UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

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

Sistema para la administración de la gestión de la configuración del Nodo Central de la Universidad de las Ciencias Informáticas.

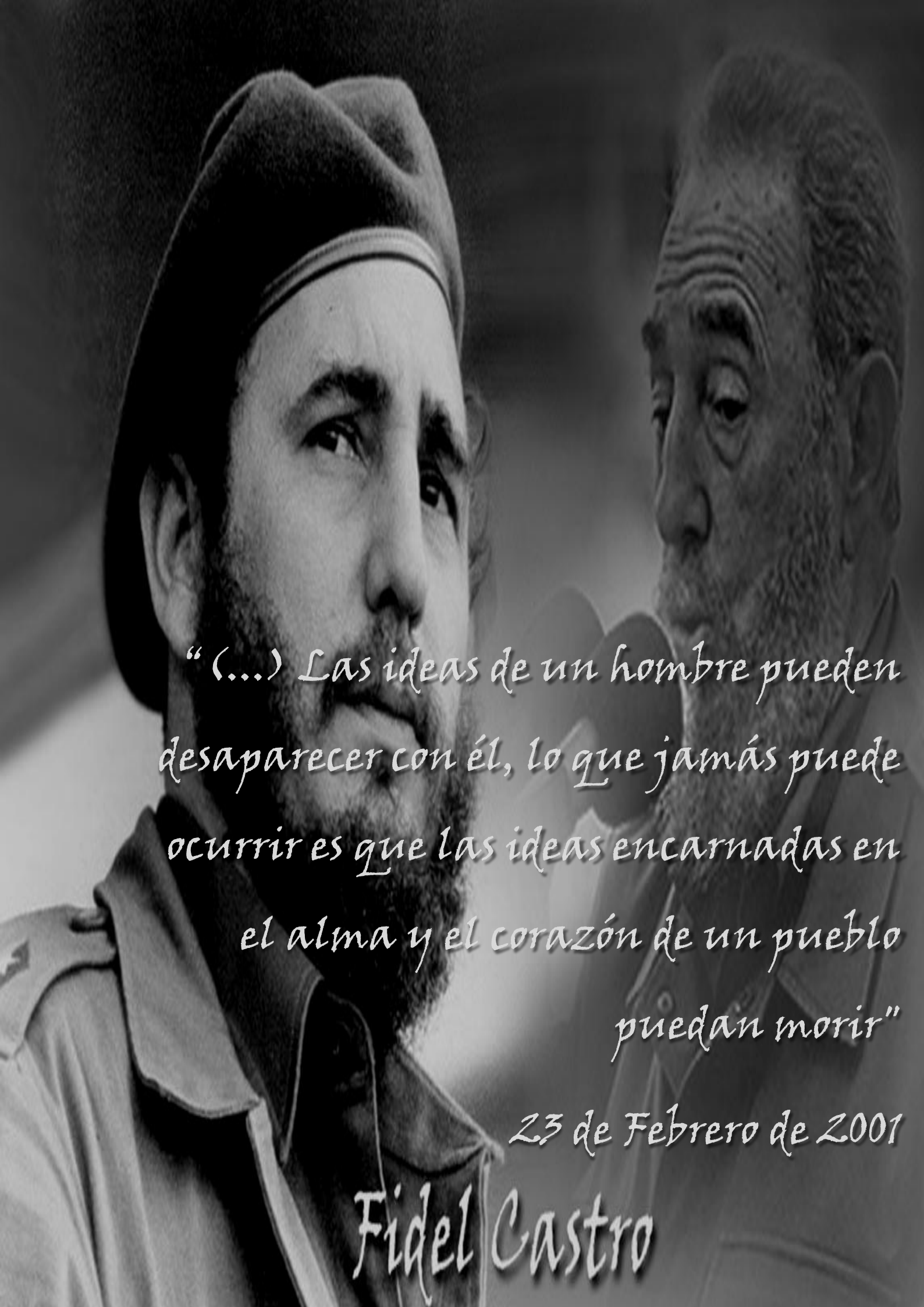
Autores: Julién Pérez García

Raidel Placencia Rodríguez

Tutor: Omar Pimentel Alfonso.

Co-Tutor: Osmar Capote Vázquez

La Habana, Cuba
Enero, 2017
"Año 59 de la Revolución"


“(...) Las ideas de un hombre pueden desaparecer con él, lo que jamás puede ocurrir es que las ideas encarnadas en el alma y el corazón de un pueblo puedan morir”
23 de Febrero de 2001
Fidel Castro

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste, firmamos la presente a los ____ días del mes de _____ del año 20____.

Julien Pérez García

_____

Firma del Autor

Raidel Placencia Rodríguez

_____

Firma del Autor

Omar Pimentel Alfonso

_____

Firma del Tutor

Osmar Capote Vázquez

_____

Firma del Co-Tutor

Agradecimientos

Dedicatoria

Resumen

Los Sistemas de gestión de la configuración de servidores en centros de datos se han convertido en potentes herramientas que facilitan la gestión y administración de la información que se maneja en los mismos, a pesar de que la curva de aprendizaje para el trabajo con las mismas es pronunciada. El objetivo de esta investigación es mejorar la eficacia con la que se realizan los procesos mencionados anteriormente en el Nodo central de la Universidad de las Ciencias Informáticas(UCI). Para ello se desea desarrollar una aplicación web que permita una navegación simple e intuitiva para los usuarios a la hora de realizar las tareas de configuración en los servidores y además permita monitorear el estado de los mismos. Con este fin se analizan los elementos teóricos – metodológicos que sustentan la propuesta de solución. Se realiza un estudio de homólogos para sistemas de gestión de la configuración a partir de un conjunto de indicadores con la finalidad de determinar los elementos que conforman la propuesta para el Nodo Central. A través del sistema los usuarios pueden monitorear los servidores (minions) existentes, aceptar, rechazar y denegar minions, editar y aplicar ficheros de estado, así como acceder a una interfaz de línea de comandos si prefieren realizar los trabajos desde la misma. Para la implementación del sistema se selecciona el marco de trabajo (Framework) Django y *eXtreme Programming* (XP) como metodología de desarrollo de *software*.

Palabras clave: aplicación web, centro de datos, configuración de servidores, sistema de gestión de la configuración.

Índice

Introducción

A partir de 1950 durante lo que se denominó la Era de la Información, las tecnologías comienzan a jugar un papel cardinal en la vida diaria del hombre. Surge un nuevo concepto que hoy en día se encuentra presente en casi todos los sectores de la sociedad denominado: Tecnologías de la Información y las Comunicaciones (TIC), tecnologías cuyas aplicaciones suponen un gran cúmulo de oportunidades, pero a la vez grandes desafíos.

En la actualidad, las organizaciones operan de forma distribuida en diferentes países y regiones, y cuentan con cientos o miles de dispositivos. Esto conlleva que los administradores de redes y sistemas se enfrenten, día a día, al reto de gestionar la infraestructura tecnológica. Esta presenta distintos escenarios que, por lo general, cuentan con menos recursos de los que requieren y demandan capacidad de respuesta en poco tiempo. (Castillo, 2014)

Cuba no ha desestimado esfuerzos para mantenerse a la par del desarrollo de las TIC. Con vistas a ubicarse entre los principales países productores de software y poder así desarrollar herramientas que contribuyan a nuestro trabajo, creó la Universidad de las Ciencias Informáticas. (Carbó, 2016)

Para lograr este fin se brinda a toda la comunidad universitaria una serie de servicios telemáticos en apoyo a sus actividades en la docencia, la investigación, la producción y la extensión. Dichos servicios se encuentran alojados en los diferentes servidores distribuidos en los nodos de la red de la universidad.

La Dirección de Redes y Servicios Telemáticos (DRST) es el encargado de la administración y gestión de la configuración de los servidores, así como de garantizar la disponibilidad e integridad de los mismos además de los servicios telemáticos que se brindan a toda la comunidad universitaria. Este proceso se realiza y controla desde el nodo central y para ello utiliza la herramienta de automatización: SaltStack.

Salt es un software 100% open source para la gestión en remoto de servidores basado en Python. Se trata de una forma sencilla y rápida de gestionar cualquier infraestructura que tengamos, por muy grande o pequeña que esta sea, ya sea formada por unos pocos servidores o bien por miles de ellos. Una de las principales características de Salt es que es posible comunicarse con cualquier servidor en cuestión de segundos. (Salt, 2013)

Llevar a cabo la administración y la gestión de la configuración de los servidores y de los servicios telemáticos significa instalar, configurar y mantener los recursos necesarios para su explotación en producción. Mientras estas acciones

se ejecuten manualmente, siempre se corre el riesgo de generar complejidad innecesaria en el sistema, haciendo el mantenimiento difícil y costoso. Desafortunadamente las aplicaciones de software suelen fallar y el costo del tiempo fuera de servicio aumenta con cada segundo. Estos errores frecuentemente son causados por problemas de configuración y cuando se manejan sistemas distribuidos donde existen dependencias entre los servicios, la probabilidad de error aumenta drásticamente.

Para la gestión de la configuración del nodo central de la UCI, a pesar de que se usa la herramienta SaltStack que agiliza y facilita este proceso, todo el trabajo se hace de forma manual introduciendo los comandos desde consola sobre la herramienta. La curva de aprendizaje para trabajar con esta, es pronunciada. La misma no cuenta con una interfaz visual que posibilite realizar este trabajo por lo que este proceso se vuelve lento y engorroso, además en casos donde se desea realizar configuraciones similares es necesario repetir toda la configuración desde el principio y esto puede conllevar a la ocurrencia de errores humanos durante el proceso.

De acuerdo a la **situación problemática** planteada anteriormente, surge como **problema a resolver**: ¿Cómo mejorar la gestión de la configuración del nodo central de la UCI de forma tal que permita realizarse con mayor eficacia?

Luego del análisis de la situación actual, la investigación enmarca su **objeto de estudio** en: la gestión de la configuración en centro de datos, centrándose el **campo de acción** en: la gestión de la configuración en el nodo central de la UCI.

El **objetivo general** de la investigación es: Desarrollar una aplicación que permita la administración de la gestión de la configuración del nodo central de la UCI de forma más eficaz.

Para dar cumplimiento al objetivo general se plantean las siguientes **tareas de investigación** (Presman, 2010):

1. Análisis bibliográfico para generar el marco teórico conceptual en lo referente al desarrollo de la gestión de la configuración y sistemas que la administren.

2. Análisis de sistemas homólogos para conocer aspectos regulares en el diseño de aplicaciones que administren la gestión de la configuración aplicados al nodo central de la UCI.

3. Análisis de las principales deficiencias y necesidades, presentes en la forma actual de desarrollarse el proceso de gestión de la configuración en el nodo central de la UCI, para diseñar el sistema.

4. Análisis de las herramientas informáticas y metodologías de desarrollo de software existentes para realizar la implementación del sistema.
5. Identificación de las principales funcionalidades del sistema para la posterior implementación del mismo, teniendo en cuenta los requisitos definidos anteriormente.
6. Investigación de los diferentes tipos de pruebas funcionales, para su posterior aplicación sobre el sistema desarrollado.

Métodos investigativos (Coello, 2012)**:**

Para llevar a cabo la investigación, se emplean los métodos de nivel teórico y empírico.

Métodos teóricos:

- ✓ **Método lógico:** se basa en el estudio histórico del fenómeno, pone de manifiesto la lógica interna de su desarrollo, de su teoría y encuentra el conocimiento más profundo de su esencia.

  En el trabajo de diploma, este método permite realizar una evaluación cronológica centrada en la evaluación de las aproximaciones desarrolladas a nivel mundial para el desarrollo de sistemas para la administración de la gestión de la configuración.

- ✓ **Método Analítico-Sintético:**

  Análisis: permite la división mental del fenómeno en sus múltiples relaciones y componentes para facilitar su estudio.

  Síntesis: establece mentalmente la unión entre las partes previamente analizadas, posibilita descubrir sus características generales y las relaciones esenciales entre ellas.

  En el trabajo de diploma, este método permite realizar un análisis exhaustivo de toda la teoría y la literatura existente relacionada con el desarrollo de los Sistemas de Gestión de la Configuración.

  Modelación: La modelación es el método mediante el cual se crean abstracciones con el objetivo de explicar la realidad. El modelo como sustituto del objeto de investigación es semejante a él, existiendo una correspondencia objetiva entre el modelo y el objeto, siendo el investigador quien elabora dicho modelo. El modelo es el eslabón entre el sujeto y el objeto intermedio.

En el trabajo de diploma se puede apreciar el uso de este método cuando se crea el modelo de datos..

Métodos empíricos:

- ✓ **Observación:** La observación científica es la percepción planificada dirigida a un fin y relativamente prolongada de un hecho o fenómeno. Es el instrumento universal del científico, se realiza de forma consciente y orientada a un objetivo determinado.

  En el trabajo de diploma, este método posibilita obtener información relacionada con el comportamiento de los indicadores de eficacia durante el proceso de desarrollo del sistema para la administración de la gestión de la configuración utilizando la propuesta de solución.

Para una mejor comprensión de la investigación, se decidió definir una estructura capitular que aporte cierto grado de organización y facilite el estudio del documento. Los capítulos que la conforman, son los siguientes:

Capítulo 1: Fundamentación teórica del Sistema para la administración de la gestión de la configuración del Nodo Central de la UCI.

Incluye el estudio del tema a tratar, tanto a nivel internacional como nacional, así como una descripción de conceptos que son esenciales para la comprensión de este trabajo. Además, una fundamentación de las tecnologías, herramientas y metodologías seleccionadas para utilizar en la solución del problema planteado.

Capítulo 2: Análisis y diseño del sistema.

En este capítulo se lleva a cabo el proceso de Arquitectura de Información. Se presentan las características del sistema y los roles que interactúan con el mismo. Se modela la información y generan los artefactos correspondientes a las fases de Exploración y Planificación para de esta forma dar paso a la posterior fase de Implementación y Validación.

Capítulo 3: Implementación y validación de la propuesta.

En este capítulo se implementan las historias de usuarios determinadas por el cliente y se realizan pruebas de aceptación a la aplicación, con las cuales se verifica que cumpla con los requerimientos del cliente.

Capítulo 1: Fundamentación teórica del Sistema

En el presente capítulo, se brinda información acerca de los principales conceptos referentes a los sistemas de gestión de la configuración. Además, se describe la justificación del conjunto de herramientas, tecnologías y metodología utilizadas para desarrollar la aplicación.

1.1. Conceptos generales

Sistemas de gestión: Son estructuras interactivas formadas por personas, equipos y métodos, destinados a crear un flujo de información capaz de proporcionar un fundamento adecuado para la toma de decisiones o sencillamente la solución a problemas específicos. (Jordi Pau i Cos, 1998)

Gestión de la configuración:

La gestión de la configuración es el registro y actualización detallados de la información que describe el hardware y software de una empresa. Dicha información incluye típicamente las versiones y actualizaciones que se han aplicado a los paquetes de software instalados, y la ubicación y las direcciones de red de los dispositivos de hardware. (Rouse, 2014)

Un objetivo importante de la gestión de la configuración es asegurar que los cambios realizados en un sistema no afectarán negativamente a cualquiera de los otros sistemas. **Sistema de Gestión de la Configuración:** Es el sistema encargado de proporcionar herramientas, técnicas y procedimientos para la realización de la Gestión de la Configuración.

Master: nombre asignado al servidor central.

Minions: nombre asignado a los nodos, los cuales son gestionados por medio de línea de comandos, donde se interpreta y genera las soluciones de administración de dispositivos remotos. El Master identifica a los Minions por medio de una etiqueta de identificación única, la cual viaja a través de Secure Shell.[1] (Castillo, 2014)

Grains(Granos): son una estructura que brindan una descripción detallada de cada una de los Minions, entre la información que almacenan están la familia pertenecen, sistemas operativos. Para gestionar los granos, SaltStack cuenta con una serie de módulos precargados, y cada uno de los elementos que componen la infraestructura (Castillo, 2014).

[1] Protocolo intérprete de red seguro.

Formulas: archivos de configuración donde se indican las distintas tareas que deben de realizar. Estos archivos siguen una estructura YAML y su funcionamiento es muy sencillo. Todas las fórmulas que se crean deberán estar ubicadas dentro del Master en la ruta "/srv/salt/" (Castillo, 2014).

1.2. Estudio del estado del arte

En la industria mundial actual, el nivel informatización de los diferentes procesos de una empresa marca la diferencia entre quienes son los números uno en el mercado y quiénes no. Para alcanzar estas metas es necesario que cada proceso se realice con la mayor calidad y en el menor tiempo posible.

Los sistemas de gestión se han hecho necesarios en el quehacer diario de cualquier institución. Estos reducen los costos de una gran cantidad de operaciones, tanto en el ámbito monetario como del personal necesario. Para guiar el desarrollo del presente trabajo se realizó un estudio de diferentes sistemas de gestión de la configuración, tanto a nivel internacional como en Cuba, persiguiendo el objetivo de encontrar similitudes y diferencias entre ellos y tomar las mejores prácticas que se tendrán en cuenta en la solución que propone el presente trabajo.

1.2.1. Sistemas de gestión de configuraciones a nivel nacional

A nivel nacional se analizó el siguiente sistema de gestión de la configuración.

GEReport: Es una herramienta destinada al diseño, generación y configuración de los reportes relacionados con los datos históricos almacenados en una fuente de datos. Además, es posible exportar la información del reporte como imagen y en los formatos HTML, PDF y Excel. Para la interacción con las aplicaciones externas, el sistema implementa un servicio que expone los metadatos de los reportes para poder utilizarlos sin restricciones de lenguajes y plataformas. Es un sistema web desarrollado siguiendo lo establecido por el Proceso Unificado de Desarrollo (RUP), utilizando UML como lenguaje de modelado. Para la gestión de la base de datos se seleccionó PostgreSQL, en la implementación se utilizaron los lenguajes de programación PHP, con CodeIgniter 2.0 como marco de trabajo del lado del servidor y JavaScript con Dojo Toolkit 1.8 para el trabajo del lado del cliente. Se utiliza en la Universidad de Cienfuegos por analistas y programadores del Grupo de Estudios y Desarrollo de Ingeniería y Sistemas (GEDIS), perteneciente a la Facultad de Ingeniería (Rodriguez Hernández, y otros, 2014).

1.2.3. Sistemas para la administración de gestores de configuración a nivel internacional.

Foreman es una herramienta completa de gestión del ciclo de vida para servidores físicos y virtuales. Proporciona a los administradores de sistemas la capacidad de automatizar fácilmente las tareas repetitivas, implementar rápidamente las aplicaciones y administrar proactivamente los servidores, tanto en las instalaciones como en la nube, como se muestra en la Imagen 1.

Se incluye un clasificador de nodos externos, parámetros like-hiera y supervisión de informes para Puppet[2], Salt-Stack[3] y Chef[4]. Totalmente listo para ajustar grupos de hosts en su centro de datos.

Los informes de Foreman muestran lo que sucede en sus nodos, y notifican al usuario cuando las cosas salgan mal. Permite observar desde su panel de control qué hosts son saludables y cuáles están desfasados.

Brinda un fácil acceso a todas las llamadas de API que necesita para mantenerse en la parte superior de su centro de datos.

Proporciona una API RESTful que le permite automatizar la mayoría de las tareas, como registrar hosts, asignar roles a los usuarios y más.

Un sistema de autorización completo basado en el control de acceso basado en funciones (RBAC) permite políticas estrictas para los usuarios de Foreman. Si utiliza LDAP[5] o FreeIPA, puede seguir usándolos para la autenticación y la autorización (Castillo, 2014).

[2] Sistema gestor de configuraciones

[3] Sistema gestor de configuraciones

[4] Sistema gestor de configuraciones

[5] Autenticación mediante el directorio activo

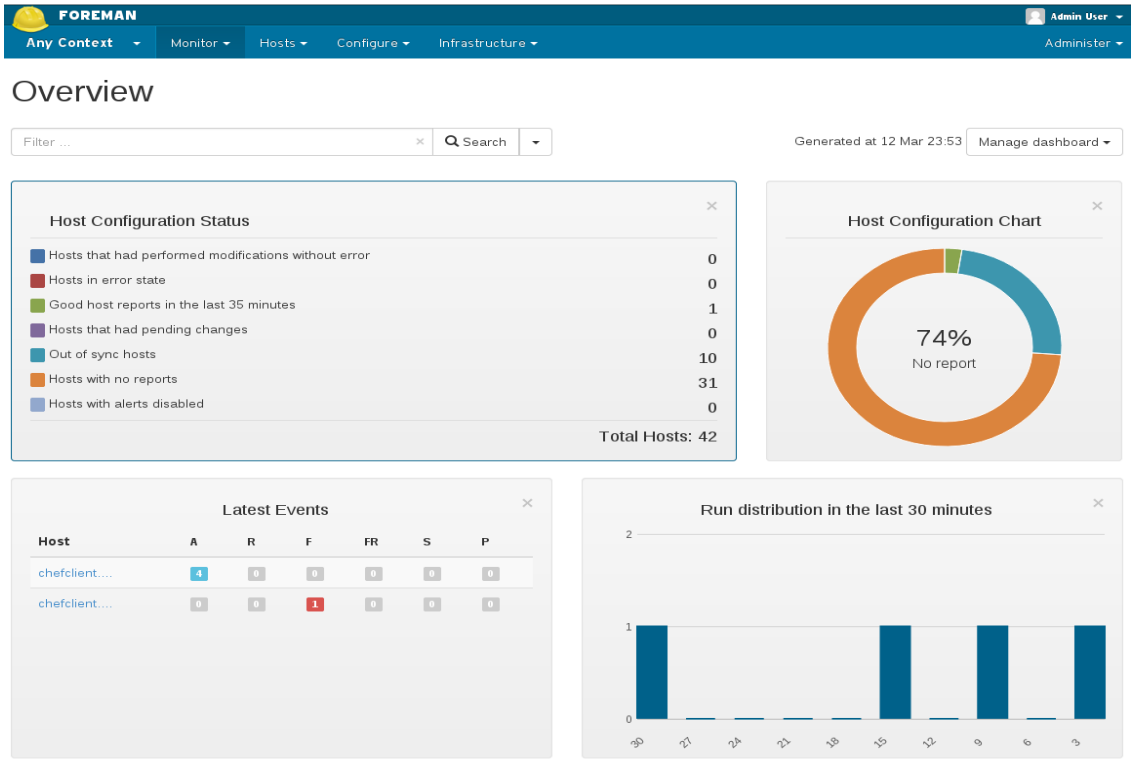


Imagen 1 Foreman: Interfaz gráfica de Puppet.

SaltStack - Halite es una de las herramientas para desplegar y orquestar infraestructuras de nube, optimizar la configuración de sistemas y controlar una vasta cantidad de entornos.

Adicionalmente mantiene público en github una colección de fórmulas y además cuenta con un gestor de fórmulas Salt Package Manager (SPM) que permite construir y desplegarlas de forma sencilla y análoga a los sistemas de empaquetado. Una de sus ventajas es que soporta ambos mecanismos de distribución (push y pull) según sean las necesidades del cliente. Cuenta con diferentes componentes desde los que podemos ejecutar las acciones remotamente sobre los nodos administrados o distribuir los cambios mediante aplicaciones instaladas previamente en los nodos a gestionar.

Salt emplea dos enfoques de prueba: pruebas de integración y pruebas unitarias, para las que provee bibliotecas en aras de minimizar las pruebas manuales. Adicionalmente cuenta con servicios de consulta y entrenamiento que pueden ser incluidos en el contrato de soporte e incluso un servicio de certificación.

SaltStack trabaja con un modelo cliente-servidor, donde el servidor central se conoce como Master y los nodos se llaman Minions. Cuenta con una detallada guía para probar las funcionalidades básicas de las herramientas de su arquitectura, y se hace necesario, para probar el despliegue, de un master y 2

minions (Pérez, 2016).

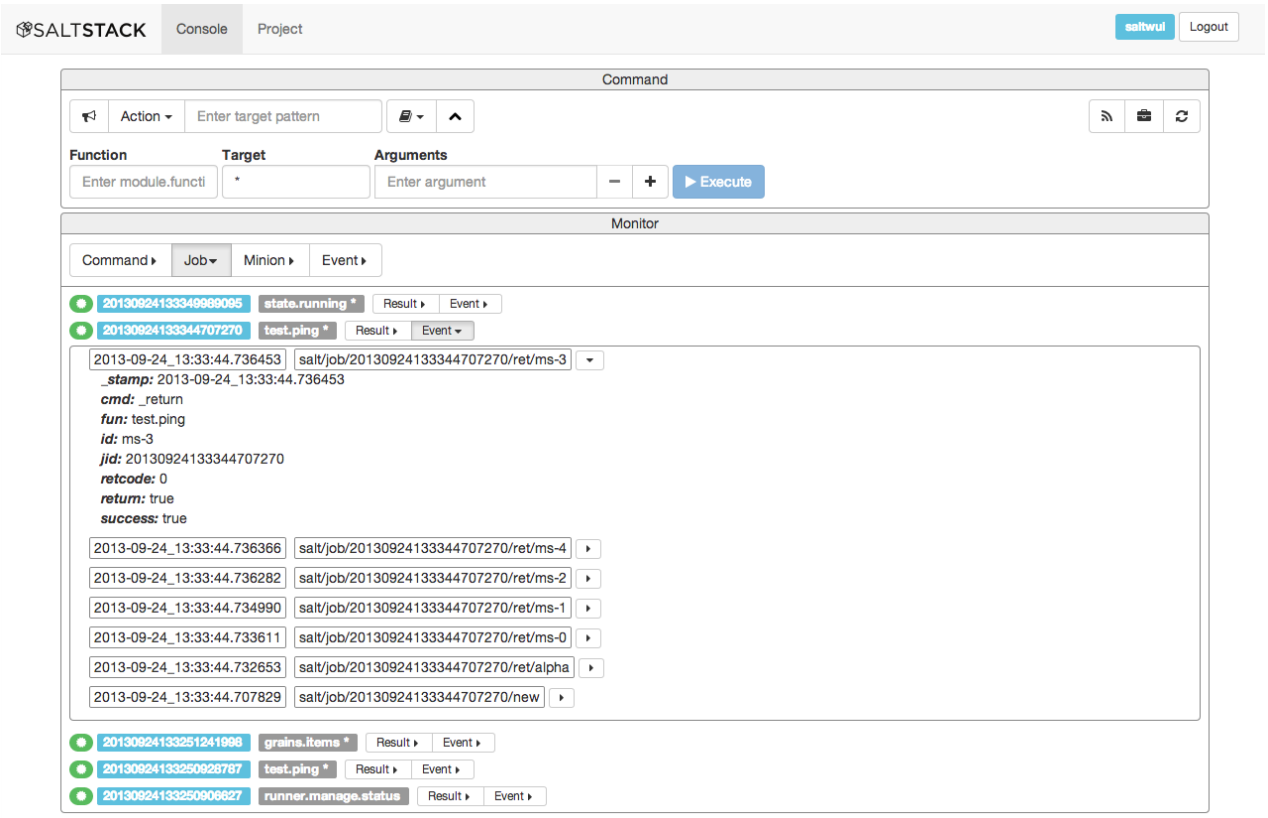


Imagen 2: Halite: Interfaz Gráfica de SaltStack

1.2.4. Conclusiones del estudio

Luego de analizado los sistemas anteriores, se concluye que:

Foreman, además de ser un sistema privado, funciona sobre Puppet, y en la UCI actualmente se cuenta con SaltStack como Sistema Gestor de Configuraciones; por tanto, se decide no utilizar Foreman. Por otra parte, Halite fue montado en un servidor de pruebas en la universidad, y quedo muy por debajo de las necesidades de la DRST.

Por lo que se decide desarrollar una aplicación capaz de integrar algunas de las mejores prácticas de los sistemas anteriores, tomando características como la supervisión de informes, monitor de hosts, la Interfaz de Línea de Comandos (CLI), programación de tareas, integración con el Directorio Activo, así como poseer una interfaz gráfica amigable tanto para usuarios experimentados como para nuevos usuarios.

1.3. Herramientas y lenguajes informáticos

1.3.1. Herramientas CASE

Visual Paradigm v8.0: herramienta CASE empleada para visualizar y diseñar elementos de software, para ello utiliza el lenguaje UML (por sus siglas en inglés Unified Modeling Language), proporciona a los desarrolladores una plataforma que les permite diseñar un producto con calidad de forma rápida. Se integra con diversos Entornos de Desarrollo Integrados como: NetBeans (de Sun), Eclipse (de IBM 2), JDeveloper (de Oracle), JBuilder (de Borland). Está disponible en varias ediciones: Enterprise, Professional, Community, Standard, Modeler y Personal. Genera código y realiza ingeniería inversa para diferentes lenguajes de programación como: Java, C++, PHP, XML. En adición se genera código para C#, Visual Basic.net, ODL (Object Definition Language), Flash Action Script, Delphi, Perl y Python. Además, exporta e importa los diagramas en el estándar XML (PARADIGM, 2010).

Ventajas de Visual Paradigm:

- Ofrece entorno de creación de diagramas para UML 2.1.
- Disponibilidad en múltiples plataformas.
- Generación de bases de datos: Permite la generación automática de bases de datos a partir de un Modelo entidad-relación.
- Interoperabilidad entre diagramas: Permite a partir de un diagrama obtener otro que guarde relación con el mismo.
- Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.
- Disponibilidad de integrarse a los principales IDEs.
- Soporta una gama de lenguajes en la Generación de Código e Ingeniería Inversa en Java, C++, PHP, Esquema de XML, Ada y Python.

1.3.2. Lenguajes de programación del lado del servidor

Python v2.7.11: Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos (Duque, 2008).

Django v1.8: Django es un framework web de alto nivel, escrito en Python, que ayuda al desarrollo rápido y a un diseño limpio y pragmático. Construido por desarrolladores experimentados, resuelve una buena parte de los problemas del desarrollo web de tal manera que uno se pueda enfocar en escribir su app sin necesidad de reinventar la rueda. Es gratis y de código abierto. (Django, 2015).

Características de Django (Django, 2015):

- Django fue diseñado para ayudar a los desarrolladores a llevar las aplicaciones desde el concepto hasta su finalización lo más rápido posible.
- Django incluye docenas de extras que puede utilizar para manejar tareas comunes de desarrollo Web. Django se encarga de la autenticación de usuarios y administración de contenido.
- Django toma la seguridad en serio y ayuda a los desarrolladores a evitar muchos errores de seguridad comunes, como la inyección de SQL, el cross-site scripting, la falsificación de solicitudes entre sitios y el clickjacking. Su sistema de autenticación de usuarios proporciona una forma segura de administrar cuentas de usuario y contraseñas.
- Algunos de los sitios más concurridos del planeta utilizan la capacidad de Django para escalar rápida y flexiblemente para satisfacer las demandas de tráfico más pesadas.
- Su increíble versatilidad ha permitido que las empresas, las organizaciones y los gobiernos utilicen Django para construir todo tipo de cosas, desde sistemas de gestión de contenidos a redes sociales hasta plataformas de computación científica.

1.3.3. Lenguajes del lado del cliente

HTML5: Quinta versión de HTML que remplaza a HTML4 corrigiendo problemas con los que se encontraban los desarrolladores. Con la aparición del mismo se introducen cambios en la semántica que hacen que la estructura de la web sea más coherente y de fácil entendimiento. Se utiliza para describir texto presentado de forma estructurada y agradable, con enlaces que conducen a otros documentos o fuentes de información relacionadas, y con inserciones de multimedia (gráfico, imágenes, sonido, video). Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web, es decir, se utiliza para crear las páginas web y les indica a los navegadores cómo deben mostrar el contenido (DE LUCA, 2011).

JavaScript: Lenguaje de programación que surgió con el objetivo inicial de programar ciertos comportamientos sobre las páginas web, respondiendo a la interacción del usuario. Hoy es el motor de las aplicaciones más conocidas en el ámbito de Internet: Google, Facebook, Twitter. La Web 2.0 se basa en el uso de JavaScript para implementar aplicaciones enriquecidas que son capaces de realizar todo tipo de efectos, interfaces de usuario y comunicación asíncrona con el servidor (FLANAGAN, 2002).

1.3.4. Gestores de base de datos

PostgreSQL v9.3: es un Sistema de Gestión de Base de Datos Relacionales orientadas a Objetos (ORDBMS) basado en POSTGRES, desarrollado en la Universidad de California en Berkeley Computer Science Department. Fue pionero en muchos conceptos que sólo se hicieron disponibles en algunos sistemas de bases de datos comerciales mucho más tarde. PostgreSQL es un descendiente de código abierto de este código original de Berkeley. Soporta una gran parte del estándar SQL y ofrece muchas características modernas:

• consultas complejas

• llaves foráneas

Además, PostgreSQL puede ser ampliado por el usuario de muchas maneras, por ejemplo, añadiendo

• tipos de datos

• funciones

• operadores

• funciones agregadas

• índices

• lenguajes procedimentales

1.3.5. Administración de la base de datos

PgAdmin vIII: es la plataforma de administración y desarrollo Open Source[6] más popular y con más características para PostgreSQL. Puede utilizarse en plataformas Linux, FreeBSD, Solaris, MacOS y Windows para gestionar PostgreSQL 9.2 y superiores en cualquier plataforma, así como versiones comerciales y derivadas de PostgreSQL como EDB Postgres Advanced Server. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL sencillas hasta desarrollar bases de datos complejas. La

[6] Término usado para referirse a las aplicaciones que son de código abierto

interfaz gráfica puede ejecutarse en el escritorio o en un servidor web y admite todas las características comunes de PostgreSQL y además incluye una sintaxis que resalta el editor de SQL. Es desarrollado por una comunidad de expertos de PostgreSQL en todo el mundo. Es un Software Libre publicado bajo la Licencia PostgreSQL (pgAdmin, 2016).

1.3.6. Metodología de desarrollo de software

Actualmente, los negocios operan en un entorno global que cambia rápidamente. Tienen que responder a nuevas oportunidades y mercados, condiciones económicas cambiantes y la aparición de productos y servicios competidores. El software es parte de casi todas las operaciones de negocio, por lo que es fundamental que el software nuevo se desarrolle rápidamente para aprovechar nuevas oportunidades y responder a la presión competitiva. Por lo tanto, actualmente el desarrollo y entrega rápidos son a menudo los requerimientos más críticos de los sistemas de software. Debido a que las compañías operan en un entorno cambiante, a menudo es prácticamente imposible obtener un conjunto completo de requerimientos de software estables. Los requerimientos que se proponen cambian inevitablemente. Por esto surgen los procesos de desarrollo rápido de software, los cuales están diseñados para producir software útil de forma rápida. Generalmente, son procesos iterativos en los que se entrelaza la especificación, el diseño, el desarrollo y las pruebas. Están pensados para entregar software funcional de forma rápida a los clientes, quienes pueden entonces proponer que se incluyan en iteraciones posteriores del sistema nuevos requerimientos o cambios en los mismos. Son muchos los métodos de desarrollo ágil, entre ellos: Scrum, Cristal, Desarrollo de Software Adaptable, DSDM, Desarrollo Dirigido por Características y el más conocido actualmente y que se utilizara a lo largo de este trabajo, Extreme Programming.

Extreme Programming (XP): metodología ágil para el desarrollo de software, muy popular, se centra en las necesidades del cliente con el objetivo de lograr un producto de calidad y aminorar el tiempo de entrega de la solución. Su principal objetivo es el logro de una relación de comunicación armoniosa entre el cliente y el equipo de desarrollo para así llegar a un producto informático que cumpla con las peticiones del primero y garantizar el éxito en el desarrollo de software. Promueve el trabajo en equipo estableciendo un buen clima. Esta metodología es la adecuada para proyectos pequeños y con requisitos imprecisos y muy cambiantes.

Se basa en:

- Re fabricación: se desarrolla en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

- Programación en pares: consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

    - Pruebas de aceptación o pruebas funcionales: destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final (J. J. Gutiérrez, 2014).

Ventajas del uso de XP:

- Se obtiene gran optimización en el código, ya que el mismo es generado para una arquitectura específica. Esto trae consigo un ahorro de hardware y tiempo de ejecución.

- Las posibilidades de fracasar el proyecto son muy bajas, ya que el cliente participa continuamente en el mismo y ante cualquier cambio de última hora, se vuelven a replantear los objetivos.

- Los errores son encontrados tempranamente ya que el producto se prueba continuamente.

Dada la falta de recursos humanos, poca envergadura del proyecto y el corto tiempo de entrega del mismo, se decidió el uso de la metodología ágil XP. Las metodologías ágiles son básicamente para proyectos pequeños, siempre están preparadas para cambios durante el desarrollo y sobre todo al cliente se le considera parte del equipo.

Conclusiones parciales

En este capítulo se abordaron los elementos asociados a un estudio del arte a nivel internacional y nacional de sistemas de gestión de la configuración. Además, se estudiaron varias tecnologías, herramientas informáticas y metodologías de desarrollo de software; definiéndose XP como metodología de desarrollo de software; Visual Paradigm como herramienta de modelado, como lenguajes de programación Python, y Django como marco de trabajo.

Capítulo 2: Análisis y diseño del sistema.

2.1. Descripción detallada del negocio.

El trabajo de gestión de la configuración en el nodo central se realiza en su totalidad por los administradores de redes, estos son los encargados de, a través de la herramienta SaltStack, definir las configuraciones pertinentes a los servidores que administran mediante la edición de los ficheros de estado de los mismos. El proceso de realizar una configuración a un grupo de servidores se inicia cuando el administrador de red se conecta a la herramienta SaltStack, luego define qué características y reglas se van a tener en cuenta para realizar el agrupamiento de dichos servidores antes de realizar la configuración. Una vez definidos estos criterios de agrupamiento se modifica el o los ficheros de estado correspondientes y para finalizar se aplica la regla, que no es más que aplicar dichos ficheros de estado en la configuración.

2.2. Características del sistema.

Las características del sistema son propiedades o cualidades que el producto presenta. A diferencia de las funcionalidades, estas son de mayor peso para la arquitectura del sistema y de no cumplirse afectan el funcionamiento del software. Definen apariencia, usabilidad, rapidez y confiabilidad.

Usabilidad: Cada sección del sistema, en la interfaz, debe comenzar con un título o encabezamiento que describa el contenido de la pantalla. En procesos con formularios y pantallas múltiples, cada página debe estar etiquetada para mostrar su relación con las otras. Cuando se introducen datos en la pantalla la terminología utilizada debe ser familiar para los usuarios.

Apariencia o interfaz externa: La presente investigación genera el artefacto Arquitectura de Información para la aplicación con la finalidad de estructurar, organizar y etiquetar el contenido y los elementos de navegación para facilitar la navegación de los usuarios.

Seguridad: La información manejada por el sistema está protegida de acceso no autorizado y divulgación. Existe un único usuario con el rol de administrador del sistema que otorga los permisos a los demás usuarios según su rol.

Portabilidad: La aplicación debe ser multiplataforma, es decir, debe poder ejecutarse sobre los Sistemas Operativos Windows y GNU Linux.

Legal: La plataforma escogida para el desarrollo del portal web, está basada en la Licencia Pública General (GNU/GPL).

Requisitos de la aplicación:

Servidor:

- Sistema Operativo Centos 7.
- Servidor web nginx 1.2.0.
- Python v2.7.11 y Django v1.8
- Postgres v9.3 o superior.

Cliente:

- Sistema operativo GNU Linux / Windows XP o superior.
- Navegador web Firefox 4.0 o superior, Internet Explorer 7.0 o superior.
- Microprocesador Intel Pentium IV 2.0 GHz, 512 MB RAM y 20 GB disco duro o superior.

2.3. Roles de usuario.

Los roles son los diferentes tipos de usuarios que interactúan con el sistema. Cada uno posee permisos de acuerdo a sus funciones en el sistema. La presente investigación identificó dos tipos de usuarios fundamentales:

- **Administrador:** Es el encargado de brindar soporte técnico al sistema, gestionar los usuarios y sus permisos. Tiene privilegios para realizar cualquier acción.
- **Gestor de configuraciones:** Es el encargado de gestionar grupos y configuraciones. Lleva acabo el monitoreo y administración de los minions.

2.4. Fase de exploración

“La fase de exploración marca el inicio de la metodología XP. En esta fase, los clientes plantean a grandes rasgos las historias de usuarios que son de interés para la primera entrega del producto y los programadores estiman el tiempo de desarrollo de cada una. Las primeras estimaciones son superficiales, ya que están basadas en el leguaje de alto nivel y pueden variar cuando se analicen más en detalle por cada iteración. Al mismo tiempo, el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto” (Ruiz, 2014).

En esta fase se realiza la entrevista al cliente con el objetivo de determinar las funcionalidades que deben estar presente en el sistema.

Entrevista realizada al cliente

Fecha: 1 de marzo de 2017.

Nombre del entrevistado: Ing. Omar Pimentel Alfonso.

Preguntas:

- ¿A qué tipo de usuarios está dirigido el sistema?
- ¿Por qué usar SaltStack?
- ¿Por qué CentOS para desplegar la aplicación?
- ¿Cuáles son las funcionalidades que deben estar presentes en el sistema?
- ¿Cuáles estándares y normas deben cumplir el sistema?
- ¿Qué roles intervienen en el sistema?

2.5. Funcionalidades del sistema

- Realizar un login utilizando el Directorio Activo y controlador de dominio.
- Gestionar minions.
- Integrar interfaz de línea de comando.
- Gestionar ficheros de estado.
- Aplicar ficheros de estado
- Monitorear la gestión de la configuración en los servidores

2.6. Historias de Usuario(HU)

Las historias de usuarios definen en XP los requisitos funcionales del sistema. Son escritas por el cliente en un lenguaje no técnico, sin hacer mucho hincapié en los detalles y especificando el tiempo estimado de duración de la funcionalidad que describe. Además, se define la prioridad para el negocio y el riesgo en desarrollo de tiene cada una de las historias de usuario. Estos parámetros se definen a continuación:

La prioridad en el negocio:

Alta: historias de usuario que resultan funcionalidades fundamentales en el desarrollo del sistema, es decir a las que el cliente define como principales para sistema.

Media: historias de usuario que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.

Baja: historias de usuario que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el sistema en desarrollo.
El riesgo en el desarrollo:

Alto: Cuando en la implementación de las historias de usuario se considera la posible existencia de errores que lleven a la inoperatividad del código.

Medio: Cuando pueden aparecer errores en la implementación de la historia de usuario que puedan retrasar la entrega de la versión.

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

A continuación, se muestra el formato de una historia de usuario:

Tabla 1: Maqueta de una historia de usuario

Historia de Usuario	
Número: número de la historia de usuario	Nombre: nombre de la historia de usuario.
Usuario: involucrados en el desarrollo de la historia de usuario.	
Prioridad en Negocio:	Riesgo en Desarrollo:

(Alta / Media / Baja)	(Alto / Medio / Bajo)
Puntos Estimados: tiempo estimado que se demora el desarrollo de la historia de usuario en semanas.(1 punto estimado equivale a 5 días que es una semana de trabajo)	Iteración Asignada: número de la iteración en que se desarrolla la historia de usuario.
Responsable: programador responsable.	
Descripción: breve descripción de la historia de usuario.	

Tabla 2: Historia de Usuario Autenticación

Historia de Usuario	
Número: 1	Nombre: Autenticación
Usuario: Administrador de Red	
Prioridad en Negocio: Alta	Riesgo en Desarrollo: Alto
Puntos Estimados: 1	Iteración Asignada: 1
Responsable: Julién Pérez García y Raidel Placencia Rodríguez	
Descripción: La página de autenticación estará compuesta de dos campos de texto para introducir el usuario y la contraseña y un botón de aceptar para enviar dichos datos.	

Tabla 3: Historia de Usuario Gestionar Minions

Historia de Usuario	
Número: 2	Nombre: Gestionar Minions
Usuario: Administrador de Red	
Prioridad en Negocio: Alta	Riesgo en Desarrollo: Alto
Puntos Estimados: 2	Iteración Asignada: 1
Responsable: Julién Pérez García y Raidel Placencia Rodríguez	

Descripción: Esta funcionalidad mostrará los minions existentes en un listado separados por su estado, aceptados, no aceptados, rechazados y denegados y permitirá cambiar el estado de cada uno de ellos.

Tabla 4: Historia de Usuario Gestionar Ficheros de Estado

Historia de Usuario	
Número: 3	Nombre: Gestionar ficheros de estado
Usuario: Administrador de Red	
Prioridad en Negocio: Alta	Riesgo en Desarrollo: Alto
Puntos Estimados: 2	Iteración Asignada: 1
Responsable: Julién Pérez García y Raidel Placencia Rodríguez	
Descripción: Esta funcionalidad mostrará todos los ficheros de estado existentes en un listado desde el cual se podrán modificar o aplicar a un servidor o grupo de servidores a través de un botón a la derecha de cada entrada del listado y crear un nuevo fichero de estado a través de un botón en la parte inferior del listado, el cual al ser pulsado abrirá un fichero de estado en blanco para ser llenado con la configuración deseada.	

Tabla 5: Historia de Usuario Aplicar Ficheros de Estado

Historia de Usuario	
Número: 4	Nombre: Aplicar ficheros de estado
Usuario: Administrador de Red	
Prioridad en Negocio: Alta	Riesgo en Desarrollo: Alto
Puntos Estimados: 1	Iteración Asignada: 1
Responsable: Julién Pérez García y Raidel Placencia Rodríguez	

Descripción: Para aplicar un fichero de estado se selecciona la opción aplicar en el listado de ficheros de estado, este lanza una ventana que permite seleccionar el o los servidores o grupo de servidores a los cuales se les va a aplicar el fichero de estado, luego se selecciona el botón aplicar en la parte inferior de la ventana.

Tabla 6: Historia de Usuario Integrar interfaz de línea de comandos

Historia de Usuario	
Número: 5	Nombre: Integrar interfaz de línea de comandos
Usuario: Administrador de Red	
Prioridad en Negocio: Alta	Riesgo en Desarrollo: Alto
Puntos Estimados: 2	Iteración Asignada: 1
Responsable: Julién Pérez García y Raidel Placencia Rodríguez	
Descripción: Esta funcionalidad mostrará una interfaz de línea de comando en la aplicación, de la cual se pueden realizar todas las tareas que comúnmente se ejecutaban en la terminal del sistema operativo.	

Tabla 7: Historia de Usuario Monitorear la gestión de la configuración en los servidores

Historia de Usuario	
Número: 6	Nombre: Monitorear la gestión de la configuración en los servidores
Usuario: Administrador de Red	
Prioridad en Negocio: Alta	Riesgo en Desarrollo: Alto
Puntos Estimados: 2	Iteración Asignada: 1
Responsable: Julién Pérez García y Raidel Placencia Rodríguez	

Descripción: Se muestran y registran en tiempo real los cambios (jobs) que se realizan sobre todos los servidores del sistema.

2.7. Estimación de esfuerzo por historia de usuario.

"Un punto dentro de la fase de planificación es la estimación del esfuerzo que costará implementar cada historia de usuario. La estimación del costo de la implementación de las HU es establecida por los programadores tomando como medida el punto. Un punto se considera como una semana ideal de trabajo, donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción" (Ruiz, 2014).
A continuación, se recoge en una tabla la estimación de esfuerzo por HU.

Tabla 8: Estimación del esfuerzo por historias de usuario

Historia de Usuario	Estimación
Autenticación	1
Gestionar Minions	2
Gestionar Ficheros de Estado	2
Aplicar Ficheros de Estado	1
Integrar interfaz de línea de comando	2
Monitorear Servidores por un conjunto de parámetros definidos	2

Conclusiones parciales

En este capítulo de se definieron las características y funcionalidades del sistema a partir de una entrevista realizada al cliente. Se definieron los roles de usuario que interactúan con el sistema. Se identificaron 6 HU y con alta prioridad para el negocio y alto riesgo de desarrollo, todas a ser implementadas en una sola iteración.

Capítulo 3: Implementación y Validación de la Propuesta.

La metodología XP propone que en la fase de implementación al alcanzar el final de cada iteración se debe examinar y mostrar el producto obtenido al cliente. Este proceso garantiza una constante retroalimentación entre desarrolladores y cliente, además aumenta la visión del equipo de trabajo sobre cuán cerca están de obtener el producto final. En el actual capítulo se exponen la implementación de las 6 HU previstas para la etapa de construcción del sistema, desglosando cada una en tareas. Además, se explican las pruebas realizadas al producto.

3.1. Tarjetas de Cargo o Clase, Responsabilidad y Colaboración (CRC).

La metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando Lenguaje Modelado Unificado (UML), en su lugar se usan las tarjetas CRC. Las tarjetas consisten en una estructura que permite representar las características más importantes de una clase. Por lo general el nombre de la clase o cargo se pone en el borde superior en forma de título, las clases asociadas a la que hace referencia se ubican en el lado derecho bajo el identificador de colaboraciones (constituyen las clases que trabajan en conjunto para lograr una responsabilidad) y las responsabilidades implicadas por esta clase se posicionan en el lado izquierdo (métodos y funciones que realiza la clase).

Tabla 9: Tarjeta CRC 1

jobsList	
Responsabilidades	Colaboraciones
• Listar Jobs • Ver detalles de los Jobs	• login • jobsListDetails

Tabla 10: Tarjeta CRC 2

minionsAlive	
Responsabilidades	Colaboraciones
• Listar Minions • Ver detalles de los Minions	• login • List_Minions

Tabla 11: Tarjeta CRC 3

minionStates	
Responsabilidades	Colaboraciones

• Listar Minions en sus diferentes estados • Cambiar el estado de los Minions	• login • Minion_States • Accept

Tabla 12: Tarjeta CRC 4

minionStatus	
Responsabilidades	Colaboraciones
• Listar Minions activos • Listar Minions inactivos	• login • List_MinionStatus

Tabla 13: Tarjeta CRC 5

statesFiles	
Responsabilidades	Colaboraciones
• Listar Ficheros de estado • Editar ficheros de estado	• login • States_Files

3.2. Modelo de Datos

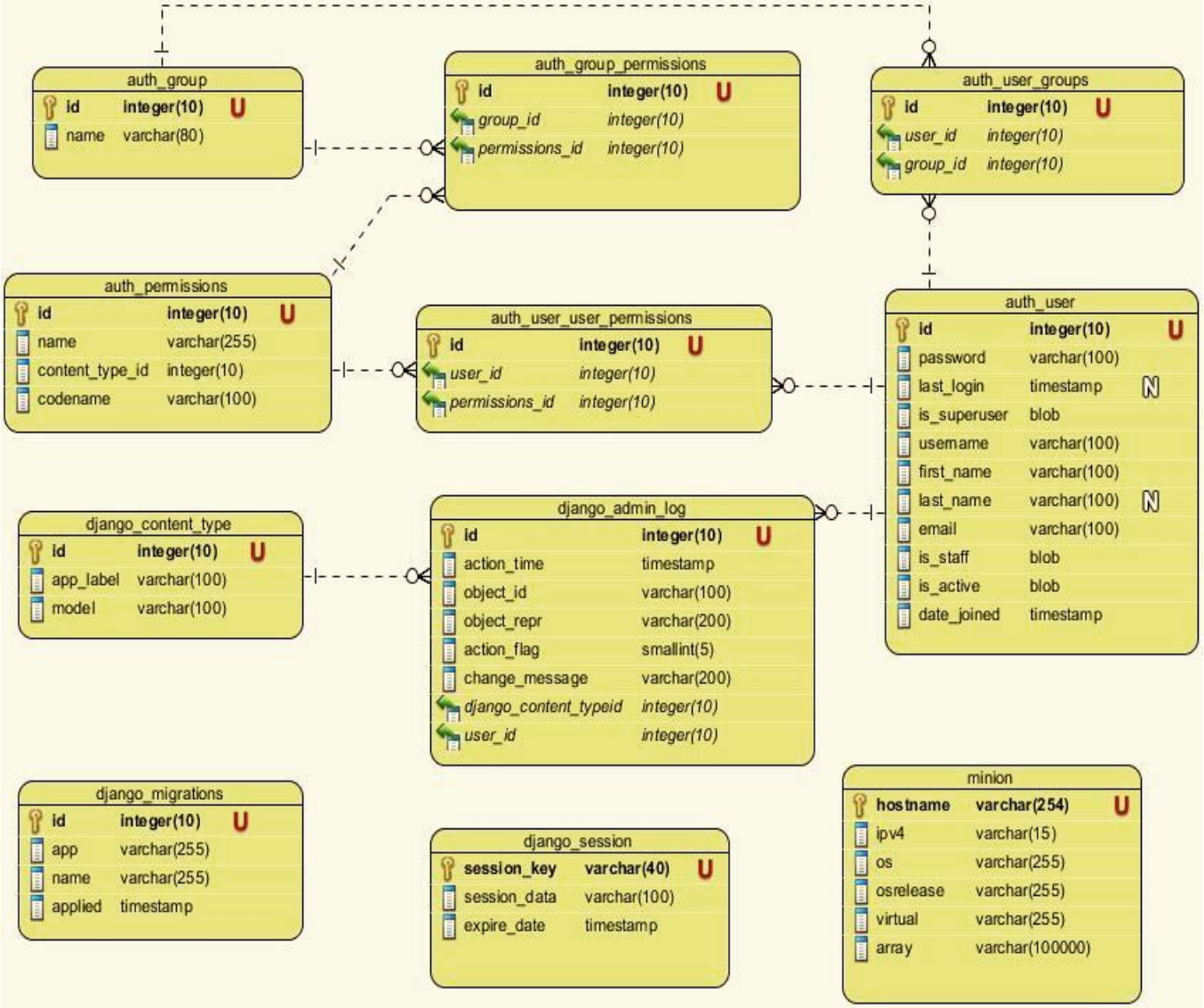


Imagen 3: Modelo de la base de datos

3.3. Estándar de codificación.

Para la implementación del sistema, la presente investigación se apoya sobre Django, un framework de desarrollo web basado en Python que usa los mismos estándares de codificación que este lenguaje, por lo que se mantendrá el uso de los estándares definidos para el mismo. A continuación, se exponen los elementos más significativos.

- Importaciones:

Las importaciones deben estar en líneas separadas y siempre se colocan al comienzo del archivo, simplemente luego de cualquier comentario o documentación del módulo, y antes de globales y constantes.

- Espacios en blanco en Expresiones y Sentencias

Evitar el uso de espacios en blanco extraños en las siguientes situaciones:

~ Inmediatamente dentro de paréntesis, corchetes o llaves

~ Inmediatamente antes de una coma, un punto y coma o dos puntos

~ Inmediatamente antes del paréntesis que comienza la lista de argumentos en la llamada a una función

~ Inmediatamente antes de un corchete que empieza una indexación

~ Más de un espacio alrededor de un operador de asignación (u otro) para alinearlo con otro

Otras recomendaciones

~ Siempre rodear estos operadores binarios con un espacio en cada lado: asignación (=), asignación de aumentación (+= , -= , etc.), comparaciones (==, <, >, !=, <>, <=, >=, in, not in, is, is not), booleanos (and, or, not).

~ Si se utilizan operadores con prioridad diferente, considera agregar espacios alrededor del operador con la menor prioridad.

~ No usar espacios alrededor del = (igual) cuando es utilizado para indicar un argumento en una función o un parámetro con un valor por defecto.

~ Las sentencias compuestas (múltiples sentencias en la misma línea) son generalmente poco recomendadas.

- Comentarios

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. ¡Nunca cambiar las mayúsculas/minúsculas de los identificadores! (Nombres de clases, objetos, funciones, etc.).

Comentarios en bloque generalmente consisten en uno o más párrafos compuestos por oraciones completas, por lo que cada una de ellas debe finalizar en un punto.

Los comentarios en bloque generalmente se aplican a algunos (o todos) códigos que los siguen, y están identados al mismo nivel que ese código. Cada línea de un comentario en bloque comienza con un # (numeral) y un espacio (a menos que esté identado dentro del mismo comentario).

Los párrafos dentro de un comentario en bloque están separados por una línea que contiene únicamente un # (numeral).

Un comentario en línea es aquel que se encuentra en la misma línea que una sentencia. Los comentarios en línea deberían estar separados por al menos dos espacios de la sentencia. Deberían empezar con un # (numeral) seguido de un espacio.

- Convenciones de nombramiento

Se utilizarán como estilos de nombramiento para variables, métodos y clases los

siguientes:

- *minúscula* (lowercase)
- *minúscula_con_guiones_bajos* (lower_case_with_underscores)
- *PalabrasConMayúscula* (CapWords o CamelCase).
- *minúsculaMayúscula* (mixedCase)

Los módulos deben tener un nombre corto y en minúscula. Guiones bajos pueden utilizarse si mejora la legibilidad (Python, 2013)

3.4. Patrón de arquitectura.

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

M, la porción de acceso a la base de datos, es manejada por la capa de la base de datos de Django, la cual describiremos en este capítulo.

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

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

Debido a que la "C" es manejada por el mismo framework y la parte más importante se produce en los modelos, las plantillas y las vistas, Django es conocido como un Framework MTV. En el patrón de diseño Model Template View(MTV) como se muestra en la Imagen 4.

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: como 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: puedes pensar en esto como un puente entre el modelo y las plantillas (Django 1.0, 2015).

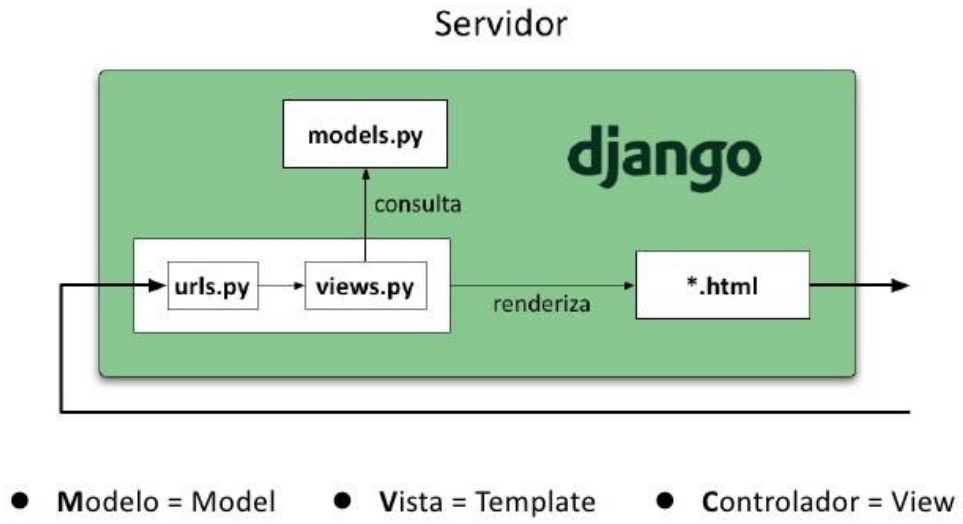


Imagen 4: Django: esquema interno

3.5. Fase de implementación.

En esta fase, se realiza un análisis de cada una de las historias de usuarios en conjunto con el plan de iteraciones y se modifican en caso de ser necesario. Luego las historias de usuario se descomponen en tareas de ingeniería y son asignadas a un grupo de desarrollo o a una persona como responsable de su implementación. Dichas tareas, al ser asignadas a programadores, se escriben en lenguaje técnico y no en un lenguaje entendible por el cliente. Las historias de usuario seleccionadas para ser implementadas en cada iteración se van realizando durante el transcurso de la iteración a la cual pertenecen. A continuación, se muestran las HU desarrolladas.

Tabla 14: Tarea de ingeniería autenticación

No. HU: 1	**No. Tarea:** 1
Nombre de la tarea: Autenticación	
Tipo de tarea: Implementación	**Putos estimados:** 1
Fecha inicio: 16-01-2017	**Fecha fin:** 20-01-2017
Programador responsable: Julién Pérez García y Raidel Placencia Rodríguez	
Descripción: El sistema debe permitir la autenticación de los usuarios autorizados a acceder al sistema	

Tabla 15: Tarea de ingeniería Gestionar Minions

No. HU: 2	**No. Tarea:** 1
Nombre de la tarea: Gestionar minions	
Tipo de tarea: Gestión	**Putos estimados:** 2
Fecha inicio: 23-01-2017	**Fecha fin:** 03-02-2017
Programador responsable: Julién Pérez García y Raidel Placencia Rodríguez	
Descripción: El sistema debe permitir la creación, modificación y eliminación de minions	

Tabla 16: Tarea de ingeniería gestionar ficheros de estado

No. HU: 3	**No. Tarea:** 1
Nombre de la tarea: Gestionar Ficheros de estado	
Tipo de tarea: Gestión	**Putos estimados:** 2
Fecha inicio: 20-02-2017	**Fecha fin:** 03-03-2017
Programador responsable: Julién Pérez García y Raidel Placencia Rodríguez	
Descripción: El sistema debe permitir la creación, modificación y eliminación de ficheros de estado	

Tabla 17: Tarea de ingeniería aplicar ficheros de estado

No. HU: 4	**No. Tarea:** 1
Nombre de la tarea: Aplicar Ficheros de estado	
Tipo de tarea: Implementación	**Putos estimados:** 1
Fecha inicio: 06-03-2017	**Fecha fin:** 10-03-2017
Programador responsable: Julién Pérez García y Raidel Placencia Rodríguez	
Descripción: El sistema debe permitir aplicar los ficheros de estado con una configuración especificada al servidor o grupo de servidores deseado	

Tabla 18: Tarea de ingeniería Integrar Interfaz de línea de comandos

No. HU: 5	**No. Tarea:** 1
Nombre de la tarea: Integrar interfaz de línea de comando	
Tipo de tarea: Gestión	**Putos estimados:** 2
Fecha inicio: 13-03-2017	**Fecha fin:** 24-03-2017
Programador responsable: Julién Pérez García y Raidel Placencia Rodríguez	

Descripción: El sistema lanza una terminal integrada a la aplicación.

Tabla 19: Tarea de ingeniería monitorear servidores

No. HU: 6	**No. Tarea:** 1
Nombre de la tarea: Monitorear servidores	
Tipo de tarea: Implementación	**Putos estimados:** 2
Fecha inicio: 27-03-2017	**Fecha fin:** 7-04-2017
Programador responsable: Julién Pérez García y Raidel Placencia Rodríguez	
Descripción: El sistema debe permitir monitorear servidores según un conjunto de parámetros definidos	

3.6. Pruebas de aceptación

Las pruebas de aceptación son pruebas de caja negra que se realizan a partir de las historias de usuarios. En ella se especifican la perspectiva del cliente y los escenarios para probar que la historia de usuario ha sido implementada correctamente. Una historia de usuario puede tener todas las pruebas de aceptación que desee para asegurar su funcionamiento. El objetivo específico de estas pruebas es garantizar que los requerimientos han sido cumplidos y que el sistema funcione correctamente (PINO, 2013).

A continuación, se muestran las pruebas realizadas:

Tabla 20: Prueba número 1 a la historia de usuario 1

Código: HU1-P1	**No. de HU:** 1
Nombre: Autenticación	
Descripción: Prueba para la funcionalidad de autenticación	
Condiciones de Ejecución:	
Pasos de ejecución: Se introduce el usuario y la contraseña correctos	
Resultado esperado: Se autoriza al usuario especificado y se muestra la página principal	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 21: Prueba número 2 a la historia de usuario 1

Código: HU1-P2	**No. de HU:** 1
Nombre: Autenticación	

Descripción: Prueba para la funcionalidad de autenticación
Condiciones de Ejecución:
Pasos de ejecución: Se introduce el usuario y la contraseña incorrectos
Resultado esperado: No se autoriza al usuario especificado y se muestra un mensaje indicando que el usuario o la contraseña son incorrectos.
Evaluación de la prueba: Prueba satisfactoria

Tabla 22: Prueba número 1 a la historia de usuario 2

Código: HU2-P1	**No. de HU:** 2
Nombre: Gestionar minions	
Descripción: Prueba para la funcionalidad de aceptar minions	
Condiciones de Ejecución:	
Pasos de ejecución: Se selecciona el botón aceptar en la correspondiente entrada de la tabla para uno de los minions no aceptados (Unacepted)	
Resultado esperado: El minion seleccionado es eliminado de la tabla actual y se muestra en la tabla de minions aceptados cambiando su estado a aceptado (Acepted)	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 23: Prueba número 2 a la historia de usuario 2

Código: HU2-P2	**No. de HU:** 2
Nombre: Gestionar minions	
Descripción: Prueba para la funcionalidad de rechazar minions	
Condiciones de Ejecución:	
Pasos de ejecución: Se selecciona el botón rechazar en la correspondiente entrada de la tabla para uno de los minions no aceptados (Unacepted)	
Resultado esperado: El minion seleccionado es eliminado de la tabla actual y se muestra en la tabla de minions rechazados cambiando su estado a rechazado (Rejected)	
Evaluación de la prueba: Se muestra el minion en la tabla de rechazados pero no se elimina el minion de la tabla de no aceptados	

Tabla 24: Prueba número 3 a la historia de usuario 2

Código: HU2-P3	**No. de HU:** 2
Nombre: Gestionar minions	

Descripción: Prueba para la funcionalidad de rechazar minions
Condiciones de Ejecución:
Pasos de ejecución: Se selecciona el botón rechazar en la correspondiente entrada de la tabla para uno de los minions no aceptados (Unacepted)
Resultado esperado: El minion seleccionado es eliminado de la tabla actual y se muestra en la tabla de minions rechazados cambiando su estado a rechazado (Rejected)
Evaluación de la prueba: Prueba satisfactoria

Tabla 25: Prueba número 4 a la historia de usuario 2

Código: HU2-P4	**No. de HU:** 2
Nombre: Gestionar minions	
Descripción: Prueba para la funcionalidad de denegar minions	
Condiciones de Ejecución:	
Pasos de ejecución: Se selecciona el botón denegar en la correspondiente entrada de la tabla para uno de los minions aceptados (Acepted)	
Resultado esperado: El minion seleccionado es eliminado de la tabla actual y se muestra en la tabla de minions denegados cambiando su estado a denegado (Denied)	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 26: Prueba número 5 a la historia de usuario 2

Código: HU2-P5	**No. de HU:** 2
Nombre: Gestionar minions	
Descripción: Prueba para la funcionalidad de mostrar detalles de los minions existentes	
Condiciones de Ejecución:	
Pasos de ejecución: Se selecciona el botón con el símbolo (+) en la correspondiente entrada de la tabla para uno de los minions	
Resultado esperado: Se despliega el contenido de la fila mostrando más detalles del minion seleccionado	
Evaluación de la prueba: El botón con el símbolo (+) no se muestra y los detalles aparecen fuera de lugar en la tabla	

Tabla 27: Prueba número 6 a la historia de usuario 2

Código: HU2-P6	**No. de HU:** 2
Nombre: : Gestionar minions	
Descripción: Prueba para la funcionalidad de mostrar detalles de los minions existentes	
Condiciones de Ejecución:	
Pasos de ejecución: Se selecciona el botón con el símbolo (+) en la correspondiente entrada de la tabla para uno de los minions	
Resultado esperado: Se despliega el contenido de la fila mostrando mas detalles del minion seleccionado	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 28: Prueba número 7 a la historia de usuario 2

Código: HU2-P7	**No. de HU:** 2
Nombre: : Gestionar minions	
Descripción: Prueba para la funcionalidad de mostrar minions en funcionamiento	
Condiciones de Ejecución:	
Pasos de ejecución: Se selecciona la pestaña Minions Up en la tabla	
Resultado esperado: Se muestra la información de los minions con estado en ejecución (Up)	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 29: Prueba número 8 a la historia de usuario 2

Código: HU2-P8	**No. de HU:** 2
Nombre: : Gestionar minions	
Descripción: Prueba para la funcionalidad de mostrar minions fuera de servicio	
Condiciones de Ejecución:	
Pasos de ejecución: Se selecciona la pestaña Minions Down en la tabla	
Resultado esperado: Se muestra la información de los minions con estado fuera de servicio (Down)	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 30: Prueba número 1 a la historia de usuario 3

Código: HU3-P1	**No. de HU:** 3
Nombre: Gestionar ficheros de estado	
Descripción: Prueba para la funcionalidad editar fichero de estado	
Condiciones de Ejecución:	
Pasos de ejecución: Se selecciona el botón editar fichero de estado	
Resultado esperado: Se muestra el contenido del fichero y se permite editarlo y guardar los cambios realizados	
Evaluación de la prueba: Se muestra el contenido pero no permite editarlo	

Tabla 31: Prueba número 2 a la historia de usuario 3

Código: HU3-P2	**No. de HU:** 3
Nombre: Gestionar ficheros de estado	
Descripción: Prueba para la funcionalidad editar fichero de estado	
Condiciones de Ejecución:	
Pasos de ejecución: Se selecciona el botón editar fichero de estado	
Resultado esperado: Se muestra el contenido del fichero y se permite editarlo y guardar los cambios realizados	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 32: Prueba número 1 a la historia de usuario 4

Código: HU4-P1	**No. de HU:** 4
Nombre: Aplicar ficheros de estado	
Descripción: Prueba para la funcionalidad aplicar fichero de estado	
Condiciones de Ejecución:	
Pasos de ejecución: Se selecciona el botón aplicar fichero de estado	
Resultado esperado: Se ejecuta el contenido del fichero aplicándose la configuración contenida en el mismo.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 33: Prueba número 1 a la historia de usuario 5

Código: HU5-P1	**No. de HU:** 5
Nombre: Integrar interfaz de línea de comandos	

Descripción: Prueba para la funcionalidad interfaz de línea de comandos
Condiciones de Ejecución:
Pasos de ejecución: Se selecciona la opción Terminal
Resultado esperado: Se muestra una nueva página desde la cual se accede a la terminal
Evaluación de la prueba: Prueba satisfactoria

Tabla 34: Prueba número 1 a la historia de usuario 6

Código: HU6-P1	**No. de HU:** 6
Nombre: Monitorear servidores	
Descripción: Prueba para la funcionalidad de mostrar detalles de jobs	
Condiciones de Ejecución:	
Pasos de ejecución: Se selecciona el botón con el símbolo (+) en la correspondiente entrada de la tabla para uno de los jobs	
Resultado esperado: Se despliega el contenido de la fila mostrando mas detalles del Job realizados	
Evaluación de la prueba: Prueba satisfactoria	

3.7. Prueba de Eficacia para la Eliminación de Defectos (EED)

Una métrica de la calidad que proporciona beneficios tanto a nivel del proyecto como del proceso, es la eficacia de la eliminación de defectos (EED) En particular el EED es una medida de la habilidad de filtrar las actividades de la garantía de calidad y de control al aplicarse a todas las actividades del marco de trabajo del proceso.

Cuando se toma en consideración globalmente para un proyecto, EED se define de la forma siguiente:

EED = E / (E + D)

donde E= es el número de errores encontrados antes de la entrega del software al usuario final y D= es el número de defectos encontrados después de la entrega. El valor ideal de EED es 1, donde simbolizando que no se han encontrado defectos en el software. De forma realista, *D* será mayor que cero, pero el valor de EED todavía se puede aproximar a 1 cuando *E* aumenta.

Se midieron los aspectos basados en un día laboral en el Nodo Central de la UCI, encontrándose un total de 5 errores durante el proceso de gestión de

configuración de los servidores. Luego de desplegada la aplicación, se repitió este proceso y se logró mitigar 4 de estos 5 errores, quedando un valor para EED de aproximadamente 0.83.

En una escala del 1 al 5.

Tabla 35: Distribución de puntos de eficacia respecto al valor de la variable EED

Rangos	Puntos
0-20%	1
21-40%	2
41-60%	3
61-80%	4
81-100%	5

Se determina que la aplicación mejoró la eficacia de la gestión de la configuración en el nodo central de la UCI con el valor más alto que puede tomar la escala.

3.8. Beneficios, Impacto y Pertinencia

La presente investigación responde al cumplimiento de lo plasmado en los lineamientos de la política económica y social del partido y la revolución.
Lineamiento 131: "Sostener y desarrollar los resultados alcanzados en el campo de la biotecnología, la producción médico-farmacéutica, la industria del software y el proceso de informatización de la sociedad, las ciencias básicas, las ciencias naturales, los estudios y el empleo de las fuentes de energía renovables, las tecnologías sociales y educativas, la transferencia tecnológica industrial, la producción de equipos de tecnología avanzada, la nanotecnología y los servicios científicos y tecnológicos de alto valor agregado"
Lineamiento 132: "Perfeccionar las condiciones organizativas, jurídicas e institucionales para establecer tipos de organización económica que garanticen la combinación de investigación científica e innovación tecnológica, desarrollo rápido y eficaz de nuevos productos y servicios, su producción eficiente con estándares de calidad apropiados y la gestión comercializadora interna y exportadora, que se revierta en un aporte a la sociedad y en estimular la reproducción del ciclo. Extender estos conceptos a la actividad científica de las universidades" (Cuba, 2011)

La contribución al aumento de la eficacia con que se realizan los procesos en la Dirección de redes y servicios telemáticos también va reflejada en el desarrollo de este trabajo. Esto se evidencia con la disminución de la cantidad de errores que se cometían en el proceso de gestión de la configuración así como con un aumento de la facilidad de uso del sistema gestor SaltStack. Gracias al sistema

propuesto será posible una mejor gestión y administración de los servicios que se brindan a la comunidad universitaria.

Conclusiones parciales

En este capítulo se elaboraron 5 tarjetas CRC donde se recogen los módulos más importantes del sistema. Se decidió utilizar el estándar de codificación de Python por ser el lenguaje de programación de Django. Se descompusieron las 6 HU en igual número de tareas de ingeniería de software para implementarlas. Se realizaron las pruebas de aceptación encontrándose 3 errores los cuales fueron erradicados.

Conclusiones Generales:

1. Los elementos teóricos-metodológicos identificados ofrecieron:

   - Características y funcionalidades relevantes de otros sistemas similares con las cuales debe contar el sistema desarrollado para en Nodo Central de la UCI.
   - Tecnologías, herramientas y metodologías más idóneas para el desarrollo de dicho sistema definiéndose XP como metodología de desarrollo; Visual Paradigm como herramienta de modelado, como lenguajes de programación Python y Django como marco de trabajo.

2. Se identificaron los siguientes elementos a desarrollar en el sistema para la Administración de la Gestión de la Configuración del Nodo Central de la UCI:

   - Autenticación utilizando el Directorio Activo y controlador de dominio
   - Gestión de Minions
   - Gestión de ficheros de estado
   - Integrar Interfaz de Línea de Comando
   - Aplicar ficheros de estado
   - Monitoreo de la gestión de la configuración.

3. Con la implementación del sistema para la Administración de la Gestión de la Configuración del Nodo Central de la UCI se logró:

   - Mejorar la gestión de la configuración del nodo central de la UCI de forma tal que esta es mucho más eficaz.
   - Disminuir la curva de aprendizaje del sistema de gestión de la configuración SaltStack brindando una interfaz desde la cual se pueden realizar los mismos procesos de forma más intuitiva.

4. Se realizaron 15 pruebas de aceptación y se erradicaron todos los errores detectados.

Recomendaciones

Una vez concluida la investigación y desarrollada la propuesta de solución los autores recomiendan implementar un analizador léxico para garantizar que los ficheros de estados mantengan la estructura YAML en la edición de los mismo.

Referencias Bibliográficas:

Aguascalientes, Universidad Autónoma de. 2010. *Métricas de Software.* 2010.

Alvarez, Miguel Angel. 2014. Desarrolloweb.com. [En línea] 2014. [Citado el: 2 de 5 de 2017.] https://desarrolloweb.com/articulos/que-es-mvc.html.

Carbó, Ing. Yosvany Medina. 2016. *Cuba y el impacto de las TIC en la informatización de la sociedad.* Consolación del Sur : s.n., 2016.

Castillo, Kenneth Díaz Siles y Marlon Ramírez. 2014. *Análisis de herramientas para la automatización de la administración de configuración de la infraestructura.* San José, Costa Rica : Urbanización Tournón, 2014.

—. 2014. *Análisis de Herramientas para la Automatización de la Administración de Configuración de la Infraestructura.* San José, Costa Rica : ULACIT, Urbanización Tournón, 2014. 10235-1000.

Coello, Sayda Coello González Y Rolando Alfredo Hernández León. 2012. *El PROCESO DE INVESTIGACIÓN CIENTIFICA 2da EDICIÓN.* La Habana : EDITORIAL UNIVERSITARIA, 2012.

Cuba, VI Congreso del Partido Comunista de. 2011. *Lineamientos de la Política Económica y Social del Partido y la Revolución.* 2011.

DE LUCA, D. N. 2011. *HTML5: entienda el cambio, aproveche su potencial. .* s.l. : USERSHOP, 2011.

Django 1.0, Fundation Software. 2015. *El libro de Django 1.0.* 2015.

Django, Software Foundation. 2015. *Django Documentation.* 2015.

Duque, Raúl González. 2008. *Python para todos.* s.l. : Creative Commons Reconocimiento, 2008.

FLANAGAN, D. 2002. *JavaScript: the definitive guide.* s.l. : O'Reilly Media, Inc., 2002.

J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres. 2014. *PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA.* Sevilla : University of Sevilla , 2014.

Jordi Pau i Cos, Ricardo de Navascués y Gasca, Marta Yubero Esteban. 1998. *Manual de logística integral.* s.l. : Dias De Santos, 1998.

PARADIGM, V. 2010. *Visual paradigm for uml Visual Paradigm for UML-UML tool for software application development.* 2010.

Pérez, Enrique Carbonell Muela y Ana María García. 2016. *COMPARACIÓN DE HERRAMIENTAS DE GESTIÓN DE LA CONFIGURACIÓN.* Habana : DESOFT, 2016.

pgAdmin. 2016. pgAdmin PostgreSQL Tools. *Lastest news.* [En línea] 27 de octubre de 2016. https://www.pgadmin.org/.

PINO, LINA DE LA CARIDAD LORET DE MOLA. 2013. *Portal web de la Comunidad Cubana de Realidad.* La Habana : s.n., 2013.

PostgreSQL, The PostgreSQL Global Development Group. 2016. *PostgreSQL 9.3.14 Documentation.* s.l. : The PostgreSQL Global Development Group, 2016.

Presman, Roger S. 2010. *Ingenieria del Software. Un Enfoque Practico.* New York : McGraw-Hill, 2010. 978-607-15-0314-5.

Python, Recursos. 2013. Guía de estilo para el código Python – PEP 8 en Español. [En línea] 2013. [Citado el: 2 de 5 de 2017.] http://recursospython.com/pep8es.pdf.

Rodriguez Hernández, Cinthya, y otros. 2014. Revista Cubana De Ciencias Informáticas. *GeReport: Sistema de Gestión de Reportes Dinámicos.* [En línea] 2014. [Citado el: 10 de enero de 2017.] http://rcci.uci.cu/?journal=rcci&page=article&op=view&path%5B%5D=726.

Rouse, Margaret. 2014. WhatIs. [En línea] enero de 2014. WhatIs.com.

Ruiz, Javier Heredia. 2014. *Comparación y tendencias entre metodologías ágiles y formales. .* s.l. : Serie Científica, 2014.

Salt. 2013. Blog de Digital Valley. [En línea] 3 de octubre de 2013. http://www.digitalvalley.com.

SaltStack. 2017. SaltStack. [En línea] SaltStack, 20 de 5 de 2017. https://saltstack.com/.

SL, Hipertextual. 2014. Hipertextual : La interseccion de la tecnología, la ciencia y las humanidades. *PyCharm, IDE de Python.* [En línea] 10 de junio de 2014.

Sommerville, Ian. 2005. *Ingeniería de software 7ma edición.* Madrid : Pearson Educación, 2005. 84-7829-074-5.

WHITE, S. A. 2004. *Introduction to BPMN.* s.l. : IBM Cooperation, 2004.

Bibliografía Consultada:

- **COELLO, SAYDA COELLO GONZÁLEZ Y ROLANDO ALFREDO HERNÁNDEZ LEÓN. 2012.** *El PROCESO DE INVESTIGACIÓN CIENTIFICA 2da EDICIÓN.* La Habana : EDITORIAL UNIVERSITARIA, 2012.
- **DJANGO, SOFTWARE FOUNDATION. 2015.** *Django Documentation.* 2015.
- **POSTGRESQL, THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. 2016.** *PostgreSQL 9.3.14 Documentation.* s.l. : The PostgreSQL Global Development Group, 2016.
- **PRESMAN, ROGER S. 2010.** *Ingenieria del Software. Un Enfoque Practico.* New York : McGraw-Hill, 2010.
- **SOMMERVILLE, IAN. 2005.** *Ingeniería de software 7ma edición.* Madrid : Pearson Educación, 2005.

Anexos:

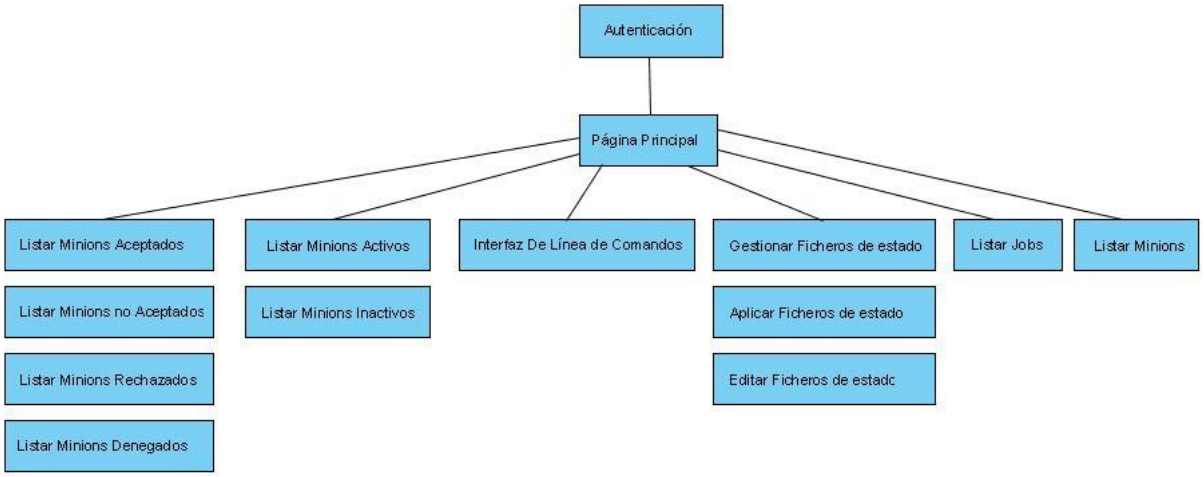


Imagen 5: Mapa de navegación de la aplicación