Universidad de las Ciencias Informáticas

Facultad 6

Sistema de Administración de Configuración de servidores de la producción de la Universidad de Ciencias Informáticas.

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

Autor:

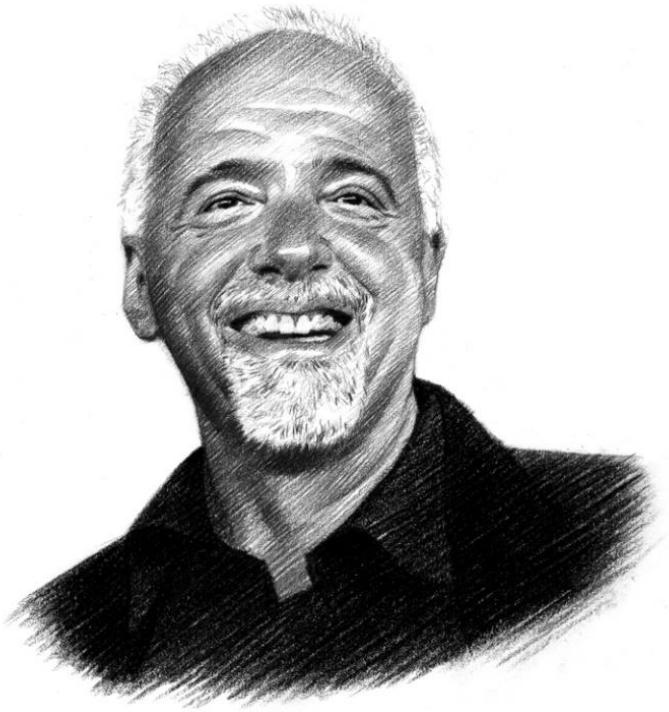
Yeneisy de la Torre Pérez.

Tutor:

Ing. Omar Pimentel Alfonso.

La Habana, Marzo 2015.

Año 57 de la Revolución.



¨El éxito no proviene del reconocimiento ajeno. Es el resultado de lo que sembraste con amor ¨

Paulo Coelho

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis que tiene por título: y reconocemos 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 ____ días del mes de _____ del año _____.

Yeneisy de la Torre Pérez

_____
Firma del Autor

Ing.Omar Pimentel Alfonzo.

_____
Firma del Tutor

DATOS DE CONTACTO

Datos del Autor:

Yeneisy de la Torre Pérez

Universidad de las Ciencias Informáticas, Cuba.

Email: ydperez@estudiantes.uci.cu

Datos del Tutor:

Ing. Omar Pimentel Alfonso. Graduado de Ingeniéro en Ciencias Informáticas en el año 2009.Pertenece al área de Redes y Servicios Telemáticos.

Universidad de la Ciencias Informáticas, La Habana , Cuba.

Email: opimentel@uci.cu

Agradecimientos

A mi papá por enseñarme a ser una mujer fuerte y guiarme.

A mi hermana por siempre animarme y darme fuerzas.

A mi pollo por aguantarme, amarme y mimarme siempre.

A mis primas por todo el cariño incondicional que me han dado.

A mi tia chacha por correr siempre conmigo y cuidarme tanto.

A mi tutor por nunca dudar de mi y siempre estar a mi lado en los buenos y malos momentos, Gracias.

A Matilde por aguantarnos y ayudarnos en cada momento de la tesis.

A mi Mary por estar al pendiende de cada paso que he dado.

A Soy por siempre aguantar mis malos tratos, y quereme de una forma muy loca.

A mis amigas de la vieja escuela por estar al pendiente en todo momento a Mary, Argelis,Yehimy y Zaily.

A mis nuevos amigos Juan, Orelvis, que me daban fuerzas en todo momento.

A mi mamá por darme la vida, enseñarme a vivirla, por quererme como soy, por apoyarme siempre, por ser mi motor impulsor. Te Amo.

A mis padres por ser los mejores, los más capaces, por guiarme, por educarme, por darme todo su amor.

A mi hermana hermosa por ser mí soporte para luchar por mis sueños.

A mi pollito por sus desvelos, sus carreras, su apoyo incondicional, y por soportar todo de mí.

A mi familia en general por estar conmigo siempre.

A mi tutor Omar por apoyarme desde el primer momento, por enseñarme cada día algo nuevo, y tenerme tanta paciencia.

A mis suegros, a mis tios postizos, mis amistades, y a todo aquel que me aprecia y de alguna forma u otra ha contribuido a lograr mi sueño, les estoy eternamente agradecida.

RESUMEN

La Administración de Configuración de Servidores se ha vuelto en la actualidad un recurso indispensable para empresas que brindan abundantes servicios informáticos. La presente investigación tiene como objetivo el desarrollo de un Sistema de Administración de Configuración de Servidores para la producción de la UCI, basado en una herramienta de administación llamada Salt. Para guiar el desarrollo del sistema se utilizó como metodología de desarrollo la Programación Extrema, como Marco de Trabajo Django y Bootstraps, utilizando a Python y JavaScript como lenguaje de programación. Los lenguajes de marcado utilizados son HTML5 Y CSS3. Se generaron los artefactos correspondientes a las fases de la metodología utilizada como: historias de usuario, plan de iteraciones, plan de duración de iteraciones, plan de entregas y tarjetas CRC. El Sistema de Administración de Configuración de los Servidores de la producción de la UCI, permite contribuir en la automatización de la administración de grandes grupos de servidores, permitiendo que el trabajo de los administradores sea mucho más rápido y eficiente.

PALABRAS CLAVE:

Administración de servidores, Servidores informáticos, Gestión de la Configuración, Administrador de la Configuración,Servidores, Configuración de servidores,Salt.

Abstract

Servers Configuration Management has become an indispensable resource for companies that provide abundant informatics services nowadays. This research aims to develop a Servers Configuration Management System for the production at the UCI; it is based on a tool called Salt Administration. For guiding the system development, Extreme Programming was used as software development methodology; Django and Bootstraps were used as frameworks and Python and JavaScript were used as programming languages, using HTML5 and CSS3 as markup languages. User stories, iteration plan, duration plan of iterations, delivering plan and CRC cards:

corresponding to the phases of the used methodology were generated as artifacts. Servers Configuration Management System for production at the UCI allows to contribute in automating the management of large groups of servers, allowing administrators to work much faster and efficient.

Key Words: Servers Administration, Informatics Servers, Configuration Management, Configuration Administrator, Servers, Servers Configuration, Salt

Índice de Contenido

Índice de Contenido:

Índice de Tablas y Figuras

INTRODUCCIÓN

El mundo de hoy está en constante cambio, sobre todo en las Tecnologías de la Información y las Comunicaciones (TIC), mediante las cuales la sociedad ha podido mirar hacia el futuro y desarrollarse aplicando las nuevas herramientas que van surgiendo con el transcurso de los años. Con el paso de los años son más las personas que utilizan la Internet. La difusión del empleo de las TIC, ha marcado un cambio significativo y de gran importancia en la sociedad. Con el paso del tiempo las personas y las empresas se hacen más dependientes de estas, siendo utilizadas en su beneficio y facilitando el trabajo diario.

En general empresas con un amplio número de miembros adoptan como práctica ofrecerle servicios a los mismos. Algunos de los servicios son correo, mensajería instantánea y publicación de información utilizando servidores centrales. En la actualidad existen dos factores que han dado lugar a un aumento en el número de servidores a administrar, el crecimiento de la información almacenada de manera virtual y del uso masivo de la computación. Incluso en pequeñas y medianas empresas, no es insólito tener una persona que administre entre cuatro o más servidores, números que podrian aumentar en dependencia de la cantidad de servicios que se ofrescan.

En la actualidad el crecimiento y expansión de los servicios informáticos ha llevado a cabo la utilización de diversos sistemas de gestión de administración de configuración de servidores. Buscando cada día nuevas formas de automatizar de manera mucho más eficiente y productiva el trabajo del personal encargado de administar los servidores. En empresas que brindan una gran variedad de servicios, han adoptado el uso de tales sistemas. Entre dichas empresas se encuentran: Cisco, Whatsapp, Avaya, Google, Microsoft, entre otras (Bécares, 2013). Sin quedar descartado hacerlo manualmente, ya que para algunos administradores les parece muchos más factible.

Introducción

En Cuba el proceso de gestión de administración de la configuración de servidores en la mayoría de los casos es gestionado manualmente y no se cuenta con registros históricos actualizados de los mismos. Existen problemas con el conocimiento de la información referente a la ubicación y detalles de los servicios y aplicaciones que se ejecutan en los servidores de los proyectos productivos. En la mayoria de las empresas cubanas, donde brindan a sus trabajadores servicios informáticos, los administradores usan la consola para crear sus propias soluciones proporcionando demora en el trabajo, estos deben tener un mayor conocimiento de comandos a ejecutar, ya sea para cambiar las propiedades del sistema del servidor e instalar o quitar funciones y sus características.

La Universidad de las Ciencias Informáticas (UCI) tiene entre sus misiones la producción de *software* y servicios informáticos a partir de la vinculación estudio-trabajo como modelo de formación. La misma cuenta con un gran número de personas, entre las que se encuentran especialistas, estudiantes y profesores. Los cuales interactuan a diario con diferentes servicios informáticos, entre los que se encuentran navegación, mensajería instantanéa y correo, estos se encuentrán alojados en servidores web, servidores de audio/video, servidores de correo, servidores de BD, entre otros.

La producción de la UCI está organizada por centros, en estos muchas veces cuentan con servidores de Base de datos, servidores de archivo, servidores web y otros servidores de despliegue de aplicaciones que tienen que ver con el software que se desarrolla en el centro. Los administradores de sistemas informáticos, plantean que el número de servidores en la infraestructura de la producción ronda los ciento ochenta. Los cuales en su mayoría son distintos en cuanto a su sistema operativo, recursos asignados, aplicaciones instaladas, versiones en aplicaciones, versiones en los sistemas operativos, entre otros. La UCI tiene varias personas dedicadas y capacitadas para la administración de servidores, realizando diariamente el mantenimiento y monitoreo de estos.

Introducción

Para la administración de servidores de producción se debe cumplir con el siguiente esquema, donde cada centro puede tener un administrador de configuración o varios, con limitadas funciones de administración de sus servidores. En la dirección de proyecto existe un administrador de configuración con acceso a todos los servidores de los centros, el cual tiene entre sus responsabilidades la de gestionar la configuración de los mismos. Por el nodo central existe un administrador de red que se dedica a la administración de estos servidores también.

Los administradores de los servidores para realizar su trabajo diario, suelen apoyarse en pequeños programas o *scripts* que ejecutan determinadas funciones, entre las que se encuentran: la configuración, el monitoreo, el mantenimiento y el despliegue de aplicaciones, entre otros. En entornos como la UCI donde se cuenta con una amplia variedad de sistemas y servicios informáticos, los administradores deben tener un amplio conocimiento de los comandos y/o *script* que ejecutarán en cada servidor. Actualmente la administración de configuración de servidores se realiza de uno en uno, es decir, realizando la misma operación manualmente en cada uno de los servidores. Este trabajo de los administradores podría resultar engorroso y repetitivo si necesitase administrar un gran grupo de servidores.

Además, los servidores de la producción pueden tener diferentes características como por ejemplo: sistemas operativos heterogéneos, diferente versión, diferentes aplicaciones instaladas, entre otros. La manera de realizar una determinada acción sobre los servidores podrían variar en dependencia de las características de los mismos. Esto implicaría mantener un registro de los datos de cada servidor o preguntar cada vez que sea necesario realizar una acción. Provocando que el trabajo de los administradores sea más lento.

La situación actual de la administración de los servidores de la producción de la UCI, descrita anteriormente, permite plantear el siguiente **problema a resolver:** ¿Cómo contribuir a la automatización de la configuración de servidores de producción de la UCI?

Introducción

Para la presente investigación se plantea como **objeto de estudio:** Los Sistemas de Administración de Configuración de Servidores; enmarcado en el **campo de acción:** Los Sistemas de Administración de Configuración de los Servidores de producción de la UCI.

Dada la necesidad de contribuir a la administración de la configuración en los servidores de la producción de la UCI se plantea como **objetivo general:** Desarrollar un Sistema para la Configuración de Servidores de la producción de la UCI.

Para guiar la investigación se definen las siguientes **preguntas científicas:**

- ¿Cuáles son los fundamentos teóricos del Sistema de Administración de Configuración de los Servidores de la Produción de la UCI?
- ¿Cuáles son las características o funcionalidades que debe tener el Sistema de Administración de Configuración de servidores de la producción de la UCI?
- ¿ Cómo validar el correcto funcionamiento del Sistema de Administración de Configuración de los servidores de la producción de la UCI?
- ¿El sistema desarrollado contribuye a la automatización de la administración de configuración de los servidores de producción de la UCI?

Para cumplir el objetivo propuesto se establecieron las siguientes **tareas de la investigación:**

- Redacción del diseño teórico de la investigación para identificar el campo de acción sobre el cual estará enmarcada la investigación.

- Elaboración del estudio del estado del arte para identificar conceptos, característcas, sistemas asociados a la configuración de servidores.
- Construcción del análisis y el diseño para estudiar las diferentes funcionalidades del Sistema de Administración de Configuración de los Servidores de la producción de la UCI.
- Implementación del sistema para la Administración de Configuración de Servidores.
- Comprobar el correcto funcionamiento utilizando las pruebas de rendimiento y las unitarias a nivel de sistema al software para validarlo.

Métodos de investigación:

Histórico-Lógico: Se utiliza en la investigación para determinar los antecedentes históricos relacionados con los sistemas de administración de configuración de servidores, posibilitando el análisis de la trayectoria de estos sistemas.

Analítico-Sintético: Permite el estudio de diferentes fuentes bibliográficas para extraer los elementos más importantes que se relacionen con la configuración de servidores. Se realizaron además resúmenes y valoraciones de conceptos relevantes relacionados con la administración de la configuración de servidores.

Modelación: Se utiliza para representar por medio de diagramas el proceso del sistema de administración de la configuración de servidores, teniendo como resultado un mejor entendimiento de la posible solución a implementar.

Análisis Documental: Se utiliza en la investigación para seleccionar las ideas informativamente relevantes de un documento, a fin de expresar su contenido sin ambigüedades para recuperar la información en él contenida.

Introducción

Caso de estudio: Usado a la hora de tomar como punto de partida para el desarrollo de la solución, del Sistema de Administración de la Configuración. Permitió obtener datos que determinaron la efectividad de la herramienta.

Posibles resultados:

Como posibles resultados se definen:

- Documentación sobre sistemas de administración de configuración de servidores.
- Un Sistema para la Gestión de la Configuración de Servidores de la producción de la UCI que contribuirá al trabajo de los administradores, permitiendo trabajar con una interfaz web sencilla y eficiente, donde podra administrar muchos servidores a la vez.

Estructura del Documento

El documento se encuentra dividido en 3 capítulos, en el primero titulado “Fundamentos Teóricos del Sistemas para la Administracíón de Servidores” se realiza la elaboración del marco teórico donde se exponen los conceptos asociados a la solución de la problemática. De igual manera se describen y caracterizan las herramientas y el lenguaje de programación para el diseño e implementación de la solución, así como la metodología de desarrollo más recomendable. Seguidamente el capítulo 2 que lleva por titulo “Análisis y Diseño del Sistema para la Administración de Servidores” se definen las principales características que debe cumplir el sistema a desarrollar, es decir, se precisan las funcionalidades y características que debe cumplir, así como el diseño y la estructuración del mismo. Se generan además los artefactos que propone la metodología escogida. Concluyendo la investigación en el capítulo 3 titulado “Implementación y Prueba del Sistema para la Administración de Servidores” se definen los patrones

utilizados. Se implementan las funcionalidades identificadas a través de las tareas de la ingeniería describiéndose cada una de ellas. Se detallan las pruebas realizadas al *software* con el objetivo de asegurar la calidad y eficiencia de la solución.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS DE SISTEMAS PARA LA ADMINISTRACIÓN DE SERVIDORES

1.1 Introducción

En el presente capítulo se investigan definiciones relacionadas con los sistemas de administración de configuración de servidores, así como sus características, ventajas y desventajas. Además se hace un estudio de los sistemas de administración de configuración. Se analizan las herramientas, tecnologías, metodologías de desarrollo y los lenguajes de programación a utilizar para la realización de la aplicación, que pudieran darle cumplimiento al objetivo general de la investigación.

1.2 Gestión de la Configuración

La Gestión de la Configuración se basa en llevar el control de todos los elementos de configuración, en la infraestructura de la TI y asegurar la calidad de todo producto. Además proporciona información precisa y fiable. La gestión de la configuración se realiza durante todas las fases del desarrollo de un sistema de información, incluyendo el mantenimiento y control de cambios, una vez realizada la puesta en producción. Estas configuraciones actualmente se realizan de manera manual en la mayoría de las instituciones, por los administradores de sistemas informáticos, por lo que se ven obligados a buscar diferentes formas de automatizar su trabajo, una de estas maneras son los sistemas de gestión de la administración de la configuración (Escobar, 2009).

1.3 Sistemas de Gestión de la Administración de Configuración

Los Sistemas de Administración de Configuración, son sistemas capaces de administrar y configurar muchos servidores a la vez. En la actualidad debido a la propagación de información en las computadoras ha habido un aumento exesivo de servicios informáticos, que ha llevado a cabo un despunte significativo en los servidores que necesitan ser administrados. En muchos casos se administran grandes grupos de servidores similares, con funciones de aplicaciones y servicios similares. La capacidad de los administradores de sistemas de incorporar a su trabajo diario formas de automatización es el único modo de manejar grandes y cada vez más crecientes infraestructuras (Venezia, 2013). Sistemas de

Administración de Configuración como Puppet, Chef, Ansible y Salt fueron creados con el objetivo de hacer fácil la configuración y el mantenimiento de muchos servidores. También es aplicable en organizaciones más pequeñas donde estas herramientas favorezcan la automatización y orquestación en general, logrando con esto que el trabajo diario de los administradores sea más eficaz.

1.4 Análisis de los Sistemas de gestión de Configuración de Servidores

Se han seleccionado como base para el estudio los Sistemas de Gestión de la Configuración Puppet, Chef, Ansible y Salt ya que son utilizados en empresas como , Google, Microsoft, (Bécares, 2013), son unos de los sistemas más potentes que se han desarrollado actualmente, además son de código abierto, multiplataformas, permite ejecutarse en multiples servidores y permiten mantener un control de versiones de la configuración, son los más adecuados para automatizar las tareas del administrador del servidores.

1.4.1 ***Puppet***

Puppet abarca casi todos los sistemas operativos. Su configuración inicial requiere la instalación de un servidor maestro y agentes clientes en cada sistema que se va a administrar. Todos los módulos y configuraciones de Puppet están construidas con un lenguaje específico basado en Ruby, y por lo tanto se requieren conocimientos de programación, además de habilidades de administración del sistema. Puppet tiene una interfaz de usuario web que aún está disponible en forma limitada para la versión libre (Portolani, 2014). Una de las desventajas de Puppet, es que requiere de aprendizaje del lenguaje de programación Ruby. Además de que le resulta muy complicado el traseo de los errores de configuración o en el desarrollo de los módulos.

1.4.2 ***Chef***

Chef cuenta con un servidor maestro y agentes instalados en los nodos administrativos. Su configuración gira en torno a un Sistema de Control de Versiones, por lo que el conocimiento de sus funciones es un requisito previo para el funcionamiento de Chef. Sus módulos se escriben en Ruby. Chef no tiene una aplicación web, aunque hay una versión beta disponible. La interfaz web de Chef es funcional pero no brinda la capacidad de modificación de configuraciones, tiene carencia de memoria y otras, aunque si permite el control de inventario y la organización de los nodos. Chef tiene disponibilidad de un gran número de módulos y recetas de configuración, pero estas dependen en gran medida del conocimiento que tenga el

administrador de Ruby (Jaynes, 2014). Teniendo como desventaja carencia de comandos de entrada y escasa documentación.

1.4.3 **Ansible**

La perspectiva de Ansible es ser simplificado y rápido. Ansible no requiere instalación en el nodo. Realiza todas las funciones a través de Intérprete de Órdenes Segura (SSH, por sus siglas en inglés) y está programado sobre Python. Tiene una interfaz web aunque esta no vincula directamente a la Interfaz de Línea de Comando (CLI, por sus siglas en inglés), esto significa que los elementos de configuración presentes en la CLI no aparecerán en la interfaz web. La interfaz web es funcional, pero no es tan completa como la CLI, por lo que se encontrará trabajando entre los dos en el uso general, o simplemente utilizando la CLI (Venezia, 2013). Una de las desventajas de Ansible es que carece de soporte para los clientes de *Windows*. Y su interfaz web tiene limitaciones ya que no vincula automáticamente la CLI.

1.4.4 **Salt**

Salt es una herramienta basada en CLI. Es un sistema de gestión de la configuración, capaz de mantener los nodos remotos en estados definidos. Fue desarrollado con el fin de aportar una mejor solución a la ejecución remota. Tiene una gran capacidad para el manejo de grandes cargas de información. Proporciona versatilidad en despliegues a grandes escalas. Salt es muy fácil de instalar y mantener, su arquitectura está diseñada para trabajar con cualquier número de servidores, desde un puñado de sistemas de redes locales hasta despliegues internacionales a través de diferentes centros de datos. Su topología es simple, modelo cliente / servidor. Las rutinas de ejecución de Salt están escritas como módulos de Python. Salt puede ser llamado desde una simple Interfaz de Programación de Aplicaciones (API, por sus siglas en inglés) de Python, o desde la CLI. Su resultado es un sistema que puede ejecutar comandos a alta velocidad, en grupos de servidores de destino. Salt es muy rápido, fácil de configurar, e increíblemente flexible y proporciona una arquitectura de ejecución remota única, que puede manejar las numerosas necesidades de cualquier cantidad de servidores. La infraestructura de Salt reúne ejecución remota, amplifica sus capacidades y extiende su gama, lo que resulta en un sistema que es tan versátil como es de práctico (Jaynes, 2014).

Salt puede comunicarse con los clientes a través de SSH o mediante un agente local instalado. Salt también introduce controles más granulares para las ejecuciones remotas, lo que permite que los sistemas se dirijan no sólo por nombre de host, sino también por las propiedades del sistema (Venezia, 2013).

El Sistema de Gestion de la Administración Salt utiliza un servidor master que será el servidor principal para controlar y comunicarse con los servidores de destino llamados minions. Incluso puede configurar varios niveles de master, lo que resulta en una disposición escalonada de carga compartida y una mayor redundancia. Salt permite administrar cualquier cantidad de servidores minions, a una mayor velocidad de respuesta.

1.4.5 ***Fundamentos de la elección***

De los sistemas estudiados el más idoneo para automatizar el proceso de Administración de la Configuración de servidores de la producción de la UCI es la herramienta Salt. Debido a que esta es capaz de administrar muchos servidores a la vez a mayor velocidad, ya sea automatizando las secuencias de comandos en varios servidores, el despliegue de aplicaciones y la instalación.Además minimíza la monotonía de configuración manual del servidor y ofrece la creación de un gran número de módulos para hacer frente a un *software* específico. Sus módulos pueden ser escritos en Python. Salt se centra principalmente en la administración de servidores Linux, Unix y Centos, aunque también ofrece administración de Windows. Mientras que Salt puede parecer simple a primera vista, es sorprendentemente potente y extensible, y se ha diseñado para manejar un gran número de clientes. La herramienta Salt tiene su propia interfaz , pero esta realiza sus ejecuciones a travéz de la terminal, por lo que resulta engorroso para los administradores realizar su trabajo , ya que se ven obligados a tener un amplio conocimiento de los comandos a ejecutar. Como Salt no cumple con todas las espectativas para lograr el objetio de la investigación se dedice utilizar Salt como base para el desarrollo de un Sistema de Administración de la Configuración de servidores de la producción de la UCI, creando una aplicación web que permita al administrador de manera visual configurar grandes cantidades de servidores.

1.5 Resultados obtenidos en el trabajo precedente

El Sistema de Administración de Configuración de servidores de la producción de la UCI que estará basado en la herramienta Salt podrá garantizar la automatización de la configuración de los servidores y optimizará

el trabajo diario de los administradores de servidores. El sistema propuesto favorece la administración de los servidores, puesto que hoy en día es complicado y se necesita de conocimiento avanzado para administrar un servidor, este sistema va a permitir con solo dar clic, administrar cientos de servidores a la vez a mayor velocidad, de una manera más amena y eficaz.

Para el funcionamiento del sistema se debe instalar en el servidor principal que será el servidor master y en las pc clientes los servidores minions, este servidor master sera el encargado de proveerle servicios y dar órdenes a estos servidores minions. Esta sería una interfaz visual,a los administradores les será sencillo realizar su trabajo diario, ya que podrán instalar, configurar, monitoriar mediante esta, sin la necesidad de utilizar la terminal en todo momento y tener que memorizar grandes cargas de comandos, además de ser un sistema multiplataforma. Brindará la capacidad de tener una lista de tareas donde el administrador podrá ver todas las funcionalidades que han estado ejecutando hasta el momento. Y para aquel administrador con conocimientos avanazados podra utilizar la terminal.

1.6 Metodología de desarrollo

Las metodologías de desarrollo se clasifican en tradicionales y ágiles. Entre las principales metodologías tradicionales están el Proceso de Desarrollo Unificado (RUP, por sus siglas en inglés) y Marco de trabajo para Soluciones de Marco de Trabajo de Soluciones de Microsoft (MSF, por sus siglas en inglés), estas se recomiendan en proyectos complejos y de larga duración, obteniendo resultados satisfactorios cuando el equipo de trabajo tiene experiencia en su aplicación. Estas características hacen engorroso el desarrollo de proyectos pequeños, haciendo lenta su implementación y generando documentación que no será utilizada.

Como se quiere realizar una aplicación en un corto plazo de tiempo y el equipo de desarrollo es pequeño, quedan descartadas las metodologías robustas, ya que estas se centran en proyectos complejos de larga duración. Entre las metodologías ágiles se destacan SCRUM, Crystal y Programación Extrema (XP, por sus siglas en inglés). Estas hacen menos énfasis en la documentación, puesto que el funcionamiento y el desarrollo del *software* son más importantes que la misma. La regla a seguir es no realizar documentos a menos que sean necesarios de forma inmediata, el cliente es muy importante el cual pasa a formar parte del equipo de desarrollo y también son muy flexibles ante los cambios y no siguen estrictamente un plan (Letelier, 2006).

1.6.1 ***SCRUM***

La Metodología SCRUM define un marco para la gestión de proyectos. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos, el ciclo de vida está dividido en tres fases: planificación del *sprint*, seguimiento del *sprint* y revisión del *sprint*. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. La segunda característica importante se refiere a las reuniones a lo largo del proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (Palacios, 2008).

1.6.2 ***Crystal***

Se trata de un conjunto de metodologías para el desarrollo de *software* caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. El desarrollo de *software* se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas (Letelier, 2006).

1.6.3 ***XP***

La Programación Extrema es una metodología de desarrollo ligero (o ágil) basada en una serie de valores y prácticas de buenas maneras que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas. Este modelo de programación da prioridad a los trabajos que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación.

Se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

Consiste en un conjunto de prácticas que a lo largo de los años han demostrado ser las mejores prácticas de desarrollo de software, llevadas al extremo y fundamentadas, en un conjunto de valores (Kenneth, 2005).

A continuación se presentan algunas características de XP que se adaptan a las necesidades de un proyecto, así como a las condiciones de trabajo:

- Desarrollo iterativo e incremental: Pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas: Frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- Simplicidad en el código: la programación extrema apuesta que es más sencillo hacer un código simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar funciones complicadas y quizás nunca utilizarlas.

1.6.4 ***Fundamentos de la elección****.*

Se selecciona la metodología XP para el desarrollo del debido a que se trata de un proyecto pequeño, donde todo el trabajo se realiza por un programador, no necesita la generación de tantos artefactos, los requisitos pueden cambiar con el tiempo a medida que avanza el trabajo, el intercambio de opiniones con el cliente juega un papel fundamental a lo largo de todo el proceso de desarrollo de software por lo que está integrado al proyecto y se emplea también con el propósito de alcanzar un producto que satisfaga las necesidades del cliente en el menor tiempo posible y con la calidad requerida.

1.7 Tipos de Aplicación

Teniendo en cuenta que se pretende desarrollar un Sistema de Administración de Configuración, y que estos se basan en aplicaciones web y de escritorios, por lo que se estudiarán estos dos tipos de aplicaciones, para luego poder seleccionar, uno de ellos.

1.7.1 ***Aplicación de Escritorio***

Una aplicación de Escritorio (también llamada *Desktop*) es aquella que está instalada y configurada en el ordenador de un usuario y es ejecutada directamente por su sistema operativo.

Las aplicaciones de escritorio no se conectan a un servidor web, se pueden adaptar a las necesidades de una empresa y su rendimiento depende de diversas configuraciones de hardware como memoria RAM o capacidad del disco duro (Niño, 2010).

Una ventaja fundamental de la aplicación de escritorio es que su ejecución no necesita comunicarse con un servidor web, sino que se realiza de forma local. Esto repercute en mayor velocidad de procesamiento y por tanto en mayores capacidades a la hora de programar herramientas más complicadas o funcionales. Además que el tiempo de respuesta es rápido y suelen ser muy seguras. Mientras que una de las desventajas es que son dependientes del sistema operativo en que están instaladas, además que sus instalaciones y actualizaciones se realizan de forma personalizada.

1.7.2 ***Aplicaciones Web***

Se denomina aplicación web a aquellas aplicaciones que los usuarios utilizan accediendo a un servidor web a través de Internet o de una Intranet mediante un navegador web. Las aplicaciones web se caracterizan por dar soluciones puntuales expresadas en un *software* navegable por lo que los usuarios acceden a la aplicación sin la necesidad de su instalación (Martin, 2014).

Las aplicaciones web tiene con ventaja que es mutliplataformas, su acceso es inmediato y se puede acceder desde cualquier lugar. Pueden ser accedidas desde cualquier ordenador conectado a la red. Son fáciles de actualizar y mantener. No necesitan tantos requerimientos de hardware para su funcionamiento

Pero si es impresindible la conexión a una red, para la comunicación constante con el servidor que ejecuta la aplicación. Además, el servidor debe tener las prestaciones necesarias para ejecutar la aplicación de manera fluida, no solo para un usuario sino para todos los que la utilicen de forma concurrente.

1.7.3 ***Fundamentos de la elección***

Teniendo en cuenta que para que el sistema cumpla con sus funcionalidades correctamente se seleccionó el uso de la aplicación web, debido a que la misma brinda la posibilidad de poder acceder a ella desde cualquier ordenador sin importar en que lugar del mundo estés, simplemente debe estar conectado a una red. De esta manera el administrador podrá gestionar los servicios que se encuentren en cualquier lugar.

1.8 Marco de Trabajo

Un marco de trabajo normalmente integra componentes variados para desarrollo de aplicaciones, pero dependen del lenguaje y ambiente de desarrollo. Está orientado a la reutilización de componentes permitiendo el desarrollo rápido de aplicaciones. Puesto que la herramienta Salt en la que se basó el Sistema esta escrito en Python, el marco de trabajo que se utiliza es Django. Tambien se utiliza como marco de trabajo Bootstrap.

1.8.1 ***Django 1.6***

Django es un marco de trabajo web de código abierto escrito en Python que permite construir aplicaciones web de forma rápida y con menos código (15). Usa una modificación de la arquitectura Modelo-Vista-Controlador (MVC), llamada MTV (Model – Template – View), que sería Modelo-Plantilla-Vista.

Se utilizará para la implementación del subsistema el marco de trabajo Django 1.6, esta nueva versión aparte de presentar las propiedades de las versiones anteriores, incluye nuevas características que permiten mejorar el desarrollo en aplicaciones mediante su uso, entre ellas la mejora de la gestión de transacciones, simplificación y modernización de plantillas de proyectos y aplicaciones predeterminadas. Además este marco de trabajo es compatible con diversos gestores de datos (Bennett, 2009).

1.8.2 ***Bootstrap 3.3.2***

Bootstrap es un marco de trabajo que simplifica el proceso de creación de diseños web combinando CSS y JavaScript. Es un marco de trabajo que incluye numerosos componentes web que permiten ahorrar esfuerzo y tiempo en el desarrollo de aplicaciones. Bootstrap se divide en (Davison, 2010):

✓ **Scaffolding:** estructura con estilos globales, variables, además utiliza elementos HTML y propiedades CSS.

✓ **CSS:** dispone de un conjunto de clases para aplicar tanto al formateo de código como a las tablas, formularios, botones, tipografía e incluso íconos.

✓ **Componentes:** para ofrecer mayor interactividad al sitio cuenta con componentes como grupos de botones, alertas, navegación y tipografía.

- ✓ ***Plugins* de JavaScript:** brinda la posibilidad de crear *plugins* de JQuery personalizados.

En el desarrollo de la aplicación se empleará Boostrap 3.0 para la creación de las interfaces, proporcionando un diseño sólido basado en estándares como CSS3 y HTML5 que también serán usados en la solución.

1.9 Lenguaje de programación

Un lenguaje de programación permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador, para que este pueda comunicarse con los dispositivos de hardware y software existentes. Teniendo en cuenta que el Sistema de Administración de Configuración que se ha utilizado, sus módulos se escriben Python 2.7, se ha seleccionado el mismo como lenguaje del lado del servidor y HTML5, CSS3 y JavaScript como lenguajes del lado del cliente. A continuación una breve descripción de estos lenguajes.

1.9.1 ***Python 2.7***

Python es un lenguaje de programación que soporta múltiples paradigmas, incluyendo programación orientada a objetos, programación imperativa y funcional. El código Python define objetos incorporados como listas enlazadas, tuplas, tablas hash y enteros de longitud arbitraria. Debido al soporte que brinda y su integración con otros lenguajes y herramientas, se utiliza para la solución propuesta en conjunto con el marco de trabajo Django a Python en su versión 2.7 ya que el mismo permite la integración de numerosas librerías que facilitarán la implementación de algunas funcionalidades del subsistema (Montoro, 2012).

1.9.2 ***JavaScript 1.5***

JavaScript es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web. Permite crear diferentes efectos e interactuar con los usuarios, gran parte de la programación en este lenguaje está centrada en describir objetos y escribir funciones que respondan a utilización de teclas, aperturas y cargas de páginas.

Se utilizará este lenguaje en el desarrollo del software para que exista mayor interacción en la aplicación con respecto a los formularios y dinamismo ante la respuesta a los eventos que se manejarán, también se empleará para validar los datos de entrada en las interfaces del subsistema (Gauchat, 2012).

1.9.3 ***HTML5***

HTML5 es la última evolución de la norma que define el lenguaje de marcas de hipertexto (HTML por sus siglas en inglés). Se trata de una nueva versión del lenguaje, con nuevos elementos, atributos y comportamientos, y un conjunto amplio de tecnologías que permite crear sitios web y las aplicaciones más diversas y de gran alcance (Alvarez, 2009). Este nuevo estándar de HTML permite que el formato de las webs formado por elementos como cabecera, pie y navegadores se agrupe en nuevas etiquetas que representan cada una de las partes típicas de una página. Se utilizará el lenguaje HTML5 para el diseño de las interfaces del subsistema, aprovechando las características del soporte para CSS3 y el manejo mejorado de formularios en el navegador (Aubry, 2012).

1.9.4 ***Hojas de estilo en cascada (CSS 3)***

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. Además de que separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo. Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

El lenguaje CSS se utiliza para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, entre otros (Perez J, 2009).

1.9.5 ***Lenguaje Unificado de Modelado (UML)***

El lenguaje unificado para la construcción de modelos se define como un leguaje que permite especificar, visualizar y construir los artefactos de los sistemas informáticos. Está destinado a los sistemas de modelado

que utilizan conceptos orientados a objetos .Este lenguaje permitirá el diseño de los diagramas de clases persistentes y de paquetes del sistema propuesto.

1.9.6 ***Herramienta CASE***

Las herramientas CASE (Computer Aided Software Engineering por sus siglas en inglés) de modelado con UML (Lenguaje Unificado de Modelado) permiten aplicar la metodología de análisis y diseño orientado a objeto y abstraerse del código fuente, en un nivel donde la arquitectura y el diseño se tornan más obvios y más fáciles de entender y modificar.

1.9.7 ***Visual Paradigm 5.0***

Visual Paradigm es una herramienta de modelado que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objeto, construcción, pruebas y despliegue. Propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y diseño. Permite crear todos los tipos de diagramas de clases, realizar ingeniería inversa, generar documentación y código desde diagramas.

En este trabajo se emplea la herramienta Visual Paradigm en su versión 5.0, porque permitirá la modelación del Modelo de Dominio, en el cual se reprensentará de manera gráfica el problema que existe actualmente.

1.10 Servidores web

Un servidor web no es más que un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet. El servidor web se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo de acuerdo a los comandos solicitados (Abreu, 2011). Para la creación de la aplicación se realizó un estudio de los servidores web Nginx y Apache. A continuacón se muestra el estudio realizado.

1.10.1 ***Apache 2.2.22***

Apache es un servidor web de Protocolo de Transferencia de Hipertexto *(*HTTP, por sus siglas en inglés) de código abierto, para plataformas Unix (Distribución de *Software* Berkeley (BSD, por sus siglas en inglés), GNU/Linux), *Microsoft Windows, Macintosh* y otras, que implementa el protocolo HTTP/1.1 y la noción de

sitio virtual. Apache es usado principalmente para enviar páginas web estáticas y dinámicas en la *World Wide Web*. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a *Apache*

Apache dentro de Internet es capaz de producir aplicaciones de calidad profesional difíciles de igualar. Funciona en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal. Apache es una tecnología gratuita, de código fuente abierta. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia al *software* de manera, que permite ver qué es lo que se está instalando como servidor (Ciberaula, 2013).

1.10.2 ***Nginx 1.4.4***

Nginx es un servidor web de proxy inverso ligero de alto rendimiento, es de código abierto y posee licencia BSD simplificada. Posee características importantes como:

Multiplataforma**:** Inicialmente se creó para funcionar en sistemas operativos Unix, pero más tarde también apareció una versión compatible con Windows, por tanto se utiliza en sistemas como GNU/Linux, FreeBSD, *Solaris, Mac OS X y Windows* (Dar, 2013).

Se caracteriza además por otros aspectos como:

- Incluye servicios de correo electrónico con acceso al Protocolo de Red de Acceso a Mensajes (IMAP, por sus siglas en inglés*)* y al servidor (*POP,* por sus siglas en inglés)
- Está listo para ser utilizado como un proxy inverso con opciones de caché.
- Tiene soporte para Capa de Conexión Segura (SSL, por sus siglas en inglés), *streaming* de vídeo, autenticación y balanceo de carga.
- Manejo de archivos estáticos, archivos de índices y auto indexado.
- Posee módulo de reescritura de Localizador de Recursos Uniforme (URL, por sus siglas en inglés).

1.10.3 ***Fundamentos de la elección***

Se ha seleccionado Nginx como servidor web puesto que actualmente es el servidor que se está utilizando en la producción de la UCI. Es un servidor HTTP, proxy en reversa, balanceador de carga, basado en el concepto de código abierto. Enfocado en una arquitectura de manejo de eventos asincrónicos en lugar de hilos, es reconocido por su alto desempeño, bajo consumo de recursos, estabilidad, escalabilidad y configuración sencilla. Este servidor web al manejar los requerimientos basados en eventos permitirá al software consumir bajos recursos y asegurar un funcionamiento óptimo bajo mucha carga.

1.11 Sistemas Gestores de Base de Datos

Un sistema gestor de base de datos (SGBD) es un conjunto de programas que permite crear y mantener una base de datos, realizar todas las tareas de administración necesarias para mantenerlas operativas, mantener su integridad, confidencialidad y seguridad (Ramos, 2008).

La UCI en el proceso de desarrollo de sofware esta optando por el uso de aplicaciones de codigo abierto, por lo tanto a la hora de seleccionar un Gestor de Base de Datos, por lo tanto se han seleccionado para el estudio SQLite y PostgreSQL.

1.11.1 ***SQLite 3.8.2***

SQLite es un sistema de gestión de datos muy ligero y potente, es un proyecto de dominio público. A diferencia de otros sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En vez de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Por sus características se utiliza en una gran variedad de aplicaciones, Mozilla *Firefox*, *Adobe Photoshop Elements*, el navegador web Opera (Kreibich, 2010).

1.11.2 ***PostgreSQL 9.1***

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Desde su creación, hace más de 16 años la estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. Es de código abierto, al igual que todo el *software* libre. Cuenta con dos ventajas claras: un código fuente optimizado que puede ser modificado y adaptado, y una baja inversión por implementación, ya que no existen costos por licencia (Obe, 2012).

1.11.3 ***Fundamentos de la elección***

Luego de haber analizado los dos SGBD se decide utilizar SQLite para el desarrollo del Sistema de Administración de Configuración ya que se adapta satisfactoriamente, puesto que brinda la posibilidad de poder ejecutarse en multiples plataformas y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración. Además de contar con diferentes interfaces del API, las cuales permiten trabajar Python. SQLite realiza operaciones de manera eficiente y es más rápido que Oracle y PostgreSQL.

1.12 Conclusiones parciales del capítulo

En el capítulo se han analizado los conceptos fundamentales de desarrollo de *software*, de manera que se pueda seguir el curso de la investigación, llegándose a las siguientes conclusiones:

- El estudio realizado a los sistemas de administración de configuración de servidores permitió identificar características propias de estos sistemas y seleccionar la herramienta sobre la cual se desarrollará la propuesta de solución.

- Al contar con un equipo de trabajo pequeño, con un corto plazo para la entrega, se dedice utilizar como metodología de desarrollo XP, como marco de trabajo Django y Bootstrap ,como lenguaje de programación Python, JavaScript,HTML5 Y CSS3, además del servidor web Nginx y el gestor de BD SQLite. Luego de conocer las principales características de las tecnologías y herramientas empleadas, se puede asegurar que el uso de las mismas aporta fortaleza, organización, flexibilidad

y control durante todo el desarrollo de software. Además con el empleo de herramientas libres, se logra independizar la solución de licencias propietarias que resultan generalmente costosas.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA DE ADMINISTRACIÓN DE CONFIGURACIÓN DE SERVIDORES DE PRODUCCIÓN DE LA UCI.

2.1 Introducción

En este capítulo se describen las fases principales de la metodología XP: Planificación y Diseño para la solución propuesta. Se desarrollarán los artefactos importantes de estas fases como la Definición de la audiencia, Historia de Usuarios, Plan de Iteraciones, Plan de Duración de Iteraciones, Plan de Entregas y Tarjetas CRC (Clases, Responsabilidad y Colaboración).

2.2 Modelo de dominio

El Modelo de Dominio o Modelo Conceptual es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Se realiza cuando no se logra determinar el proceso del negocio con fronteras bien establecidas y donde los flujos de información son difusos (múltiples orígenes, sólo eventos, sucesos), cuando existe solapamiento de responsabilidades, así como múltiples responsabilidades. Además es difícil establecer reglas de funcionamiento (Larman, 2003).

Este modelo representa clases conceptuales del dominio del problema que vienen a ser las ideas u objetos físicos y el enlace de unos objetos con otros. Para la realización del mismo se utiliza un lenguaje común facilitando la comunicación entre los desarrolladores y usuarios de la aplicación y un mayor entendimiento del contexto en que se desarrolla el sistema. El modelo de dominio se describe mediante diagramas UML (diagrama de clases). Estos diagramas muestran las clases del dominio y cómo se relacionan unas con otras mediante asociaciones.

Después de analizado el uso y las características del modelo de dominio se arriba a la conclusión de que debido a no tener una estructura definida de los procesos del negocio, o sea, al no contar con fronteras bien definidas y un manual de procedimientos, y al tener flujos de información difusa, se plantea un modelo de dominio. Se realizará identificando los conceptos asociados al negocio y representándolos en un

diagrama de clases UML de forma tal que se pueda lograr un fácil entendimiento de cómo se realizan los procesos actualmente.

La metodología XP no utiliza diagramas de Modelo del Dominio pero para un mejor entendimiento de la problemática, se ha realizado el mismo.

2.2.1 Descripción de clases del Modelo de Dominio.

Como el modelo de dominio o modelo conceptual contribuye posteriormente en el proceso de desarrollo del software a identificar las clases que se utilizarán para modelar el sistema, a continuación se identifican los conceptos fundamentales y las entidades que se emplean en el modelo a través de un glosario de términos:

Administrador: Es el encargado de ejecutar comandos de configuración en los servidores.

Servidor: Es quien recibe provee servicios a los clientes.

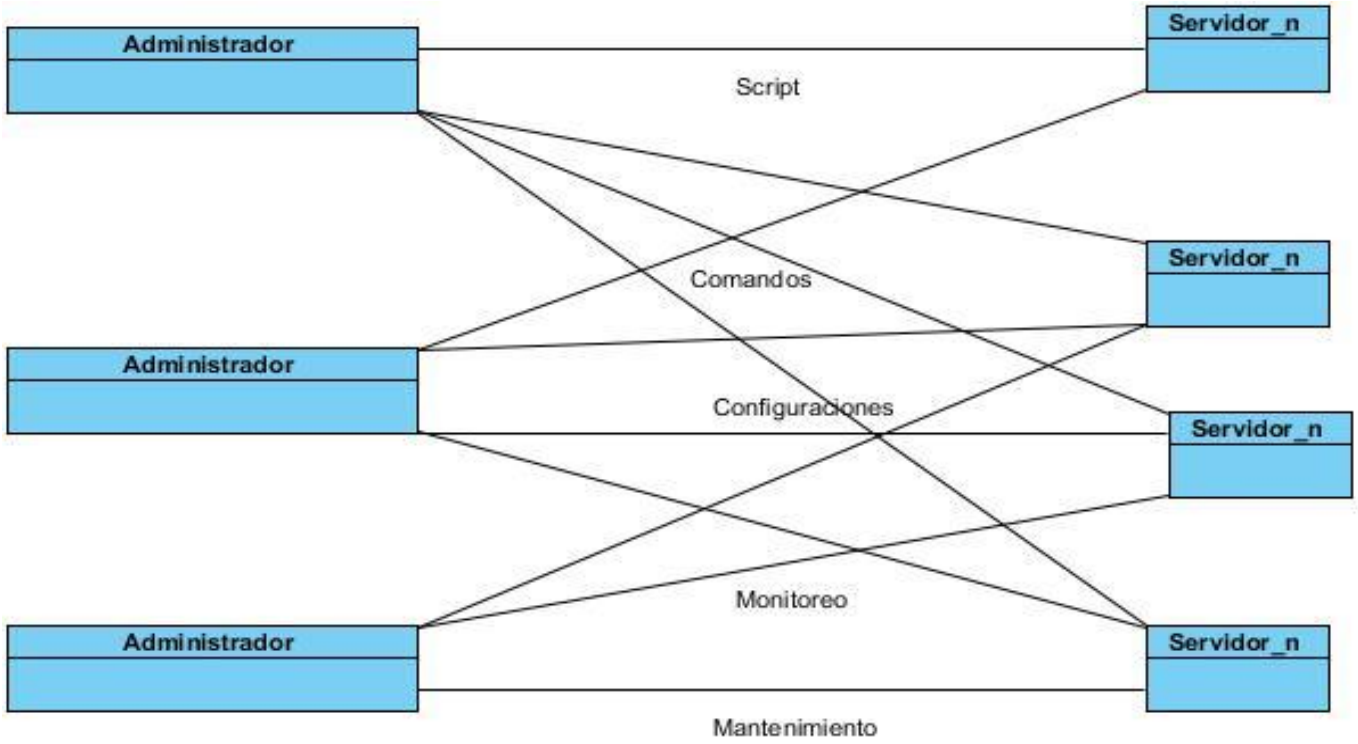


Ilustración 1 Modelo de Dominio del Sistema de Administración de Configuración

2.3 Definición de la audiencia

La audiencia constituye uno de los elementos más importantes de esta fase de desarrollo del sistema, ya que consiste en definir al público a quien va dirigida la solución del Sistema de Administración de Configuración de servidores de la producción, en este caso la solución va dirigida a los Administradores de los servidores de producción de la UCI.

2.4 Usuarios relacionados con el sistema

La aplicación web que se desarrolla muestra un grupo de funcionalidades para cumplir con los objetivos trazados. En la implementación se establecen roles para asignar los diferentes permisos para su acceso.

Los roles y permisos son utilizados para restringir el nivel de acceso a las funciones del sistema de cada usuario que interactúe con el sistema, es decir que en dependencia de los roles de cada usuario y los permisos que tengan cada uno de los roles será su acceso a las funciones del sistema. Se denomina usuario a cualquier persona relacionada con el sistema, ya sea vinculada al desarrollo del mismo o que de una forma u otra interactúa con la aplicación, incluyendo a los que mantendrán el sistema funcionando y actualizado (Amo, 2005).

En la tabla 1 se muestran los Usuarios relacionados con el sistema:

Tabla 1 Usuarios relacionados con el sistema.

Usuarios	Descripción
Administrador de la Configuración	Es el usuario que puede ejecutar módulos, administrar *minions*, gestionarlos, entre otros.
Reportador	Es el que puede obtener los reportes estadísticos de los *minions*.
Administrador del Sistema	Es el usuario que tiene permiso para administrar todas las funcionalidades del sistema.

2.5 Historias de Usuarios

La metodología XP utiliza la técnica de las Historias de Usuario (HU) para sustituir a los documentos de especificación funcional y a los casos de uso. Estas HU son escritas por el cliente en su propio lenguaje como descripciones cortas de lo que el sistema debe realizar. El tratamiento de las HU es muy dinámico y flexible, permite que en cualquier momento se puedan romper, reemplazar por otras más específicas o generales, añadirse nuevas o ser modificadas. Para ser implementadas las HU, el cliente y los desarrolladores se reúnen para detallar las funcionalidades de cada una. El tiempo de desarrollo ideal para una HU varía entre 1 y 3 semanas (Joskowicz, 2008).

En las Tablas 2, 3, 4 y 5 se muestran las Historias de Usuario.

Tabla 2 HU Ejecutar Comandos.

Historia de usuario	
Numero: 1	Nombre de la HU: Ejecutar Comandos
Usuario: Administrador de la Configuración	
Prioridad en el negocio: Alta	Estimación: 1
Riesgo de Desarrollo: Alto	Iteración: 1
Descripción: Primero se debe seleccionar el comando de salt, luego se ejecuta, en la terminar de la interfaz.	

Tabla 3 HU Agregar *o* Quitar Minions

Historia de usuario	
Numero: 2	Nombre de la HU: Agregar o quitar Minions
Usuario: Administrador de Configuración.	
Prioridad en el negocio: Alta	Estimación: 1
Riesgo de Desarrollo: Alto	Iteración: 1

Descripción: Se obtiene una lista de todos los minions que estan aceptados, luego se guardan en una variable todas las configuraciones del master, se guarda en otra variable los minions que estan aceptados y no aceptados, recorro la lista minions, y voy adicionando los que estan aceptados y los que no estan aceptados.

Tabla 4 HU Seguridad.

Historia de usuario	
Numero: 10	Nombre de la HU: Seguridad
Descripción: La autenticación en el sistema está definida para cada uno de los usuarios mediante sus roles y permisos.	
Observación: Para la utilización del sistema, la aplicación sólo podrá ser accedida por medio de un usuario y una contraseña. El usuario registrado podrá realizar las operaciones correspondientes de acuerdo a su rol definido en el sistema	

Tabla 5 HU. Usabilidad

Historia de usuario	
Numero: 11	Nombre de la HU: Usabilidad.

Descripción: El Sistema de Administración de la Configuración de los servidores de producción de la UCI, podrá ser empleado por personas con pocos, medios o avanzados conocimientos de informática.
Observación: El sistema muestra una interfaz sencilla y amigable. Brinda los botones con un tamaño adecuado y con nombres claros que permiten a los usuarios realizar las operaciones que deseen de forma sencilla.

En los Anexos se hace alución a las restantes HU.

2.6 Plan de iteraciones

Todo proyecto que emplea metodología XP debe dividirse en iteraciones. Las iteraciones son fases o etapas de la implementación donde se obtienen resultados en un tiempo estimado. En el plan de iteraciones se especifican detalladamente el orden de desarrollo de las HU dentro de cada iteración conjuntamente con la duración de las mismas.

Iteración 1: Esta iteración tiene como objetivo la implementación de las historias de usuario de prioridad alta. Al finalizar se contará con las funcionalidades descritas en las historias de usuario asignadas.

Iteración 2: El objetivo de esta iteración es la implementación de las restantes funcionalidades con prioridad media. Con la culminación de la misma se tendrán implementadas las peticiones del cliente descritas en las historias de usuario asignadas.

Iteración 3: En esta iteración serán implementadas las funcionalidades de prioridad baja. Estas funciones tienen el propósito de brindar al cliente comodidad en la gestión de otras tareas asociadas a las de baja prioridad. Estas funciones están descritas en las historias de usuario asignadas.

2.7 Plan de duración de iteraciones

Siguiendo el desarrollo de la metodología XP se crea el plan de duración de las iteraciones. En este plan se especifica más detalladamente el orden de desarrollo de las historias de usuarios dentro de cada iteración así como la estimación completa de dicha iteración.

En la tabla 6 se muestra el Plan de duración de iteraciones.

Tabla 6 Plan de duración de Iteraciones.

Iteraciones	Desripción de la iteración	Orden de las HU a Implementar	Duración(Semanas)
Iteración 1	Se desarrollan las historias de usuario que tienen prioridad muy alta. Permitiendo la instalacion del sistema.	Agregar o quitar minions Ejecutar módulos. Ejecutar comándos.	8 semanas
Iteración 2	Se desarrollan las historias de usuarios con prioridad media, que permiten Obtener los repores de los clientes.	Gestionar grupo. Obtener el reporte estadístico de los minions. Lista de trabajos	8 semanas
Iteración 3	Se desarrollan las historias de usuarios de prioridad baja, que permiten agregar o listar los grains del servidor minions.	Gestionar usuario Listar grains de los minions. Agregar grains de los minions.	8 semanas

2.8 **Plan de Entregas**

En el plan de entregas se establecen qué HU son agrupadas para conformar una entrega y el orden de las mismas.

En la Tabla 7 se muestra el Plan de entregas:

Tabla 7 Plan de Entregas.

Historia de Usuario.	Primera Iteración.	Segunda Iteración	Tercera Iteración
Agregar o quitar minions	Desarrollándose	Finalizado	-
Ejecutar módulo	Desarrollándose	Finalizado	-
Ejecutar comandos	Desarrollándose	Finalizado	-
Obtener el reporte estadístico de los minions.	-	Desarrollándose	Finalizado
Gestionar grupo	-	Desarrollándose	Finalizado
Lista de tareas	-	-	Finalizado
Gestionar Usuario.	-	-	Finalizado
Agregar Grains de los Minions	-	-	Finalizado
Listar Grains de los Minions	-	-	Finalizado

2.9 **Tarjetas CRC**

Las tarjetas CRC (Clase-Responsabilidad-Colaboración) son en la práctica pequeñas tarjetas de cartón que se elaboran para ser mostradas al cliente, de manera que se pueda llegar a un acuerdo sobre la validez de las abstracciones propuestas, lo que ayuda al equipo durante el diseño e implementación del sistema. Estas constituyen documentación adicional que será adjuntada a las HU (Vallespir, 2002).

Las tarjetas CRC trabajan con la técnica de modelado basada en objetos, representando cada tarjeta CRC a un objeto, identificando las clases y sus responsabilidades. Las tarjetas están compuestas por el nombre de la clase colocado como título, en la parte izquierda se colocan las responsabilidades (funcionalidades) y en la parte derecha las clases que se implican en cada una.

En las Tablas 8 y 9 se muestran las Tarjetas CRC:

Tabla 8 Tarjeta CRC Ejecutar Comandos

Clase: *Ejecute_Comands*	
Responsabilidad.	Colaboración:
Ejecutar Comandos	*TempleteView.*

Tabla 9 Tarjeta CRC Agregar o quitar minions

Clase: *Aceppt_Minions*	
Responsabilidad.	Colaboración
Adicionar minions	*TempleteView.*

Eliminar minions	

En los Anexos se hace alución a las restantes Targetas CRC.

2.10 Conclusiones parciales del capítulo

- Al culminar el proceso de análisis y diseño del Sistema de Administración de la Configuración de servidores de producción de la UCI se logró una mayor comprensión de los requisitos, desglosados en trece Historias de Usuarios que permiten a los desarrolladores una mejor planificación de las tareas.
- Por otra parte, se obtuvo otro artefacto como el Modelo de Dominio; el cual permite a los desarrolladores obtener una visión más detallada de los principales componentes que deben conformar el sistema.
- Se concluye, además, que todos los artefactos generados por la metodología en esta etapa guiarán de forma efectiva el desarrollo del Sistema de Administración de Configuración de producción de la UCI, o prácticamente cualquier otro subsistema que pueda ser concebido en el futuro.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA DEL SISTEMA DE ADMINISTRACIÓN DE CONFIGURACIÓN DE SERVIDORES DE PRODUCCIÓN DE LA UCI.

3.1. Introducción

En el presente capítulo se elaboran las fases de implementación y prueba. La metodología XP plantea que la implementación debe realizarse de forma iterativa e incremental, que estará dedicada al desarrollo de los requisitos especificados por el cliente, y una vez concluida la misma se da paso a la fase de pruebas, donde se exponen los resultados arrojados por las pruebas realizadas a las diferentes funcionalidades implementadas.

3.2. Patrones

Un patrón es un modelo a seguir, surgen de la experiencia de seres humanos al tratar de lograr ciertos objetivos, capturan la experiencia existente y probada para promover buenas prácticas. Para que una solución sea considerada un patrón debe haber sido comprobada su efectividad resolviendo problemas similares en ocasiones anteriores. Ademá de que debe ser reusable (Moraga, 2010). Existen varias clasificaciones de patrones, como los de Arquitectura y los de Diseño.

Los patrones arquitectónicos son los que definen la estructura de un sistema de *software*, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema Los patrones arquitectónicos fijan la arquitectura global de una aplicación.

Por su parte, los patrones de diseño son una solución a un problema de diseño. Son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces, es aplicable a diferentes problemas de diseño en distintas circunstancias.

3.2.1. ***Patrón Arquitectónico***

Una de las tecnologías que se emplea en el desarrollo del software es el *framework* Django el cual usa una modificación de la arquitectura Modelo-Vista-Controlador (MVC), llamada Modelo-Plantilla-Vista (MTV por sus siglas en inglés), empleado en la implantación de la solución. El modelo en Django continúa siendo modelo, la vista se llama Plantilla y el controlador se nombra Vista.

- **La Capa Modelo:** Se refiere a la capa de acceso a datos. Esta capa contiene todo lo referido a los datos: cómo acceder a ellos, cómo validarlos, qué comportamiento tienen y las relaciones entre ellos.

- **La Capa Plantilla:** Se refiere a la capa de presentación. Esta capa contiene las decisiones relacionadas con la presentación: cómo debería mostrarse algo en una página web u otro tipo de documento.

- **La Capa Vista:** Se refiere a la capa de lógica del negocio. Esta capa contiene el acceso al modelo y delega en las plantillas apropiadas.

Como parte del funcionamiento de este patrón, primeramente el navegador manda una solicitud a la vista, luego la vista interactúa con la modelo para obtener los datos, después hace una llamada a la plantilla y la plantilla se encarga de renderizar la respuesta a la solicitud del navegador. Seguidamente se muestra en la ilustración dicha relación (Parrra, 2008):

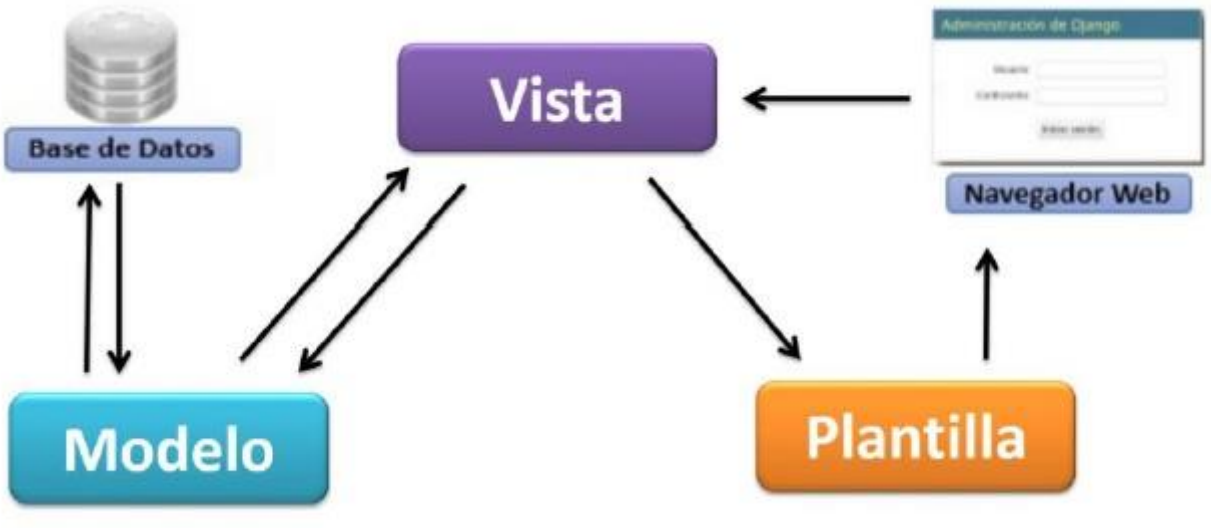


Ilustración 2 Modelo Vista Template

3.2.2. ***Patrones de Diseño***

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Estos patrones identifican Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades. Los patrones de diseño se dividen en dos grandes grupos los Patrones Generales de S*oftware* para Asignar Responsabilidades GRASP (por sus siglas en inglés*)* y la Banda de los cuatro (GOF, por sus siglas en inglés). A continuación se realiza un estudio de cómo fueron utilizados en el Sistema de Administración de la Configuración de los servidores de producción de la UCI (Debrauwer, 2012).

3.2.2.1. ***Patrones GRASP***

Los Patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

En el diseño del sistema se destacan el uso de 3 patrones principales que son (Alvarez, 2009):

Alta Cohesión: Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto identificable. Cada clase realiza una función especializada que va de acuerdo con el rol asignado.

Bajo Acoplamiento: Debe haber pocas dependencias entre las clases. Si todas las clases dependen de todas ¿Cuántos software se pueden extraer de un modo independiente y reutilizarlo en otro proyecto? El gestor de consultas es completamente independiente del módulo de base de datos, solo utilizando la conexión a la base de datos que está en uso, las clases poseen mínimas dependencias de otras.

Creador: Se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando:
B contiene a A.
B es una agregación (o composición) de A.
B almacena a A.
B tiene los datos de inicialización de A (datos que requiere su constructor)
B usa a A.

3.2.2.2. ***Patrones GOF***

Los patrones GOF se clasifican según el libro, GOF en tres categorías: de creación, estructurales y de comportamiento. Los patrones de creación abstraen el proceso de creación de instancias, los estructurales se ocupan de cómo clases y objetos son utilizados para componer estructuras de mayor tamaño y los de comportamiento atañen a los algoritmos y a la asignación de responsabilidades entre objetos.

Algunos de estos patrones utilizados en el Sistema de Administración de Configuración de los servidores de producción de la UCI son (Debrauwer, 2012):

Creación:

- Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Se utiliza en el objeto de base de datos que es una instancia única para el proyecto en uso.

Estructurales:

- Facade(Fachada): Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema. Se utiliza al unificar las interfaces en una sola ventana, manejando esta varias interfaces (mensajes, datos, consulta, entre otros). Específicamente en la herencia de las plantillas y en la redefinición de los procesadores de contexto para la plantilla base y en el uso de la ORM para acceso a la BD.

- Decorator (Decorador): Añade funcionalidad a una clase dinámicamente. Se utiliza al decorar las clases para la autenticación y para la creación del cliente para ejecutar acciones sobre la API.

Comportamiento:

- Command (Orden): Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma. En el uso del decorador para la autenticación.

3.3. Implementación

Durante la fase de implementación se vinculan las Historias de Usuarios con las tareas concretas de desarrollo. Debido a que el desarrollo de las mismas fue dividido en iteraciones se proponen una serie de tareas por iteración que darán cumplimiento a lo especificado en la fase. Para el desarrollo del módulo se concretaron tres iteraciones de forma tal que al concluir con cada iteración se pudieran obtener las características deseadas y la realización de las pruebas a las mismas. Para la decisión de los puntos de

estimación se asumió que un punto equivale a una semana la cual cuenta con cinco días de trabajo y cada día cuenta con ocho horas laborales.

3.4. Tareas de la Ingeniería

Las tareas de la ingeniería son las distintas funcionalidades operativas que conforman una historia de usuario y que permiten testear si se está trabajando bien. Este trabajo conjunto constituye un paso decisivo para comenzar la implementación del *software*, pues permite organizar el trabajo en pasos lógicos, de acuerdo a la planificación correspondiente a esa historia de usuario (Amo, 2005). Las tareas de la ingeniería se realizan con el objetivo de resolver las HU que pueden tener una o más tareas de ingeniería en dependencia de la complejidad de la funcionalidad a desarrollar.

En la Tabla 10 se identificaron las siguientes Tareas de la Ingeniería:

Tabla 10 Tareas de ingeniería.

No.HU	Nombre HU	No. TI	Tarea de ingeniería
1	Ejecutar Comandos.	1	Ejecuta comandos de salt.
2	Agregar o quitar minions	1	Adiciona minions
		2	Elimina minions
3	Ejecutar módulos	1	Ejecutar módulos de salt
4		1	Adicionar grupo

	Getionar grupo	2	Eliminar grupo
		3	Modificar grupo
5	Obtener el reporte estadístico de los minions	1	Obtiene todos los datos estadísticos de los minions
6	Lista de tareas	1	Lista las tareas realizadas hasta el momento.
7	Getionar Usuario	1	Adicionar Usuario
		2	Eliminar Usuario
		3	Modificar Usuario
8	Listar grains de los minions.	1	Lista todos los grains de los minions
9	Agregar grains de los minions	1	Adicionar grains de los minions

3.4.1. ***Desarrollo de las Tareas de la ingeniería***

A continuación se detallan el desarrollo de las tareas de la ingeniería divididas en las iteraciones correspondientes a cada historia de usuario.

Iteración 1.

Tabla 11 Descripción de la Tarea de la Ingeniería Agregar o quitar minions

Número de tarea:1	HU (No.2): Adicionar minions
Nombre de la tarea: Adicionar minions	
Tipo de tarea: Desarrollo	Estimación: 1
Fecha Inicio: 1/10/2014	Fecha Fin: 20/10/2014
Programador Responsable: Yeneisy de la Torre Pérez	
Descripción: Se muestra una interfaz donde se adicionan los minions en la base de datos.	

Tabla 12 Descripción de la tarea de ingeniería Ejecutar comandos

Número de tarea: 1	HU (No.1): Ejecutar comandos de salt.
Nombre de la tarea: Ejecutar comandos de salt	
Tipo de tarea: Desarrollo	Estimación: 1
Fecha Inicio:20/10/2014	Fecha Fin: 10/11/2014
Programador Responsable: Yeneisy de la Torre Pérez	
Descripción: Se muestra una interfaz donde se realizan los comandos de salt, ya sea en la propia aplicación o en una consola.	

3.5. Instalación y configuración del Sistema de Administración de Configuración de Servidores de producción de la UCI.

Para el uso del Sistema de Administración de Configuración de servidores de producción de la UCI, se deberán instalar determinados componentes y herramientas. El sistema está concebido para emplearse en los servidores de la producción de la UCI, los cuales poseen en su mayoría sistemas operativos tales como: Ubuntu y Centos. Como la aplicación está basada en Salt, será necesario como primer paso instalar dicha herramienta.

La configuración de Salt se basa en un ordenador central que será el servidor master (que en términos de Salt se conocen como salt master), y los ordenadores clientes (nombrados como salt minions) (Myers, 2015). El salt master es el servidor central al que todos los salt minions se conectan. Los comandos se ejecutan en los salt minions mediante órdenes enviadas por el salt master y los salt minions envian los datos al salt master. Los salt minions son los potencialmente cientos o miles de servidores que pueden ser consultados y controlados desde el master. Para el empleo de Salt se requiere seguir determinados pasos, los cuales son descritos a continuación de manera general:

- Instalar el paquete *salt-master* en el ordenador central.
- Instalar el *salt-minion* en los ordenadores clientes.
- Configurar el salt minion para decirle dónde se encuentra el master.
- Reiniciar los servicios que ofrece Salt master y los minions.

La manera exacta de realizar las acciones anteriores depende del sistema operativo sobre el cual se esté trabajando.

Además es necesario tener instalado en el salt master el lenguaje de programación Python y el servidor web Nginx. La apliación web deberá estar ubicada dentro del servidor Nginx. Finalmente se podrá acceder a la aplicación desde el navegador web.

3.6. Pruebas

Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo. El objetivo de las pruebas es presentar información sobre la calidad del producto a las personas responsables de este.

La metodología XP enfatiza en los aspetos relacionados con las pruebas, clasificandolas en diferentes tipos y funcionalidades; indicando, quién, cuándo y cómo deben ser implementadas y ejecutadas (Oré, 2009) .

3.6.1. ***Pruebas de Unitarias***

Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, revisar el código, que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código. En este sentido, el sistema y el conjunto de pruebas deben ser guardados junto con el código, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo (Joskowicz, 2008). Estas pruebas fueron llevadas a cabo por los programadores, encargadas de verificar el código de forma automática, se aplicaron a todas las funcionalidades del sistema comprobando el correcto funcionamiento de cada una de estas y obteniendo resultados satisfactorios.

3.6.2. ***Pruebas de rendimiento***

Pruebas dirigidas a evaluar la conformidad de un sistema o componente con requerimientos de desempeño específicos. Normalmente esto se lleva a cabo usando una herramienta de prueba automática para simular un gran número de usuarios, carga y volumen de información y para monitorear el

desempeño del hardware, al no ser Pepper un portal web sino un front_end de Salt las pruebas de rendimiento se realizan enfocadas a las acciones de Salt sobre los servidores.

Plan de pruebas

Los objetivos fundamentales para diseñar un plan de pruebas radican en la obtención de las configuraciones óptimas para el despliegue de la aplicación, la validación del estimado de carga y la concurrencia que se espera soporte la misma en condiciones extremas. Por otra parte, permiten la detección y localización de posibles errores existentes en el software.

Las pruebas fueron diseñadas para probar la estabilidad soportada por la aplicación, por encima de la velocidad de respuesta; debido a que se considera que la estabilidad y fiabilidad de la aplicación posee mayor importancia en situaciones extremas (obtención de grains, ejecución de comandos, tareas y informes estadísticos) que la rapidez en los tiempos de respuesta que depende de Salt.

Para la realización de las pruebas se utilizó un entorno de hardware y software consistente en:

- Servidor Virtual con procesador Core 2 duo a 2.20 GHZ y 3 GB de RAM.
- Sistema operativo Centos 6.4.
- Servidor Nginx en su versión 1.4.4.

Antes de iniciar la prueba se identifican un conjunto comandos y estados que se caracterizan por su alto consumo de recursos para ser utilizadas en la misma. En la solución se identificaron 3 comandos fundamentales que son solicitadas con mayor frecuencia por los administradores (test.ping, grains, salt-key -L). El plan de pruebas fue diseñado para simular un escenario que supere la ejecución concurrente real estimada para el sistema. La prueba simulará una ejecución concurrente de 30 trabajos realizándose sobre 200 servidores.

Resultados

Al culminar la prueba se puede concluir que la aplicación se encuentra lista para soportar una

ejecución concurrente muy superior a la que estará sometida; dígase 50 trabajos concurrentes sobre 500 servidores A continuación se mostraran los resultados de las pruebas.

- Tiempo minimo de respuesta a (test.ping) 0m11.122s
- Tiempo minimo de respuesta a (grains.items) 0m11.037s
- Tiempo minimo de respuesta a (salt-key -L) 0m0.779s

Los resultados anteriores muestran el tiempo mínimo tomado por cada uno de los trabajos seleccionados. Es importante destacar que estos casos extremos son difíciles de repetir en un ambiente donde el sistema se use por varios trabajos simultáneos:

- Tiempo maximo de respuesta a (test.ping) 0m19.122s.
- Tiempo maximo de respuesta a (grains.items) 0m23.037s.
- Tiempo maximo de respuesta a (salt-key -L) 0m06.011s.
  Los tiempos máximos de respuesta de los trabajos realizados a cada uno de los trabajos seleccionados siendo el tiempo más alto de respuesta de 23 segundos. Teniendo en cuenta que las condiciones de hardware utilizadas para realizar las pruebas y la prueba simula un ambiente con mayor demanda que la mayor estimada para el sistema en un entorno real, el tiempo máximo resultante es considerablemente más bajo del esperado.

3.7. Conclusiones

- Con el desarrollo del capítulo, aplicando la metodología de desarrollo de software XP se generaron las Tareas de Ingeniería que sirvieron como base para la implementación del sistema diseñado en el capítulo anterior.
- Partiendo del alcance del desarrollo se determinó la prueba de unitarias y de rendimiento como las más indicadas, comprobando que el sistema responde a una velocidad mayor de lo que se esperaba.

- Las pruebas realizadas permitieron determinar el correcto funcionamiento del sistema implementado, así como analizar que el mismo es adaptable a las diferentes empresas e instituciones donde administren gran catidad de servidores.

CONCLUSIONES GENERALES

Luego de analizar las principales deficiencias existentes en la Administración de servidores de la UCI se evidenció la necesidad de implementar un sistema para contribuir a dicho proceso. Para ello se realizó un proceso de investigación que propició la obtención de conocimiento como base para el desarrollo del sistema. Luego de concluida la investigación se arribó a las siguentes conclusiones:

- El estudio de Sistemas para la Administración de servidores permitió identificar caraterísticas propias de estos sistemas y sentar las bases para el desarrollo del Sistema de Administración de Configuración de los servidores de la producción de la UCI.
- Se identificaron herramientas y tecnologías que apoyaron adecuadamente el desarrollo del sistema.
- XP es una metodología flexible que se ajustó satisfactoriamente al desarrollo.
- La implementación del Sistema de Administración de Configuración de servidores de produccion de la UCI, cumplió con las trece Historias de Usuario identificados en la fase de análisis y diseño.
- El diseño y ejecución de las pruebas unitarias y de rendimiento, permitió comprobar el correcto funcionamiento del Sistema de Administración de Configuración de los servidores de producción de la UCI.
- Se obtuvo un Sistema de Administración de Configuración de los servidores de la producción de la UCI, que permite contribuir a la automatización de la administración de la configuración de los servidores.

- Se logró un Sistema de Administración de Configuración de servidores de la producción de la UCI basado en la herramienta Salt que permite a los administradores mediante una interfaz web , realizar su trabajo diario de forma rapida y eficáz.

RECOMENDACIONES

Para continuar en la profundización de la propuesta de este trabajo se recomienda:

- Continuar desarrollando el sistema con el objetivo de hacer su uso más flexible, posibilitando su aplicación en otras instituciones del país.
- Seguir extendiendo la implementación del módulo del sistema.
- Extender las funcionalidades para Windows.

REFERENCIA BIBLIOGRÁFICA:

Larman, . 2003. *UML y Patrones. 2da Edición.* s.l. : Prentice Hall, Prentice Hall, 2003.

Abreu, . 2011. *Gestión de servidores web.* 2011.

Alvarez, Miguel Angel. 2001. Desarrolloweb.com. [Online] 2001. http://www.desarrolloweb.com/articulos/541.php.

—. 2009. Novedades de HTML 5 - ¿ Qué es HTML 5? *Novedades de HTML 5 - ¿ Qué es HTML 5?* [Online] 14 Octubre 2009. http://www.desarrolloweb.com/articulos/que-es-html5.html..

Amo, Fernando Alonso. 2005. *Introducción a la Ingeniería del Software.* España : s.n., 2005.

Aubry, . 2012. *HTML5 Y CSS3.* Barcelona : s.n., 2012.

Bécares, . 2013. Ansible Nueva Plataformade comunicación. *Ansible Nueva Plataformade comunicación.* [Online] 17 julio 2013. [Cited: 20 febrero 2015.] http://www.channelbiz.es/2013/07/17/ansible-la-plataforma-de-siemens/.

Beck, . 1999. Extreme Programming Explained. [Online] 1999.

Bennett, . 2009. *Practical Django Projects.* s.l. : Board, 2009.

Ciberaula. 2013. Ciberaula. *Ciberaula.* [Online] 25 enero 2013. [Cited: 17 diciembre 2014.] http://linux.ciberaula.com/articulo/linux_apache_intro..

Collado, Cecilia Castaño. 2010. *Género y TIC.* Barcelona : UOC, 2010.

Dar, . 2013. *Nginx Module Extension.* 2013.

Davison, . 2010. *Bootstrap Methods and their Applications.* Estados Unidos : s.n., 2010.

Debrauwer, . 2012. *Patrones de Diseño.* Barcelona : ENI, 2012.

Escobar, Carlos Javier Perez. 2009. Administración de la Configuracion. *Administración de la Configuracion.* [Online] lunes junio 2009. [Cited: 21 febrero 2015.] http://asprotech.blogspot.com/2009/06/administracion-de-la-configuracion.html.

Gauchat, Juan Diego. 2012. *El gran libro de HTML, CCS y Javascript.* Barcelona : s.n., 2012.

Henney, . 2010. Developer Fusion. [Online] 23 Julio 2010. http://www.developerfusion.com/article/84899/what-is-software-architecture/.

Herrera, . 2008. Que es java. [Online] 2008. http://www.iec.csic.es/criptonomicon/java/quesjava.html..

Infante, . 2012. Maestros del web. [Online] 16 Abril 2012. http://www.maestrosdelweb.com/editorial/curso-django-introduccion/.

informática, Unidad Docente de Ingeniería del Software. Facultad de. *Patrones del "Gang of Four".* Madrid : s.n.

Jalón, Javier García de. 2000. Aprenda Java como si estuviera en primero. [Online] Enero 2000. http://www.matematica.ciens.ucv.ve/files/Manuales/Manuales/Programacion%20Web%20-%20Aprenda%20Java%20como%20si%20estuviera%20en%20primero.pdf.

Jaynes, . 2014. Taste Test. *Taste Test.* [Online] june 2014. [Cited: 23 febrero 2015.] https://devopsu.com/books/taste-test-puppet-chef-salt-stack-ansible.html.

Joskowicz, . 2008. Scribd. [Online] 02 octubre 2008. http://es.scribd.com/doc/208173814/Xp-Jose-Joskowicz#scribd.

Joskowicz., . 2008. Reglas y Prácticas en eXtremeProgramming. [Online] 2008.

Katz, . 2009. *El papel de las TIC en el desarrollo.* Madrid : Ariel,SA, 2009.

Kenneth, E. Kendall. 2005. *Análisis y diseño de Sistemas .* mexico : s.n., 2005.

Kickbill, . 2009. Tus primeros pasos con Zend Framework. [Online] 2009. http://www.kickbill.com/?p=1232..

Kreibich, Jay A. 2010. *Using SQLite.* Estados Unidos : s.n., 2010.

Lehey, . 2013. Que es BSD? [Online] 13 11 2013. https://www.freebsd.org/doc/es/articles/explaining-bsd/article.html.

Letelier, . 2006. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [Online] Junio 2006. http://www.cyta.com.ar/ta0502/b_v5n2a1.htm.

Martin, Alicia Ramos. 2014. *Aplicaciones Web.* Madrid : Paraninfo, 2014.

Medina, . 2014. *Pruebas de Rendimiento TIC.* Murcia : s.n., 2014.

Montero, Sergio Infante. 2012. Maestros del Web. [Online] 16 Abril 2012. http://www.maestrosdelweb.com/editorial/curso-django-introduccion/ .

—. 2012. Maestros del Web .Django, el web framework para perfeccionistas. [Online] 16 Abril 2012. http://www.maestrosdelweb.com/editorial/curso-django-introduccion/.

Montoro, Alturo Fernández. 2012. *Python al descubierto .* Madrid : s.n., 2012.

Moraga, Maria de los Angeles. 2010. *Calidad del producto y proceso del sofware.* España : RA-MA, 2010.

Myers, . 2015. *Learnning Saltstack.* 2015.

Niño, . 2010. *Aplicaciones web de escritorio.* 2010.

Obe, . 2012. *PostgreSQL Up and Running.* Estados Unidos : s.n., 2012.

Oré, . 2009. Unit Testing-Pruebas Unitarias. *Unit Testing-Pruebas Unitarias.* [Online] 2009. [Cited: 15 enero 2015.] http://www.calidadysoftware.com/testing/pruebas_unitarias1.php.

Palacios, . 2008. ScrumManager: Gestión de proyectos. [Online] Septiembre 2008. www.etnassoft.com/biblioteca/scrummanager-gestion-de-proyectos/.

Parrra, Jose David. 2008. *Guia de patrones y arquitectura.net.* 2008.

PBworks. 2013. Patrones arquitectónicos. [Online] 14 Junio 2013. http://isg3.pbworks.com/w/page/7624479/Patrones%20Arquitect%C3%B3nicos..

Perez, Isaias Carrillo. 2008. Metodologia de desarrollo del software . [Online] 10 Septiembre 2008. http://plataforma.edu.pe/pluginfile.php/246542/mod_resource/content/1/Metodologias%20de%20desarrollo%28RUP-METODOLOS%20AGILES%29.pdf.

Perez, Javier Eguíluz. 2009. Introduccion a CSS. [Online] 2009. www.jesusda.com/docs/ebooks/introduccion_css.pdf.

Portolani, . 2014. *Learning Ansible.* 2014.

Ramos, . 2008. *Operaciones con base de datos .* Madrid : s.n., 2008.

Rodríguez, . 2012. Gembeta: dev. [Online] 20 Agosto 2012. http://www.genbetadev.com/frameworks/bootstrap.

Ron Jeffries, Ann Anderson y Chet Hendrickson. 2000. Extreme Programming Installed. [Online] 26 Octubre 2000. http://www.amazon.com/Extreme-Programming-Installed-Ron-Jeffries/dp/0201708426.

Sommerville, . 2006. *Ingenieria de sofware.* Madrid : s.n., 2006.

Tupe, Cirley Cisneros y Juan. *Evaluación y Selección de Framework de Desarrollo PHP.*

Vallespir, . 2002. *CRC y Taller.* Urugay : s.n., 2002.

Venezia, . 2013. InfoWord. [Online] 21 Noviembre 2013. http://www.Review Puppet vs. Chef vs. Ansible vs. Salt _ InfoWorld.htm.

BIBLIOGRAFÍA

1. Anón. Características. [citado 1 octubre 2012a]. Available from world wide web:<http://acsblog.es/articulos/trunk/LinuxActual/Apache/html/x31.html>.
2. Anón. Metodologías tradicionales y metodologías ágiles. [citado 25 octubre 2012d]. Available from
3. world wide web: <http://www.eumed.net/libros-gratis/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm>.
4. S. Pressman, Roger. *Ingeniería de software. Un enfoque práctico.* Quinta edición.
5. *Python_para_todos.pdf (objeto application/pdf)* [online]. S.l.: s.n. [Citado el: 8 de Junio de 2012]. Disponible en: http://dspace.universia.net/bitstream/2024/919/1/Python_para_todos.pdf.
6. PostgreSQL: About. [En línea]. [Citado el: 8 de Junio de 2012]. Disponible en: http://www.postgresql.org/about/.
7. Apache Tomcat - Welcome! [En línea]. [Citado el: 12 de Junio de 2012]. Disponible en: http://tomcat.apache.org/.
8. *METODOLOGIAS_TRADICIONALES_VS._METODOLOGIAS_AGILES.pdf (objeto application/pdf)* [online]. S.l.: s.n. [Citado el: 15 de Junio de 2012]. Disponible en: http://eva.uci.cu/file.php/161/Documentos/Materiales_complementarios/UD_1_Procesos/Metodologias/METODOLOGIAS_TRADICIONALES_VS._METODOLOGIAS_AGILES.pdf.
9. Introduction to UML 2 Class Diagrams. [En línea]. [Citado el: 30 de Abril de 2012]. Disponible en: REFERENCIAS BIBLIOGRÁFICAS http://www.agilemodeling.com/artifacts/classDiagram.htm.
10. **Selley Rojas, Héctor Julián.** *Monitoreo del comportamiento de servidores de aplicaciones.* Ciudad de México: s.n., 2008. A060432.
11. **Álvarez S., Nicolás y Monsalve Z., Juan.** *Instalación y configuración de un servidor web.* Santa María : s.n., 2008.
12. **Pressman, Roger S.** *Ingeniería de software un enfoque práctico.* Madrid : Madrid. MacGraw-Hill, 2002.
13. The Apache Tomcat Connector. *AJP Protocol Reference - AJPv13.* [En línea] [Citado el: 24 de Abril de 2013.] http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html.
14. PostgreSQL. *About postgres.* [En línea] [Citado el: 20 de Mayo de 2013.] http://www.postgresql.org/about/.

15. *Guión Visual Paradigm for UML.* [En línea] 2011. [Citado el: 7 de Febrero de 2013.] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.
16. AlegsaOnLine.Lenguaje de Programación. [En línea] [Citado el: 18 de Enero de 2014.] http://www.alegsaonline.com/art/11.php.
17. Tutorial de Python v2.7. Tutorial de Python v2.7. [En línea] Python Software Foundation. [Citado el: 20 de Enero de 2014.] http://docs.python.org.ar/tutorial/2/appetite.html.
18. Ingeniería de Software. Extreme Programing. [En línea] [Citado el: 20 de Enero de 2014.] http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.
19. norfiPC. Cómo y por qué usar HTML5. [En línea] [Citado el: 26 de Enero de 2014.] http://norfipc.com/web/como-usar-html5-codigo-paginas-web-ventajas.html.
20. W3Schools. CSS3 Introduction. [En línea] [Citado el: 27 de Enero de 2014.] http://www.w3schools.com/css/css3_intro.asp.
21. LibrosWeb. Introducción al JavaScript. [En línea] [Citado el: 29 de Enero de 2014.] http://librosweb.es/javascript/capitulo_1.html.
22. Django. Django Documentation. [En línea] 6 de Noviembre de 2013. [Citado el: 29 de Enero de 2014.] https://docs.djangoproject.com/en/dev/releases/1.6/.
23. Genbeta. Bootstrap. [En línea] 16 de Junio de 2012. [Citado el: 30 de Enero de 2014.] http://www.genbetadev.com/frameworks/bootstrap.
24. León, Hermán. Desarrollo web.Características de un sistema gestor de base de datos relacional. [En línea] [Citado el: 3 de Febrero de 2014.] http://www.desarrolloweb.com/articulos/12-reglas-sgbdr.html.
25. Universidad de Belgrano de Argentina.Características del SGBD. [En línea] [Citado el: 3 de Febrero de 2014.] http://www.ub.edu.ar/catedras/ingenieria/Datos/capitulo1/cap13.htm.
26. PostgreSQL. Sobre PostgreSQL. [En línea] 2 de Octubre de 2010. [Citado el: 4 de Febrero de 2014.] http://www.postgresql.org.es/sobre_postgresql.
27. PostgreSQL. Características, limitaciones y ventajas. [En línea] [Citado el: 4 de Febrero de 2014.] http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html.

28. MySQL. Las principales características de MySQL. [En línea] 2011. [Citado el: 8 de Febrero de 2014.] http://dev.mysql.com/doc/refman/5.0/es/features.html.

29. Pérez, Damián. Maestros del web. Javascript. [En línea] 3 de Julio de 2007. [Citado el: 8 de Febrero de 2014.] http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/

30. Wiesel, Jonathan . CodeHero. Cómo Instalar Nginx . [En línea] [Citado el: 9 de Febrero de 2014.] http://codehero.co/como-instalar-nginx/.

31. Ganbeta. Historias de Usuario. [En línea] 28 de Febrero de 2012. [Citado el: 11 de Febrero de 2014.] http://www.genbetadev.com/metodologias-de-programacion/historias-de-usuario-una-forma-natural-de-analisis-funcional.

32. Letelier, Patricio y Penadés, Carmen. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea] [Citado el: 12 de Febrero de 2014.] http://www.cyta.com.ar/ta0502/b_v5n2a1.htm.

33. Priolo, Sebastián. Programación Extrema. Programación Extrema. [En línea] [Citado el: 12 de Febrero de 2014.] http://www.fcad.uner.edu.ar/jai/6JAI/XP_6JAI.pdf.

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

35. Django Project. [En línea] [Citado el: 6 de Enero de 2014.] http://www.djangoproject.com/.

36. Pressman, Roger S. Ingeniería de Software. Un enfoque práctico. Capítulo 8 Modelado de Análisis. 2005.

37. Sommerville, Ian. Ingeniería del Software. Madrid : Pearson Educacion S.A., 2005. 84-7829-074-5.

38. Larman, C. UML y patrones. México : Prentice Hall, 2003. 590 p.

39. Mis primeros pasos por el mundo de la arquitectura del software. León, Jeimy. Octubre de 2009.

40. Escribano, Gerardo Fernández. Introducción a Extreme Programming. 2002.

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

GLOSARIO DE TÉRMINOS:

Traseo: Se llama traseo cuando valida un código para ver que todo este en correcto funcionamiento.

JSON*:* Acrónimo de *JavaScript Object Notation*, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

Servidor: Un servidor es un ordenador remoto que provee los datos solicitados por parte de los navegadores de otras computadoras.

MSF: Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos

Script: En informática un guión, archivo de órdenes o archivo de procesamiento por lotes, vulgarmente referidos con el barbarismo *script*, es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano. Los guiones son casi siempre interpretados, pero no todo programa interpretado es considerado un guión.

Unix: Es un Sistema Operativo no libre muy popular, porque está basado en una arquitectura que ha demostrado ser técnicamente estable.

GNU: Fue diseñado para ser totalmente compatible con UNIX. El hecho de ser compatible con la arquitectura de UNIX implica que GNU esté compuesto de pequeñas piezas individuales de software, muchas de las cuales ya estaban disponibles.

Minions: Es el servidor que se instala en las pc clientes, los cuales obedecen todas las ordenes que le mande el master.

ANEXOS

Historias de Usuario.

Tabla 13 HU Ejecutar Módulos

Historia de usuario	
Numero: 3	Nombre de la HU: Ejecutar Módulos
Usuario: Administrador de Configuración	
Prioridad en el negocio Alta	Estimación: 1
Riesgo de Desarrollo: Alto	Iteración: 1
Descripción: Primero se debe seleccionar el módulo de salt, luego se ejecuta, y por ultimo se renderiza para devolverlo a la vista	

Tabla 14 HU Gestionar Usuario.

Historia de usuario	
Numero: 7	Nombre de la HU: Gestionar Usuario
Usuario: Administrador	
Prioridad en el negocio : Baja	Estimación: 1
Riesgo de Desarrollo: Alto	Iteración: 3
Descripción: Permite adicionar, eliminar o modificar cualquier usuario que haya interactuado con el sistema.	

Tabla 15 HU Obtener el reporte estadístico de los minions.

Historia de usuario

Numero: 5	Nombre de la HU: Obtener el reporte estadístico de los minions
Usuario: Administrador de la Configuración	
Prioridad en el negocio: Media	Estimación: 1
Riesgo de Desarrollo: Alto	Iteración: 2
Descripción: Para obtener el reporte estadístico de los minions, primero se debera escoger el minion del cual se quiere obtener su reporte, luego de haber seleccionado el minion, se accede a los grains del mismo.	

Tabla 16 HU Listar Grains de los Minions.

Historia de usuario	
Numero: 8	Nombre de la HU: Listar grains de los minions
Usuario: Administrador de la Configuración	
Prioridad en el negocio: Baja	Estimación: 1
Riesgo de Desarrollo: Alto	Iteración: 3
Descripción: Se crea un agente local de salt, se crea una variable de tipo diccionario que va a contener todos los grains de los minions, y se crea otra variable vacia, que más adelante se utilizará para guardar los nuevos datos de los grains, se recorren estas 2 variables y luego se devuelve una lista con un nuevo formato de grains.	

Tabla 17 HU Gestionar Grupo

Historia de usuario	
Numero: 4	Nombre de la HU: Gestionar Grupo
Usuario: Administrador	

Prioridad en el negocio: Media	Estimación: 1
Riesgo de Desarrollo: Alto	Iteración: 2
Descripción: Permite adicionar, modificar, eliminar los datos de los grupos que han sido creados para acceder al sistema.	

Tabla 18 HU Agregar Grains de los Minions.

Historia de usuario	
Numero: 9	Nombre de la HU: Agregar Grains de los Minions
Usuario: Administrador de la Configuración	
Prioridad en el negocio: Baja	Estimación: 1
Riesgo de Desarrollo: Alto	Iteración: 3
Descripción: Se escoge el minion al cual se le quiere agregar los grains,luego de tener el minion adecuado se crea una variable vacia que más adelante se utilizará para guardar los nuevos datos de los grains que se agregara a este minions.	

Tabla 19 HU Lista de Trabajos

Historia de usuario	
Numero: 6	Nombre de la HU: Lista de Trabajos
Usuario: Administrador de la Configuración	
Prioridad en el negocio: Baja.	Estimación: 1
Riesgo de Desarrollo: Alto	Iteración: 3

Descripción: Muestra todos los trabajos que se han ido realizando hasta el momento por el administrador del sistema.

Tarjeta CRC.

Tabla 20 Tarjeta CRC Ejecutar Módulos.

Clase: ModulesIptables	
Responsabilidad.	Colaboración:
Ejecuta módulos de Salt.	TempleteView.

Tabla 21 Tarjeta Gestionar Grupo CRC

Clase: Gestion_Grup.	
Responsabilidad.	Colaboración:
Gestionar Grupo	TempleteView.

Tabla 22 Tarjeta CRC Obtener el reporte estadístico de los Minions

Clase: Report_Stadist_Minions	
Responsabilidad.	Colaboración:
Obtiene el reporte estadístico de los minions.	TempleteView.

Tabla 23 Tarjeta CRC Gestionar Usuario.

Clase: Gestion_User.	
Responsabilidad.	Colaboración:
Gestionar Usuario.	TempleteView.

Tabla 24 Tarjeta CRC Listar Grains de los minions

Clase: List_grains_minions.	
Responsabilidad.	Colaboración:
Lista todos los grains de los minions.	TempleteView.

Tabla 25 Tarjeta CRC Agregar Grains de los Minions.

Clase: Agrega_minions	
Responsabilidad.	Colaboración:
Agregar o quitar grains de los minions.	TempleteView.

Tabla 26 Tarjeta CRC Lista de Trabajos

Clase: List_job	
Responsabilidad.	Colaboración:
Lista de Trabajos	TempleteView.

Tareas de Ingeniería.

Tabla 27 Descripción de la Tarea de ingeniería Ejecutar módulos

Número de tarea: 1	HU (No.3): Ejecutar módulos de salt
Nombre de la tarea: Ejecutar módulos de salt.	
Tipo de tarea: Desarrollo	Estimación: 1
Fecha Inicio: 15/10/2014	Fecha Fin: 20/10/2014
Programador Responsable: Yeneisy de la Torre Pérez	

Descripción: Se muestra una interfaz donde se se seleccionan los módulos de salt, y luego se ejecutan.

Iteración 2.

Tabla 28 Descripción de la Tarea de Ingeniería Gestionar grupo.

Número de tarea: 1	HU (No.4): Adicionar grupo.
Nombre de la tarea: Adicionar grupo	
Tipo de tarea: Desarrollo	Estimación: 1
Fecha Inicio: 23/10/2014	Fecha Fin: 30/10/2014
Programador Responsable: Yeneisy de la Torre Pérez	
Descripción: Se muestra una interfaz donde se adicionan grupos en la base de datos.	

Tabla 29 Descripción de la Tareas de Ingeniería Gestionar grupo.

Número de tarea: 2	HU (No.4): Eliminar grupo.
Nombre de la tarea: Eliminar grupo	
Tipo de tarea: Desarrollo	Estimación: 1
Fecha Inicio: 2/11/2014	Fecha Fin: 9/11/2014
Programador Responsable: Yeneisy de la Torre Pérez	
Descripción: Se muestra una interfaz donde se modifican los grupos en la base de datos.	

Tabla 30 Descripción de la Tarea de Ingeniería Gestionar grupo.

Número de tarea: 3	HU (No.4): Modificar grupo.
Nombre de la tarea: Modificar grupo	
Tipo de tarea: Desarrollo	Estimación: 1
Fecha Inicio: 15/11/2014	Fecha Fin: 20/11/2014
Programador Responsable: Yeneisy de la Torre Pérez	
Descripción: Se muestra una interfaz donde se modifican los grupos en la base de datos.	

Tabla 31 Descripción de la Tarea de Ingeniería Obtener el reporte estadístico de los minions.

Número de tarea: 1	HU (No.5): Obtiene el reporte estadístico de los minions
Nombre de la tarea: Obtiene el reporte estadístico de los minions	
Tipo de tarea: Desarrollo.	Estimación: 1.
Fecha Inicio: 24/11/2014	Fecha Fin: 29/11/2014
Programador Responsable: Yeneisy de la Torre Pérez	
Descripción: Se muestra una interfaz donde se obtienen los reportes estadísticos de todos los minions.	

Tabla 32 Descripción de Tarea de Ingeniería Lista de Trabajos

Número de tarea: 1	HU (No.6): Lista de Trabajos
Nombre de la tarea: Lista de Trabajos	
Tipo de tarea: Desarrollo	Estimación: 1
Fecha Inicio: 3/12/2014	Fecha Fin: 7/12/2014
Programador Responsable: Yeneisy de la Torre Pérez	
Descripción: La lista de trabajos va a mostrar todas las funciones y trabajos de salt realizados por el administrador del sistema.	

Iteración 3

Tabla 33 Descripción de la Tarea de Ingeniería Gestionar usuario.

Número de tarea:1	HU (No.7): Adicionar usuario
Nombre de la tarea: Adicionar usuario.	
Tipo de tarea: Desarrollo	Estimación: 1
Fecha Inicio: 12/12/2014	Fecha Fin: 18/12/2014
Programador Responsable: Yeneisy de la Torre Pérez	

Tabla 34 Descripción de la tarea de Ingeniería Gestionar usuario.

Número de tarea: 2	HU (No.7): Eliminar usuario
Nombre de la tarea: Adicionar usuario. ese usuario.	
Tipo de tarea: Desarrollo.	Estimación: 1
Fecha Inicio: 25/12/2014	Fecha Fin: 30/12/2014
Programador Responsable: Yeneisy de la Torre Pérez	
Descripción: Se muestra una interfaz donde el usuario debe introducir un usuario y una contraseña, el sistema verifica que los datos estén correctos y que permita el acceso al sistema según los permisos definidos para ese usuario.	

Tabla 35 Descripción de la tarea de Ingeniería Gestionar Usuario.

Número de tarea: 3	HU (No.7): Modificar usuario
Nombre de la tarea: Adicionar usuario.	
Tipo de tarea: Desarrollo	Estimación: 1
Fecha Inicio: 5/01/2015	Fecha Fin: 10/01/2015
Programador Responsable: Yeneisy de la Torre Pérez	
Descripción: Se muestra una interfaz donde se modifican usuarios introducidos con acceso al sistema.	

Tabla 36 Descripción de la tarea de Ingeniería Listar grains de los minions

Número de tarea: 1	HU (No.8): Listar grains de los minions.
Nombre de la tarea: Listar grains de los minions.	
Tipo de tarea: Desarrollo	Estimación: 1
Fecha Inicio:12/01/2015	Fecha Fin: 15/01/2015
Programador Responsable: Yeneisy de la Torre Pérez	
Descripción: Se muestra una interfaz donde se mostrar grains en la base de datos.	

Tabla 37 Descripción de la tarea de Ingeniería Agregar grains de los minions.

Número de tarea: 1	HU (No.9): Adicionar grains de los minions
Nombre de la tarea: Adicionar grains de los minions.	
Tipo de tarea: Desarrollo	Estimación: 1
Fecha Inicio: 17/01/2015	Fecha Fin: 20/01/2015
Programador Responsable: Yeneisy de la Torre Pérez	
Descripción:Se muestra una interfaz donde se adicionan grains en los minions en la base de datos.	