UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 1

Sistema informático para la gestión de recursos de hardware y software en la Universidad de la Habana

Trabajo final presentado en opción al título de

Ingeniero en Ciencias Informáticas

Autor: Alejandro Soto Morales

Tutor: Dr C. José Ortiz Rojas

MSc. Aneyty Martín García

Ciudad de La Habana, Noviembre de 2021

Pensamiento

“Los proveedores de software están intentando hacer sus productos más amigables para el usuario. Su mejor aproximación hasta el momento ha sido tomar sus antiguos folletos y estampar las palabras 'amigable para el usuario' en la portada”

Bill Gates

“Los ordenadores son buenos siguiendo instrucciones, no leyendo tu mente”

Donald Knuth

Declaración jurada de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los _6_ días del mes de ___diciembre__ del año 2021.

Alejandro Soto Morales	**Dr C. José Ortiz Rojas**	**MSc. Aneyty Martín García**
Autor	**Tutor**	**Tutora**

Datos de contacto

Datos del Autor:

Alejandro Soto Morales

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: asoto@estudiantes.uci.cu

Datos del Tutor:

Dr C. José Ortiz Rojas. Pertenece a la Dirección de Formación de Pregrado.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: jortiz@uci.cu

Datos de la Tutora:

MSc. Aneyty Martín García, graduada de Ingeniero en Ciencias Informáticas. Pertenece al Dpto. de Informática. Facultad 1.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: amartin@uci.cu

Dedicatoria

Agradecimientos

Resumen

Mantener un seguimiento y control detallado de los software y hardware 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 sistema para monitorear los software y hardware de las computadoras en la Dirección de Informatización de la Universidad de la Habana (UH), capaz de monitorear todo el equipamiento de un ordenador y gestionar adecuadamente el inventario de software, es decir, obtiene el inventario del software, datos del sistema operativo, el antivirus, programas instalados, dominio y usuarios registrados en las computadoras, logrando así una aplicación que cumpla con las expectativas del cliente.

Palabras Clave: hardware, inventario, monitoreo, software.

Índice

Índice de Tablas

Índice de Figuras

Índice de Gráficos

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 (ALBALADEJO, 2009).

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 (ARAUZO AZOFRA & REVILLA, 2004). 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 (Arquitectura cliente servidor, 2010), 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 la Dirección de Informatización de la Universidad de la Habana (UH), el cual se encarga de monitorear todo el equipamiento de un ordenador y gestiona adecuadamente el inventario de software, es decir, obtiene el inventario del software, datos del sistema operativo, el antivirus, programas instalados, dominio y usuarios registrados en las computadoras, a través de la

herramienta Open Computer and Software Inventory (OCS Inventory). OCS 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.

El sistema aún no es capaz de gestionar la información referente a los software y hardware de las computadoras de forma intuitiva. Del mismo modo, este elemento ha dificultado la capacidad para controlar todos los recursos que se poseen, ya que resulta muy engorroso descubrir y determinar toda la información que se desea saber de las computadoras ya sea que estén defectuosos o que en determinado momento no se encuentran disponibles en el ámbito de trabajo. Surgiendo así la necesidad de crear inventarios de hardware y software para este tipo de recursos. Por consiguiente, disponer de un inventario de estos medios es tener la información, que puede ser vasta y muy detallada, necesaria para la toma de decisiones.

Ejemplo elocuente del tema en cuestión puede ser la información del modelo, procesador y la RAM (Memoria de Acceso Aleatorio), así como la lista de los software instalados disponibles con su versión, entre otros de igual relevancia. En tal sentido, la utilidad del inventario será superior en la medida que contenga mayor cantidad de datos, detallados de la mejor manera posible y con la calidad máxima; teniendo en cuenta la relevancia que estos datos puedan representar en el momento de efectuar el inventario. Por ende, para tal finalidad, sería de gran utilidad una herramienta que recoja los datos de un modo automático.

Otra de las problemáticas detectadas es la poca frecuencia con que se realizan los inventarios de estos recursos, hecho que incide en la antigüedad de los mismos, así como en que la información se encuentre en su mayoría descentralizada. Además, el hurto y las violaciones de privilegios y políticas de seguridad son cada vez más frecuentes y con perspectivas de crecimiento, debido a la débil presencia del control.

Pueden añadirse otros, como la posibilidad de ocasionar pérdidas monetarias, ya que la universidad se vería obligada a destinar parte de su presupuesto a adquirir nuevamente estos recursos, gastos cuya influencia afectaría la economía de la institución y, por consiguiente, la del país; ya que podría ser empleado este recurso monetario para la inversión en otro tipo de tecnología cuya utilidad sea más apremiante para el centro.

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.

A partir de todo lo planteado con anterioridad surge como **problema a resolver**: ¿Cómo contribuir al control de los recursos de hardware y software de la Universidad de la Habana?

Para llevar a cabo la investigación se plantea como **objeto de estudio**: Gestión de recursos de hardware y software y como **campo de acción:** Monitoreo de software instalado de las computadoras en la red.

Para contribuir a la solución del problema planteado se propone como **objetivo general**: Desarrollar un sistema informático para la gestión de los recursos de hardware y software de la Universidad de la Habana.

Para darle cumplimiento al objetivo se plantean las siguientes **preguntas científicas**:

1. ¿Cuáles son los referentes teóricos del proceso de monitoreo de los software y hardware en las computadoras de la Universidad de la Habana?
2. ¿Qué metodologías, herramientas, tecnologías y lenguajes resultan adecuadas para el desarrollo del sistema informático propuesto?
3. ¿Qué funcionalidades debe tener la aplicación para poder contribuir al proceso de monitoreo de los software y hardware en las computadoras de la Universidad de la Habana?
4. ¿Qué tipos de pruebas se pueden utilizar en la validación de la propuesta de solución y cómo se deben aplicar?

En correspondencia con las preguntas científicas enunciadas anteriormente se declaran las siguientes **tareas de investigación**:

1. Caracterización del proceso de monitoreo de los software y hardware en las computadoras para identificar las necesidades del sistema.
2. Caracterización de las principales herramientas, tecnologías, lenguajes y metodologías a utilizar para el desarrollo del sistema informático.
3. Desarrollo de las funcionalidades del sistema informático para el monitoreo del uso de los software y hardware en la red.
4. Definición de los tipos de pruebas para comprobar el funcionamiento del sistema.

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

Métodos Teóricos:

Histórico-Lógico: Se utilizó 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 utilizó para realizar el modelado del sistema informático a través de los artefactos correspondientes a las fases: Modelo del Negocio, Requisitos, Análisis y Diseño e Implementación.

Inducción – deducción: Para llegar de los elementos específicos observados a los generales y viceversa a partir de la observación y de las referencias bibliográficas estudiadas.

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 realizará una serie de entrevistas con el cliente involucrado y en base a éstas se trabajará para satisfacer sus necesidades.

La investigación se encuentra estructurada en: introducción, tres capítulos, conclusiones, recomendaciones, bibliografía y anexos. El primer capítulo está dedicado a la fundamentación teórica, se realiza un estudio sobre soluciones informáticas similares existentes en sistemas para la gestión de recursos de hardware y software en las computadoras, en Cuba y a escala mundial. Además, se analizan los principales elementos teóricos que constituyen la base de la investigación, entre los cuales se encuentran la metodología de desarrollo para el sistema, las tecnologías y herramientas a utilizar en la propuesta de solución.

En el segundo capítulo se definió la solución que se propone y se realizó la selección de los requerimientos del sistema que se pretende implementar. Se hizo el análisis de la solución que se propone, se modelan y describen los recursos ingenieriles necesario acorde a la metodología XP que representan las funcionalidades del sistema, aplicando los patrones de arquitectura y diseño seleccionados.

En el tercer capítulo se describió la implementación y posterior validación realizada al producto obtenido como solución.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

Capítulo 1: Fundamentación teórica

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.

1.1 Conceptos fundamentales

A continuación se enuncian conceptos fundamentales para ayudar al entendimiento del proceso de gestión de la información de recursos de hardware y software.

Gestión de Información: La gestión de información es el proceso que se encarga de suministrar los recursos necesarios para la toma de decisiones, así como para mejorar los procesos, productos y servicios de la organización. Con el objetivo de mostrar cómo influye la gestión de información en el desarrollo y fortalecimiento de las instituciones. Se define como el conjunto de actividades realizadas con el fin de controlar, almacenar y, posteriormente, recuperar adecuadamente la información producida, recibida o retenida por cualquier organización en el desarrollo de sus actividades. (Bustelo Ruesta & Amarilla Iglesias, 2021)

Hardware: El término hardware se refiere a todas las partes físicas de un sistema informático (Alegsa, 2021). Son cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado. El término es propio del idioma inglés (literalmente traducido: partes duras); la Real Academia Española lo define como el conjunto de componentes que integran la parte material de una computadora (Española, 2021). No solamente se aplica a las computadoras; del mismo modo, también un robot, un dispositivo móvil, una cámara fotográfica, un reproductor multimedia o cualquier otro electrónico que procese datos poseen hardware. El hardware por sí solo no puede hacer nada, pues es necesario que exista el software, que es el conjunto de instrucciones que hacen funcionar al hardware.

Software: El software es un conjunto de programas, documentos, procedimientos, y rutinas asociados con la operación de un sistema de cómputo. Distinguiéndose de los componentes físicos llamados hardware. Comúnmente a los programas de computación se les llama software; el software asegura que el programa o sistema cumpla por completo con sus objetivos, opera con eficiencia, está adecuadamente documentado, y suficientemente sencillo de operar. Es simplemente el conjunto de instrucciones individuales que se le proporciona al microprocesador para que pueda procesar los datos y generar los resultados esperados (Alegsa, 2021). Un software

puede ser ejecutado por cualquier dispositivo capaz de interpretar y ejecutar las instrucciones para lo cual es creado.

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 (CHAUVIN, 2013), 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. 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. (DATOS, 2012)

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 (CHAUVIN, 2013). 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.

Gestión de Información de Recursos de Hardware y Software: Se define la gestión de información de recursos de hardware y software como el proceso de planificación y control de todos los recursos de hardware y software, destinado a facilitar la toma de decisiones disponiendo de información precisa sobre estos (Tamayo Betancourt & Ciervide Cabalé, 2015).

1.2 Sistemas para la gestión de recursos de hardware y software

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.

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. ManicTime 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 (Gaikar, 2013).

A pesar de que ManicTime 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.

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. Dicha herramienta brinda una serie de ventajas y funcionalidades (referentes al monitoreo de software) (Shepherd, Toloza, McClung, & Schmalzried, 1999):

- 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.

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 (Yoanni, Ernesto, Julio, & Odaysa):

- 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 (Yoanni, Ernesto, Julio, & Odaysa):

- 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).

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.

Open Computer and Software Inventory (OCS Inventory)

OCS 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. 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 (Tipantuña Heredia, 2017):

- 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.

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, el sistema aún no es capaz de gestionar la información referente a los software y hardware de las computadoras de forma intuitiva. Del mismo modo, este elemento ha dificultado la capacidad para controlar todos los recursos que se poseen, ya que resulta muy engorroso descubrir y determinar toda la información que se desea saber de las computadoras ya sea que estén defectuosos o que en determinado momento no se encuentran disponibles en el ámbito de trabajo.

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 (Tamayo Betancourt & Ciervide Cabalé, 2015).

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.

1.2.1 Resultado del estudio de las soluciones homólogas

Para el estudio y comparación de las herramientas anteriores se resumen en la siguiente tabla alguna de sus características fundamentales:

Tabla 1.2.1 Comparativa de las soluciones homólogas.

Indicadores comparativos	ManicTime	Activity Monitor	VEO Ultimate	OCS Inventory	GRHS
Versiones para Windows.	x	x	x	x	x
Versiones para GNU/Linux.				x	x
Inventario de los recursos de hardware y software.	x	x	x	x	x
Gestiona la información de forma intuitiva.					x
Software propietario.	x	x	x		
Reporte equipos fuera de servicios.					
Visualización de los datos de los software por computadoras.		x	x		
Visualización de las estadísticas generales de los software.					
Posibilidad de exportar a .csv los datos de los software por computadoras.					

Por lo mostrado anteriormente, las soluciones en estudio responden a necesidades del entorno productivo para el cual fueron creadas, por lo que no son lo suficientemente flexibles como para adaptarse a las necesidades planteadas en la problemática de esta investigación.

Luego de realizar el estudio de soluciones homólogas y determinar la necesidad de confeccionar una nueva solución, se hace necesario hacer un análisis de la metodología que regirá el proceso de desarrollo, así como las tecnologías y herramientas a emplear. La adopción y puesta en marcha de lo antes expuesto estará vinculado a las experiencias adquiridas en el estudio de las soluciones homólogas.

1.3 Estudio de metodologías, tecnologías y herramientas

Un proceso de software se define como un marco de trabajo para las tareas que se requieren en la construcción de software de alta calidad. Un proceso define el enfoque que se adopta mientras el software está en desarrollo (Labroots, 2017).

A continuación, se realiza un análisis de las metodologías con el propósito de determinar la más conveniente para la solución propuesta. Además, se describen las tecnologías y herramientas seleccionadas para el desarrollo de la solución.

1.3.1 Fundamentación de la selección de los modelos

Para lograr la consecución exitosa de un proyecto de software, existen un conjunto de modelos o metodologías que permiten perfilar el proceso de desarrollo, los cuales deben ser analizados cuidadosamente. La selección de uno de estos modelos o de la unión de varios responde a las necesidades y condiciones específicas del proyecto. Estos modelos o metodologías se pueden clasificar en cinco grandes grupos (Malea, Orduna, Martín y López Cózar, & Delgado, 2016):

- Modelo de cascada: es un proceso secuencial de desarrollo en el que los pasos son vistos hacia abajo (como en una cascada de agua).
- Modelo iterativo incremental: provee una estrategia para controlar la complejidad y los riesgos, desarrollando una parte del producto de software reservando el resto de los aspectos para el futuro.
- Modelo espiral: en cierto modo, es como una versión más grande e intensa del modelo incremental.
- Modelos ágiles: son aquellas que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno.
- Modelo de prototipos: en el método de creación de prototipos, el software es desarrollado en forma similar a una cebolla. Cuando el proyecto se inicia, el equipo se centra en la construcción de un prototipo con todos o la mayoría de las características y construye todo un programa para que el cliente lo utilice.

Se escogió el modelo iterativo incremental puesto que responde a una situación donde los requisitos están bien definidos, pero es necesario satisfacer al cliente de forma rápida con conjuntos limitados de funcionalidad en pequeñas porciones que aumentan gradualmente y se refinan y expanden en cada entrega.

1.3.2 Fundamentación de la selección de la metodología

Una metodología de desarrollo de software se refiere al marco que se utiliza para estructurar, planificar y controlar el proceso de desarrollo de un sistema de información. Una amplia variedad de dichos marcos, han evolucionado a lo largo de los años con sus propias fortalezas y debilidades reconocidas. Cada una de las metodologías disponibles se adapta mejor a tipos específicos de

proyectos, en función de diversas consideraciones técnicas, organizativas, de proyecto y de equipo (Pressman, 2011).

Para guiar el proceso de desarrollo de la solución planteada se empleó la Metodología eXtreme Programming (XP) ya que es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. La metodología XP se basa en la 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. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (Varios, 2017).

Su selección se basa en las características que posee el proyecto y su equipo de desarrollo para la obtención de entregas funcionales con la mayor calidad requerida. La interrelación entre los clientes, grupos de proyectos y desarrolladores se hacen semanalmente con una comunicación entre todas las entidades beneficiadas.

La metodología XP especifica las siguientes fases que guiarán el proceso de desarrollo y será determinada en cada acápite de este documento:

- Fase I: Exploración
- Fase II: Planificación de las entregas
- Fase III: Diseño
- Fase IV: Implementación (Desarrollo)
- Fase V: Pruebas

En la Figura 1.3.1, se puede observar una mejor visualización de estas fases, según las especificaciones dadas por la Metodología XP.

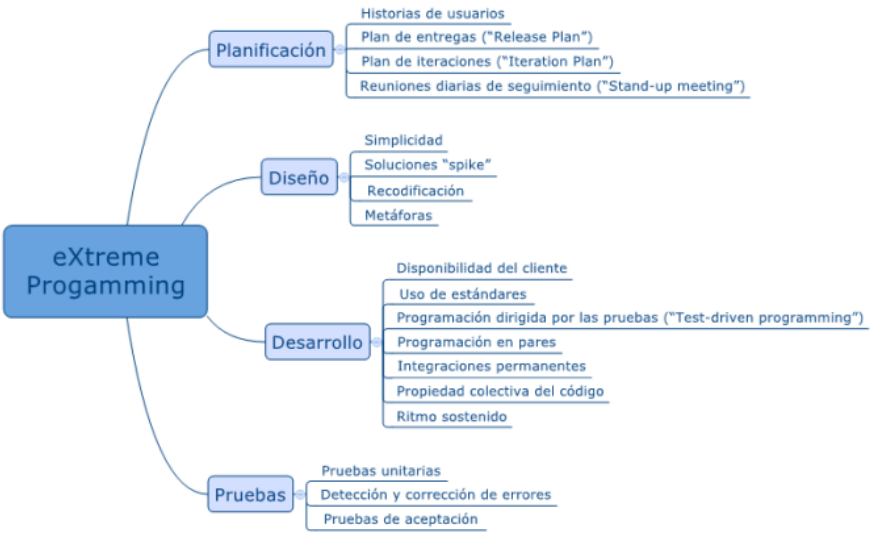


Figura 1.3.1 Fases de desarrollo de la metodología XP.

Fuente: "Introducción a las Metodologías de Desarrollo de Software" realizado por (Varios, 2017).

1.4 Modelado, lenguajes de programación y otras herramientas

En este epígrafe se habla sobre la herramienta de modelado empleada para crear algunos artefactos, así como los diferentes lenguajes de programación y el uso de otras herramientas conocidas por bibliotecas y marcos de trabajo, que ayudaron a la confección de la solución propuesta.

1.4.1 Selección del framework de desarrollo

Para el desarrollo de la aplicación es necesario el uso de los frameworks[1], y primero hay que entender su significado y para qué sirve. En el desarrollo de software, un framework o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos concretos de software, 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.

Los frameworks tienen como objetivo principal ofrecer una funcionalidad definida, auto contenido, siendo construidos usando patrones de diseño, y su característica principal es su alta cohesión y bajo acoplamiento. Para acceder a esa funcionalidad, se construyen piezas, objetos, llamados objetos calientes, que vinculan las necesidades del sistema con la funcionalidad que este presta.

[1] Del inglés: infraestructura, armazón, marco.

Esta funcionalidad, está constituida por objetos llamados fríos, que sufren poco o ningún cambio en la vida del framework, permitiendo la portabilidad entre distintos sistemas (Framework, s.f.). 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 3.2.

El **framework 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 ModeloPlantilla-Vista, esta forma de trabajar permite que sea pragmático (MARTÍNEZ GUAITA, 2012). Django se encarga de gran parte de las complicaciones del desarrollo web, por lo que puedes concentrarte en escribir tu aplicación sin necesidad de reinventar la rueda. Es gratuito y de código abierto, tiene una comunidad próspera y activa, una gran documentación y muchas opciones de soporte gratuito y de pago.

Django te ayuda a escribir software que es (Django, 2021):

- Completo: Django sigue la filosofía "Baterías incluidas" y provee casi todo lo que los desarrolladores quisieran que tenga "de fábrica". Porque todo lo que necesitas es parte de un único "producto", todo funciona a la perfección, sigue principios de diseño consistentes y tiene una amplia y actualizada documentación.
- Versátil: Django puede ser (y ha sido) usado para construir casi cualquier tipo de sitio web — desde sistemas manejadores de contenidos y wikis, hasta redes sociales y sitios de noticias. Puede funcionar con cualquier framework en el lado del cliente, y puede devolver contenido en casi cualquier formato (incluyendo HTML, RSS feeds, JSON, XML, etc).
- Seguro: Django ayuda a los desarrolladores evitar varios errores comunes de seguridad al proveer un framework que ha sido diseñado para "hacer lo correcto" para proteger el sitio web automáticamente. Por ejemplo, Django, proporciona una manera segura de administrar cuentas de usuario y contraseñas, evitando así errores comunes como colocar informaciones de sesión en cookies donde es vulnerable (en lugar de eso las cookies solo contienen una clave y los datos se almacenan en la base de datos) o se almacenan directamente las contraseñas en un hash de contraseñas.
- Mantenible: El código de Django está escrito usando principios y patrones de diseño para fomentar la creación de código mantenible y reutilizable. En particular, utiliza el principio No te repitas "Don't Repeat Yourself" (DRY) para que no exista una duplicación innecesaria, reduciendo la cantidad de código. Django también promueve la agrupación de la funcionalidad relacionada en "aplicaciones" reutilizables y en un nivel más bajo, agrupa código relacionado en módulos (siguiendo el patrón Model View Controller (MVC)).

- Portable: Django está escrito en Python, el cual se ejecuta en muchas plataformas. Lo que significa que no está sujeto a ninguna plataforma en particular, y puede ejecutar sus aplicaciones en muchas distribuciones de Linux, Windows y Mac OS X. Además, Django cuenta con el respaldo de muchos proveedores de alojamiento web, y que a menudo proporcionan una infraestructura específica y documentación para el alojamiento de sitios de Django.

1.4.2 Selección de la biblioteca JavaScript

La biblioteca **jQuery 3.5** es de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM[2], manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Es la biblioteca de JavaScript más utilizada. (The Write Less, Do More, JavaScript Library, s.f.)

Esta biblioteca de JavaScript es un software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT[3] y la Licencia Pública General de GNU[4] v2, permitiendo su uso en proyectos libres y privados. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. (jQuery, s.f.)

1.4.3 Selección del framework CSS

Para la realización del diseño de una aplicación es necesario utilizar un framework de diseño o CSS, se realizó una investigación con algunos de los frameworks CSS más usados en el mundo.

El **framework Gumby** fue desarrollado a la base de Saas, que es un pre-procesador potente de CSS. Actualmente Gumby está disponible en la versión 2.6 con muchas características Útiles, estilos predefinidos y elementos de interfaz comunes, tales como palancas, menús desplegables, botones, pestañas etc. Con la ayuda de Gumby, se puede personalizar y formar una cuadrícula (grid) para su diseño. Por otra parte, se puede adaptar a cualquier tamaño y resolución de pantalla. (Gumby, s.f.)

En el caso de **Kube**, es adaptable y compatible con todo lo necesario para el diseño del sitio web. Presenta la mayoría de las características de Gumby, pero a diferencia del anterior tiene una cuadrícula (grid) flexible revolucionaria con asombrosa tipografía. (Kube CSS Framework, s.f.)

[2] Del inglés: Document Object Model, en español Modelo de Objetos del Documento o Modelo en Objetos para la Representación de Documentos.
[3] Licencias de software que ha empleado el Instituto Tecnológico de Massachusetts.
[4] Del inglés: General Public License, garantiza a los usuarios finales la libertad de usar, estudiar, compartir (copiar) y modificar el software.

La biblioteca **YAML**[5] es ideal para los sitios web flexibles, accesibles y adaptables (responsive). El framework ofrece un conjunto de bloques para la construcción de sitios web. Los elementos incluyen la funcionalidad de tipografía, la navegación, los formularios etc. Este framework contiene un buen Ajax constructor que es una herramienta muy útil para el desarrollo visual del diseño CSS. (YAML CSS Framework, s.f.)

A pesar de las buenas propiedades de los frameworks anteriormente mencionados se ha elegido como el indicado para el desarrollo de la aplicación el siguiente:

El framework o conjunto de Herramientas de software libre **Bootstrap** en su versión 4 es para el diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales (Bootstrap, 2013).

Algunas de las ventajas que nos brinda Bootstrap se desglosan a continuación:

- Ofrece un paquete de elementos web personalizables: con Bootstrap se puede diseñar una web jugando con sus elementos compuestos por diferentes combinaciones de HTML, CSS y Javascript, de manera que las piezas siempre encajan.
- Utiliza componentes vitales para los desarrolladores: como HTML5, CSS3, jQuery o GitHub, entre otros.
- Sus plantillas son de sencilla adaptación. Se desarrolló con la idea de facilitar el proceso de adaptación web a todo tipo de dispositivos.
- Incluye Grid system: muy útil para maquetar por columnas.
- Se integra con librerías JavaScript.
- Usa Less: un lenguaje de las hojas de estilo CSS preparado para enriquecer los estilos de la web.
- Es una herramienta de uso ágil y sencillo: facilita enormemente el diseño de interfaces y además incluye por defecto una plantilla bastante optimizada.
- De manera general Bootstrap es perfecto para los diseñadores de todos los niveles, los proyectos de todos los tamaños, así como dispositivos de todas las formas. (Bootstrap, s.f.)

1.4.4 Lenguajes de programación

En este epígrafe se habla sobre diversos lenguajes de programación empleados en la confección de la solución deseada. Estos son utilizados como un sistema para dar forma a la plataforma web y obtener una aplicación funcional que resuelva la problemática de esta investigación.

[5] Del inglés: Yet Another Multicolumn Layout, en español OtroDiseñode Varias Columnas

CSS3

CSS[6], en español "Hojas de estilo en cascada", es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. CSS3 no solo abarca diseño y estilos web sino también forma y movimiento. La especificación de CSS3 es presentada en módulos que permiten a la tecnología proveer una especificación estándar por cada aspecto involucrado en la presentación visual del documento (Carlos & Zapata, 2010).

Se decide emplear en el proyecto para crear una interfaz fresca, amena e intuitiva para los usuarios de la plataforma cumpliendo con las normas de diseño establecidas por Google para Material Design.

Python 3.5.1

Python es un lenguaje de programación poderoso y fácil de aprender. Posee estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. Tiene una cómoda sintaxis y un tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas (Reitz, 2016).

El intérprete de Python y la extensa biblioteca estándar están a libre disposición en forma binaria y de código fuente para las principales plataformas desde el sitio web de Python, http://www.python.org/, y puede distribuirse libremente. El mismo sitio contiene también distribuciones y enlaces de muchos módulos libres de Python de terceros, programas y herramientas, y documentación adicional (Reitz, 2016).

Se decidió emplear el lenguaje de programación Python, ya que es muy efectiva su utilización en el procesamiento de grandes volúmenes de datos. Usados en la ciencia de datos para la toma de decisiones, hacer predicciones con bases en los datos, entender los datos y extraer información que pueda generar valor.

A la hora de optar por usar Python como lenguaje de programación, se presenta la problemática de escoger entre dos grandes versiones, la 2.x y las 3.x. Para la implementación de la solución de esta investigación se ha seleccionado Python 3.5.1 por las siguientes razones:

- Todas las clases son ahora de nuevo estilo “range()” mejorando la eficiencia en memoria comparado con 2.x.

[6] Del inglés: Cascading Style Sheets, en español Hoja de Estilo en Cascada

- Dentro de la 3.x la versión 3.5 es la versión más extendida y estable en el mundo.
- Soporte Unicode (con todas las cadenas con texto Unicode por defecto), así como una separación bytes/Unicode más coherente.
- Crecimiento en el uso del lenguaje sobretodo en su difusión en SO como Linux.
- Existen distribuciones de Anaconda para Python 3.5.1 con todas las librerías necesarias para hacer ciencia de datos y beneficiarse de los recursos de Python 3.
- Python 2.x es un lenguaje antiguo con muchos errores.

HTML5

HTML5 no es una nueva versión del antiguo lenguaje de etiquetas, sino un nuevo concepto para la construcción de sitios web y aplicaciones en una era que combina dispositivos móviles, computación en la nube y trabajos en red (Carlos & Zapata, 2010). Es un sistema para formatear las páginas y hacer ajustes a su aspecto. A través de este, los navegadores como Mozilla Firefox, Google Chrome, Internet Explorer y otros pueden saber cómo mostrar una página web determinada, ubicar los elementos, texto e imágenes. Se puede reducir la dependencia de complementos y amplía el horizonte del desarrollo de aplicaciones que pueden ser usadas en múltiples dispositivos.

Se decidió su implementación debido a que es el lenguaje de estructuración básica actual de los elementos que se emplea en el diseño y maquetado de las interfaces de la plataforma.

Javascript

Javascript es un lenguaje interpretado usado para múltiples propósitos, pero solo considerado como un complemento hasta ahora. Una de las innovaciones que ayudó a cambiar el modo en que vemos Javascript fue el desarrollo de nuevos motores de interpretación, creados para acelerar el procesamiento de código. La clave de los motores más exitosos fue transformar el código Javascript en código máquina para lograr velocidades de ejecución similares a aquellas encontradas en aplicaciones de escritorio (Carlos & Zapata, 2010).

En la plataforma se decide utilizar debido a que este lenguaje permite darle dinamismo y versatilidad a la plataforma, así como el chequeo de determinados parámetros que pueden dañar al servidor del sistema.

1.4.5 Entorno de desarrollo integrado

Un ambiente de desarrollo integrado o entorno de desarrollo interactivo, IDE[7], es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el

[7] Del inglés: Integrated Development Environment, en español Ambiente de Desarrollo Integrado

desarrollo de software. Normalmente, un IDE consiste de un editor de código fuente, Herramientas de construcción automáticas y un depurador. La mayoría de los IDE tienen auto-completado inteligente de código (IntelliSense). Algunos IDE contienen un compilador, un intérprete, o ambos (Caravaca-Mora, 2010).

El límite entre un IDE y otras partes del entorno de desarrollo de software más amplio no está bien definido. Muchas veces, a los efectos de simplificar la construcción de la interfaz gráfica de usuario (GUI, por sus siglas en inglés) se integran un sistema controlador de versión y varias Herramientas. Muchos IDE modernos también cuentan con un navegador de clases, un buscador de objetos y un diagrama de jerarquía de clases, para su uso con el desarrollo de software orientado a objetos. (KOSKELA, 2008)

El IDE de desarrollo más adecuado para la realización de la aplicación según sus desarrolladores debido a su usabilidad y funcionalidades es: El **IDE Visual Studio Code** es un editor de código fuente. Es compatible con varios lenguajes de programación y un conjunto de características que pueden o no estar disponibles para un lenguaje dado, como se muestra en la siguiente tabla. Muchas de las características de Visual Studio Code no están expuestas a través de los menús o la interfaz de usuario. Más bien, se accede a través de la paleta de comandos o a través de archivos .json (por ejemplo, preferencias del usuario). La paleta de comandos es una interfaz de línea de comandos. Sin embargo, desaparece si el usuario hace clic fuera de él o presiona una combinación de teclas en el teclado para interactuar con algo que está fuera de él. Esto también se aplica a los comandos que requieren mucho tiempo. Cuando esto sucede, el comando en progreso se cancela.

En el rol de editor de código fuente, Visual Studio Code permite cambiar la página de códigos en la que se guarda el documento activo, el carácter que identifica el salto de línea (una opción entre LF y CRLF) y el lenguaje de programación del documento activo (Lardinois, 2015).

1.4.6 Herramienta de Ingeniería de Software Asistida por Computación

Usualmente para desarrollar una solución informática con calidad y competitiva, el uso de lenguajes de modelado es prácticamente indispensable. Esta necesidad responde a que el uso de estos lenguajes permite que el desarrollo de software se formalice a través de estándares unificados. Los lenguajes de modelado utilizan modelos visuales y diagramas que realizan esa representación de manera precisa, entendible y económica, lo que facilita su uso en las herramientas de Ingeniería de Software Asistida por Computación (CASE) convencionales (Selecting a Developement Approach., 2017).

Visual Paradigm

Visual Paradigm es una herramienta CASE. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (Canós, Letelier, & Penadés, 2005).

Al ser una de las mejores y más completas herramientas CASE, se decidió emplear en el modelado de la solución a desarrollar. Se evidencia su uso en el modelo de entidad relación que se encuentra en el Anexo 1.

1.4.7 Servidor de Bases de Datos

Los Servidores de Bases de Datos. También conocidos como RDBM[8], son programas que permiten organizar datos en una o más tablas relacionadas. Los servidores de Bases de Datos se utilizan en todo el mundo en una amplia variedad de aplicaciones (ALBA Software, s.f.). Para bases de datos con múltiples usuarios sirve un servidor de base de datos. Las bases de datos están situadas en un servidor y se puede acceder a ellas desde terminales o equipos con un programa - llamado cliente- que permita el acceso a la base o bases de datos. Los gestores de base de datos de este tipo permiten que varios usuarios hagan operaciones sobre ella al mismo tiempo (ASENSIO & Menéndez-Barzanallana, 2011).

El servidor seleccionado por la familiarización del equipo de desarrollo y por su sistema de gestión es el RDBM **MySQL**. MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM[9], pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones. Sea cual sea el entorno en el que va a utilizar MySQL, es importante monitorizar de antemano el rendimiento para detectar y corregir errores tanto de SQL[10] como de programación. (MySQL, s.f.)

1.4.8 Servidor Web

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web.

[8] Del inglés: Relational Data Base Management System, en español Servidor de Bases de Datos
[9] Es el mecanismo de almacenamiento de datos usado por defecto por el sistema administrador de bases de datos relacionales MySQL hasta su versión 5.5.
[10] Del inglés: Structured Query Language, en español Lenguaje de Consulta Estructurado. Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas

Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI. El término también se emplea para referirse al ordenador que ejecuta el programa. (World Wide Web Consortium (W3C), s.f.)

El servidor web que se usó para desplegar la aplicación de esta investigación fue **Apache** en su versión 2.4.48 es un servidor de aplicaciones web con licencia de software libre creada por la Apache Software Foundation (ASF). La licencia Apache requiere la conservación del aviso de copyright, no requiere la redistribución del código fuente cuando se distribuyen versiones modificadas. Todo el software producido por la ASF o cualquiera de sus proyectos está desarrollado bajo los términos de esta licencia. Además, algunos proyectos que no pertenecen a la ASF también siguen la licencia Apache. (Apache License, s.f.)

1.5 Herramienta de validación

Las pruebas de validación en la ingeniería de software son el proceso de revisión que verifica que el sistema de software producido cumple con las especificaciones y que logra su cometido. La validación es una parte del proceso de pruebas de software de un proyecto, que también utiliza técnicas tales como evaluaciones, inspecciones y tutoriales. Es el proceso de comprobar que las especificaciones del usuario fueron cumplidas (Pressman, 2011). Para realizar estas pruebas se utilizan herramientas, en el desarrollo de la solución se empleó Acunetix y Apache JMeter.

1.5.1 Acunetix

Acunetix WVS por sus siglas en inglés (Web Vulnerability Scanner) es una herramienta de seguridad de aplicaciones web automatizada. Es capaz de escanear cualquier sitio o aplicación web que sea accesible a través del protocolo HTTP/HTTPS. Sin embargo, no todas las pruebas se pueden realizar de forma automática, por lo tanto, Acunetix WVS proporciona herramientas de penetración manuales para pruebas particulares. Comprueba diferentes vulnerabilidades como inyecciones SQL, XSS, XXE, SSRF, Host Header Injection y otras 4500 vulnerabilidades web.

En el Anexo 5 se describe los resultados emitidos por esta herramienta al escanear el sistema una vez concluidas las iteraciones. Los diferentes errores detectados fueron solucionados y otros para su solución dependen del despliegue en un entorno real.

1.5.2 Apache JMeter

Apache JMeter, es una herramienta de código abierto que se utiliza para realizar pruebas de rendimiento y resistencia, habitualmente a aplicaciones web. JMeter permite realizar simulaciones de gran carga en el servidor, red o aplicación para comprobar su “fuerza” y para analizar el

rendimiento ante diferentes tipos de sobrecarga. Permite cargar y realizar pruebas sobre distintos tipos de servidores mediante un sistema de pruebas distribuido. JMeter es multiplataforma y permite realizar pruebas de forma concurrente. A esto se añade su interfaz gráfica permitiendo realizar las operaciones a una mayor velocidad (Apache JMeter, s.f.).

1.6 Conclusiones del capítulo

- El estudio de los conceptos asociados de la investigación permitió obtener un mayor entendimiento de la misma.
- Se detectaron las características, ventajas y desventajas de los sistemas homólogos, permitió demostrar la necesidad de implementar una nueva propuesta de solución.
- El análisis y estudio de la metodología XP permitió identificar sus características y artefactos para guiar el proceso de desarrollo del software.

CAPÍTULO 2

DISEÑO Y DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN

Capítulo 2: Diseño y descripción de la propuesta de solución

En este capítulo se aborda la propuesta de solución a la problemática planteada. Se describen los requisitos funcionales y no funcionales, para dar cumplimiento a los objetivos planteados. Se realizan las historias de usuario, plan de iteraciones, tarjetas CRC y demás artefactos exigidos por la Metodología XP.

2.1 Fase I: Exploración

Se cumple con lo establecido en la Fase I de la Metodología XP, donde el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2.1.1 Propuesta de solución

La solución propuesta está diseñada para la Dirección de Informatización de la Universidad de la Habana (UH). El objetivo del sistema que se propone en la presente investigación es contribuir a la gestión de inventario que este realiza monitorizando cada uno de los procesos de cada software que se utiliza en las áreas de trabajo, detectando modificaciones en estos y respondiendo ante las incidencias detectadas. De igual manera, será capaz de mostrar toda la información de hardware y software que poseen.

El sistema que se propone en la presente investigación estará formado por dos scripts (cliente y servidor). El objetivo del script cliente, una vez instalado en las computadoras clientes, es contribuir al monitoreo de software y hardware de cada computadora en las áreas de trabajo. El objetivo del script servidor es procesar la información recibida por los clientes.

El flujo de ejecución del sistema inicia en las computadoras clientes cuando se obtiene la información de los software y hardware a monitorear, enviando periódicamente la información al servidor. Por otro lado, el servidor procesa la información recibida por los clientes para detectar nuevos equipos de cómputos en la red o modificaciones en los ya existentes. Toda la información es almacenada en la base de datos de donde la aplicación web obtiene la información y la muestra al usuario final en forma de gráficos y tablas. A continuación, se muestra una imagen con el flujo de ejecución del funcionamiento del sistema para un mejor entendimiento de la propuesta de solución:

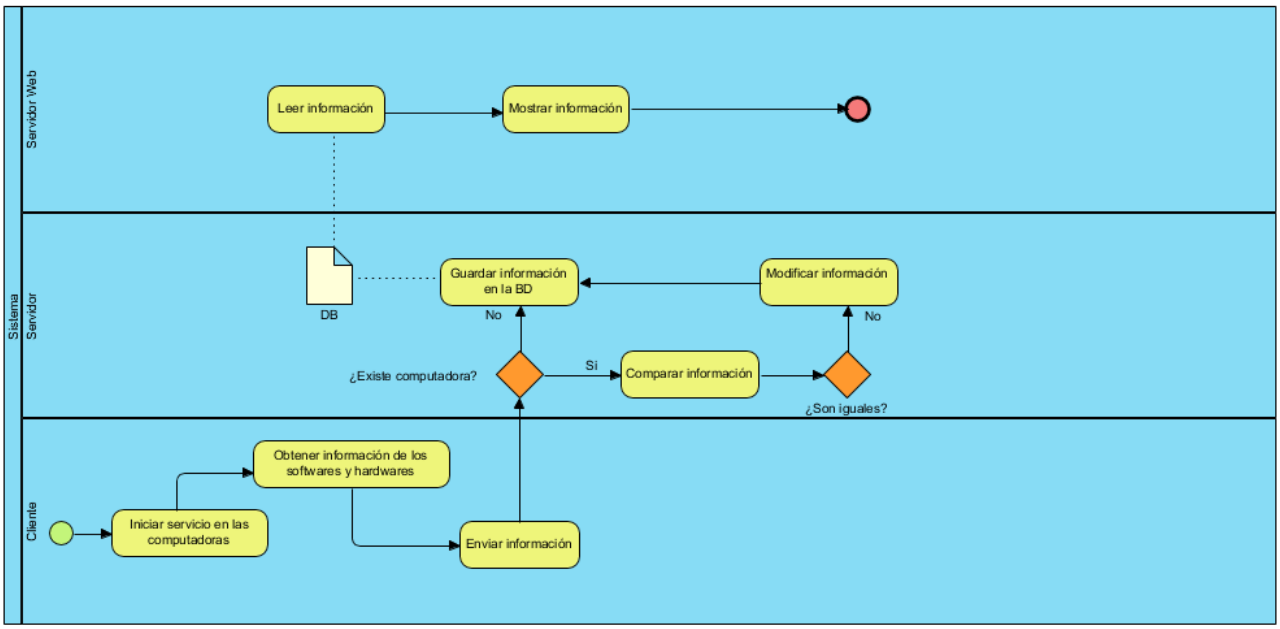


Figura 2.1.1 Modelo de negocio del sistema

2.1.2 Requisitos del sistema

El análisis de requisitos es una de las tareas más importantes en el ciclo de vida del desarrollo de software, puesto que en ella se determinan los "planos" de la nueva aplicación. El análisis de requisitos se puede definir como el proceso del estudio de las necesidades de los usuarios para llegar a una definición de los requisitos del sistema, hardware o software, así como el proceso de estudio y refinamiento de dichos requisitos (Rossum, 2009).

Requisitos funcionales

Los requisitos funcionales describen la interacción entre el sistema y su ambiente independientemente de su implementación. El ambiente incluye al usuario y cualquier otro sistema externo que interactúa con el sistema. Los requisitos definidos por el cliente se listan a continuación:

RF1_Identificar software instalados.
RF2_Visualizar software.
RF3_Modificar software.
RF4_Identificar propiedades del CPU, la RAM, el disco y la red.
RF5_Visualizar las propiedades del CPU, la RAM, el disco y la red.
RF6_Modificar las propiedades del CPU, la RAM, el disco y la red.
RF7_Reportar computadoras fuera de servicios.
RF8_Eliminar computadora.
RF9_Realizar monitoreo de software.

RF10_Realizar monitoreo de hardware.
RF11_Visualizar datos de los software por computadoras.
RF12_Exportar a .csv los datos de los software por computadoras.

Requisitos no funcionales

Los requisitos no funcionales describen aspectos del sistema que son visibles por el usuario que no incluyen una relación directa con el comportamiento funcional del sistema. Los requerimientos no funcionales incluyen restricciones como el tiempo de respuesta (desempeño), la precisión, recursos consumidos, seguridad, etc.

Interfaz:

RNF1_La comunicación entre el cliente y el servidor de aplicaciones se lleva a través del protocolo HTTPS.

RNF2_La comunicación entre el servidor de aplicaciones y la base de datos se lleva a través del protocolo TCP/IP.

RNF3_El diseño de la interfaz se regirá por la filosofía establecida en Material Design para la confección de páginas web dinámicas. Esto enfoca el uso de los colores, las sombras, la profundidad, las superficies y los bordes en un diseño limpio y agradable a la vista.

RNF4_El sistema debe garantizar una correcta organización de la información para permitir una adecuada interpretación.

Hardware:

RNF5_El servidor de aplicación y el de base de datos deberán tener los siguientes componentes de hardware para su funcionamiento en el ambiente en que se encuentra desplegado el sistema: Microprocesador de 8 núcleos, 4GB de memoria RAM, 250 GB disco duro y una fuente de 800 W como mínimo.

RNF6_Para la ejecución del sistema se requiere que la PC cliente tenga los siguientes componentes de hardware: Pentium 4 o superior, 512 MB RAM como mínimo.

RNF7_Los hardware clientes y servidor deben poseer una tarjeta de interconexión de red que permita acceder y responder a las peticiones respectivamente.

Software:

RNF8_Para el despliegue del sistema se contará en el servidor de base de datos con MySql 8.0 o superior, servidor de aplicaciones web Apache 2.4 o superior, Python 3.5.1 o superior.

RNF9_Es necesario tener instalada los paquetes de Python: python3-pip, python3-listresources, python3-mysqldb, pyspectator, tornado, pylist.

RNF10_Para el uso del sistema se requiere una PC cliente con el navegador web Mozilla Firefox 3.6 o superior.

Eficiencia:

RNF11_El sistema soportará la conexión simultánea de un promedio de 50 y un máximo de 500.

RNF12_El tiempo promedio de respuesta del sistema no excederá los 8 segundos de manera general.

Portabilidad:

RNF13_El sistema debe desarrollarse de forma tal que cuente con la posibilidad de ser multiplataforma.

Soporte:

RNF14_El sistema se regirá por un estándar de código debidamente documentado en el expediente de proyecto.

RNF15_Los productos de trabajo generados en el desarrollo del software se regirán por las pautas de configuración debidamente documentadas en el expediente de proyecto.

Usabilidad:

RNF16_El sistema debe estar dirigida a registrar y gestionar la información referente al proceso de gestión de recursos de SW y HD.

RNF17_Solo se mostrarán a los usuarios aquellas acciones o informaciones a las que por su responsabilidad o rol dentro del negocio necesitan acceder mostrando en la vista mediante iconos y menús el acceso.

RNF18_La información que se registra estará protegida de modificaciones no deseadas de acuerdo a la estrategia de seguridad definida.

RNF19_El sistema debe presentar un menú lateral que permitan el acceso rápido a la información por parte de los usuarios, es decir la interfaz del sistema debe ser amigable a los usuarios finales.

RNF20_El sistema debe contar con una adecuada combinación de color y tipografía.

2.2 Fase II: Planificación de las entregas

Se cumple con la Fase II de la Metodología XP donde 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. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres semanas.

2.2.1 Historias de Usuario

Las historias de usuario (HU) son la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

A continuación, se muestran las HU de los RF9 y RF10 por el nivel de prioridad para ser desarrollados, el resto de las historias de usuario pueden verse en el Anexo 2.

Tabla 2.2.1 Historia de Usuario HU-09

Historia de Usuario

Código	HU-09	**Nombre**	RF9_Realizar monitoreo de software
Usuario	Sistema	**Tipo de actividad**	Nueva
Riesgo	Alto	**Prioridad**	Alta
Iteración	2	**Puntos estimados**	2
Descripción	Permitirá monitorear los software por cada computadora, para conocer información de aquellos que se instalan o desinstalan, además permitirá realizar un monitoreo referente al sistema operativo en el que corre la aplicación, programas instalados, antivirus, usuarios conectados, dominio de la computadora, información del bios.		

Tabla 2.2.2 Historia de Usuario HU-10

Historia de Usuario			
Código	HU-10	**Nombre**	RF10_Realizar monitoreo de hardware
Usuario	Sistema	**Tipo de actividad**	Nueva
Riesgo	Alto	**Prioridad**	Alta
Iteración	2	**Puntos estimados**	2
Descripción	Permitirá monitorear los hardware por cada computadora, para conocer información de RAM, CPU, disco, marca, modelo, entre otros.		

2.2.2 Estimación de tiempo por historia de usuario

La programación bajo la metodología XP (programación extrema) basa sus procesos de planificación en estimaciones temporales de las historias de usuario, las cuáles deben ser realizadas por los desarrolladores durante las diversas reuniones de planificación.

Una historia es lo suficientemente pequeña como para que el equipo la desarrolle durante una entrega; de una a tres semanas, más de tres semanas implica que se debe señalar al cliente que debe dividir una historia de usuario y menos de una semana implica que la historia es demasiado sencilla y por lo que se deben unir dos o más de ellas para su mejor interpretación.

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto de estimación. Un punto de estimación, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos de estimación. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

En la Tabla 2.2.3 se muestra un ejemplo de denotación del punto de estimación del tiempo de las historias de usuario, se usó la denotación de punto de estimación por semana y la denotación de puntos flotantes para indicar días.

Tabla 2.2.3 Denotación del punto de estimación

Denotación de punto de estimación	**Interpretación**
1 punto	1 semana

1.5 puntos	1 semana y 5 días

Tabla 2.2.4 Estimación del tiempo de las HU

Historia de Usuario	**Punto de estimación**
RF1_Identificar software instalados.	1
RF2_Visualizar software.	0.5
RF3_Modificar software.	0.5
RF4_Identificar propiedades del CPU, la RAM, el disco y la red.	1
RF5_Visualizar las propiedades del CPU, la RAM, el disco y la red.	0.5
RF6_Modificar las propiedades del CPU, la RAM, el disco y la red.	0.5
RF7_Reportar computadoras fuera de servicios	1
RF8_Eliminar computadora	1
RF9_Realizar monitoreo de software	2
RF10_Realizar monitoreo de hardware	2
RF11_Visualizar datos de los software por computadoras	1
RF12_Exportar a .csv los datos de los software por computadoras	1

Las estimaciones realizadas permitieron confeccionar una evaluación puntual del tiempo de implementación de cada historia de usuario para la posterior elaboración del plan de iteración. Una vez realizadas las estimaciones es preciso construir un plan de iteración donde se pueden agrupar estas historias y dar su cumplimiento de manera paulatina.

2.2.3 Plan de iteraciones

Un plan de iteración está constituido por un conjunto secuencial de actividades y tareas, cada una tiene recursos asignados y pueden depender a su vez de otras tareas. El plan de iteración se realiza una vez por cada iteración.

El contenido de la iteración puede variar, dependiendo de la posición dentro del ciclo de vida y de la naturaleza del proyecto. El plan de iteración incluye porciones relevantes de todas las disciplinas

para cada iteración en particular, la prioridad de implementación se evalúa en base al cliente y el equipo de desarrollo, y el impacto del riesgo en base al juicio de experto. Los planes de iteración por lo general muestran un planeamiento detallado de quién va a realizar una tarea/actividad de acuerdo en conformidad a que criterios de evaluación.

Tabla 2.2.5 Plan de iteraciones

Iteración	Número de Historia	Historia de Usuario	Duración (semanas)	
Iteración No. 1	RF1_Identificar software instalados.	HU-01	1	4
	RF2_Visualizar software.	HU-02	0.5	
	RF3_Modificar software.	HU-03	0.5	
	RF4_Identificar propiedades del CPU, la RAM, el disco y la red.	HU-04	1	
	RF5_Visualizar las propiedades del CPU, la RAM, el disco y la red.	HU-05	0.5	
	RF6_Modificar las propiedades del CPU, la RAM, el disco y la red.	HU-06	0.5	
Iteración No. 2	RF9_Realizar monitoreo de software	HU-09	2	4
	RF10_Realizar monitoreo de hardware	HU-10	2	
Iteración No. 3	RF7_Reportar computadoras fuera de servicios	HU-07	1	4
	RF8_Eliminar computadora	HU-08	1	
	RF11_Visualizar datos de los software por computadoras	HU-11	1	
	RF12_Exportar a .csv los datos de los software por computadoras	HU-12	1	

Una vez realizado el plan de iteración se pudo agrupar las diferentes historias de usuario en 3 iteraciones teniendo en cuenta las características que rigen la metodología XP. Con este plan de iteraciones ya es posible realizar un plan de entrega que será entregado al cliente y que el grupo de desarrollo está obligado a hacer cumplir.

2.2.4 Plan de entrega

Determinada la duración de cada iteración se presenta el plan de entrega elaborado para la fase de implementación teniendo en cuenta que el desarrollo del proyecto inicia el 01/04/2021 y concluirá el 28/06/2021:

Tabla 2.2.6 Plan de entrega

	Iteración No. 1	Iteración No. 2	Iteración No. 3
Cantidad de HU	6	2	4
Fecha de inicio	01/04/2021	30/04/2021	31/05/2021
Fecha de entrega	29/04/2021	28/05/2021	28/06/2021

Una vez realizado el plan de entrega se puede confirmar con el cliente que el proyecto durará 12 semanas, lo que significa que tendrá un tiempo de desarrollo de 3 meses aproximadamente.

2.2.5 Resumen de tareas

La siguiente tabla muestra un resumen de las tareas de ingeniería a desarrollar por cada una de las historias de usuario en las iteraciones.

Tabla 2.2.7 Resumen de tareas de ingeniería por HU

Iteración	Historia de Usuario	Número de tarea	Tarea de Ingeniería
Iteración No. 1	RF1_Identificar software instalados.	HU-01	Identificar los software instalados de una computadora.
	RF2_Visualizar software.	HU-02	Mostrar los software instalados de una computadora.
	RF3_Modificar software.	HU-03	Funcionalidades para modificar los software instalados de una computadora.
	RF4_Identificar propiedades del CPU, la RAM, el disco y la red.	HU-04	Identificar las propiedades del CPU, la

			RAM, el disco y la red de una computadora.
	RF5_Visualizar las propiedades del CPU, la RAM, el disco y la red.	HU-05	Mostrar las propiedades del CPU, la RAM, el disco y la red de una computadora.
	RF6_Modificar las propiedades del CPU, la RAM, el disco y la red.	HU-06	Funcionalidades para modificar las propiedades del CPU, la RAM, el disco y la red de una computadora.
Iteración No. 2	RF9_Realizar monitoreo de software	HU-09	Funcionalidades para el monitoreo de software.
	RF10_Realizar monitoreo de hardware	HU-10	Funcionalidades para el monitoreo de hardware.
Iteración No. 3	RF7_Reportar computadoras fuera de servicios	HU-07	Sistema de notificación de las computadoras que están fuera de servicio.
	RF8_Eliminar computadora	HU-08	Funcionalidades para eliminar una computadora.
	RF11_Visualizar datos de los software por computadoras	HU-11	Visualizar la relación de los datos de los software por computadoras.
	RF12_Exportar a .csv los datos de los software por computadoras	HU-12	Mostrar la relación de los datos de los software por computadoras en un archivo tipo .csv.

Una vez determinadas las tareas ingenieriles a implementar por cada iteración, se posee un espectro más detallado para el desarrollo de la solución planteada y la posibilidad de planificar con mayor detenimiento las reuniones diarias.

2.2.6 Reuniones diarias de seguimientos

El planeamiento es esencial para cualquier tipo de metodología, es por ello que XP requiere de una revisión continua del plan de trabajo. A pesar de ser una metodología que evita la documentación exagerada, es muy estricta en la organización del trabajo.

Para la misma, el grupo de desarrollo definió desde los primeros momentos espacios en los que el equipo y el cliente realizarían encuentros semanales para la evaluación del cumplimiento de los resultados de las diferentes iteraciones, así como el cumplimiento parcial, total o nuevas redefiniciones a los conceptos establecidos. Las pequeñas entregas facilitarán el trabajo de los desarrolladores, haciendo posible suplir las necesidades del cliente.

Tabla 2.2.8 Reuniones de seguimientos

Reunión de / Fecha		**Descripción**
Plan de entregas	24/03/2021	Se realiza entre el equipo de trabajo y los clientes y se define el marco temporal de la realización del sistema. El cliente expone las historias de usuario a los integrantes de grupo, quienes estimarán el grado de dificultad de la implementación de cada historia. A partir de las historias de usuario, el cliente plantea las pruebas de aceptación con las cuales se comprueba que cada una de éstas ha sido correctamente implementada.
Inicial de Iteración	28/03/2021	Esta reunión es realizada previo a iniciar una iteración donde se organizan las actividades de programación a realizar. Las historias de usuario son traducidas a tareas y asignadas a los desarrolladores.
Diarias	Al inicio de cada jornada de trabajo	Estas reuniones se realizan al comenzar la jornada laboral, donde todo el equipo de desarrollo se reúne para exponer los problemas e ideas que se estén presentando. Es de vital importancia evitar las discusiones largas, ya que se está utilizando tiempo laboral que puede ser destinado a la construcción del sistema.
Fin de iteración	29/04/2021 28/05/2021 28/06/2021	Estas reuniones se realizan al finalizar cada iteración en conjunto con los clientes para presentar los avances en

		cada iteración y demostrar la aceptación por los mismos como muestra de una correcta implementación.

Con el plan de reuniones queda determinado la secuencia de las distintas actividades a realizar dentro del proceso de desarrollo de la solución, contribuyendo a la calidad del producto deseado. Una vez determinado los artefactos organizativos, se procede a la fase de diseño de la solución propuesta.

2.3 Fase III: Diseño

Teniendo en cuenta lo que plantea la metodología XP, en esta fase se confeccionan las tarjetas Clase-Responsabilidad-Colaborador (CRC) para la descripción de las principales clases de los módulos desarrollados. Se define la arquitectura del sistema y los estándares de codificación, así como los patrones de diseño utilizados en el desarrollo de la propuesta.

2.3.1 Arquitectura de software

La arquitectura de un software es el diseño de más alto nivel de la estructura de un sistema. Toda arquitectura de software debe definir diversos aspectos del software, y generalmente, cada uno de estos se describe de una manera más comprensible si se utilizan distintos modelos o vistas (Márquez Avendaño & Zulaica Rugarcía). La arquitectura determinada para el desarrollo del sistema es la 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 2.3.1). 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 (GUILLERMO VALLE & GILDARDO GUTIERREZ, 2005).

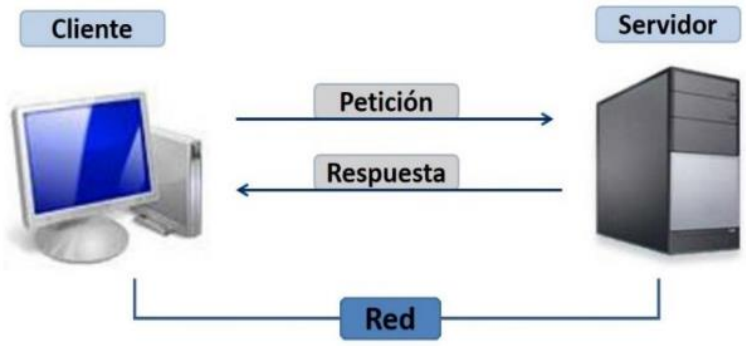


Figura 2.3.1 Arquitectura Cliente – Servidor diseñada para la propuesta de solución.

Fuente: "Definición arquitectura cliente-servidor", realizado por (GUILLERMO VALLE & GILDARDO GUTIERREZ, 2005)

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.

2.3.2 Modelo-Vista-Plantilla

Django fue diseñado para promover el acoplamiento débil y la estricta separación entre las piezas de una aplicación. Debido a esta filosofía, es fácil realizar cambios en un lugar particular de la aplicación sin afectar otras piezas.

Por lo general los sistemas que dividen la lógica de acceso a la base de datos, la lógica de negocios y la lógica de presentación comprenden un concepto llamado el patrón de arquitectura de software Modelo-Vista-Controlador (MVC). En este patrón, el "Modelo" referencia al acceso a la capa de datos, la "Vista" se refiere a la parte del sistema que selecciona qué mostrar y cómo mostrarlo, y el "Controlador" implica la parte del sistema que decide qué vista usar, dependiendo de la entrada del usuario, accediendo al modelo si es necesario.

Django aplica de manera similar este modelo. El Controlador es manejada por el mismo marco de trabajo y la parte más importante se produce en los modelos, las plantillas y las vistas, Django redefine este modelo como MVT: Model-View-Template (Modelo-Vista-Plantilla):

- M: significa "Model" (Modelo), la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.
- T: significa "Template" (Plantilla), la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: ¿cómo algunas cosas son mostradas sobre una página web u otro tipo de documento?

- V: significa "View" (Vista), la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada. Es como un puente entre los modelos y las plantillas.

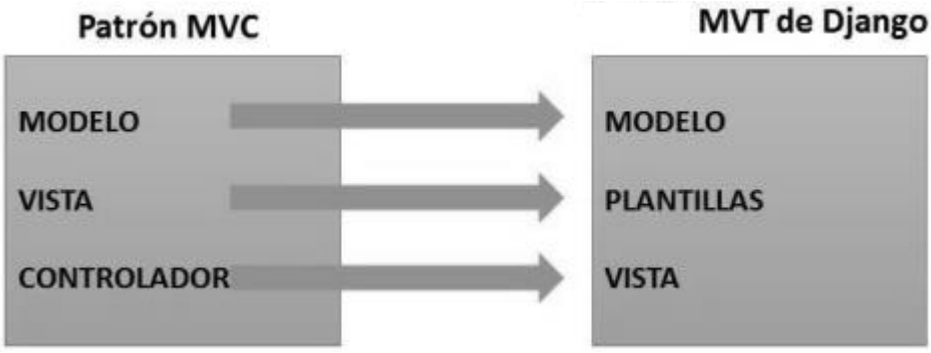


Figura 2.3.2 Relación estructural entre los MVC y MVT.

Fuente: "Curso Django: Entendiendo como trabaja Django", realizado por (Infante Montero, 2012)

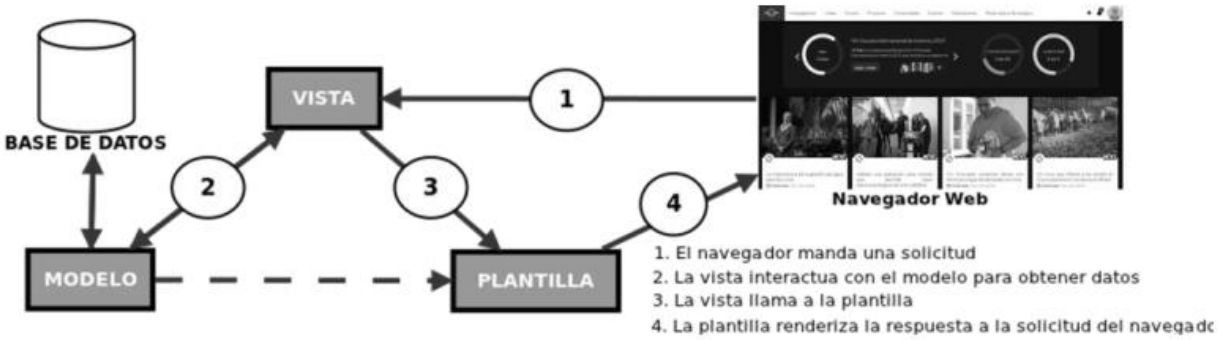


Figura 2.3.3 Ciclo del MVT.

Fuente: "Curso Django: Entendiendo como trabaja Django", realizado por (Infante Montero, 2012)

Cuando se realiza una solicitud desde un navegador, las vistas realizan la lógica del negocio retroalimentándose con el modelo de BD, envía los datos a la plantilla y esta es la que se devuelve como resultado del proceso (Figura 2.3.3)

Tabla 2.3.1 Estructura de la solución dependiendo del MVT.

Model(Modelos)	Views(Vistas)	Templates(Plantillas)
	Home_View	Home_Template
General	General_View	General_Template
CPU	Cpu_View	Cpu_Template
Disco	Disco_View	Disco_Template

Memory	Memory_View	Memory_Template
Network	Network_View	Network_Template
Software	Software_View	Software_Template

2.3.3 Tarjetas CRC

Las tarjetas CRC se elaboran durante la fase de diseño de la metodología XP para describir las entidades existentes en la aplicación. El uso de este tipo de tarjetas es una técnica de modelado que permite identificar las clases, responsabilidades y colaboraciones. El objetivo es obtener un diseño simple, elegante y fácil de comprender por parte de los programadores.

Tabla 2.3.2 Tarjeta CRC: General

Tarjeta CRC	
Clase: General	
Responsabilidad	**Colaboración**
listar_software() listar_hardware() reportar_computadoras() mostrar_monitoreo_software() mostrar_monitoreo_hardware(mac_address) visualizar_datos_computadoras() visualizar_estadisticas_generales()	CPU Disco Network Memory Software

Tabla 2.3.3 Tarjeta CRC: CPU

Tarjeta CRC	
Clase: CPU	
Responsabilidad	**Colaboración**
adicionar_cpu() visualizar_cpu()	General

Tabla 2.3.4 Tarjeta CRC: Disco

Tarjeta CRC

Clase: Disco	
Responsabilidad	**Colaboración**
adicionar_disco() visualizar_disco()	General

Tabla 2.3.5 Tarjeta CRC: Memory

Tarjeta CRC	
Clase: Memory	
Responsabilidad	**Colaboración**
adicionar_memory() visualizar_memory()	General

Tabla 2.3.6 Tarjeta CRC: Network

Tarjeta CRC	
Clase: Network	
Responsabilidad	**Colaboración**
adicionar_network() visualizar_network()	General

Tabla 2.3.7 Tarjeta CRC: Software

Tarjeta CRC	
Clase: Software	
Responsabilidad	**Colaboración**
adicionar_software() visualizar_software()	General

Con la realización de las tarjetas CRC se evidencia la interrelación existente entre cada clase de la solución y sus diferentes funcionalidades. Teniendo en cuenta la interrelación de las entidades se

hace necesario realizar prototipos que evidencien al cliente y al grupo de desarrollo el resultado que se obtendría con la solución.

2.4 Fase IV: Implementación

El desarrollo o la codificación es un proceso que se realiza en forma paralela con el diseño y la cual está sujeta a varias observaciones por parte de XP. Dentro de esta fase se describen los estándares de codificación, así como los patrones de diseño que se emplearon dentro del sistema.

2.4.1 Estándares de codificación

En el proceso de desarrollo de un software siguiendo la metodología XP, es necesario que exista una adecuada comunicación entre los programadores del equipo de desarrollo. Para lograr esto, se establecen un conjunto de estándares definidos por las convenciones de escritura de código Python abarcando la librería estándar en la principal distribución de Python (PEP).

El PEP8 y PEP257 (convenciones para la documentación del código) establecen una serie de reglas a seguir en la escritura y denominación del código de programación en lenguaje Python y por el cual se definirá el sistema. Algunas de estas reglas básicas a tener en cuenta son descritas a continuación:

- Se deben usar 4 espacios para ordenar el código o aplicar tabulaciones para todas las líneas excepto la primera.
- El paréntesis/corchete/llave que cierre una asignación debe estar alineado con el primer carácter que no sea un espacio en blanco.
- Nunca se deben mezclar tabulaciones y espacios.
- Límite máximo de 79 caracteres por líneas. Separa funciones de alto nivel y definiciones de clase con dos líneas en blanco.
- Uso de codificación UTF-8 para versiones de Python posterior a la 3.0.
- Los comentarios deben ser oraciones completas. Si un comentario es una frase u oración, su primera palabra debe comenzar con mayúscula, a menos que sea un identificador que comienza con minúscula.
- Se comentará en idioma español para la comprensión de futuros desarrolladores.
- Definir variables en minúscula y las variables compuestas separadas por _.
- Los módulos o paquetes deben tener un nombre corto y en minúscula. Guiones bajos pueden utilizarse para mejora la legibilidad.
- Casi sin excepción, los nombres de clases deben utilizar la convención “CapWords” (palabras que comienzan con mayúsculas). Las clases para el uso interno tienen un guion bajo como prefijo.

2.4.2 Patrones de diseño

El autor Larman (2004) establece que un patrón es "…un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos." (Agusto Monferrer, 2001). Los patrones de diseño comunican los estilos y soluciones consideradas como buenas prácticas, que los expertos en el diseño orientado a objetos utilizan para la creación de sistemas. En los sub-epígrafes siguientes se describen los patrones utilizados para el desarrollo de la propuesta de solución.

Patrones GRASP

Estos patrones constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objetos, un patrón es una descripción de un problema y la solución a la que se le da un nombre, y que además se puede aplicar a nuevos contextos, los patrones GRASP se dividen en seis ellos son: experto, creador, controlador, fachada, alta cohesión y bajo acoplamiento. (Cam, 2003)

Experto: Este patrón busca la forma de responder a la pregunta ¿A quién se le debe asignar una responsabilidad? La solución a esto es: Designarle la responsabilidad al que tiene la información para realizarla. Esto significa asignarle la responsabilidad a la clase que tiene la información necesaria para llevar a cabo la responsabilidad.

En el sistema un ejemplo de ello se evidencia en las clases modelo de Django vistas en la Tabla 2.3.1, las cuales son las encargadas y las que poseen la información necesaria para comunicarse con la base de datos del sistema.

```python
class Computer(AbcMonitor):

    def __init__(self):
        self.datetime_format = '%H:%M:%S %d/%m/%Y'
        self.__raw_boot_time = psutil.boot_time()
        self.__boot_time = datetime.fromtimestamp(self.raw_boot_time)
        self.__boot_time = self.__boot_time.strftime(self.datetime_format)
        self.__hostname = platform.node()
        self.__os = Computer.__get_os_name()
        self.__architecture = platform.machine()
        self.__python_version = '{} ver. {}'.format(
            platform.python_implementation(), platform.python_version()
        )
        self.__processor = Cpu(monitoring_latency=1)
        self.__nonvolatile_memory = NonvolatileMemory.instances_connected_devices(monitoring_latency=10)
        self.__nonvolatile_memory_devices = set(
            [dev_info.device for dev_info in self.__nonvolatile_memory]
        )
        self.__virtual_memory = VirtualMemory(monitoring_latency=1)
        self.__swap_memory = SwapMemory(monitoring_latency=1)
        self.__network_interface = NetworkInterface(monitoring_latency=3)
        super().__init__(monitoring_latency=3)

    @property
    def processor(self):
        return self.__processor

    @property
    def raw_boot_time(self):
        return self.__raw_boot_time

    @property
    def boot_time(self):
        return self.__boot_time

    @property
    def raw_uptime(self):
        return datetime.now() - datetime.fromtimestamp(self.raw_boot_time)

    @property
    def uptime(self):
        return str(self.raw_uptime).split('.')[0]

    @property
    def os(self):
        return self.__os
```

Figura 2.4.1 Clase Computer() es la experta en obtener la información del hardware.

Creador: Responde a la pregunta: ¿Quién debe ser el responsable de la creación de una nueva instancia? La respuesta a esto: Se le asigna la responsabilidad a quien agrega, contiene, registra o utiliza la instancia.

En el sistema todas las acciones se definen y ejecutan en las clases contenidas en las vistas de los módulos respondiendo al modelo MVT de Django y visualizadas en la Tabla 2.3.1.

Alta cohesión: Busca responder a la pregunta: ¿Cómo mantener la complejidad manejable? La solución a esto es: Asignar responsabilidades de manera que la información que almacena una clase sea coherente y esté relacionada con la clase.

Todas las clases de la solución propuesta deberán cumplir con lo establecido por este patrón. De esta forma las clases de modelo realizan sus funciones de conexión con las bases de datos y las clases de vistas la lógica del negocio.

Bajo Acoplamiento: Busca responder a la pregunta: ¿Cómo asignar las responsabilidades de manera que existan pocas dependencias? La respuesta a esta interrogante es: Manteniendo bajo acoplamiento.

Esto lo podemos evidenciar en el sistema donde cada clase se comunica con un número relativamente pequeño de clases.

Controlador: Brinda solución a la interrogante: ¿Quién debe manejar los eventos del sistema? La solución a esto es: Uso de los controladores del MVT.

Todos los eventos del sistema son gestionados por los controladores del MVT de Django, haciendo posible la identificación de las Uniform Resource Locator (URL) con el respectivo controlador de la vista.

Patrones GOF

Los patrones GOF constituyen una herramienta fundamental para cualquier programador. Se utilizan para solucionar problemas de creación de instancias, ayudando a encapsular y abstraer dicha creación. Según lo descrito por la publicación Patrones del “Gang of Four” (LARMAN, 2005) se han identificado 2 patrones GOF empleados en el sistema:

Observador: es un patrón de diseño de comportamiento que te permite definir un mecanismo de suscripción para notificar a varios objetos sobre cualquier evento que le suceda al objeto que están observando.

```
def handle_client(conn, addr):
    print(f"[NEW CONNECTION] {addr} connected")

    connected = True
    while connected:
        msg_length = conn.recv(HEADER_LENGTH)
        if msg_length:
            msg_length = int(msg_length)
            buffer = io.BytesIO()
            recibidos = 0
            while recibidos < msg_length:
                packet = conn.recv(msg_length - recibidos)
                buffer.write(packet)
                recibidos += len(packet)
            buffer.seek(0)
            msg = pickle.loads(buffer.read())
```

Figura 2.4.2 Servidor observando a los clientes que se conectan.

Singleton: es un patrón de diseño creacional que nos permite asegurarnos de que una clase tenga una única instancia, a la vez que proporciona un punto de acceso global a dicha instancia.

El patrón Singleton resuelve dos problemas al mismo tiempo, vulnerando el Principio de responsabilidad única:

1. Garantizar que una clase tenga una única instancia.
2. Proporcionar un punto de acceso global a dicha instancia.

```
comp_info = Computer()


def send(client, msg):
    message = msg
    msg_length = len(message)
    send_length = str(msg_length).encode(FORMAT)
    send_length += b' '*(HEADER_LENGTH - len(send_length))
    client.send(send_length)
    client.send(message)

def cpu_info():
    return comp_info.processor

def mem_info():
    return comp_info.virtual_memory

def disk_info():
    return comp_info.nonvolatile_memory

def nif():
    return comp_info.network_interface

def _format_bytes(byte_value):
    try:
        if (byte_value is None) or (byte_value == 0):
            byte_value = '0'
        elif isinstance(byte_value, (int, float)):
            val, unit = UnitByte.auto_convert(byte_value)
            byte_value = '{:.2f}'.format(val) + \
                         UnitByte.get_name_reduction(unit)
```

Figura 2.4.3 Clase Computer() con una única instancia.

2.5 Conclusiones del capítulo

- La generación de los artefactos en la etapa de implementación del sistema siguiendo la Metodología XP, facilitaron las tareas de implementación, la reducción de errores y riesgos.
- Se realizó la captura de los principales requerimientos que contribuyó a la confiabilidad de los datos recolectados y a satisfacer las expectativas de los usuarios mediante las características de la aplicación.
- Se realizaron las tarjetas CRC para describir las entidades existentes y tener un mayor dominio de las diferentes clases a desarrollar dentro de la solución.

CAPÍTULO 3

VALIDACIÓN DE LA PROPUESTA DE SOLUCIÓN

Capítulo 3: Validación de la propuesta de solución

En este capítulo se aborda las diferentes pruebas realizadas al software y que son exigidas por la metodología XP para el buen funcionamiento y calidad de la propuesta desarrollada. Se detalla con exactitud los diferentes resultados alojados por cada prueba y la respuesta a dichas no conformidades.

3.1 Fase V: Pruebas

Las pruebas de software forman parte de la última fase que propone XP, con el objetivo de lograr una herramienta que cumpla con los requisitos previamente identificados. En el presente capítulo se describen las etapas de codificación y de pruebas de la propuesta de solución. Las pruebas se logran a través de los casos de prueba, los cuales están dirigidos a probar cada una de las funcionalidades definidas en el sistema.

3.2 Estrategia de pruebas

La Metodología XP propone que las pruebas de software sean realizadas al término de cada iteración, garantizando el funcionamiento deseado y la aceptabilidad por el cliente para realizar una entrega funcional y acorde a las exigencias de un producto con calidad. Las dos pruebas exigidas por la metodología, por su importancia y agilidad en el proceso; son las pruebas unitarias y de aceptación. Se hicieron hasta cinco repeticiones de pruebas unitarias al código al finalizar cada iteración e igualmente se realizaron las pruebas de aceptación con sus respectivos casos de prueba y con el mismo número de repeticiones.

3.2.1 Pruebas unitarias

Las pruebas unitarias son pruebas de caja blanca donde los componentes individuales del software se someten a pruebas. El propósito de estas es asegurar que cada unidad de trabajo funciona individualmente bien, responde como se espera que deba responder, o falle como y cuando se supone que debe fallar. Se supone que las pruebas unitarias deben probar la unidad mínima de trabajo de un programa que es aquella que devuelve un valor o produce un cambio en el estado del programa, generalmente hablaríamos de un solo método o una función.

Django posee un sistema de pruebas automatizadas con el marco de ejecución de pruebas y otras utilidades; puede simular solicitudes, insertar datos de prueba, inspeccionar la salida de su aplicación y en general verificar que el código esté correcto. Para la validación y completamiento de las pruebas fueron programadas doce clases de pruebas que realizan comprobaciones a los modelos y las vistas de la plataforma. Las pruebas de Django emiten poca información sobre la completitud y el estado del código, es por ello que se decidió usar las pruebas unitarias en Django

utilizando los siguientes paquetes de Python que ayudan a realizar dichas pruebas con mayor envergadura:

- django-nose (1.4.7)
- coverage (6.0.1)

Debido a que se realizaron seis iteraciones, por cada una fue sometido el software a pruebas unitarias con un máximo de cuatro repeticiones. Utilizando las herramientas descritas anteriormente, se obtuvo como resultado final el siguiente estado de completitud por cada uno de los módulos de la solución, los resultados arrojados por cada iteración pueden verse en el Anexo 6 de este documento:

Tabla 3.2.1 Estado de completitud del código luego de realizar las pruebas a los distintos módulos de la solución.

Módulos	Argumentos	Ausente	Excluido	Cobertura
apps\monitoring__init__.py	0	0	0	100%
apps\monitoring\web_app.py	0	0	0	95%
apps\monitoring\module_general__init__.py	0	0	0	100%
apps\monitoring\web_server.py	172	15	0	91.28%
apps\monitoring\module_monitor__init__.py	0	0	0	100%
apps\monitoring\module_user__init__.py	0	0	0	100%
apps\monitoring\module_general\view.py	145	23	0	84.14%
apps\monitoring\module_monitor\views.py	0	0	0	100%
apps\monitoring\module_user\views.py	56	39	0	30.36%
apps\monitoring\client.py	74	18	0	75.68%
apps\monitoring\server.py	93	7	0	92.48%
apps\monitoring\module_setting__init__.py	0	0	0	100%
apps\monitoring\module_setting\views.py	9	0	0	100%
apps\monitoring\views.py	21	3	0	85.72%
Total	**570**	**105**	**0**	**89.62%**

Este resultado evidencia un porciento aceptable de cobertura de código. El cual representa el porcentaje de ejecución de código en buen funcionamiento del software cumpliendo satisfactoriamente con las pruebas realizadas.

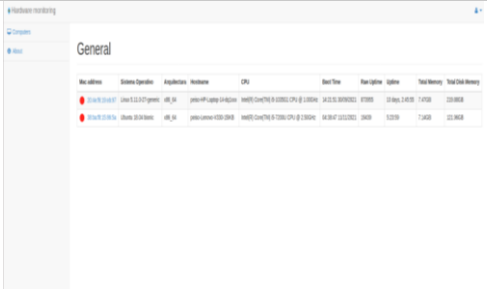
3.2.2 Pruebas de aceptación

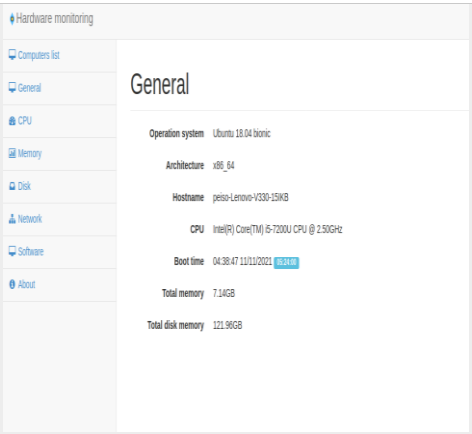
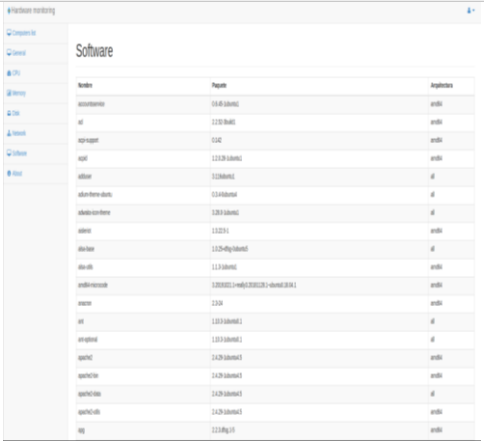
Las pruebas de aceptación son pruebas de caja negra definidas por el cliente para cada historia de usuario, y tienen como objetivo asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. En efecto, las pruebas de aceptación corresponden a una especie de documento de requerimientos en XP, ya que marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo hacia donde tiene que ir y en qué puntos o funcionalidades debe poner el mayor esfuerzo y atención (Varios, 2007).

Definición de los casos de prueba

Estas definiciones tienen como objetivo no tener un conjunto de casos escritos que cubran el 100% del código, sino poder realizar las pruebas del sistema desde el punto de vista del usuario. A continuación, se define el caso de prueba del requisito funcional RF9_Realizar monitoreo de software. Los restantes casos de prueba pueden verse en el Anexo 3.

Tabla 3.2.2 Caso de prueba del requisito funcional RF9_Realizar monitoreo de software

Caso de prueba		
Objetivo del Caso de Prueba	Monitorear los software por cada computadora, para tener un control de los mismos y conocer información de aquellos que se instalan o desinstalan.	
Identificador	HU-9	
Nombre del requisito	RF9_Realizar monitoreo de software	
Precondiciones	Estar autenticado	
Pasos	**Resultados esperados**	**Resultado real**
1. Ir a la página de inicio.	Se debe mostrar la interfaz de inicio con el listado de todas las computadoras.	

2. Elegir una computadora de las que se muestran en el listado.	Se muestra una interfaz con un menú lateral a la izquierda.	
3. Seleccionar la opción software del menú lateral.	Se muestran todos los software instalados (nombre, versión, fecha de instalación, tamaño, editor).	

3.2.3 Pruebas de rendimiento

Las pruebas de rendimiento son realizadas luego de que el sistema está completamente integrado, con el objetivo de observar el comportamiento del mismo cuando se ve sometido a una carga que actúa de manera concurrente. También se encargan de demostrar que el sistema satisface sus requerimientos y monitorizar los comportamientos en cuanto a tiempo de respuesta de la petición.

Las pruebas de rendimiento realizadas a la solución se aplicaron con la herramienta Apache JMeter para realizar mediciones exactas y evaluar el rendimiento. Las pruebas realizadas consistieron en ensayos de 50 a 500 accesos de usuarios respectivamente. Las cantidades de usuarios seleccionadas para la prueba tienen como objetivo dar cumplimiento al requisito no funcional de eficiencia RNF12, que especifica que la solución soportará la conexión simultánea promedio de 50 y un máximo de 500 usuarios conectados concurrentemente.

Se definió una lista de enlaces a los que se simuló el acceso aleatorio y se recolectaron los datos necesarios para su interpretación. La Tabla 3.1.3 se muestra los resultados de las pruebas de rendimiento.

Tabla 3.2.3 Resultados obtenidos de las pruebas de rendimiento.

Funcionalidad	Usuarios	Tiempo de ejecución				% Error	Rendimiento	
		Mín.	Máx.	Media	Mediana		Peticiones /seg	Kb/seg
RF1_Identificar software instalados.	50	0	7700	1456	647	0.0%	38.9	529.0
	500	0	8000	3022	626	0.0%	78.5	787.7
RF2_Visualizar software.	50	0	1456	1130	140	0.0%	57.8	441.6
	500	0	5546	2722	1140	0.0%	59.1	309.5
RF3_Modificar software.	50	0	1417	954	111	0.0%	113.7	1987.0
	500	0	7654	2600	148	0.0%	148.4	2196.7
RF4_Identificar propiedades del CPU, la RAM, el disco y la red.	50	0	6600	1321	345	0.0%	39.9	530.0
	500	0	7000	2356	540	0.0%	79.5	788.7
RF5_Visualizar las propiedades del CPU, la RAM, el disco y la red.	50	0	1325	190	160	0.0%	58.8	442.6
	500	0	6000	1684	874	0.0%	132.1	2150.7
RF6_Modificar las propiedades del CPU, la RAM, el disco y la red.	50	0	1578	452	210	0.0%	114.7	1900.0
	500	0	7980	1230	420	0.0%	149.4	2100.7
RF7_Reportar computadoras fuera de servicios.	50	58	1650	312	130	0.0%	48.9	429.0
	500	18	7650	2456	260	0.0%	88.5	687.7
RF8_Eliminar computadora	50	0	1104	140	147	0.0%	112.2	1940.0
	500	0	6000	1684	874	0.0%	132.1	2150.7
RF9_Realizar monitoreo de software.	50	98	1236	246	150	0.0%	67.8	341.6
	500	28	6321	3214	300	0.0%	99.1	209.5
	50	98	1987	198	160	0.0%	181.2	947.0

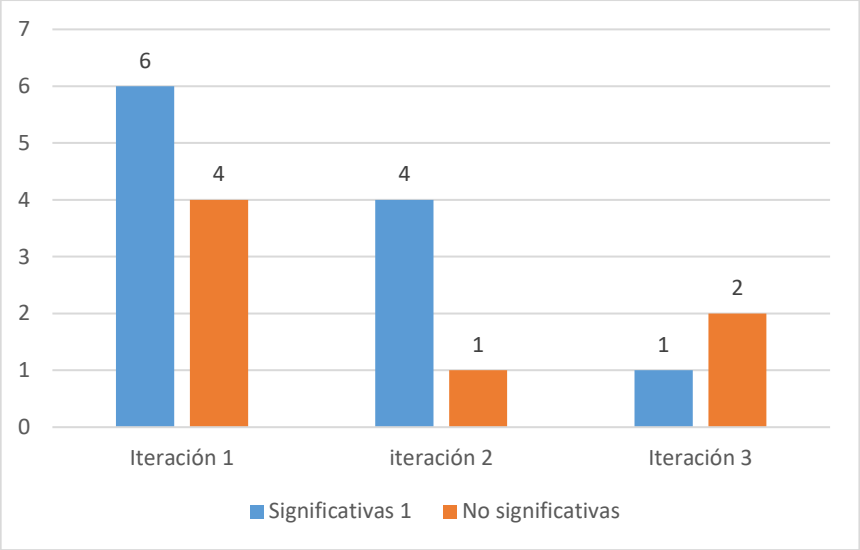
RF10_Realizar monitoreo de hardware.	500	28	8790	2980	320	0.0%	233.1	1156.7
RF11_Visualizar datos de los software por computadoras.	50	0	1020	187	150	0.0%	43.7	987.0
	500	0	3214	2870	1500	0.0%	68.4	1196.7
RF12_Exportar a .csv los datos de los software por computadoras.	50	58	2087	210	140	0.0%	161.2	917.0
	500	18	4512	1210	690	0.0%	203.1	1056.7

- Usuarios: indica la cantidad de usuarios haciendo peticiones de manera concurrente.
- Mínimo: indica el mínimo de tiempo de ejecución invertido para una petición.
- Máximo: indica el máximo de tiempo de ejecución invertido para una petición.
- Media: representa el tiempo de ejecución promedio de una petición.
- Mediana: significa que el 50% de las peticiones realizadas tardaron menos del valor reflejado.
- % Error: indica la relación entre el total de peticiones y el número de peticiones que originaron errores.
- Rendimiento (Peticiones/seg.): hace referencia al número de peticiones que el servidor puede procesar en un segundo.
- Rendimiento (Kb/seg.): hace referencia a la cantidad de datos que el servidor puede procesar en un segundo.

3.3 Análisis de los resultados de las pruebas

Al término de las tres iteraciones propuestas en el plan de entrega, se realizaron hasta cinco repeticiones de comprobación con los casos de pruebas realizados con el objetivo de hacer la entrega planificada con la calidad correspondiente. A continuación, se muestra un gráfico final con los resultados arrojados en las tres iteraciones de pruebas donde se obtuvo un total de 11 no conformidades (NC) significativas: 5 con prioridad normal, 4 con prioridad baja y 2 con prioridad alta; 7 NC no significativas: 3 de prioridad normal y de 4 de prioridad baja. Los resultados obtenidos de las pruebas de aceptación por cada iteración pueden verse en el Anexo 3 de este documento.

Gráfico 3.3.1 Resumen del resultado de las pruebas de aceptación.



Las principales NC no significativas encontradas fueron errores ortográficos, omisiones de tildes y cambio de mayúsculas por minúsculas. También se mostraban algunos mensajes innecesarios en pantalla y elementos de interfaz no sugerentes para un usuario con poca experiencia. Las principales NC significativas encontradas fueron errores de validación y errores en mostrar correctamente la información. Después de concluida cada iteración se resolvieron las NC arrojadas quedando en un estado cerrado.

Resultados de las pruebas de rendimiento aplicadas al sistema

En la Tabla 3.1.3 se puede percibir que el tiempo de respuesta del componente no excede a los 8 segundos en la mayoría de los casos, de esta forma se valida el cumplimento del requisito no funcional de eficiencia RNF12, que especifica que los tiempos de ejecución no excederán los 8 segundos de manera general. Para todas las muestras de usuarios la ocurrencia de errores se mantiene en 0.0%, lo que evidencia que las peticiones hechas se ejecutan satisfactoriamente.

3.4 Conclusiones del capítulo

- Los diferentes métodos de pruebas aplicados a la solución durante el ciclo de vida del software permitieron comprobar los errores existentes y mejorar la calidad de los resultados.
- Se realizaron prueba de aceptación que pudo evidenciar el correcto funcionamiento y la aceptabilidad de los requisitos descritos de conjunto con el cliente en cada iteración en el proceso de desarrollo de la metodología XP.
- Los resultados de estas pruebas al sistema demostraron su efectividad para ser desplegada.

Conclusiones

Considerando los resultados descritos en este informe, la necesidad y el objetivo planteado por la investigación se arriban a las siguientes conclusiones:

- El análisis de las principales fuentes bibliográficas permitió identificar los principales conceptos asociados al monitoreo de software activos en un ordenador propiciando un mayor entendimiento de los mismos, así como también el análisis de las soluciones tecnológicas existentes permitieron identificar las características que sirvieron de guía en la solución.
- El estudio de las diferentes tecnologías y soluciones homólogas permitieron identificar aquellas que por su versatilidad y agilidad en el tratamiento de los datos se adecuan mejor a soluciones para gestionar los recursos de software y hardware. Por su relevancia se destaca el uso de Python como lenguaje de servidor por su amplio uso en ciencia de datos.
- La implementación de los scripts para monitorear y obtener información de los principales procesos de los software y hardware, fue de utilidad para el control y seguimiento del uso de software de las computadoras en la red.
- Los resultados de las distintas pruebas de software aplicadas a la solución demostraron su efectividad y permitieron comprobar que el sistema cumple con las funcionalidades esperadas.

Recomendaciones

Para el desarrollo de futuras investigaciones relacionadas con la presente, se recomienda:

- Incluir nuevos reportes de estadísticas del sistema.

Referencias bibliográficas

Agusto Monferrer, R. (2001). Especificación de Requisitos Software según el estándar de IEEE 830. *Departament d'Informàtica - Universitat Jaume I*.

ALBA Software. (s.f.). Recuperado el 16 de Junio de 2021, de http://www.albasoft.com/web/#/home

ALBALADEJO, X. (2009). Estimación y planificación ágil. [en línea]. Recuperado el 1 de Junio de 2021, de http://www.proyectosagiles.org/estimacion-planificacion-agil-quinto-encuentro-agil-barcelona.

Alegsa, L. (14 de Junio de 2021). *Diccionario de Informática y Tecnología. Diccionario de Informática y Tecnología.* Obtenido de http://www.alegsa.com.ar/Dic/software.php.

ÁLVAREZ, M. Á. (Noviembre de 2003). *Qué es Python. DesarrolloWeb.com*. Recuperado el 16 de Septiembre de 2021, de http://www.desarrolloweb.com/articulos/1325.php

Apache JMeter. (s.f.). Recuperado el 05 de Julio de 2021, de http://jmeter.apache.org/

Apache License. (s.f.). Recuperado el 16 de Junio de 2021, de http://www.apache.org/licenses/LICENSE-2.0

ARAUZO AZOFRA, A., & REVILLA, E. (2004). Una pequeña introducción a Python.

Arquitectura cliente servidor. (2010). Obtenido de http://es.slideshare.net/NoeGonzalezMendoza/arquitectura-cliente-servidor.

ASENSIO, & Menéndez-Barzanallana, R. (13 de Marzo de 2011). *Artículos y enlaces de interés en informática*. Recuperado el 16 de Junio de 2021, de http://www.um.es/docencia/barzana/DIVULGACION/INFORMATICA/Rafael

Bootstrap. (s.f.). Recuperado el 16 de Junio de 2021, de http://getbootstrap.com/2.3.2/

Bootstrap. (9 de Noviembre de 2013). Recuperado el 16 de junio de 2021, de http://web.archive.org/web/20131109025010/http://www.oneskyapp.com/docs/bootstrap/es

Bustelo Ruesta, C., & Amarilla Iglesias, R. (14 de Junio de 2021). *InterContact: Gestión del Conocimiento.* Obtenido de comunidad/archivos/Gestion_del_Conocimiento-BusteloRuesta-AmarillaIglesias.pdf.

Cam, C. G. (2003). Arquitectura de la Información: diseño e implementación. Pontificia Universidad Católica del Perú.

Canós, J. H., Letelier, P., & Penadés, M. d. (2005). Metodologías Ágiles en el Desarrollo de Software. (V. U. Valencia, Ed.)

Caravaca-Mora, O. M. (2010). Diseño de un Entorno de Desarrollo Integrado para una Unidad Controladora de Procesos.

Carlos, J., & Zapata, M. (2010). Un modelo de gíalogo para la generación automática de especificaciones en un -LENCEP. . *Medellin, Colombia: Escuela de Sistemas*.

CHAUVIN, S. (2013). Notación Parael Modelado de Procesos de Negocio. Mujeres de Empresa. Obtenido de http://www.mujeresdeempresa.com/notacion-para-el-modelado-de-procesos-de-negocio/.

CodeIgniter Web Framework. (s.f.). Recuperado el 15 de Junio de 2021, de http://www.codeigniter.com/

DATOS, D. D. (2012). INFORMÁTICA APLICADA. Recuperado el 14 de Junio de 2021, de https://irfeyal.wordpress.com/bases-de-datos/modelamiento-de-bdd/.

Django. (8 de Octubre de 2021). *Introducción a Django*. Obtenido de https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction

Española, R. A. (14 de Junio de 2021). *Real Academia Española. Diccionario de la Lengua Española*. Obtenido de http://lema.rae.es/drae/?val=hardware.

FINLAND. (s.f.). Recuperado el 15 de Junio de 2021, de Invest in Finland News (archive): http://www.investinfinland.fi/articles/news/2News

Framework. (s.f.). Recuperado el 15 de Junio de 2021, de English-Spanish Dictionary - WordReference.com: http://www.wordreference.com/es/translation.asp?tranword=frameworkframework

Gaikar, V. (2013). *5 Free Apps for Time Tracking.*

GUILLERMO VALLE, J., & GILDARDO GUTIERREZ, J. (2005). Definición arquitectura cliente servidor. *Monografias.com*. Recuperado el 19 de Junio de 2021, de http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml

Gumby. (s.f.). Recuperado el 16 de Junio de 2021, de A Flexible, Responsive CSS Framework: http://www.gumbyframework.comCreate

Infante Montero, S. (30 de abril de 2012). *Curso Django: Entendiendo como trabaja Django*. Recuperado el 19 de noviembre de 2021, de http://www.maestrosdelweb.com/curso-django-entendiendo-como-trabaja-django/

jQuery. (s.f.). Recuperado el 15 de Junio de 2021, de Ejemplo de ABM usando Ajax -PHP -MySQL.: http://ideaschile.cl/abm/

JQUERY.ORG. (s.f.). Recuperado el 15 de Junio de 2021, de jquery Foundation: http://jquery.com/jQuery

KOSKELA, L. (2008). TEST DRIVEN, Practical TDD and Acceptance TDD for Java Developers. *Manning Publication Co.*

Kube CSS Framework. (s.f.). Recuperado el 16 de Junio de 2021, de https://imperavi.com/kube/

Labroots. (2017). Media Kit 2017.

Laravel. (s.f.). Recuperado el 15 de Junio de 2021, de The PHP Framework For Web Artisans: https://laravel.com/

Lardinois, F. (2015). *Microsoft Launches Visual Studio Code, A Free Cross-Platform Code Editor For OS X, Linux And Windows*. Recuperado el 20 de Junio de 2021

Larman, C. (2004). Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development. *New Jersey: Prentice Hall PTR Upper Saddle River*.

LARMAN, C. (2005). UML y Patrones. (C. P. Hall, Ed.) 520.

Malea, E., Orduna, M., Martín y López Cózar, A., & Delgado, E. (2016). ResearchGate como fuente de evaluación científica: desvelando sus aplicaciones bibliométricas. . *España: El Profesional de la Informacion*.

Márquez Avendaño, B. M., & Zulaica Rugarcía, J. M. (s.f.). Implementación de un reconocedor de voz gratuito a el sistema de ayuda a invidentes Dos-Vox en español. *Colección de Tesis Digitales Universidad de las Américas Puebla*. Recuperado el 19 de Junio de 2021, de http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/

MARTÍNEZ GUAITA, A. (Abril de 2012). *Nuevo Django 1.4.* Recuperado el 15 de Septiembre de 2021, de http://www.desarrolloweb.com/actualidad/nuevo-django-1-4-6755.html

MySQL. (s.f.). Recuperado el 16 de Junio de 2021, de http://www.mysql.com/

PHP Framework Laravel ESPAÑOL. (s.f.). Recuperado el 15 de Junio de 2021, de https://plus.google.com/communities/111797011764886461382

PHP: Hypertext Preprocessor. (s.f.). Recuperado el 15 de Junio de 2021, de http://php.net/

Potencier, F., & Zaninotto, F. (2008). Symfony, la guía definitiva. *Librosweb*.

Pressman, R. S. (2011). Ingeniería de Software -Un enfoque práctico. (MEXICO, Ed.) *7ma*(s.n.).

QUEVEDO, V. D. (2016). Solución para sistemas de control logístico. *15 Convención Científica de Ingeniería y Arquitectura. Instituto Superior Politécnico José Antonio Hecheverría*.

Reitz, K. (2016). *Requests Documentation.* EEUU: Requests Documentation. .

Releases. (s.f.). Recuperado el 15 de Junio de 2021, de vaadin/vaadin · GitHub: https://github.com/vaadin/vaadin/releases

Reynoso, C. B. (2004). Introducción a la arquitectura de software Versión 1.0.

Rossum, G. (2009). El tutorial de Python.

Selecting a Developement Approach. (2017). *Center for Medicare & Medicaid Services*.

Shepherd, E. F., Toloza, E., McClung, C. D., & Schmalzried, T. P. (1999). Step activity monitor: increased accuracy in quantifying ambulatory activity. *Journal of Orthopaedic Research, 17*(5), 703-708.

Symfony. (s.f.). Recuperado el 15 de Junio de 2021, de Drupal 8 integra los primeros componentes de Symfony2: http://symfony.es/noticias/2011/10/26/drupal-8-integra-los-primeros-componentes-de-symfony2/

Symfony.es. (s.f.). Recuperado el 15 de Junio de 2021, de El mejor framework PHP para crear aplicaciones web: http://symfony.es/

Tamayo Betancourt, Y. E., & Ciervide Cabalé, R. E. (2015). *Cliente del Gestor de Recursos de Hardware y Software para dispositivos que operan con Android.* Bachelor's thesis, Universidad de las Ciencias Informáticas. Facultad 2.

The technological benefits of Symfony in 6 easy lessons. (s.f.). Recuperado el 15 de Junio de 2021, de http://symfony.com/six-good-technical-reasons

Tipantuña Heredia, E. L. (2017). *istema de gestión integral de inventario informático, aplicando la herramienta GLPI con ocs-inventory en el departamento de tecnologías de la información y comunicación de la Universidad Técnica de Cotopaxi.* Bachelor's thesis, LATACUNGA/UTC/2017.

Varios. (2007). Patrones GRASP. *Universidad del Valle*.

Varios. (2017). Introducción a las Metodologías de Desarrollo de Software. (España, Ed.) (s.n.).

World Wide Web Consortium (W3C). (s.f.). Recuperado el 16 de Junio de 2021, de http://www.w3c.es/

YAML CSS Framework. (s.f.). Recuperado el 16 de Junio de 2021, de http://www.yaml.de/

Yoanni, O. L., Ernesto, A. V., Julio, H. P., & Odaysa, R. H. (s.f.). GRHS: Gestor de Recursos de Hardware y Software.

Anexos

Anexo 1 Modelo Entidad Relación

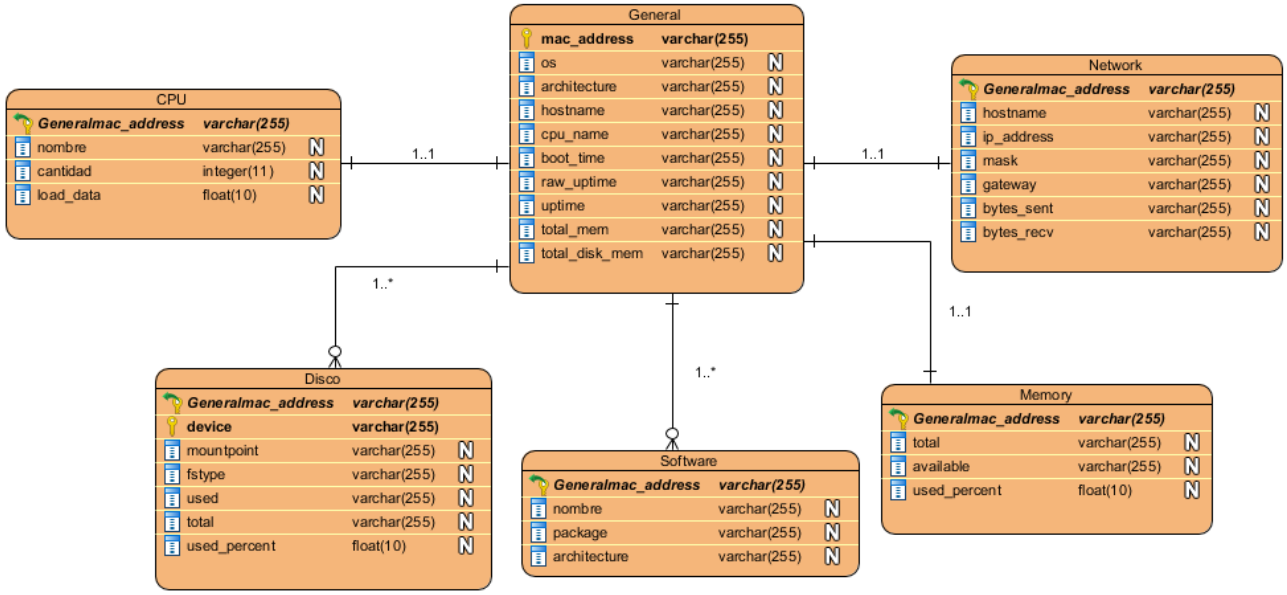


Figura 3.4.1 Modelo Entidad Relación

Anexo 2 Historias de Usuarios

Historia de Usuario			
Código	HU-01	**Nombre**	RF1_Identificar software instalados.
Usuario	Sistema	**Tipo de actividad**	Nueva
Riesgo	Alto	**Prioridad**	Alta
Iteración	1	**Puntos estimados**	1
Descripción	Permitirá identificar los software instalados de una computadora.		

Tabla 3.4.1 Historia de Usuario HU-01.

Historia de Usuario			
Código	HU-02	**Nombre**	RF2_Visualizar software.
Usuario	Sistema	**Tipo de actividad**	Nueva
Riesgo	Alto	**Prioridad**	Media
Iteración	1	**Puntos estimados**	0.5
Descripción	Permitirá mostrar los software instalados de una computadora.		

Tabla 3.4.2 Historia de Usuario HU-02.

Historia de Usuario			
Código	HU-03	**Nombre**	RF3_Modificar software.
Usuario	Sistema	**Tipo de actividad**	Nueva
Riesgo	Alto	**Prioridad**	Media
Iteración	1	**Puntos estimados**	0.5
Descripción	Permitirá modificar los software instalados de una computadora.		

Tabla 3.4.3 Historia de Usuario HU-03.

Historia de Usuario

Código	HU-04	**Nombre**	RF4_Identificar propiedades del CPU, la RAM, el disco y la red.
Usuario	Sistema	**Tipo de actividad**	Nueva
Riesgo	Alto	**Prioridad**	Alta
Iteración	1	**Puntos estimados**	1
Descripción	Permitirá identificar las propiedades del CPU, la RAM, el disco y la red de una computadora.		

Tabla 3.4.4 Historia de Usuario HU-04.

Historia de Usuario

Código	HU-05	**Nombre**	RF5_Visualizar las propiedades del CPU, la RAM, el disco y la red.
Usuario	Sistema	**Tipo de actividad**	Nueva
Riesgo	Alto	**Prioridad**	Media
Iteración	1	**Puntos estimados**	0.5
Descripción	Permitirá mostrar las propiedades del CPU, la RAM, el disco y la red de una computadora.		

Tabla 3.4.5 Historia de Usuario HU-05.

Historia de Usuario

Código	HU-06	**Nombre**	RF6_Modificar las propiedades del CPU, la RAM, el disco y la red.
Usuario	Sistema	**Tipo de actividad**	Nueva
Riesgo	Alto	**Prioridad**	Media
Iteración	1	**Puntos estimados**	0.5
Descripción	Permitirá modificar las propiedades del CPU, la RAM, el disco y la red de una computadora.		

Tabla 3.4.6 Historia de Usuario HU-06.

Historia de Usuario

Código	HU-07	**Nombre**	RF7_Reportar computadoras fuera de servicios
Usuario	Sistema	**Tipo de actividad**	Nueva
Riesgo	Alto	**Prioridad**	Alta
Iteración	3	**Puntos estimados**	1
Descripción	Permitirá reportar las computadoras que estén fuera de servicio, es decir las computadoras que por un motivo u otro llevan días sin encender.		

Tabla 3.4.7 Historia de Usuario HU-07.

Historia de Usuario

Código	HU-08	**Nombre**	RF8_Eliminar computadora.
Usuario	Sistema	**Tipo de actividad**	Nueva
Riesgo	Alto	**Prioridad**	Alta
Iteración	3	**Puntos estimados**	1
Descripción	Permitirá eliminar las computadoras que estén fuera de servicio.		

Tabla 3.4.8 Historia de Usuario HU-08.

Historia de Usuario

Código	HU-011	**Nombre**	RF11_Visualizar datos de los software por computadoras
Usuario		**Tipo de actividad**	Nueva
Riesgo	Medio	**Prioridad**	Media
Iteración	3	**Puntos estimados**	1
Descripción	Permitirá visualizar los datos de los software por cada computadora, referente al sistema operativo en el que corre la aplicación, programas instalados y sus versiones, antivirus, usuarios conectados, dominio de la computadora, información del bios.		

Tabla 3.4.9 Historia de Usuario HU-11.

Historia de Usuario			
Código	HU-12	**Nombre**	RF12_Exportar a .csv los datos de los software por computadoras.
Usuario		**Tipo de actividad**	Nueva
Riesgo	Bajo	**Prioridad**	Baja
Iteración	3	**Puntos estimados**	1
Descripción	Permitirá exportar a .csv los datos de los software por computadoras.		

Tabla 3.4.10 Historia de Usuario HU-12.

Anexo 3 Casos de pruebas

Caso de prueba		
Objetivo del Caso de Prueba	Permitirá identificar los software instalados de una computadora.	
Identificador	HU-1	
Nombre del requisito	RF1_ Identificar software instalados.	
Precondiciones	Estar autenticado	
Pasos	**Resultados esperados**	**Resultado real**
1. Se ejecuta la aplicación cliente una vez encendida la computadora y se mantiene corriendo en segundo plano.	Se debe crear en la base de dato una tupa con los datos de los software de cada computadora.	

Tabla 3.4.11 Caso de prueba del requisito funcional RF1_Identificar software instalados.

Caso de prueba		
Objetivo del Caso de Prueba	Permitirá mostrar los software instalados de una computadora.	
Identificador	HU-2	
Nombre del requisito	RF2_Visualizar software.	
Precondiciones	Estar autenticado	
Pasos	**Resultados esperados**	**Resultado real**

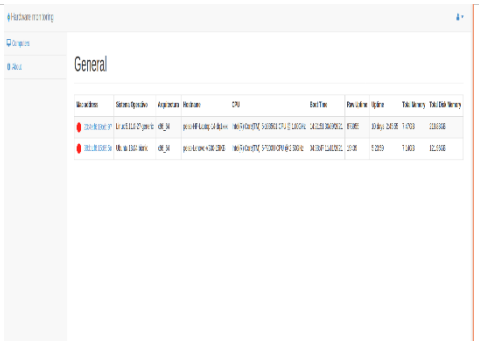
1. Ir a la página de inicio.	Se debe mostrar la interfaz de inicio con el listado de todas las computadoras.	
2. Elegir una computadora de las que se muestran en el listado.	Se muestra una interfaz con un menú lateral a la izquierda.	
3. Seleccionar la opción software del menú lateral.	Se muestran todos los software instalados (nombre, versión, fecha de instalación, tamaño, editor).	

Tabla 3.4.12 Caso de prueba del requisito funcional RF2_Visualizar software.

Caso de prueba

Objetivo del Caso de Prueba	Permitirá modificar los software instalados de una computadora.
Identificador	HU-3

Nombre del requisito	RF3_Modificar software.	
Precondiciones	Estar autenticado	
Pasos	**Resultados esperados**	**Resultado real**
1. Se ejecuta la aplicación cliente en segundo plano, cada 10 minutos.	Se debe actualizar los datos en la base de dato de los software de cada computadora.	

Tabla 3.4.13 Caso de prueba del requisito funcional RF3_Modificar software.

Caso de prueba

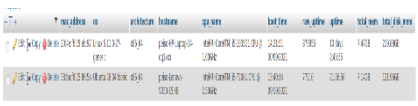
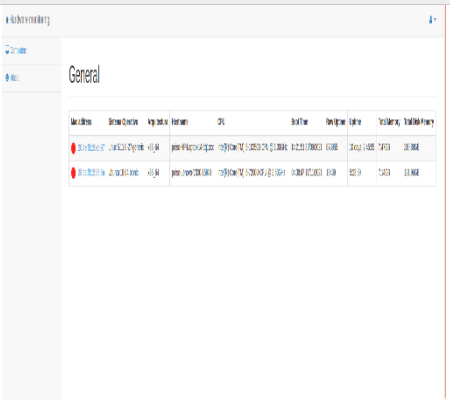
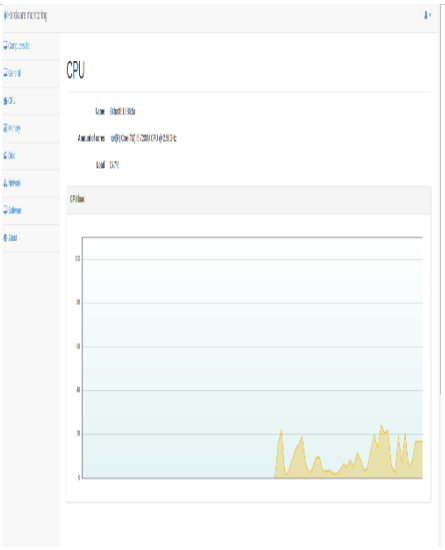
Objetivo del Caso de Prueba	Permitirá identificar las propiedades del CPU, la RAM, el disco y la red de una computadora.	
Identificador	HU-4	
Nombre del requisito	RF4_Identificar propiedades del CPU, la RAM, el disco y la red.	
Precondiciones	Estar autenticado	
Pasos	**Resultados esperados**	**Resultado real**
1. Se ejecuta la aplicación cliente una vez encendida la computadora y se mantiene corriendo en segundo plano.	Se debe crear en la base de dato una tupa con los datos de los hardware de cada computadora.	

Tabla 3.4.14 Caso de prueba del requisito funcional RF4_Identificar propiedades del CPU, la RAM, el disco y la red.

Caso de prueba

Objetivo del Caso de Prueba	Permitirá mostrar las propiedades del CPU, la RAM, el disco y la red de una computadora.	
Identificador	HU-5	
Nombre del requisito	RF5_Visualizar las propiedades del CPU, la RAM, el disco y la red.	
Precondiciones	Estar autenticado	
	Pasos	**Resultados esperados**	**Resultado real**
1.	Ir a la página de inicio.	Se debe mostrar la interfaz de inicio con el listado de todas las computadoras.	
2.	Elegir una computadora de las que se muestran en el listado.	Se muestra una interfaz con un menú lateral a la izquierda.	

3. Seleccionar la opción (CPU, memory, disco o network) del menú lateral.	Se muestran todos los datos de la computadora seleccionada en dependencia a la opción seleccionada (CPU, memory, disco, network).	  

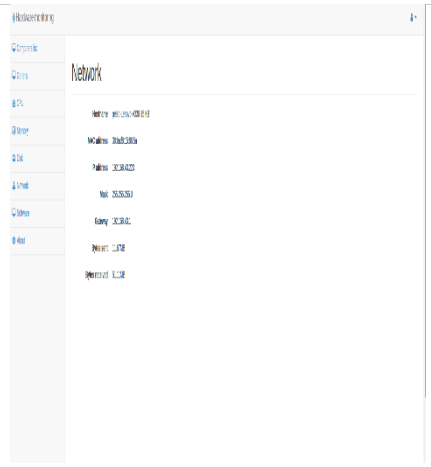
		

Tabla 3.4.15 Caso de prueba del requisito funcional RF5_Visualizar las propiedades del CPU, la RAM, el disco y la red.

Caso de prueba

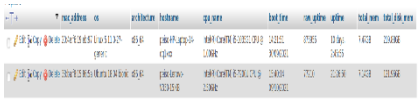
Objetivo del Caso de Prueba	Permitirá modificar las propiedades del CPU, la RAM, el disco y la red de una computadora.	
Identificador	HU-6	
Nombre del requisito	RF6_Modificar las propiedades del CPU, la RAM, el disco y la red.	
Precondiciones	Estar autenticado	
Pasos	**Resultados esperados**	**Resultado real**
1. Se ejecuta la aplicación cliente en segundo plano, cada 10 minutos.	Se debe actualizar los datos en la base de dato de los hardware de cada computadora.	

Tabla 3.4.16 Caso de prueba del requisito funcional RF6_Modificar las propiedades del CPU, la RAM, el disco y la red.

Caso de prueba

Objetivo del Caso de Prueba	Permitirá reportar las computadoras que estén fuera de servicio, es decir las computadoras que por un motivo u otro llevan días sin encender.
Identificador	HU-7
Nombre del requisito	RF7_Reportar computadoras fuera de servicios.

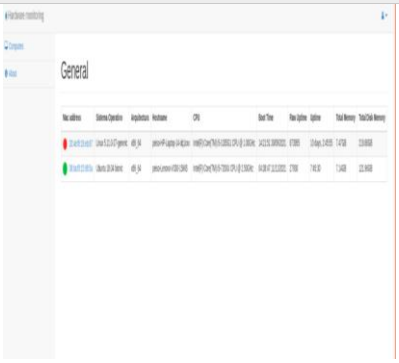
Precondiciones	Estar autenticado	
Pasos	**Resultados esperados**	**Resultado real**
1. Ir a la página de inicio.	Se debe mostrar la interfaz de inicio con el listado de todas las computadoras, las computadoras que están fuera de servicio se identifican a través del círculo rojo mientras las que están encendidas a través del circulo verde.	

Tabla 3.4.17 Caso de prueba del requisito funcional RF7_Reportar computadoras fuera de servicios.

Caso de prueba

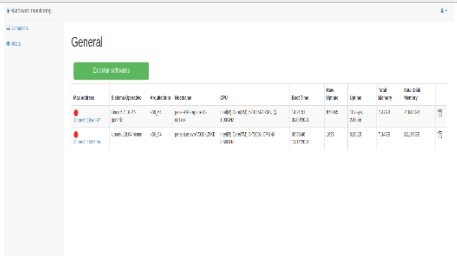
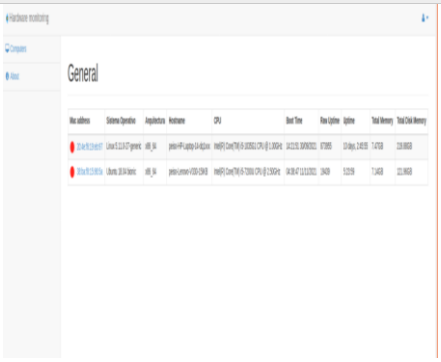
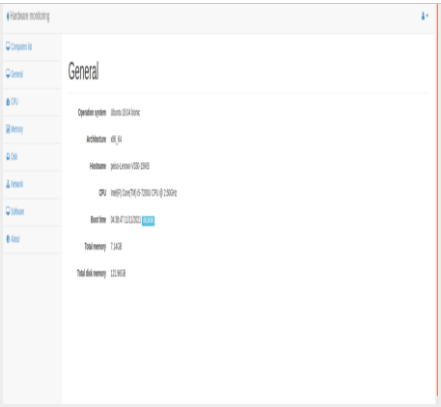
Objetivo del Caso de Prueba	Permitirá eliminar las computadoras que estén fuera de servicio.	
Identificador	HU-8	
Nombre del requisito	RF8_Eliminar computadora	
Precondiciones	Estar autenticado	
Pasos	**Resultados esperados**	**Resultado real**
1. Ir a la página de inicio.	Se debe mostrar la interfaz de inicio con el listado de todas las computadoras.	
2. Elegir la computadora que se desea eliminar y hacer clic en icono “Eliminar”.	Se dirige a la interfaz de inicio.	

Tabla 3.4.18 Caso de prueba del requisito funcional RF8_Eliminar computadora

Caso de prueba

Objetivo del Caso de Prueba	Monitorear los hardware por cada computadora, para tener un control y conocer información de los mismos.	
Identificador	HU-10	
Nombre del requisito	RF10_Realizar monitoreo de hardware	
Precondiciones	Estar autenticado	
Pasos	**Resultados esperados**	**Resultado real**
1. Ir a la página de inicio.	Se debe mostrar la interfaz de inicio con el listado de todas las computadoras.	
2. Elegir una computadora de las que se muestran en el listado.	Se muestra una interfaz con un menú lateral a la izquierda.	

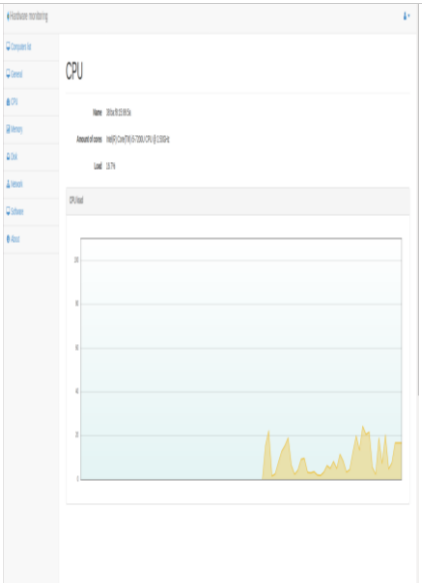
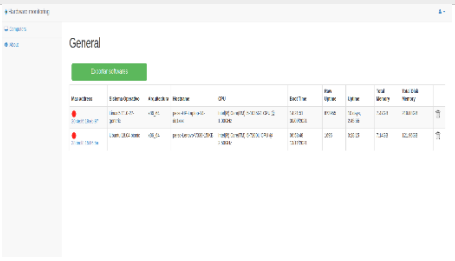
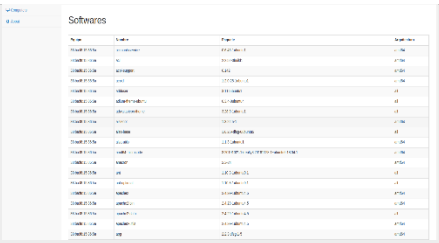
3. Seleccionar la opción (CPU, memory, disco o network) del menú lateral.	Se muestran todos los datos de la computadora seleccionada en dependencia a la opción seleccionada (CPU, memory, disco, network).	  

Tabla 3.4.19 Caso de prueba del requisito funcional RF10_Realizar monitoreo de hardware

Caso de prueba

Objetivo del Caso de Prueba	Permitirá visualizar los datos de los software por cada computadora, referente al sistema operativo en el que corre la aplicación, programas instalados y sus versiones, antivirus, usuarios conectados, dominio de la computadora, información del bios.	
Identificador	HU-11	
Nombre del requisito	RF11_Visualizar datos de los software por computadoras.	
Precondiciones	Estar autenticado	
Pasos	**Resultados esperados**	**Resultado real**
1. Ir a la página de inicio.	Se debe mostrar la interfaz de inicio con el listado de todas las computadoras.	
2. Seleccionar el apartado denominado “Exportar Software”	Se muestra una interfaz con el listado de los software por computadoras.	

ubicado en la parte superior derecha.		

Tabla 3.4.20 Caso de prueba del requisito funcional RF11_Visualizar datos de los software por computadoras

Caso de prueba

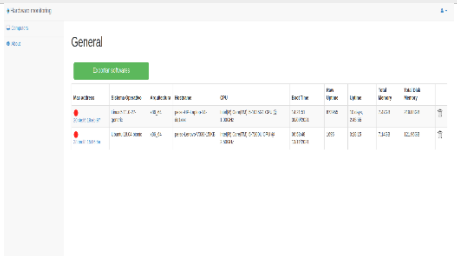
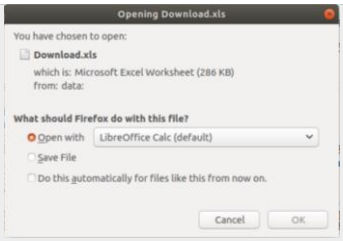
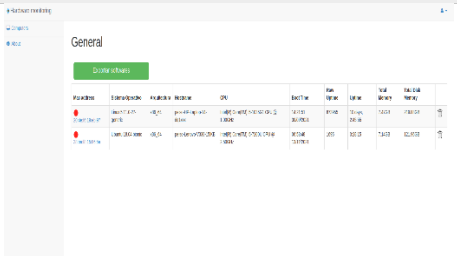
Objetivo del Caso de Prueba	Permitirá exportar a .csv los datos de los software por computadoras.	
Identificador	HU-12	
Nombre del requisito	RF12_Exportar a .csv los datos de los software por computadoras.	
Precondiciones	Estar autenticado	
Pasos	**Resultados esperados**	**Resultado real**
1. Ir a la página de inicio.	Se debe mostrar la interfaz de inicio con el listado de todas las computadoras.	
2. Seleccionar el apartado denominado “Exportar Software” ubicado en la parte superior derecha.	Se muestra un cartel de confirmación.	
3. Se presiona “Aceptar” o “Cancelar”.	Se dirige a la interfaz de inicio.	

Tabla 3.4.21 Caso de prueba del requisito funcional RF12_Exportar a .csv los datos de los software por computadoras.

Anexo 4 Resultados de las no conformidades

Gráfico 3.4.1 No conformidades de la Iteración 1.

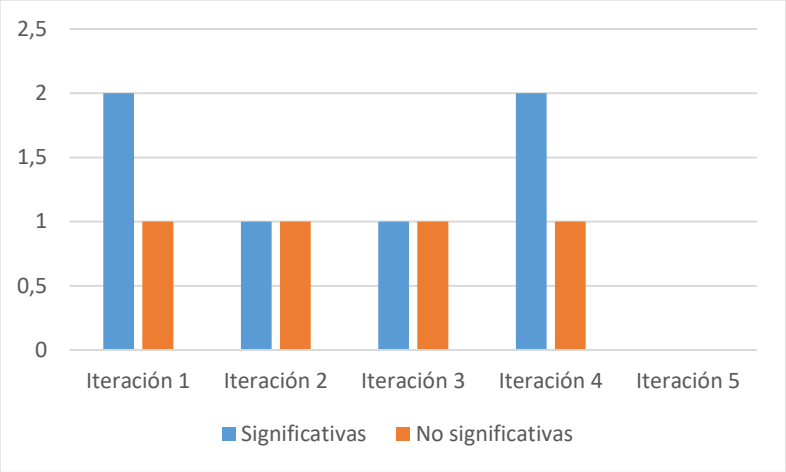


Gráfico 3.4.2 No conformidades de la Iteración 2.

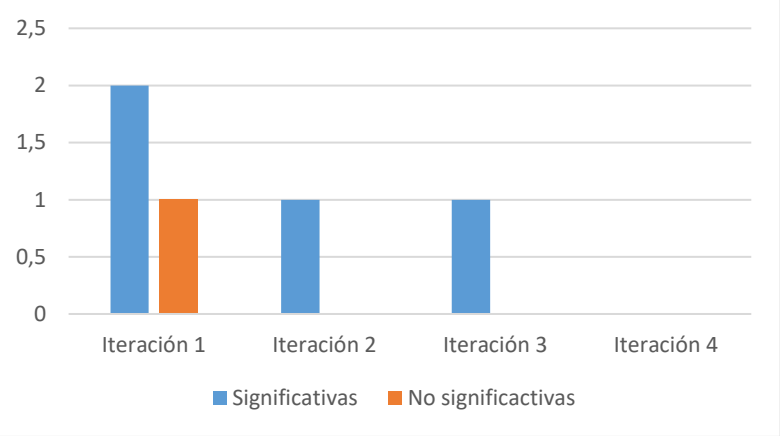
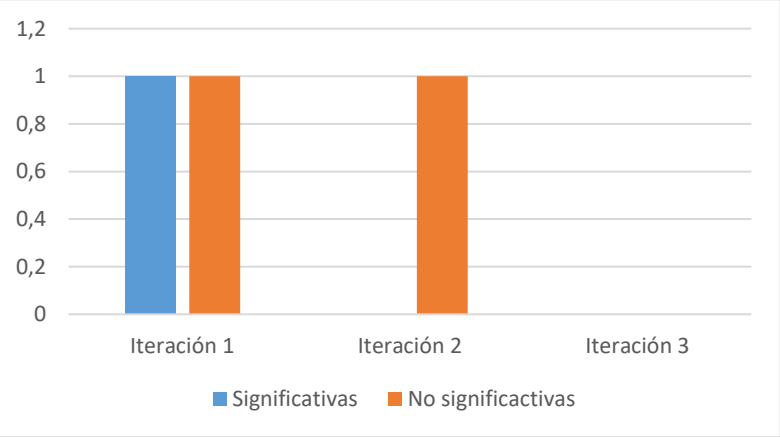


Gráfico 3.4.3 No conformidades de la Iteración 3.



Anexo 5

Luego de concluida las iteraciones y cumpliendo con lo planteado en la Metodología XP, el software fue validado funcionalmente por las pruebas unitarias y de aceptabilidad realizadas al finalizar cada iteración. Debido a que la plataforma engloba información personal de los investigadores, se hace necesario que la misma cumpla con buenas prácticas de seguridad informática para la fiabilidad de la contención de datos personales en la plataforma. Es por ello que se realizan pruebas de vulnerabilidad a la aplicación con la herramienta Acunetix WVS, por sus siglas en inglés (Web Vulnerability Scanner).

Acunetix WVS es una herramienta de seguridad de aplicaciones web automatizada. Es capaz de escanear cualquier sitio o aplicación web que sea accesible a través del protocolo HTTP/HTTPS. Sin embargo, no todas las pruebas se pueden realizar de forma automática, por lo tanto, Acunetix WVS proporciona herramientas de penetración manuales para pruebas particulares. Comprueba diferentes vulnerabilidades como inyecciones SQL, XSS, XXE, SSRF, Host Header Injection y otras 4500 vulnerabilidades web.

Luego de haber realizado el escáner de vulnerabilidades la herramienta realizó 3 887 peticiones en 37 minutos y 50 segundos, clasificándolas en:

- Vulnerabilidades de Severidad Alta
- Vulnerabilidades de Severidad Mediana
- Vulnerabilidades de Severidad Baja

Realizado el chequeo se detectaron un total de 10 posibles vulnerabilidades. De ellas 7 vulnerabilidades de severidad mediana y 3 vulnerabilidades de severidad baja, entre las que resaltan:

- Modo Debug habilitado: El modo de desarrollador en Django se encuentra activado lo que se visualiza los errores de la aplicación en las respuestas al cliente.

  Respuesta: Para trabajar en la programación de la plataforma es necesario tener habilitado el modo desarrollador que permite interactuar el programador con el marco de trabajo de Django, mostrando mensajes de error y de debugin de la plataforma al cliente junto con la petición. Este tipo de modalidad es retirada una vez se despliega la aplicación en un entorno real dependiendo de otras herramientas para su puesta en marcha.
- Mensaje de error o advertencia de la aplicación: Información sensible expuesta que puede ser vulnerable para la aplicación.

  Respuesta: Al estar el modo de desarrollador activado, todo mensaje de error y de advertencia que muestra información necesaria para corregir los errores de la plataforma

son mostrados al programador a través de una interfaz al cliente. Lo que se erradica una vez que se despliega la aplicación en un entorno real de ejecución.

- Las credenciales son enviadas en texto plano: No existe un canal de transmisión cifrada (https) y puede ser receptado el envío de contraseñas por un usuario malicioso.

  Respuesta: Básicamente el servidor que ofrece Django para correr las aplicaciones que se desarrollan es por defecto lanzado bajo protocolo http. Al montar la aplicación en un servidor donde se desplegará la aplicación; el administrador de recursos de la web deberá configurar el dominio para ofrecer seguridad en la transmisión de información sensible introducida por el usuario y el acceso a la plataforma empleando el protocolo https.

- Cookies sin insertar la bandera HttpOnly: Hay cookies que se encuentran desprotegidas y pueden ser accedidas desde un script malicioso como es el caso de una cookie de sesión.

  Respuesta: Este error se visualizaba en la dirección raíz del dominio dónde no existía respuesta para la plataforma y por tanto la devolución de una página html sin las banderas de protección para cookies sensibles. Luego de ser detectada se implementa una función de redirección bajo esta url, a la página inicial o login del sistema dependiendo de si está o no autenticado un usuario.

- Página de login no protegida contra ataques de fuerza bruta: La herramienta intentó entrar al sistema con una combinación de texto alfanumérica de forma indefinida.

  Respuesta: El formulario de entrada al sistema no posee un mecanismo para frenar el intento de entrada al sistema por fuerza bruta. Por lo que se implementó un mecanismo que permitiera introducir 5 veces la contraseña y luego se le bloqueaba al IP la posibilidad de enviar al sistema nuevos intentos de entrada hasta pasados 5minutos.

- Auto-completamiento activado en caja de contraseña: El atributo de auto-completamiento está desactivado lo que posibilita que al usuario escribir su usuario, si guardó sus credenciales pues esta se rellene dando la posibilidad al atacante de obtener sus credenciales de la cache del navegador.

  Respuesta: Detectada esta vulnerabilidad se le introdujo al formulario de inicio de sesión el atributo: <INPUT TYPE="password" AUTOCOMPLETE="off">

Anexo 6 Pruebas unitarias

Tabla 3.4.22 Resultado final tras 4 repeticiones al código de la Iteración 1.

Módulos	Argumentos	Ausente	Excluido	Cobertura
apps\monitoring__init__.py	0	0	0	100%
apps\monitoring\web_app.py	0	0	0	95%
apps\monitoring\module_general__init__.py	0	0	0	100%
apps\monitoring\web_server.py	172	15	0	91.28%
apps\monitoring\module_monitor__init__.py	0	0	0	100%
apps\monitoring\module_user__init__.py	0	0	0	100%
apps\monitoring\module_general\view.py	145	23	0	84.14%
apps\monitoring\module_monitor\views.py	0	0	0	100%
apps\monitoring\module_user\views.py	56	39	0	30.36%
apps\monitoring\views.py	21	3	0	85.72%
Total	**394**	**80**	**0**	**88.65%**

Tabla 3.4.23 Resultado final tras 4 repeticiones al código de la Iteración 2.

Módulos	Argumentos	Ausente	Excluido	Cobertura
apps\monitoring__init__.py	0	0	0	100%
apps\monitoring\web_app.py	0	0	0	95%
apps\monitoring\module_general__init__.py	0	0	0	100%
apps\monitoring\web_server.py	172	15	0	91.28%
apps\monitoring\module_monitor__init__.py	0	0	0	100%
apps\monitoring\module_user__init__.py	0	0	0	100%
apps\monitoring\module_general\view.py	145	23	0	84.14%
apps\monitoring\module_monitor\views.py	0	0	0	100%
apps\monitoring\module_user\views.py	56	39	0	30.36%

apps\monitoring\client.py	74	18	0	75.68%
apps\monitoring\module_setting__init__.py	0	0	0	100%
apps\monitoring\module_setting\views.py	9	0	0	100%
apps\monitoring\views.py	21	3	0	85.72%
Total	**570**	**105**	**0**	**89.62%**

Tabla 3.4.24 Resultado final tras 4 repeticiones al código de la Iteración 3.

Módulos	Argumentos	Ausente	Excluido	Cobertura
apps\monitoring__init__.py	0	0	0	100%
apps\monitoring\web_app.py	0	0	0	95%
apps\monitoring\module_general__init__.py	0	0	0	100%
apps\monitoring\web_server.py	172	15	0	91.28%
apps\monitoring\module_monitor__init__.py	0	0	0	100%
apps\monitoring\module_user__init__.py	0	0	0	100%
apps\monitoring\module_general\view.py	145	23	0	84.14%
apps\monitoring\module_monitor\views.py	0	0	0	100%
apps\monitoring\module_user\views.py	56	39	0	30.36%
apps\monitoring\client.py	74	18	0	75.68%
apps\monitoring\server.py	93	7	0	92.48%
apps\monitoring\module_setting__init__.py	0	0	0	100%
apps\monitoring\module_setting\views.py	9	0	0	100%
apps\monitoring\views.py	21	3	0	85.72%
Total	**570**	**105**	**0**	**89.62%**