontradas en cada iteración son las primeras tareas a resolver de la iteración siguiente, siendo el cliente el encargado de ordenarlas por prioridad. Llevando a cabo este proceso, se logran minimizar los niveles de aceptación de errores. De esta manera fueron resueltas todas las no conformidades detectadas en el módulo desarrollado.

3.11 Conclusiones del Capítulo 3

En el presente capítulo se definió la arquitectura empleada en el desarrollo del módulo, para así diseñar y guiar el proceso de desarrollo del software. Se tuvo en cuenta la utilización de patrones de diseño para lograr una mejor reutilización de código y una correcta implementación del módulo. Además se confeccionaron las tarjetas CRC permitiendo identificar y organizar las clases orientadas a objetos. Se identificaron las clases persistentes en el tiempo para obtener a

partir de las mismas el modelo físico de la base de datos. Se modeló un diagrama de paquetes para un mejor entendimiento del sistema. Se implementaron las tareas de ingeniería correspondientes a las HU definidas en cada iteración. Por último se realizaron las pruebas unitarias y de aceptación, las cuales arrojaron ciertas no conformidades que fueron solucionadas.

CONCLUSIONES GENERALES

A partir del desarrollo de la presente investigación se arribaron a las siguientes conclusiones:

- ✓ El estudio de los principales conceptos asociados al monitoreo de software activos en un ordenador propició un mayor entendimiento de los mismos.
- ✓ El análisis de los sistemas similares permitió identificar características que sirvieron de guía en la solución.
- ✓ Se definieron las herramientas y tecnologías necesarias para el desarrollo del módulo de monitoreo de software en las computadoras.
- ✓ Se realizó el plan de iteraciones que permitió la organización del tiempo y el trabajo con el fin de culminar el desarrollo del módulo para monitorear el uso de software en el tiempo requerido.
- ✓ Se implementaron los plugins para monitorear y obtener información de los principales procesos de los software activos, empleando la metodología ágil XP para guiar el proceso de desarrollo del software y generándose los artefactos necesarios para la implementación de sus funcionalidades.
- ✓ Las pruebas realizadas permitieron comprobar las funcionalidades del módulo, estas brindaron resultados satisfactorios otorgando validez a la investigación.
- ✓ Como resultado del desarrollo del módulo para monitorear el uso de los software se obtuvo un producto que permite el control y monitoreo de las aplicaciones que utilizan los usuarios en un área de trabajo.

Por todo lo anteriormente expuesto, se concluye que el objetivo propuesto para el presente trabajo de diploma se ha cumplido satisfactoriamente, poniendo en práctica todas y cada una de las tareas propuestas para el desarrollo del módulo para monitorear el uso de software en la red.

RECOMENDACIONES

Luego de haber analizado los resultados del presente trabajo de diploma, se recomienda:

- ✓ Implementar sistemas de notificaciones ante el uso excesivo o bajo uso de determinadas aplicaciones.
- ✓ Integrar el módulo desarrollado con el Módulo de Organización y Presentación para el sistema GRHS desarrollado en otro trabajo de diploma del presente curso.

REFERENCIAS BIBLIOGRÁFICAS

1. PRESSMAN, ROGGER S. *Ingeniería del Software. Un enfoque práctico.* s.l. : Félix Varela s.l, 2005.

2. MIRANDA GÓMEZ, ORLANDO. *Monitoreo en tiempo real del estado de los recursos del sistema usando el marco de trabajo jWebSocket*. Universidad de las Ciencias Informáticas, 2012.

3. GUSTAVO HIGA MIYASHIRO. [SysAdmin] Herramientas de monitoreo en infraestructuras de TI | El Mundo es Open Source. [en línea]. 2011. Disponible en: http://blogs.antartec.com/opensource/2011/05/herramientas-de-monitoreo/

4. Definición de software — Definicion.de. *Definición.de* [en línea]. abril 2015. Disponible en: http://definicion.de/software/

5. GONZALEZ DURÁN, SERGIO. Manual básico de administración de procesos en Linux. [en línea]. 2015. Disponible en: http://www.linuxtotal.com.mx/index.php?cont=info_admon_012

6. REAL ACADEMIA ESPAÑOLA. Diccionario de la lengua española. [en línea]. 2015. Disponible en: http://lema.rae.es/drae/?d=drae&val=monitorizar&x=0&y=0

7. YIRÁ, FRANCISCO. ManicTime, herramienta para gestionar el tiempo que pasamos frente al PC. [en línea]. julio 2011. Disponible en: http://www.genbeta.com/a-fondo/manictime-herramienta-para-gestionar-el-tiempo-que-pasamos-frente-al-pc

8. ¿Qué es? - Activity Monitor. [en línea]. 2014. Disponible en: http://activitymonitor.es/index.php?id_cms=6&controller=cms&id_lang=4

9. QMA SC. Inventario de Hardware y Software con el Monitoreo de Estaciones en Redes - VEO Ultimate. [en línea]. 2014. Disponible en: http://www.veo.com.mx/

10. GÓMEZ MIRANDA,ORLANDO. *Monitoreo en tiempo real del estado de los recursos del sistema usando el marco de trabajo jWebSocket*. Universidad de las Ciencias Informáticas, 2012.

11. OCS Inventory. *ASTRO Servicios informáticos* [en línea]. 2012. Disponible en: http://www.astro-corp.com/tecnologia/ocs

12. ORDOÑES, YOANNI, ÁVILES, ERNESTO, HERNÁNDEZ, JULIO y RODRÍGUEZ, ODAYSA. *GRHS: Gestor de Recursos de Hardware y Software*. 2014.

13. BAUER, FRITZ. Conferencia NATO. En: 1968.

14. CENDEJAS VALDÉZ, JOSÉ LUIS. Implementación Del Modelo Integral Colaborativo. [en línea]. mayo 2014. Disponible en: http://www.eumed.net/tesis-doctorales/2014/jlcv/

15. *UNIVERSIDAD UNION BOLIVARIANA - PROGRAMACIÓN EXTREMA.pdf* [en línea]. [Consulta: 15 abril 2015]. Disponible en: http://ingenieriadesoftware.mex.tl/images/18149/PROGRAMACI%C3%93N%20EXTREMA.pdf

16. *Metodología Actual. Metodología XP* [en línea]. [Consulta: 28 abril 2015]. Disponible en: http://blogs.unellez.edu.ve/dsilva/files/2014/07/Metodologia-XP.pdf

17. ÁLVAREZ, MIGUEL ÁNGEL. Qué es Python. *DesarrolloWeb.com* [en línea]. noviembre 2003. Disponible en: http://www.desarrolloweb.com/articulos/1325.php

18. MARTÍNEZ GUAITA, ALVARO. Nuevo Django 1.4. [En línea]. Abril 2012. Disponible en: http://www.desarrolloweb.com/actualidad/nuevo-django-1-4-6755.html

19. Python IDE & Django IDE for Web developers : JetBrains PyCharm. [En línea]. 2015. Disponible en: https://www.jetbrains.com/pycharm/

20. *BPMN (Business Process Modeling Notation –BPMN-) - BPMNbyExample.pdf* [en línea]. [Consulta: 15 abril 2015]. Disponible en: http://www.bizagi.com/esp/descargas/BPMNbyExample.pdf

21. CRAIG LARMAN. *UML y Patrones*. 2da Edición. 2003.

22. Paradigma visual para UML. [en línea]. 2007. Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/

23. ALVAREZ, SARA. Sistemas gestores de bases de datos. *DesarrolloWeb.com* [en línea]. Julio 2007. Disponible en: http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-

datos.html

24. MARTÍNEZ, RAFAEL. Sobre PostgreSQL | www.postgresql.org.es. [en línea]. 2010. Disponible en: http://www.postgresql.org.es/sobre_postgresql

25. MARTÍNEZ, DANIEL PECOS. PostGreSQL vs. MySQL - geekWare. [en línea]. 2014. Disponible en: http://danielpecos.com/documents/postgresql-vs-mysql/#AEN59

26. SQLite Home Page. [en línea]. 2014. Disponible en: http://www.sqlite.org/

27. PgAdmin – ArPug - PostgreSQL Argentina - Grupo de Usuarios. [en línea]. 2013. Disponible en: http://www.postgresql.org.ar/trac/wiki/PgAdmin

28. RAINER. Documentación generada en Extremme Programming | HananTek. [en línea]. 2010. Disponible en: http://www.hanantek.com/es/documentacion-programacion-extrema

29. ALEMÁN ESTRADA, VIVIAN DAYLIN y REMON, YORX ALEX. *Subsistema explorador de llamadas para el sistema de gestión integral de costos de llamadas en pizarras telefónicas-pabx en su versión 2.0*. 2014.

30. LETELIER, PATRICIO. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *www.cyta.com.ar/ta0502/v5n2a1.htm* [en línea]. 2006. Disponible en: http://www.cyta.com.ar/ta0502/b_v5n2a1.htm

31. Plan de entregas. *Herramientas Genéricas para la Implantación de Prácticas Ágiles de XP* [en línea]. Julio 2007. Disponible en: http://www.inf.utfsm.cl/~visconti/xp/

32. HABIT, EUGENIA. *POO y MVC en PHP*. 2011.

33. GUILLERMO VALLE, JOSE y GILDARDO GUTIERREZ, JAMES. Definición arquitectura cliente servidor - Monografias.com. [en línea]. 2005. Disponible en: http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml

34. INFANTE MONTERO, SERGIO. Curso Django: Entendiendo como trabaja Django. *Maestros del Web* [en línea]. Abril 2012. Disponible en: http://www.maestrosdelweb.com/curso-

django-entendiendo-como-trabaja-django/

35. TEDESCHI, NICOLÁS. ¿Qué es un Patrón de Diseño? [en línea]. 2015. Disponible en: https://msdn.microsoft.com/es-es/eses/library/bb972240.aspx

36. *Fundamentos de Ingeniería de Software* [en línea]. [Consulta: 1 mayo 2015]. Disponible en: http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf

37. ANAYA VILLEGAS, ADRIAN y PLAZA MARÍN, EDISON ARLEY. A propósito de programación extrema XP (eXtreme Programming) (página 2) - Monografias.com. [en línea]. 2009. Disponible en: http://www.monografias.com/trabajos51/programacion-extrema/programacion-extrema2.shtml

38. BUSTAMANTE, DAYANA y RODRÍGUEZ, JEAN C. *Metodología Actual Metodología XP* [en línea]. Marzo 2014. Disponible en: http://blogs.unellez.edu.ve/dsilva/files/2014/07/Metodologia-XP.pdf

39. ORÉ B, ALEXANDER. Pruebas Unitarias Cap. 1 - Software Testing and QA - Pruebas Unitarias. [en línea]. 2009. Disponible en: http://www.calidadysoftware.com/testing/pruebas_unitarias1.php

40. BLÉ JURADO, CARLOS. *Diseño ágil con TDD*. Primera. 2010.

BIBLIOGRAFÍA

1. ALBALADEJO, XAVIER, 2009. Estimación y planificación ágil. [en línea]. [Consulta: 17 abril 2015]. Disponible en: http://www.proyectosagiles.org/estimacion-planificacion-agil-quinto-encuentro-agil-barcelona.

2. ARAUZO AZOFRA, ANTONIO y REVILLA, ERNESTO, 2004. *Una pequeña introducción a Python*. marzo 2004. S.l.: s.n.

3. Arquitectura cliente servidor. [en línea] 2010. S.l. Disponible en: http://es.slideshare.net/NoeGonzalezMendoza/arquitectura-cliente-servidor.

4. BEAS, JOSÉ MANUEL, 2011. Historias de Usuario. [en línea]. Disponible en: http://jmbeas.es/guias/historias-de-usuario/.

5. CHAUVIN, S., 2013. Notación Para el Modelado de Procesos de Negocio. *Mujeres de Empresa* [en línea]. Disponible en: http://www.mujeresdeempresa.com/notacion-para-el-modelado-de-procesos-de-negocio/.

6. DISEÑO DE BASE DE DATOS. *INFORMÁTICA APLICADA* [en línea] 2012. [Consulta: 22 mayo 2015]. Disponible en: https://irfeyal.wordpress.com/bases-de-datos/modelamiento-de-bdd/.

7. Django 1.6 release notes. [en línea] 2013. Disponible en: https://docs.djangoproject.com/en/dev/releases/1.6/.

8. ECHEVERRY TOBON, LUIS MIGUEL y DELGADO CARMONA, LUZ ELENA, 2007. *Caso práctico de la metodología ágil XP al desarrollo de software*. 2007. S.l.: s.n.

9. Gestión de Inventario. [en línea] 2014. Disponible en: http://www.articulosinformativos.com.mx/Gestion_de_Inventario-a854148.html.

10. GONZÁLEZ, D., 2011. Daniela González - Programación: Cómo correr pruebas unitarias en Python. *Daniela González - Programación* [en línea]. Disponible en: http://lablenguajesdaniela.blogspot.com/2011/11/como-correr-pruebas-unitarias-en-python.html.

11. GONZÁLEZ, DANIEL, 2010. Tecnologías de la Información y la Comunicación (TIC´S) - Monografias.com. [en línea]. Disponible en: http://www.monografias.com/trabajos67/tics/tics.shtml.

12. GONZÁLEZ DUQUE, RAÚL, 2008. *Python para todos*. S.l.: s.n.

13. GONZÁLEZ DUQUE, RAÚL [sin fecha]. Pruebas en Python. *Mundo geek* [en línea]. [Consulta: 13 mayo 2015]. Disponible en: http://mundogeek.net/archivos/2008/09/17/pruebas-en-python/#more-1744.

14. GUILLERMO VALLE, JOSÉ, 2005. Definición arquitectura cliente servidor. [en línea]. Disponible en: http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml.

15. GUTIÉRREZ, J.J, ESCALONA, M.J, MEJÍAS, M. y TORRES, J., 2014. *Pruebas del sistema en programación extrema*. 2014. S.l.: s.n.

16. HERNÁNDEZ LEÓN, ROLANDO ALFREDO y COELLO GONZÁLEZ, SAYDA, 2002. *El paradigma cuantitativo de la investigación científica*. S.l.: Editorial Universitaria.

17. HOLOVATY, ADRIAN y KAPLAN MOSSS, JACOB, 2015. *La guía definitiva de django*. S.l.: s.n.

18. JOSKOWICZ, JOSÉ, 2008. *Reglas y Prácticas en eXtreme Programming*. Autoedición. S.l.: s.n.

19. KIOSKEA.NET, 2014. *Entorno cliente/servidor*. junio 2014. S.l.: s.n.

20. LARMAN, CRAIG, 2003. *UML y Patrones*. S.l.: s.n.

21. MARIBEL, 2012. Gestor de base de datos. [en línea]. Disponible en: http://www.estudioteca.net/universidad/telecomunicaciones/gestor-base-datos.

22. MASADELANTE.COM [sin fecha]. ¿Qué es el TCP/IP? - Definición de TCP/IP. [en línea]. Disponible en: https://www.masadelante.com/faqs/tcp-ip.

23. *Modelado de datos. Fundamento de diseño de base de datos* [en línea], [sin fecha]. S.l.: s.n. [Consulta: 8 mayo 2015]. Disponible en: http://elvex.ugr.es/idbis/db/docs/intro/C%20Modelado%20de%20datos.pdf.

24. MODELAMIENTO VISUAL Y UML. [en línea] 2007. S.l. Disponible en: http://es.slideshare.net/dersteppenwolf/modelamiento-visual-y-uml.

25. OCS INVENTORY. *infosecsapobla.foroactivo.com* [en línea] 2014. [Consulta: 28 abril 2015]. Disponible en: http://infosecsapobla.foroactivo.com/t95-ocs-inventory.

26. PÉREZ VÁLDES, DAMIÁN, 2007. ¿Qué es Javascript? *Maestros del Web* [en línea]. Disponible en: http://www.maestrosdelweb.com/que-es-javascript/.

27. PostgreSQL: POSTGRESQL. *PostgreSQL* [en línea] 2012. Disponible en: http://postgresql-dbms.blogspot.com/2012/11/blog-post.html.

28. ¿Qué es? - Activity Monitor. [en línea] 2014. Disponible en: http://activitymonitor.es/index.php?id_cms=6&controller=cms&id_lang=4.

29. QUESADA ALLUE, XAVIER, 2009. Introducción a la estimación y planificación ágil. [en línea]. [Consulta: 17 abril 2015]. Disponible en: http://www.proyectosagiles.org/introduccion-estimacion-planificacion-agil.

30. RUFERTO, 2011. Estimación ágil de historias de usuario. *Biko* [en línea]. [Consulta: 17 abril 2015]. Disponible en: http://www.biko2.com/agile/estimacion-agil-de-historias-de-usuario/.

31. SHUMWAY, CHRIS, 2010. 3.7. Procesos. [en línea]. Disponible en: https://www.freebsd.org/doc/es/books/handbook/basics-processes.html.

32. SQLITE, 2014. About SQLite. [en línea]. Disponible en: http://www.sqlite.org/about.html.

33. VISCONTI, MARCELLO y ASTUDILLO, HERNÁN ,2013. Fundamentos de Ingeniería de Software. [en línea]. S.l. Disponible en: http://es.slideshare.net/EFAR11/01-intro-isw.

34. Visual Paradigm | Manténgase actualizado. [en línea] 2015. Disponible en: http://www.visual-paradigm.com/staytuned/#IEntWPqz.

35. WINDOWS SERVER [sin fecha]. Introducción a WMI. [en línea]. [Consulta: 20 enero 2015]. Disponible en: https://technet.microsoft.com/es-es/library/cc753534.

36. wmi - Mostrando particiones en Windows. *doutdeslibertas* [en línea] 2010. Disponible en: https://doutdeslibertas.wordpress.com/2010/12/27/wmi/.

37. YIRÁ, F. 2011. ManicTime, herramienta para gestionar el tiempo que pasamos frente al PC. *Genbeta* [en línea]. Disponible en: http://www.genbeta.com/a-fondo/manictime-herramienta-para-gestionar-el-tiempo-que-pasamos-frente-al-pc.

ANEXOS

Anexo I. Modelo de entrevista aplicada a especialistas

Datos a recoger de los entrevistados:

Nombre y apellidos:

Categoría científica:

Centro de trabajo:

Área de investigación:

Años de experiencia en el área:

Consigna

Estimados compañeros, con el objetivo de precisar y definir algunos elementos para el desarrollo de esta investigación, cuyo tema es el desarrollo de un módulo para monitorear el uso de software de las computadoras en la red con el sistema GRHS, solicitamos de su colaboración.

Temas para el intercambio

- ✓ ¿Cree usted que es importante conocer el tiempo que los trabajadores de un proyecto, centro o empresa dedican a determinados software? ¿Por qué?
- ✓ ¿Qué sistemas que obtienen información de los software conoces?
- ✓ ¿Qué elementos se deben tener en cuenta, según su consideración, para monitorear el uso de los software en las computadoras?
- ✓ ¿Qué características, según su consideración, puede destacar de la estructura del sistema GRHS?
- ✓ ¿Quisiera agregar otro elemento que considere relevante y que no haya sido comprendido en las preguntas anteriores?

Cierre

Agradecimiento y despedida

Anexo II. Historias de usuario.

Iteración 1

Tabla 16: HU #1 Realizar inventario de los procesos de las aplicaciones activas

Historia de usuario	
Número: 1	**Usuario:** Sistema
Nombre de Historia de Usuario: Realizar inventario de los procesos de las aplicaciones activas	
Prioridad en negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos estimados: 1	**Iteración Asignada:** 1
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: Obtiene de cada computadora cliente y envía al servidor los procesos de las aplicaciones activas en dependencia de la configuración obtenida del servidor y los procesos obtenidos de las aplicaciones usadas por los usuarios en las computadoras.	
Observaciones:	
Prototipo de interfaz: No aplica	

Iteración 2

Tabla 17: HU#3 Enviar inventario de los procesos de las aplicaciones activas

Historia de usuario	
Número: 3	**Usuario:** Sistema
Nombre de Historia de Usuario: Enviar inventario de los proceso de las aplicaciones activas	
Prioridad en negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos estimados: 1	**Iteración Asignada:** 2
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: Envía al servidor los datos de los procesos de las aplicaciones activas utilizadas por los usuarios en las computadoras.	
Observaciones:	
Prototipo de interfaz: No aplica	

Tabla 18: HU#4 Calcular el tiempo que están activos los procesos

Historia de usuario	
Número: 4	**Usuario:** Sistema
Nombre de Historia de Usuario: Calcular el tiempo que están activos los procesos	
Prioridad en negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos estimados: 2	**Iteración Asignada:** 2
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: Calcula el tiempo que están activos los procesos de las aplicaciones utilizadas por los usuarios en las computadoras para obtener el tiempo de uso de dichas aplicaciones.	
Observaciones:	
Prototipo de interfaz: No aplica	

Tabla 19: HU#5 Almacenar inventario de los procesos de las aplicaciones activas

Historia de usuario	
Número: 5	**Usuario:** Sistema
Nombre de Historia de Usuario: Almacenar inventario de los procesos de las aplicaciones activas	
Prioridad en negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos estimados: 1	**Iteración Asignada:** 2
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: Almacena en la base de datos del servidor los datos de los procesos de las aplicaciones activas utilizadas por los usuarios en las computadoras.	
Observaciones:	
Prototipo de interfaz: No aplica	

Iteración 3

Tabla 20: HU#6 Adicionar nombre de procesos y aplicaciones

Historia de usuario	
Número: 6	**Usuario:** Administrador

Nombre de Historia de Usuario: Adicionar nombre de procesos y aplicaciones	
Prioridad en negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 4/5	**Iteración Asignada:** 3
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: Adiciona el nombre de aplicaciones y su proceso correspondiente.	
Observaciones: Es necesario adicionar el nombre correcto de los procesos.	
Prototipo de interfaz:	

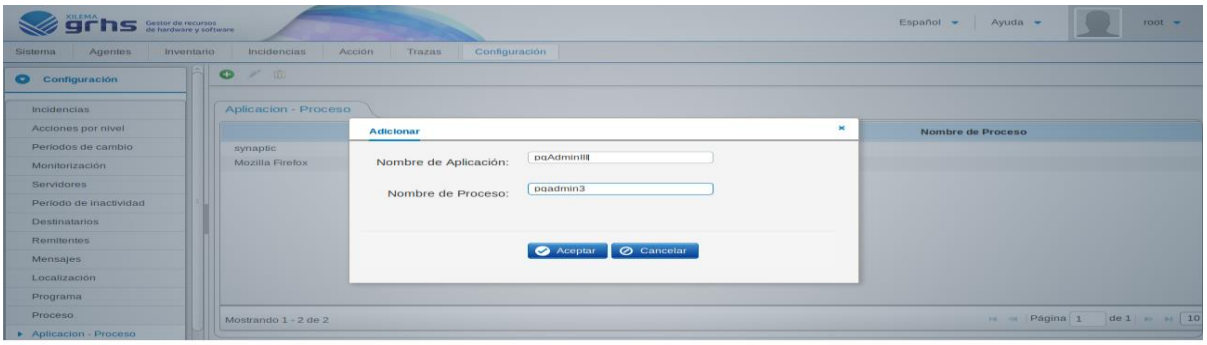


Tabla 21: HU#7 Listar nombre de procesos y aplicaciones

Historia de usuario	
Número: 7	**Usuario:** Administrador
Nombre de Historia de Usuario: Listar nombre de procesos y aplicaciones	
Prioridad en negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 4/5	**Iteración Asignada:** 3
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: Lista el nombre de aplicaciones y su proceso correspondiente.	
Observaciones: Haber adicionado al menos una aplicación con su correspondiente proceso.	
Prototipo de interfaz:	

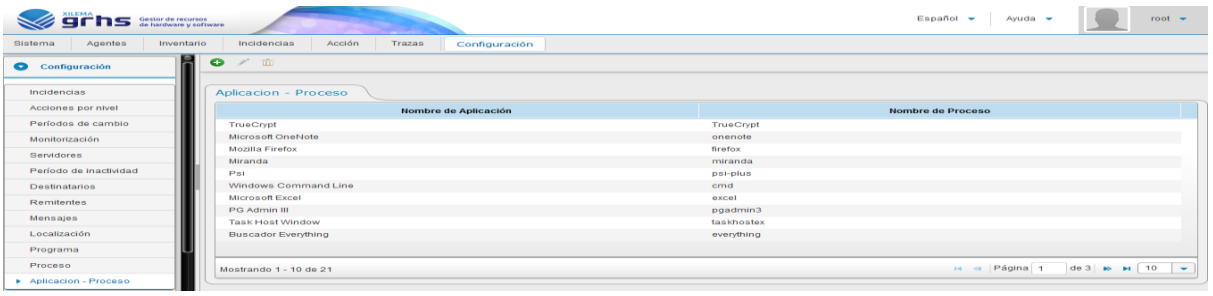


Tabla 22: HU#8 Modificar nombre de procesos y aplicaciones

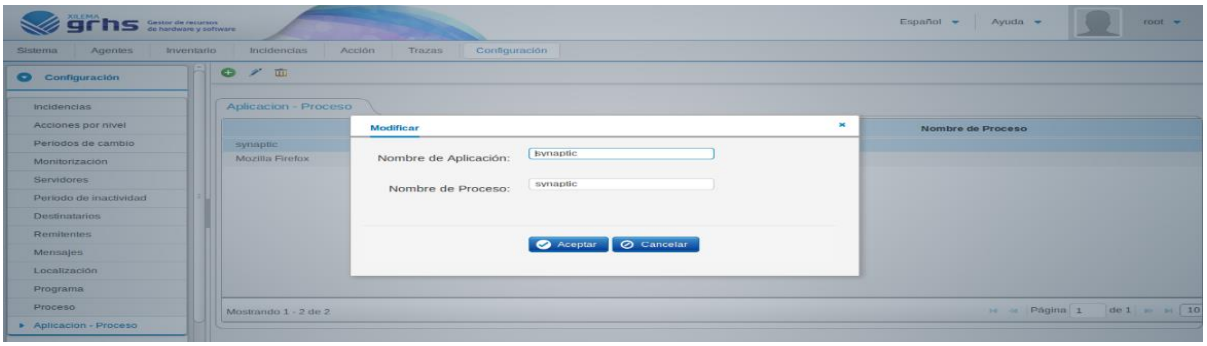
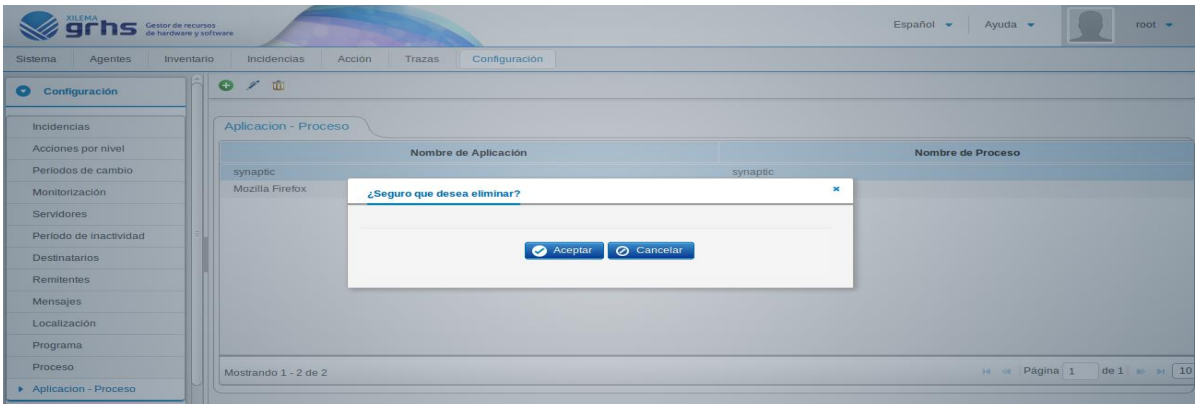
Historia de usuario	
Número: 8	**Usuario:** Administrador
Nombre de Historia de Usuario: Modificar nombre de procesos y aplicaciones	
Prioridad en negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 4/5	**Iteración Asignada:** 3
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: Modifica el nombre de la aplicación y proceso seleccionado.	
Observaciones: Haber adicionado al menos una aplicación con su correspondiente proceso.	
Prototipo de interfaz: 	

Tabla 23: HU#9 Eliminar nombre de procesos y aplicaciones

Historia de usuario	
Número: 9	**Usuario:** Administrador
Nombre de Historia de Usuario: Eliminar nombre de procesos y aplicaciones	
Prioridad en negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 3/5	**Iteración Asignada:** 3
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: Elimina el nombre de la aplicación y proceso seleccionado.	
Observaciones: Haber adicionado al menos una aplicación con su correspondiente proceso.	

Prototipo de interfaz:



Iteración 4

Tabla 24: HU#10 Adicionar los procesos a monitorear

Historia de usuario	
Número: 10	**Usuario:** Administrador
Nombre de Historia de Usuario: Adicionar los procesos a monitorear	
Prioridad en negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos estimados: 4/5	**Iteración Asignada:** 4
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: Selecciona el nombre e introduce el tiempo de inactividad y sobreactividad de los procesos que se deseen monitorear.	
Observaciones:	
Prototipo de interfaz:	

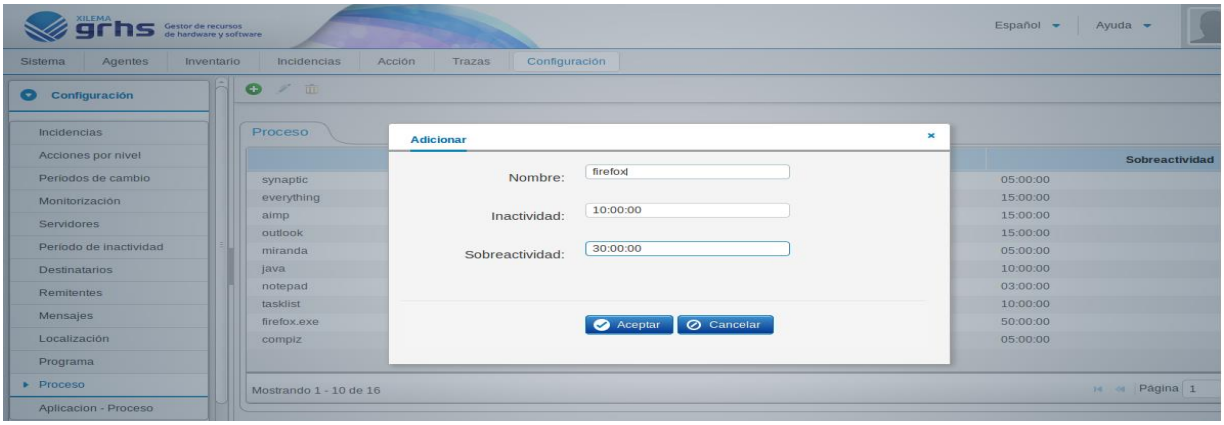


Tabla 25: HU#11 Listar los procesos a monitorear

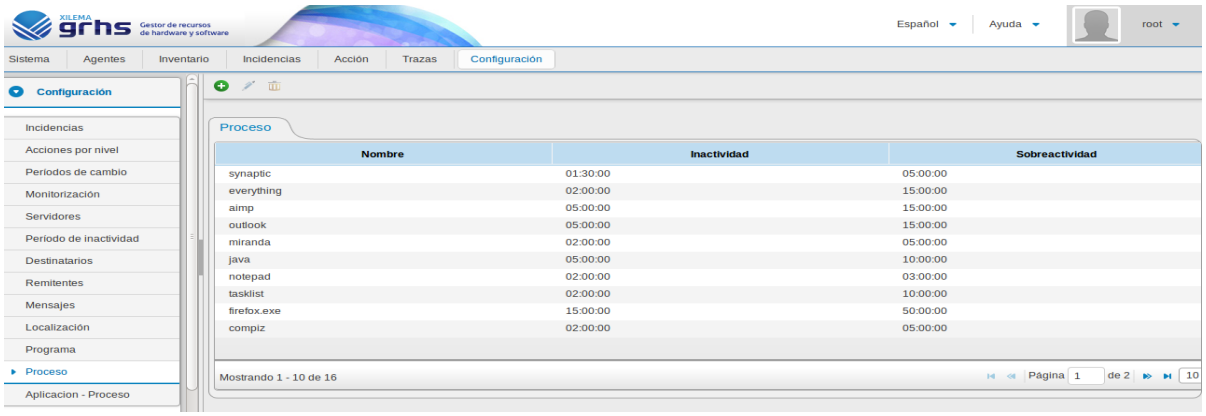
Historia de usuario	
Número: 11	**Usuario:** Administrador
Nombre de Historia de Usuario: Listar los procesos a monitorear	
Prioridad en negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 4/5	**Iteración Asignada:** 4
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: Lista datos de los procesos que se deseen monitorear.	
Observaciones: Haber adicionado al menos un proceso a la lista de procesos a monitorear.	
Prototipo de interfaz: 	

Tabla 26: HU#12 Modificar los procesos a monitorear

Historia de usuario	
Número: 12	**Usuario:** Administrador
Nombre de Historia de Usuario: Modificar los procesos a monitorear	
Prioridad en negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 4/5	**Iteración Asignada:** 4
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: Modifica los datos de los procesos que se deseen monitorear.	
Observaciones: Haber adicionado al menos un proceso a la lista de procesos a monitorear.	

Prototipo de interfaz:

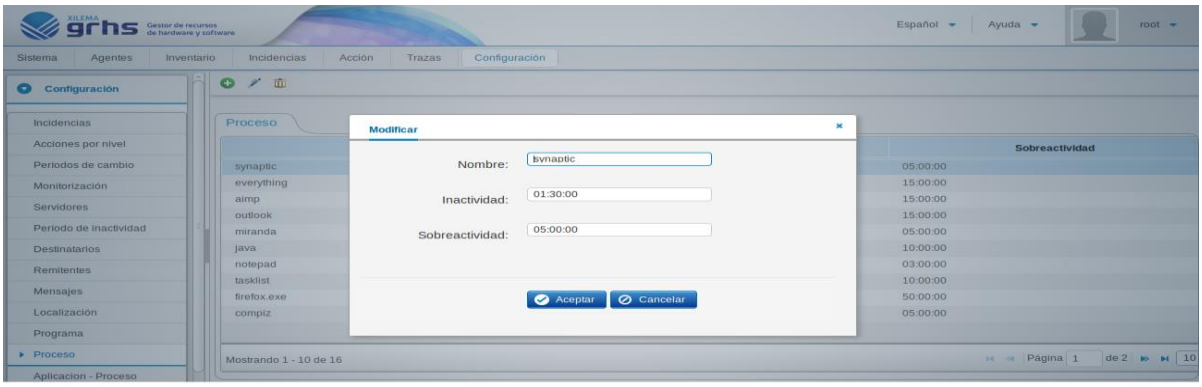
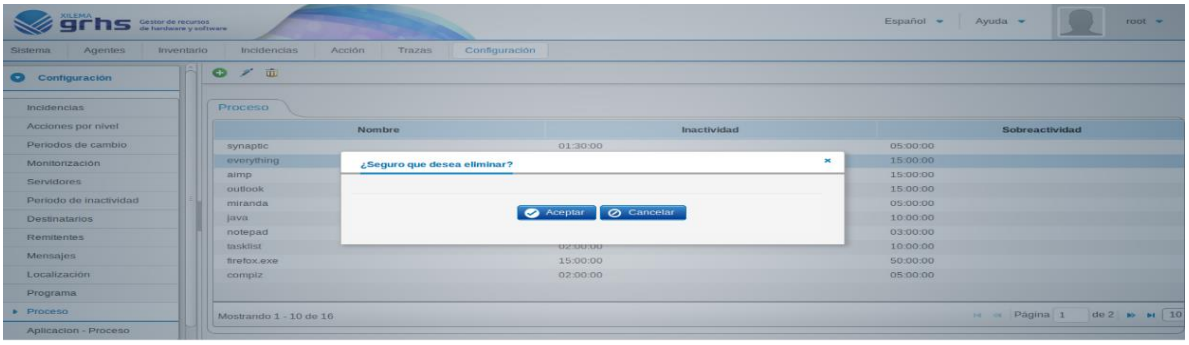


Tabla 27: HU#13 Eliminar los procesos a monitorear

Historia de usuario	
Número: 13	**Usuario:** Administrador
Nombre de Historia de Usuario: Eliminar los procesos a monitorear	
Prioridad en negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 3/5	**Iteración Asignada:** 4
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: Elimina un proceso de la lista de procesos a monitorear.	
Observaciones: Haber adicionado al menos un proceso a la lista de procesos a monitorear.	
Prototipo de interfaz: 	

Iteración 5

Tabla 28: HU# 14 Mostrar datos de los procesos de las aplicaciones activas por computadoras clientes

Historia de usuario

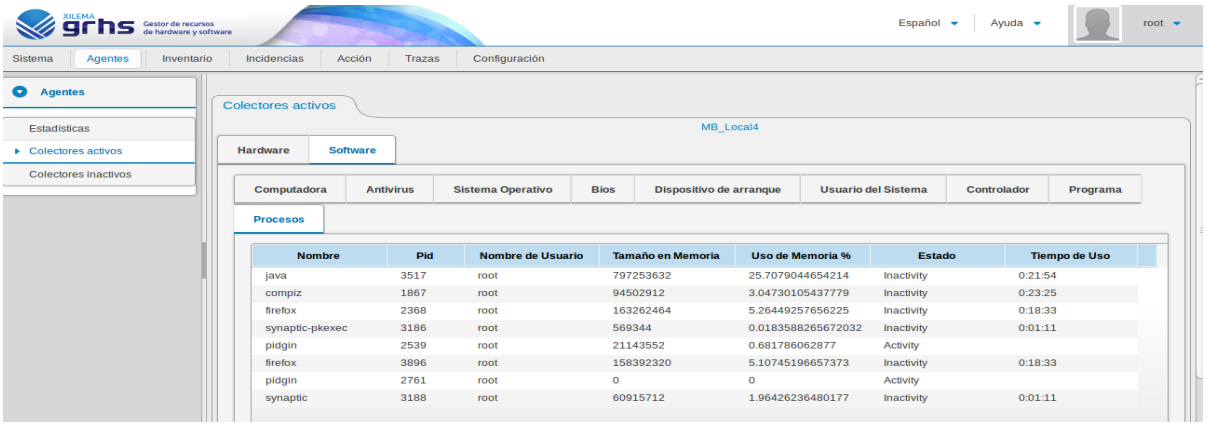
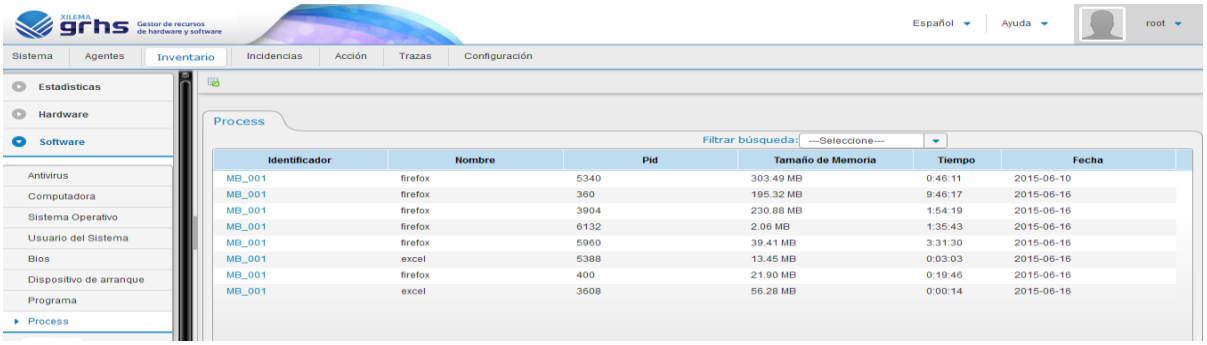
Número: 14	**Usuario:** Administrador
Nombre de Historia de Usuario: Mostrar datos de los procesos de las aplicaciones activas por computadoras clientes	
Prioridad en negocio: Alta	**Riesgo en Desarrollo:** Ato
Puntos estimados: 2	**Iteración Asignada:** 5
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: El sistema debe mostrar a través de una interfaz datos (en la semana actual) de cada proceso principal de las aplicaciones activas: nombre, Pid (identificador del proceso), usuario, tiempo de uso, uso de memoria en porciento, tamaño en memoria RAM, estado (inactivo, normal o sobre activo).	
Observaciones:	
Prototipo de interfaz: 	

Tabla 29: HU#15 Mostrar estadísticas generales de los procesos de las aplicaciones activas

Historia de usuario	
Número: 15	**Usuario:** Administrador
Nombre de Historia de Usuario: Mostrar estadísticas generales de los procesos de las aplicaciones activas.	
Prioridad en negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 4/5	**Iteración Asignada:** 5
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	

Descripción: El sistema brinda la posibilidad de mostrar las estadísticas generales de los procesos de las aplicaciones activas: identificador de la computadora, nombre, Pid, tamaño de memoria, tiempo de uso y fecha de los procesos.
Observaciones:
Prototipo de interfaz: 

Iteración 6

Tabla 30: HU#16 Exportar a excel las estadísticas generales de los procesos de las aplicaciones activas

Historia de usuario	
Número: 16	**Usuario:** Administrador
Nombre de Historia de Usuario: Exportar a excel las estadísticas generales de los procesos de las aplicaciones activas	
Prioridad en negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 1/5	**Iteración Asignada:** 6
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: El sistema brinda la posibilidad de exportar los datos de los procesos de las aplicaciones activas: identificador de la computadora, nombre, Pid, tamaño de memoria, tiempo de uso y fecha de los procesos.	
Observaciones:	
Prototipo de interfaz:	

Tabla 31: HU#17 Ordenar datos de los procesos

Historia de usuario	
Número: 17	**Usuario:** Administrador

Nombre de Historia de Usuario: Ordenar datos de los procesos	
Prioridad en negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 2/5	**Iteración Asignada:** 6
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: El sistema brinda la posibilidad de ordenar los datos de los procesos de las aplicaciones activas en las computadoras.	
Observaciones:	
Prototipo de interfaz:	

Tabla 32: HU#18 Filtrar por datos de los procesos

Historia de usuario	
Número: 18	**Usuario:** Administrador
Nombre de Historia de Usuario: Filtrar por datos de los procesos	
Prioridad en negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 2/5	**Iteración Asignada:** 6
Programador(es) responsable(s): Yoel Cornell Milián y Caridad Vinent Puig	
Descripción: El sistema brinda la posibilidad de mostrar los datos de los procesos de las aplicaciones según el tipo de filtro de búsqueda que desee, como por ejemplo: nombre, Pid, usuario, tiempo de uso, uso de memoria en porciento, tamaño en memoria RAM, estado (inactivo, normal o sobre activo).	
Observaciones:	
Prototipo de interfaz:	