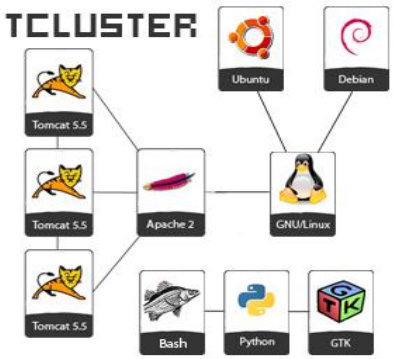
Universidad de las Ciencias Informáticas.
Facultad 3.

Título: Herramienta para la instalación y configuración de clústeres del Contenedor de Servlets Apache Tomcat.



Trabajo de Diploma para optar por el título de:

INGENIERO EN CIENCIAS INFORMÁTICAS

Autores:
Alberto Jesús La Rosa Agramonte.
Hiran Del Castillo Rodríguez.

Tutores:
Ing. Abel Arias Hernández.
Ing. Daynier Ruiz Rodríguez.

Cuba-Venezuela
2009

Declaración de Autoría:

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

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

Alberto Jesús La Rosa Agramonte.
_____
Firma del Autor

Hiran Del Castillo Rodríguez.
_____
Firma del Autor

Daynier Ruíz Rodríguez.
_____
Firma del Tutor

Abel Arias Hernández.
_____
Firma del Tutor

Opinión del Tutor del Trabajo de Diploma:

“¿El éxito es la meta? Pienso que no. La meta es conseguir, defender y mantener nuestra libertad”.
Richard Stallman.

Agradecimientos

A nuestras familias por sabernos guiar en la vida por el camino correcto.

A nuestros profesores en especial a nuestros tutores Abel Hernández Arias y Daynier Ruiz Rodríguez por su ayuda y apoyo en nuestra formación como profesionales.

A nuestros líderes de proyectos en especial a Yudier Cervantes Puga y Nahuel Massón Padilla por apoyarnos en todo momento de docencia, trabajo y actividades recreativas.

Al proyecto Convenio Cuba Venezuela de manera general por su colaboración en nuestra formación como profesionales.

A nuestros compañeros de grupo y proyecto, en especial a Ariel, Yoan, Abel, Marín, Andy, Yovany, Chung, Cori, Pepe, Martin, Hussein, Alberto y demás.

Dedicatoria:

Para mi madre y mi padre que siempre me han guiado tanto en mi vida personal como profesional y me han dado el tamaño este que tengo, por ser las personas más importantes de mi vida pues gracias a ellos existo en este mundo, gracias a ellos me esfuerzo cada día por ser mejor y que se sientan orgullosos de mí.

Para mi abuela por ser la persona que estuvo a mi lado desde pequeño y por consentirme en todos mis caprichos, por darme su apoyo en todo momento, y por ser alguien por el cual siempre pienso en hacer lo correcto.

Para mi abuelo que en paz descanse, que siempre luchaba porque estudiara y me esforzara para ser alguien en la vida, para que fuera mejor persona cada día y que lograra profesionalmente y personalmente ser mejor que él, que en mi opinión, persona inigualable.

Para mi abuela Milagros que siempre me ha querido con la vida y me ha dado un pedacito de sí misma toda su vida, y por ser una persona excepcional y dedicada en todo momento a su familia.

Para mi abuela Cuchita que a pesar de no ser de mí sangre la considero así, mi abuela, por ser como es con toda su familia y en especial conmigo.

Para mi abuelo Angelito, que cada vez que tengo un problema voy a él a verlo, por ser alguien que se acuerda y pregunta por mí constantemente y que me ha querido toda la vida inigualablemente.

Para todas mis tías, tíos; y mis primas que han sido como los hermanos que no he tenido.

Para mis compañeros de Universidad: Yoan, Abelito, Andy, Yovany, Marín y demás, que siempre se van a acordar de mí por las bromas y cosas que hacíamos juntos.

Para mis hermanos de Universidad: Yuli, Hirán y George que fueron mi apoyo en todo momento, tanto en Venezuela como en Cuba, por apoyarme siempre en todo momento incondicionalmente, por estar estos cinco años juntos llenos de problemas, fiestas y actividades extracurriculares en las farmacias.

Alberto

A la que me dio la vida, por ser tan dulce por ser refugio cuando todo lo demás se ha ido, hoy más que nunca entiendo las palabras en tu foto que siempre llevo conmigo "Esta es la única mujer que siempre te amará incondicionalmente" si esto no fuera breve te escribiría todo el día pero si tengo que resumir lo que eres para mí la palabra correcta es Amor.

A papillón, mi viejo, "Comete lo que te den" por su ideal superador por inspirarme siempre a ser mejor, a luchar, por nunca cansarse y por demostrarme que un hombre falla, reconoce sus errores y se supera, por ser mi mejor amigo por las incontables palabras de aliento y por estar siempre de mi lado aunque no tenga razón.

A mi alma gemela por las canciones que te he escrito y que nunca has escuchado, por estar siempre ausente y a la vez tan presente en cada momento de mi vida, por ir siempre delante y ser mí guía de estudio, por poner siempre los intereses de los tuyos por encima de los propios, por cada segundo juntos, por tu cariño, tu ternura inmensa, este logro te lo dedico muy especialmente a ti mi hermanita.

A Mary y Daly mis segundas madres, a todos mis primos en especial al Guille por los fines de semana donde matábamos el estrés universitario entre cuentos. A todos mis tíos y tías a mis abuelos a toda la familia por ser tan unida y ayudarse unos a los otros, para eso existimos.

A mi gente del barrio que los llevo en el corazón, a Ariel a Gelme a la pandilla eso nunca se olvida.

A todos mis compañeros y compañeras de la Universidad, a los que llegaron y a los que se quedaron en el duro camino.

A las que compartieron su amor conmigo en especial a ti por formar parte de este logro.

A mis amigos de aquí y de toda la vida.

¨vosotros sabéis quien sois¨

Hirán

Resumen:

La Universidad de las Ciencias Informaticas es un centro productivo, su misión es producir software y servicion informáticos, la política universitaria es el uso del software libre para dicho propósito de aquí que sea común el uso de la Tecnología Java por su arquitectura neutral, para brindar soporte a las aplicaciones desarrolladas en java existen diversos entornos de ejecución específicamente Apache Tomcat, también es uno de los más utilizados en la Universidad.

Apache Tomcat puede utilizarse como servidor web autónomo, pudiendose ver vinculado con el servidor web Apache 2 y para sistemas de gran envergadura se estila a utilizar las tecnologías clúster para ganar en rendimiento, seguridad y capacidad de conexiones.

El presente trabajo de diploma propone el desarrollo de una herramienta que implemente las funcionalidades necesarias para garantizar la instalación y configuración de estos clústeres de una manera más eficiente y amigable. El estudio de las tendencias tecnológicas actuales sustentan la decisión de optar por desarrollarlas con software libre, para ello se utilizan Eclipse como IDE y Python como lenguaje de programación. Para el modelado se utiliza la herramienta case Visual Paradigm, y todo el proceso de desarrollo está basado en la metodología RUP. De esta manera surgió una aplicación de escritorio que soluciona la problemática planteada.

Contenido

Introducción.

Antecedentes.

La Universidad de la Ciencias Informáticas es uno de los centros más importantes de producción de software en nuestro país, en ella existen numerosos grupos de proyectos que son los encargados de desarrollar estos productos de software basados en Software Libre, el cual proporciona las siguientes libertades:

- Libertad 0: la libertad para ejecutar el programa sea cual sea nuestro propósito.
- Libertad 1: la libertad para estudiar el funcionamiento del programa y adaptarlo a las necesidades (el acceso al código fuente es condición indispensable para esto).
- Libertad 2: la libertad para redistribuir copias y ayudar al resto.
- Libertad 3: la libertad para mejorar el programa y luego publicarlo para el bien de toda la comunidad (el acceso al código fuente es condición indispensable para esto) [1].

Una de las tecnologías más utilizadas en estos momentos tanto en el mundo como en la Universidad es la Tecnología Java debido a su arquitectura neutral, pues su código fuente es compilado en un código objeto de una máquina con arquitectura independiente, este código es interpretado por una máquina virtual en la arquitectura de destino; implementa estándares de portabilidad adicionales; posee extensiones con capacidades de manejo de redes TCP/IP y rutinas de biblioteca con protocolos de soporte tales como HTTP y FTP; tanto su compilador como el intérprete proveen extensiones para chequeo de errores y administra toda la memoria dinámica, chequea límites de arreglos y otras excepciones; no enlaza módulos invocados hasta el momento de ejecución y una de sus características más importantes es la libertad de su licencia, un elemento puntual para el desarrollo de software en nuestro país, debido a que no se tiene que luchar con los impuestos sobre patentes cada año, además de esto cabe mencionar su independencia de la plataforma y permite crear programas modulares y códigos reutilizables, permitiendo el desarrollo de programas de rendimiento seguro y alto.

Como soporte para el funcionamiento de las aplicaciones desarrolladas en Java existen varios entornos de ejecución como Jboss, GlassFish y Apache Tomcat, este último no es considerado como un servidor de aplicaciones, sino como un contenedor de servlets, pero es capaz de soportar la mayoría de las aplicaciones desarrolladas en dicha tecnología.

Apache Tomcat es uno de los más utilizados actualmente para el despliegue de las aplicaciones desarrolladas por la universidad, primero porque es gratis y además fácil de instalar, se ejecuta en máquinas más pequeñas y es compatible con las API más recientes de Java, es muy fiable por la solidez que le brindan los miles de desarrolladores que contribuyen a su estabilidad.

En el despliegue de los sistemas desarrollados en esta tecnología se pueden utilizar múltiples configuraciones y tareas de administración con Tomcat, que se puede utilizar como servidor web autónomo en entornos con alto nivel de tráfico y disponibilidad, además se puede ver muy vinculado con el servidor web Apache 2 para que este se encargue del contenido estático y Tomcat sólo del dinámico, debido a que Apache es mucho más especializado como servidor web y para sistemas de gran envergadura se estila a utilizar las tecnologías clúster para ganar en rendimiento, seguridad y capacidad de conexiones.

Los clústeres son conjuntos o conglomerados de computadoras, construidos utilizando componentes de hardware comunes jugando un papel importante en la solución de problemas de las ciencias, las ingenierías y aplicaciones comerciales. Han evolucionado para apoyar actividades en aplicaciones que van desde súper cómputo y software de misiones críticas hasta servidores Web, aplicaciones de comercio electrónico y bases de datos de alto rendimiento.

El cómputo en clústeres surge como resultado de la convergencia de varias tendencias que incluyen, la disponibilidad de microprocesadores de alto rendimiento, más económicos y redes de alta velocidad, el desarrollo de herramientas de software para cómputo distribuido de alto rendimiento y la creciente necesidad de potencia computacional para aplicaciones en las ciencias computacionales y comerciales.

Por otro lado, la evolución y estabilidad que ha alcanzado el sistema operativo Linux, ha contribuido al desarrollo de muchas tecnologías nuevas, entre ellas la de clústeres.

La tecnología de clústeres consta de dos partes:

- El primer componente, consta de un sistema operativo confeccionado especialmente para esta tarea (modificaciones al kernel[1] de Linux), un conjunto de compiladores y aplicaciones especiales.

- El segundo componente es la interconexión de hardware entre las máquinas del clúster. Se han desarrollado interfaces de interconexión especiales muy eficientes, pero comúnmente las interconexiones se realizan mediante una red Ethernet dedicada de alta velocidad. Es mediante esta interfaz que las máquinas del clúster intercambian entre sí, asignación de tareas, actualizaciones de estados y datos del programa. Además de esto, existe también otra interfaz de red que conecta al clúster con el mundo exterior.

Los beneficios de construir un clúster están presentes en varios aspectos de diversas aplicaciones y ambientes. Dentro de ellos están el incremento de velocidad de procesamiento, ofrecido por los clústeres de alto rendimiento, del número de transacciones o velocidad de respuesta ofrecida por los clústeres de balance de carga y de confiabilidad ofrecida por los clústeres de alta disponibilidad.

Actualmente en la universidad, en cuanto a este tema existe muy poca experiencia, primeramente porque se lleva migrando a Software Libre hace muy poco tiempo y no existe un conocimiento muy profundo acerca de estas tecnologías, además de los diferentes métodos que se pueden utilizar para implementar un clúster de Apache Tomcat, los cuales tienen diferentes tipos de complejidad y costos.

En cada una de las aplicaciones que se despliegan surge la necesidad de dominar este tipo de tecnologías, específicamente para implementar clústeres de servidores de aplicaciones Java, en este caso de Apache Tomcat para lograr una alta disponibilidad y rendimiento de los sistemas.

Hoy día realizar la instalación y configuración de estos clústeres es algo complicado, debido a que hay que hacerlo manualmente pudiendo contribuir a errores de configuración que afectaría

[1] el **núcleo** (también conocido en español con el anglicismo *kernel,* de la raíz germánica *Kern*) es la parte fundamental de un sistema operativo. Es el software responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora

el funcionamiento de la aplicación o, en este caso, de los servidores. Estos errores de configuración se pudieran eliminar con una herramienta que automatice este proceso de instalación y configuración el cual es muy engorroso para el personal que se encarga del despliegue y soporte dentro de los proyectos productivos.

Problema Científico.

¿Cómo optimizar el proceso de instalación y configuración de clústeres del Contenedor de Servlets Apache Tomcat?

Objeto de Estudio.

Proceso de instalación y configuración de clústeres.

Campo de Acción.

Herramientas para la instalación y configuración de clústeres del Contenedor de Servlets Apache Tomcat.

Objetivo.

Desarrollar una herramienta que facilite la instalación y configuración de clústeres del Contenedor de Servlets Apache Tomcat.

Hipótesis.

Si se desarrolla una herramienta que automatice el proceso de instalación y configuración de clústeres del Contenedor de Servlets Apache Tomcat, se logrará un óptimo funcionamiento de los mismos y de la aplicación desarrollada en Java.

Tareas de la Investigación.

1. Estudiar del estado del arte de clústeres de servidores.
2. Investigar sobre el balanceo de carga de clústeres y cuáles son las aplicaciones más utilizadas en este campo.
3. Buscar información acerca la instalación y configuración de clústeres del Contenedor de Servlets Apache Tomcat.
4. Estudiar el lenguaje Bash para sistemas GNU Linux, específicamente en Debian Etch.

5. Desarrollar una herramienta en lenguaje Bash, Python y GTK, para el logro de una mejor integración con los servidores Linux.
6. Utilizar la herramienta en el despliegue de una aplicación Java.
7. Brindar una interfaz visual que presente un fácil uso para la mayoría de los usuarios.
8. Garantizar la seguridad de todas las conexiones que se establezcan entre la herramienta y el servidor.
9. Asegurar la aplicación Java con certificados digitales a elección del cliente.

Métodos Científicos.

Para validar metodológicamente la investigación se utiliza los siguientes métodos:

Teóricos:

- *analítico – sintético:* sirvió como punto de partida en la investigación debido a que ayudó a identificar dentro del grupo de necesidades de los arquitectos del proyecto el desarrollo de una herramienta capaz de facilitar el trabajo de los mismos en la instalación y configuración de los clústeres de Apache Tomcat, que hasta el momento se hacía de forma manual.
- *inductivo – deductivo:* permitió identificar la necesidad de una herramienta para la automatización del proceso de instalación y configuración de los clústeres del Servidor de Aplicaciones Apache Tomcat a partir de los errores que se cometían en la instalación y la configuración de los mismos en la forma tradicional, es decir manualmente.
- *histórico – lógico:* permitió definir los principales errores en la instalación y configuración de los clústeres de Servidores de Aplicaciones Apache Tomcat a partir de las experiencias de administradores de red y desarrolladores de software permitiendo corregir los mismos a través de la herramienta para lograr una mayor eficiencia en el proceso de instalación y en el funcionamiento de los mismos.
- *hipotético – deductivo:* dio la posibilidad de identificar los principales factores que ayudarían a mejorar el tradicional proceso de instalación y configuración de los clústeres de Servidores de Aplicaciones de Apache Tomcat a través de una herramienta que elimine todos los errores de instalación y configuración.

Empíricos:

- *observación:* permitió tener una idea lo más objetiva posible de la situación y como iba siendo su evolución. Supone además, la interacción social entre el investigador y el objeto de investigación donde el objetivo es recoger datos, de modo sistemático, a través del contacto directo, en situaciones específicas.

Estructura del Trabajo.

En el capítulo 1 se hace referencia a la fundamentación teórica, abordando acerca de las teconologías de clústeres que existen, sus clasificaciones, beneficios e importancia. Además se mencionan las variantes para la instalación de clústeres de alta disponibilidad y como es que se realiza el proceso de instalación y configuración de las mismas de forma manual.

En el capítulo 2 se abordan las tecnologías, metodologías y entornos de desarrollo para la implementación de la herramienta, los procesos del negocio a través de un modelo de negocio, y a partir de esto se comienza hacer el análisis del sistema a desarrollar. Además se identifican y refinan los requisitos funcionales definidos, los cuales están implícitos en los casos de uso del sistema, y se describen detalladamente. Se muestran los diagramas de clases del análisis, del diseño para cada realización de caso(s) de uso(s). Además se describen las clases utilizadas en el diseño.

En el capítulo 3 se describen las pruebas realizadas al software, como las pruebas de caja negra y caja blanca y el análisis de los resultados, la implementación de la herramienta y el diagrama de despliegue.

1. Fundamentación Teórica.

1.1 Introducción.

El presente capítulo describe en qué consisten las tecnologías clústeres, los tipos existentes, cada uno de los elementos necesarios para implementarlas, los beneficios y diferentes ambientes en que pueden funcionar. Se profundiza en la tecnología de clústeres para balanceo de carga de aplicaciones desarrolladas en tecnología Java que es la base de nuestro problema.

1.2 Historia.

El origen del término clúster y el uso de este tipo de tecnología son desconocidos pero se puede considerar que comenzó a finales de los años 50 y principios de los años 60.

La base formal de la ingeniería informática de la categoría como un medio de hacer trabajos paralelos de cualquier tipo fue posiblemente inventado por Gene Amdahl[2] de IBM, que en 1967 publicó lo que ha llegado a ser considerado como el papel inicial de procesamiento paralelo: la Ley de Amdahl que describe matemáticamente el aceleramiento que se puede esperar, paralelizando cualquier otra serie de tareas realizadas en una arquitectura paralela [2].

Este artículo define la base para la ingeniería de la computación tanto multiprocesador y computación clúster, en donde el principal papel diferenciador es, si las comunicaciones interprocesador cuentan con el apoyo "dentro" de la computadora (por ejemplo, en una configuración personalizada para el bus o la red de las comunicaciones internas) o "fuera" del ordenador en una red "commodity".

En consecuencia, la historia de los primeros grupos de computadoras está ligada a la historia de principios de las redes, como una de las principales motivaciones para el desarrollo de una red para enlazar los recursos de computación, de hecho la creación de un clúster de computadoras. Las redes de conmutación de paquetes fueron conceptualmente inventados por la corporación RAND en 1962 [2].

Utilizando el concepto de una red de conmutación de paquetes, el proyecto ARPANET logró crear en 1969 lo que fue posiblemente la primera red de computadoras básico, basadas en el clúster de computadoras por cuatro tipos de centros informáticos, cada una de las cuales fue algo similar a un clúster, pero no un comodity clúster como hoy en día se entiende.

[2] Americano de origen noruego, arquitecto computacional y una de las personalidades más importantes y excéntricas de la historia de la informática y computación. Fundó cuatro compañías tecnológicas en diferentes ámbitos, dentro de ellas la famosa empresa informática IBM.

El proyecto ARPANET creció y se convirtió en lo que es ahora Internet, que se puede considerar como la madre de todos los clústeres, como la unión de casi todos los recursos de cómputo, incluidos los clústeres, que pasarían a ser conectados.

También estableció el paradigma de uso de computadoras clústeres en el mundo de hoy, el uso de las redes de conmutación de paquetes para realizar las comunicaciones, entre procesadores localizados en los marcos de otro modo desconectado.

El desarrollo de la construcción de computadoras por los clientes y grupos de investigación procedió a la par con el surgimiento de las redes y el sistema operativo Unix desde principios de la década de los años 70, como TCP/IP y el proyecto de la Xerox PARC proyecto y formalizado para protocolos basados en la red de comunicaciones.

El núcleo del sistema operativo fue construido por un grupo de DEC PDP-11 minicomputadoras llamado C.mmp en C-MU en 1971.

Sin embargo, no fue hasta alrededor de 1983 que los protocolos y herramientas para el trabajo remoto facilitasen la distribución y el uso compartido de archivos fueran definidos, en gran medida dentro del contexto de BSD Unix, e implementados por Sun Microsystems, y por tanto llegar a disponerse comercialmente, junto con una compartición del sistema de ficheros.

El primer producto comercial de tipo clúster fue ARCnet, desarrollada en 1977 por Datapoint pero no obtuvo un éxito comercial y los clústeres no consiguieron tener éxito hasta que en 1984 VAXcluster produjeran el sistema operativo VAX/VMS [2].

El ARCnet y VAXcluster no sólo son productos que apoyan la computación paralela, también comparten los sistemas de archivos y dispositivos periféricos. La idea era proporcionar las ventajas del procesamiento paralelo, al tiempo que se mantiene la fiabilidad de los datos y el carácter singular. VAXcluster, VMScluster está todavía disponible en los sistemas de HP OpenVMS corriendo en sistemas Itanium y Alpha.

Otros dos principios comerciales de clústeres notables fueron el Tandem Himalaya, alrededor de 1994 con productos de alta disponibilidad, y el IBM S/390 Parallel Sysplex también alrededor de 1994, principalmente para el uso de la empresa.

La historia de los clústeres de computadoras estaría completa sin señalar el papel fundamental desempeñado por el desarrollo del software de Parallel Virtual Machine (PVM).

Este software de fuente abierta basado en comunicaciones TCP/IP permitió la creación de un superordenador virtual, un clúster HPC, realizada desde cualquiera de los sistemas conectados TCP/IP.

De forma libre los clústeres heterogéneos han constituido la cima de este modelo logrando aumentar rápidamente en FLOPS globalmente y superando con creces la disponibilidad incluso de los más caros superordenadores.
PVM y el empleo de computadoras y redes de bajo costo llevó, en 1993, a un proyecto de la NASA para construir supercomputadoras de clústeres.
En 1995, la invención de la "beowulf", un estilo de clúster, una granja de computación diseñado en base a un producto básico de la red con el objetivo específico de "ser un superordenador" capaz de realizar firmemente cálculos paralelos HPC [2].
Esto estimuló el desarrollo independiente de la computación Grid como una entidad, a pesar de que el estilo Grid giraba en torno al del sistema operativo Unix y el Arpanet.

1.3 Tecnología Clúster.

Los clùsteres son conjuntos o conglomerados de computadoras construidos mediante la utilización de componentes de hardware y software comunes y que se comportan como si fuesen una única computadora.
Juegan hoy en día un papel importante en la solución de problemas de las ciencias, las ingenierías y del comercio moderno.
La tecnología de clústeres ha evolucionado en apoyo de actividades que van desde aplicaciones de súper cómputo y software de misiones críticas, servidores Web y comercio electrónico, hasta bases de datos de alto rendimiento, entre otros usos [3].
El cómputo con clústeres surge como resultado de la convergencia de varias tendencias actuales que incluyen la disponibilidad de microprocesadores económicos de alto rendimiento y redes de alta velocidad, el desarrollo de herramientas de software para cómputo distribuido de alto rendimiento, así como la creciente necesidad de potencia computacional para aplicaciones que la requieran.
Simplemente, un clúster es un grupo de múltiples ordenadores unidos mediante una red de alta velocidad, de tal forma que el conjunto es visto como un único ordenador, más potente que los comunes de escritorio [3].

1.3.1 Beneficios de la Tecnología Clúster.

Las aplicaciones paralelas escalables requieren: buen rendimiento, baja latencia, comunicaciones que dispongan de gran ancho de banda, redes escalables y acceso rápido a

los archivos. Un clúster puede satisfacer estos requerimientos usando los recursos que tiene asociados a él.

Los clústeres ofrecen las siguientes características a un costo relativamente bajo:

- Alto Rendimiento.
- Alta Disponibilidad.
- Alta Eficiencia.
- Escalabilidad.

La tecnología clúster permite a las organizaciones incrementar su capacidad de procesamiento usando tecnología estándar, tanto en componentes de hardware como de software que pueden adquirirse a un costo relativamente bajo.

1.3.2 Clasificación de Clústeres.

El término clúster tiene diferentes connotaciones para diferentes grupos de personas. Los tipos de clústeres, establecidos en base al uso que se dé a los clústeres y los servicios que ofrecen, determinan el significado del término para el grupo que lo utiliza. Los clústeres pueden clasificarse en base a sus características:
Se pueden tener clústeres de alto rendimiento (HPC – High Performance Clusters), clústeres de alta disponibilidad (HA – High Availability) o clústeres de alta eficiencia (HT – High Throughput).

Alto rendimiento: Son clústeres en los cuales se ejecutan tareas que requieren de gran capacidad computacional, grandes cantidades de memoria, o ambos a la vez. El llevar a cabo estas tareas puede comprometer los recursos del clúster por largos periodos de tiempo.

Alta disponibilidad: Son clústeres cuyo objetivo de diseño es el de proveer disponibilidad y confiabilidad. Estos clústeres tratan de brindar la máxima disponibilidad de los servicios que ofrecen. La confiabilidad se provee mediante software que detecta fallos y permite recuperarse frente a los mismos, mientras que en hardware se evita tener un único punto de fallos.

Alta eficiencia: Son clústeres cuyo objetivo de diseño es el ejecutar la mayor cantidad de tareas en el menor tiempo posible. Existe independencia de datos entre las tareas individuales. El retardo entre los nodos del clúster no es considerado un gran problema [3].

Los clústeres pueden también clasificar como Clústeres Comerciales (Alta disponibilidad, Alta eficiencia) y Clústeres Científicos (Alto rendimiento). A pesar de las discrepancias a nivel de requerimientos de las aplicaciones, muchas de las características de las arquitecturas de hardware y software, que están por debajo de las aplicaciones en todos estos clústeres, son las mismas. Más, un clúster de determinado tipo, puede también presentar características de los otros.

En el caso presentado se decidió implementar un clúster de alta disponibilidad debido a que no puede haber fallas de ningún tipo en el funcionamiento, ni en la estabilidad de los sistemas, ya sea por carga de conexiones o por falla de algunos de los servidores pertenecientes al clúster.

1.3.3 Componentes de un Clúster.

En general, un clúster necesita de varios componentes de software y hardware para poder funcionar.

- Nodos
- Sistemas Operativos
- Conexiones de Red
- Middleware
- Protocolos de Comunicación y Servicios
- Aplicaciones
- Ambientes de Programación Paralela

Nodos: Pueden ser simples ordenadores, sistemas multiprocesador o estaciones de trabajo. En informática, de forma muy general, un nodo es un punto de intersección o unión de varios elementos que confluyen en el mismo lugar. Ahora bien, dentro de la informática la palabra nodo puede referirse a conceptos diferentes según el ámbito [3].

En redes de computadoras cada una de las máquinas es un nodo, y si la red es Internet, cada servidor constituye también un nodo. En estructuras dinámicas de datos, un nodo es un registro que contiene un dato de interés y al menos un puntero para referenciar a otro nodo. Si la estructura tiene sólo un puntero, la única estructura que se puede construir con él es una lista, si el nodo tiene más de un puntero ya se pueden construir estructuras más complejas como árboles o grafos.

El clúster puede estar conformado por nodos dedicados o por nodos no dedicados.

En un clúster con nodos dedicados, los nodos no disponen de teclado, ni mouse, ni monitor y su uso está exclusivamente dedicado a realizar tareas relacionadas con el clúster. Mientras que, en un clúster con nodos no dedicados, los nodos disponen de teclado, mouse, monitor y el uso no está exclusivamente dedicado a realizar tareas relacionadas con el clúster, el cual hace uso de los ciclos de reloj que el usuario del computador no está utilizando para realizar sus tareas.

Cabe aclarar que a la hora de diseñar un Clúster, los nodos deben tener características similares, es decir, deben guardar cierta similitud de arquitectura y sistemas operativos, ya que si se conforma un clúster con Nodos totalmente heterogéneos (existe una diferencia grande entre capacidad de procesadores, memoria, HD) será ineficiente debido a que el middleware delegará o asignará todos los procesos al Nodo de mayor capacidad de Computo y sólo distribuirá cuando este se encuentre saturado de procesos; por eso es recomendable que los nodos sean lo más similares posible.

Sistema Operativo: Debe ser multiproceso, multiusuario. Otras características deseables son la facilidad de uso y acceso y permitir además múltiples procesos y usuarios. Un sistema operativo es un programa o conjunto de programas de computadora, destinado a permitir una gestión eficaz de sus recursos. Comienza a trabajar cuando se enciende el computador, y gestiona el hardware de la máquina desde los niveles más básicos, permitiendo también la interacción con el usuario.

Un sistema operativo se puede encontrar normalmente en la mayoría de los aparatos electrónicos que utilicen microprocesadores para funcionar, ya que gracias a estos se puede entender la máquina y que ésta cumpla con sus funciones (teléfonos móviles, reproductores de DVD, autorradios y computadoras).

Ejemplos: GNU/Linux, OpenMosix, Rocks, Kerrighed, Unix, Solaris, HP-Ux, Aix, Windows NT-2000 Server-2003 Server-2008 Server, Mac OS X, FreeBSD.

Conexiones de Red: Los nodos de un clúster pueden conectarse mediante una simple red Ethernet con placas comunes (adaptadores de red o NICs), o utilizarse tecnologías especiales de alta velocidad como Fast Ethernet, Gigabit Ethernet, Myrinet, Infiniband, SCI, etc.

1. *Ethernet.*

Son las redes más utilizadas en la actualidad, debido a su relativo bajo costo de instalación y flexibilidad. No obstante, su tecnología limita el tamaño del paquete, realizan excesivas comprobaciones de error, sus protocolos no son eficientes, y sus velocidades de transmisión pueden limitar el rendimiento de los Clústeres. Para aplicaciones con paralelismo de gran grueso puede suponer una solución acertada.

La opción más utilizada en la actualidad es Gigabit-Ethernet (1000Mbps), siendo emergente la solución 10Gigabit-Ethernet. La latencia de estas tecnologías está en torno a los 30-100 us, dependiendo del protocolo de comunicación empleado.

En todo caso, Ethernet es la red de administración por excelencia, así que aunque no sea la solución de red de altas prestaciones para las comunicaciones, es la red dedicada a las tareas administrativas.

2. *Myrinet (Myrinet 2000 y Myri-10G)*

Su latencia es de 1,3/10 μs, y su ancho de Banda de 2/10Gbps, respectivamente para Myrinet 2000 y Myri-10G.

Es la red de baja latencia más utilizada en la actualidad, tanto en clusters como en MPPs estando presente en más de la mitad de los sistemas del top500. Tiene dos bibliotecas de comunicación a bajo nivel (GM y MX). Sobre estas bibliotecas están implementadas MPICH-GM, MPICH-MX, Sockets-GM y Sockets MX, para aprovechar las excelentes características de Myrinet. Existen también emulaciones IP sobre TCP/IP, IPoGM e IPoMX.

3. *Infiniband.*

Es una red surgida de un estándar desarrollado específicamente para realizar la comunicación en clústeres. Una de sus mayores ventajas es que mediante la agregación de canales (x1, x4 y x12) permite obtener anchos de banda muy elevados. La conexión básica es de 2Gbps efectivos y con ‘quad connection’ x12 alcanza los 96Gbps. No obstante, los startups no son muy altos, se sitúan en torno a los 10 μs.

Define una conexión entre un nodo de computación y un nodo de I/O. La conexión va desde un Host Channel Adapter (HCA) hasta un Target Channel Adapter (TCA). Se está usando principalmente para acceder a arrays de discos SAS.

4. *SCI (Scalable Coherent Interface) IEEE standard 1596- 1992.*

Su latencia teórica es de 1.43 µs y su ancho de banda de 5333 Mbps bidireccional. Al poder configurarse con topologías de anillo (1D), toro (2D) e hipercubo (3D) sin necesidad de switch, se tiene una red adecuada para clústeres de pequeño y mediano tamaño.

Al ser una red de extremadamente baja latencia, presenta ventajas frente a Myrinet en clústeres de pequeño tamaño al tener una topología punto a punto y no ser necesaria la adquisición de un conmutador. El software sobre SCI está menos desarrollado que sobre Myrinet, pero los rendimientos obtenidos son superiores, destacando SCI Sockets (que obtiene startups de 3 microsegundos) y ScaMPI, una biblioteca MPI de elevadas prestaciones.

Además, mediante el mecanismo de pre-loading (LD_PRELOAD) se puede conseguir que todas las comunicaciones del sistema vayan a través de SCI-SOCKETS (transparencia para el usuario).

5. *Middleware.*

El middleware es un software que generalmente actúa entre el sistema operativo y las aplicaciones con la finalidad de proveer a un clúster lo siguiente:

- Una interfaz única de acceso al sistema, denominada SSI (Single System Image), la cual genera la sensación al usuario de que utiliza un único ordenador muy potente.

- Herramientas para la optimización y mantenimiento del sistema: migración de procesos, checkpoint-restart (congelar uno o varios procesos, mudarlos de servidor y continuar su funcionamiento en el nuevo host), balanceo de carga, tolerancia a fallos, etc.
- Escalabilidad: debe poder detectar automáticamente nuevos servidores conectados al cluster para proceder a su utilización.

Existen diversos tipos de middleware, como por ejemplo: MOSIX, OpenMOSIX, Cóndor, y OpenSSI.

El middleware recibe los trabajos entrantes al clúster y los redistribuye de manera que el proceso se ejecute más rápido y el sistema no sufra sobrecargas en un servidor. Esto se realiza mediante políticas definidas en el sistema (automáticamente o por un administrador) que le indican dónde y cómo debe distribuir los procesos, por un sistema de monitorización, el cual controla la carga de cada CPU y la cantidad de procesos en él.

El middleware también debe poder migrar procesos entre servidores con distintas finalidades:

- ***Balancear la carga*:** si un servidor está muy cargado de procesos y otro está ocioso, pueden transferirse procesos a este último para liberar de carga al primero y optimizar el funcionamiento.
- ***Mantenimiento de servidores*:** si hay procesos corriendo en un servidor que necesita mantenimiento o una actualización, es posible migrar los procesos a otro servidor y proceder a desconectar del clúster al primero.
- ***Priorización de trabajos*:** en caso de tener varios procesos corriendo en el clúster, pero uno de ellos de mayor importancia que los demás, puede migrarse este proceso a los servidores que posean más o mejores recursos para acelerar su procesamiento.

Ambientes de Programación Paralela: Los ambientes de programación paralela permiten implementar algoritmos que hagan uso de recursos compartidos: CPU (Central Processing Unit), memoria, datos y servicios.

1.4 Elementos necesarios para implementar un clúster de Alta Disponibilidad de Apache Tomcat.

1.4.1 Balanceadores de carga.

Los balanceadores de carga son servicios utilizados para compartir el trabajo a realizar varios procesos u ordenadores, íntimamente ligados a los sistemas de multiprocesamiento o que hacen uso de varias unidades de procesamiento para realizar determinadas labores.
Existen varios métodos para el balanceo de carga, la elección de los mismos va en dependencia de los requerimientos, la complejidad y el costo. Dentro de ellos se tienen:

- **Round Robin DNS**: simple, poco costoso y fácil de implementar.
- **Hardware**: provee una robusta topología con alta disponibilidad pero muy costoso.
- **Software:** menos costosos, más configurables en base a los requerimentos y pueden incorporar un ruteo inteligente, basado en parámetros.

1.4.2 Servidores de Aplicaciones.

Un servidor de aplicaciones es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general a través de Internet y utilizando el protocolo http. Los servidores de aplicación se distinguen de los servidores web por el uso extensivo del contenido dinámico y por su frecuente integración con bases de datos, además es un producto basado en un componente que se encuentra en el plano medio de la arquitectura central de un servidor y trabaja como un intermediario para la seguridad y el mantenimiento, y además de proveer acceso a los datos. Maneja la mayoría de las transacciones relacionadas con la lógica y el acceso a los datos de la aplicación.

Los servidores de aplicación típicamente incluyen también middleware (o software de conectividad) que les permite intercomunicarse con variados servicios, para efectos de confiabilidad, seguridad, no-repudio, etc. Los servidores de aplicación también brindan a los desarrolladores una Interfaz para Programación de Aplicaciones (API), de tal manera que no tengan que preocuparse por el sistema operativo o por la gran cantidad de interfaces requeridas en una aplicación web moderna.

***Servidores de aplicación J2EE**.*

Como consecuencia del éxito del lenguaje de programación Java, el término servidor de aplicaciones usualmente hace referencia a un servidor de aplicaciones J2EE. WebSphere (IBM), Oracle Application Server (Oracle Corporation) y WebLogic (BEA) están entre los servidores de aplicación J2EE privativos más conocidos. EAServer (Sybase Inc.) es también conocido por ofrecer soporte a otros lenguajes diferentes a Java, como PowerBuilder. El servidor de aplicaciones JOnAS, desarrollado por el consorcio ObjectWeb, fue el primer servidor de aplicaciones libre en lograr certificación oficial de compatibilidad con J2EE. JBoss es otro servidor de aplicaciones libre y muy popular en la actualidad, así como el GlassFish de SUN y Tomcat (The Apache Software Foundation) aunque también es llamado solamente un contenedor de servlets.

J2EE provee estándares que le permiten a un servidor de aplicaciones servir como "contenedor" de los componentes que conforman dichas aplicaciones. Estos componentes, escritos en lenguaje Java, usualmente se conocen como Servlets, Java Server Pages (JSPs) y Enterprise JavaBeans (EJBs) y permiten implementar diferentes capas de la aplicación, como

la interfaz de usuario, la lógica de negocio, la gestión de sesiones de usuario o el acceso a bases de datos remotas.

La portabilidad de Java también ha permitido que los servidores de aplicación J2EE se encuentren disponibles sobre una gran variedad de plataformas, como Unix, Microsoft Windows y GNU/Linux.

Se utilizará Apache Tomcat 5.5 debido a que con todas las expectativas de las aplicaciones Java que se puedan desarrollar. Contiene soporte de servlets y JSPs, incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets y su motor de servlets a menudo se presenta en combinación con el servidor Web Apache 2. Implementado a partir de las especificaciones Servlet 2.4 y JSP 2.0, presenta recolección de basura reducida, con análisis rápido JSP y diseñado para funcionar en Java SE 5.0 y posteriores. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

1.4.3 Sistema Operativo de los Servidores.

La utilización de un sistema operativo basado en Unix, en este caso las distribuciones Linux, Debian y Ubuntu fue debido a que superaron en los siguientes campos a los famosos servidores Windows NT/2000 Advanced Server/2003 Advanced Server:

- **Seguridad.**
  - Ya que la gran mayoría de los ataques de hackers son dirigidos a servidores Windows al igual que los virus los cuales se enfocan principalmente a servidores con éste sistema operativo.
  - La plataforma Linux es más robusta lo cual hace más difícil que algún intruso pueda violar el sistema de seguridad de Linux.
- **Eficiencia y estabilidad.**
  - Al tener una plataforma más estable, esto favorece el desempeño de aplicaciones de todo tipo tales como: bases de datos, aplicaciones XML, multimedia, etc.
  - La eficiencia de su código fuente hace que la velocidad de las aplicaciones Linux sean superiores a las que corren sobre Windows.

- **Costo.**
  - Ya que requieren menor mantenimiento. En servidores Windows es más costoso debido a que es necesaria una frecuente atención y monitoreo contra ataques de virus, hackers y errores de código, instalación y actualización de parches y service packs.
  - El software Linux así como también un sin número de aplicaciones son de código abierto.
  - No requieren supervisión tan estrecha ni pagos de pólizas de mantenimiento necesarias para obtener los Service Packs.

1.5 Seguridad.

En cuanto al tema de la seguridad de la aplicación Java contenida en los nodos Apache Tomcat se decidió utilizar un certificado digital que garantizará un nivel de encriptación (1024 bits) determinado para la información que se va a manejar en la aplicación.

1.5.1 Certificados Digitales.

Un Certificado Digital es un documento digital mediante el cual un tercero confiable (una autoridad de certificación) garantiza la vinculación entre la identidad de un sujeto o entidad y su clave pública.

Si bien existen varios formatos para certificados digitales, los más comúnmente empleados se rigen por el estándar UIT-T X.509. El certificado contiene usualmente el nombre de la entidad certificada, número de serie, fecha de expiración, una copia de la clave pública del titular del certificado, (utilizada para la verificación de su firma digital) y la firma digital de la autoridad emisora del certificado, de forma que el receptor pueda verificar que esta última ha establecido realmente la asociación.

Dentro de las herramientas para gestionar estos certificados están:

- OpenSSL
- KeyTool

Se utilizó OpenSSL debido a que es un robusto paquete de herramientas de administración y librerías relacionadas con la criptografía, que suministran funciones criptográficas a otros paquetes como OpenSSH y navegadores web (para acceso seguro a sitios HTTPS).

Estas herramientas ayudan al sistema a implementar el Secure Sockets Layer (SSL), así como otros protocolos relacionados con la seguridad, como el Transport Layer Security (TLS).

Este paquete de software es importante para cualquiera que esté planeando usar cierto nivel de seguridad en su máquina, con un sistema operativo Libre basado en GNU/Linux. OpenSSL también permite crear certificados digitales que se pueden aplicar a nuestro servidor, en nuestro caso Apache 2.

1.6 Proceso de instalación del clúster de servidores Apache Tomcat.

De los métodos anteriormente expuestos para el balanceo de carga debido al costo, los requerimientos y el tipo de implementación, se utilizará el balanceo de carga a nivel de software.

Como balanceador de carga para los nodos Apache Tomcat se eligió Apache 2 debido a su estructura modular y lo extensible que se puede convertir por los diversos módulos que se le pueden integrar, que aportan muchas de las funcionalidades que podrían considerarse básicas para un servidor web, y en el caso de que se decidiera poner en funcionamiento otros balanceadores Apache 2 para mayor disponibilidad, se decidió utilizar los llamados Linux Virtual Server que su funcionamiento es basado en una dirección IP virtual, la cual responde a cada uno de los ip de los balanceadores Apache y a diferencia del balanceo por DNS si detecta cuando uno de los servidores esta fuera de servicio.
Dentro de los módulos de Apache 2 que fueron de utilidad están:

- mod_ssl: Comunicaciones Seguras vía TLS o SSL, el cual permite la utilización de Certificados Digitales.

- mod_jk: Es un conector que permite enlazar a Apache 2 con servidores de aplicaciones J2EE, introduciendo una mayor gestión en las conexiones de los clientes y mayor seguridad en las transacciones del sistema. Así mismo se pueden enlazar varias instancias al servidor web, permitiendo así una mayor tolerancia a errores y aligerar la

carga en los servidores J2EE, además permite la réplica de sesiones lo cual es un elemento importantísimo para el sistema.

1.6.1 Instalando servicios para el funcionamiento del clúster.

Primeramente se debe instalar el servidor Apache Tomcat en cada unos de los nodos del clúster, para eso se debe instalar la Maquina Virtual de Java (JVM) que permitirá que el servicio funcione debido a que está desarrollado en Java, se debe recordar que si se está trabajando sobre Debian GNU/Linux se debe habilitar los repositorios backports para poder instalar la versión 6 de la misma, para ello se ejecuta el siguiente comando en la consola:
>> apt-get install sun-java6-jre sun-java6-fonts.

Luego se instala el compilador de servlets ANT:
>> apt-get install ant

Y se configura previamente el fichero /etc/default/tomcat5.5 **(Ver anexo 4)** para que el Tomcat reconozca donde está la JVM, se deshabilite la seguridad del Tomcat para la correcta ejecución de los servlets y se configuran los parámetros de memoria que va a utilizar.

Ahora ya se pasa a la instalación del Tomcat:
>> apt-get install tomcat5.5 tomcat5.5-webapps tomcat5.5-admin

Una vez ejecutada esta línea de comandos durante el proceso se alertará de que ya existe un fichero tomcat5.5 en /etc/default/, y se debe decir que se desea conservar el existente, de otra manera no funcionaría correctamente el servidor. Si se procedió correctamente ya se puede pasar a configurar el Tomcat para que funcione como un nodo de un clúster.

Para ello se modifican varios ficheros dentro de la configuración del Tomcat. Inicialmente se modifica el fichero /var/lib/tomcat5.5/conf/context.xml **(Ver Anexo 4)** cambiando el valor del parámetro ¨distributable¨ a ¨true¨. Luego el fichero /var/lib/tomcat5.5/conf/server.xml **(Ver Anexo 4)**, que es él que va a garantizar el funcionamiento del Tomcat como nodo del clúster. Posteriormente se reinicia el servidor Apache Tomcat para que las configuraciones tengan efecto. Esta configuración permite que las sesiones puedan ser replicadas entre los nodos en caso de fallas por lo que al más mínimo error el clúster puede que no funcione correctamente.

Culminada la instalación y configuración de los nodos Apache Tomcat, se pasa a la configuración del, o los balanceadores de carga, que en este caso sería el Apache 2.

Primeramente se instala el Apache y los módulos que serán de utilidad, para ello se ejecuta el siguiente comando:

>> apt-get install apache2 libapache2-mod-jk mod-ssl

Se activan los módulos pues sólo han sido instalados y para su funcionamiento se deben habilitar:

>> a2enmode ssl //en caso de que se vaya a utilizar Certificados Digitales para la aplicación a desplegar.

>> a2enmode jk

Se crea y configura los ficheros necesarios para el funcionamiento del, o los balanceadores.

Primeramente se crea el fichero /etc/apache2/workers.properties **(Ver Anexo 4)** que es el que tendrá la información de los nodos Tomcat.

Se edita el fichero /etc/apache2/httpd.conf **(Ver Anexo 4)**.

Y se reinicia el servidor Apache para que los cambios tengan efecto:

>> /etc/init.d/apache2 reload.

La correcta configuración de los balanceadores es muy importante debido a que estos son los que controlan cada uno de los nodos, encargándose de que cada nodo tenga la misma carga de trabajo y que en caso de que falle alguno, distribuye el trabajo de este hacia los demás.

En caso de que se decida utilizar Certificados Digitales, primeramente se debe generar el certificado de la siguiente manera, el cual se copiará dentro de la carpeta ssl dentro del directorio de configuración de Apache:

>> openssl genrsa -out cert.key 1024.

>> openssl req -new -x509 -days 365 -key cert.key -out cert.crt.

>> cat cert.key cert.crt > cert.pem

Para luego crear el Virtual Host **(Ver Anexo 4)** en el Apache 2 que contendrá dicho certificado.

Luego se reinicia nuevamente el Apache 2 y la aplicación hosteada en el clúster estará funcionando mediante conexiones seguras.

Si se decide utilizar más de un balanceador de carga Apache 2 se tendría que instalar un LVS que será el encargado de controlar cada uno de los balanceadores en caso de fallas de hardaware o software, redireccionando las conexiones al próximo balanceador disponible.

Para ello primeramente se instalan los paquetes necesarios para montar el LVS:

>> apt-get install ipvsadm ldirectord heartbead

Ahora se comienzan a configurar cada uno de los ficheros correspondientes. Primeramente se configura el fichero /etc/sysctl.conf **(Ver Anexo 4)** habilitando los parámetros necesarios.

Ahora se debe activar una interfaz de redirección local para ello se adiciona en el fichero de interfaces la nueva interfaz **(Ver Anexo 4).**

Y se ejecuta los siguientes comandos para activar la interfaz y las configuraciones del fichero sysctl.conf:

>> /sbin/ifup lo:0

>> /sbin/sysctl -p

Se crean los ficheros de configuración /etc/ha.d/ha.cf**,** /etc/ha.d/ldirectord, /etc/ha.d/haresources, /etc/ha.d/authkeys y /etc/ha.d/ldirectord **(Ver Anexo 4),** del heartbeat configurando los parámetros necesarios, definiéndose el nodo maestro y la cantidad de nodos que tendrá para controlar de Apache 2.

Por ultimo se reinicia el servicio del heartbeat para que funcione nuestro balanceador.

>> /etc/init.d/ldirectord stop //debido a que cuando se inicie el heartbeat, este lo levanta.

>> /etc/init.d/heartbeat start

1.7 Conclusiones.

En este capítulo quedaron reflejados los principales conceptos relacionados con el problema, ampliando así los conocimientos para comprender mejor los procesos de negocio que ocurren en la entidad. Se hizo un explicación de cómo es el proceso de instalación y configuración de cada uno de los servicios, mostrando cada uno de los ficheros de configuración.

2. Solución Propuesta

2.1 Introducción.

El presente capítulo describe las tecnologías y la metodología utilizada para el desarrollo de la herramienta, además se comienza a describir el proceso de negocio, definir los requisitos funcionales y no funcionales, se definen las clases de análisis y diseño incluyendo las realizaciones de cada uno de los casos de uso que contiene la herramienta.

2.2 Tecnologías utilizadas para el desarrollo de la herramienta.

2.2.1 Lenguaje Python.

La utilización de Python se debe a que permite dividir el programa en módulos reutilizables desde otros programas Python. Incluye una gran colección de módulos estándar que se pueden utilizar como base de los programas. Además incluye módulos que proporcionan E/S de ficheros, llamadas al sistema, sockets y hasta interfaces a GUI como Tk, GTK, Qt; además contiene módulos para realizar conexiones SSH lo cual es de gran utilidad para la copia y ejecución de scripts Bash en los servidores remotos. Python ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. Permite mantener de forma sencilla la interacción con el sistema operativo, y resulta muy adecuado para manipular archivos.

2.2.2 Bash Script.

La utilización de Bash Script proporcionará la creación de los scripts de instalación y configuración de cada uno de los servicios que se necesita para el funcionamiento de los clústeres de Apache Tomcat, los que serán copiados y ejecutados en cada uno de los servidores. Además permite el paso de parámetros a la hora de ejecutar un script, lo cual permite la creación de scripts mucho más eficientes y dinámicos.

2.2.3 GTK.

La utilización de la biblioteca GTK ayudó a la implementación de todas las interfaces de usuario de una manera sencilla, rápida y eficiente a través de la herramienta Glade. GTK proporciona entornos gráficos mucho más ligeros y óptimos en cuanto a rendimiento que la biblioteca Qt, además es una de las más utilizadas para el desarrollo de herramientas de administración,

navegadores como firefox, entornos de escritorio como gnome, xfce, los cuales son unos de los que más rendimiento tienen dentro del mundo de Linux. GTK se acopla a Python de una manera increíble proporcionando el desarrollo rápido de aplicaciones gráficas potentes, a través del módulo PyGTK.

Se basa en varias bibliotecas del equipo de GTK+ y GNOME:

● **GLib.** Biblioteca de bajo nivel y estructura básica de GTK+ y GNOME. Proporciona el manejo de estructuras de datos para C, portabilidad, interfaces para funcionalidades de tiempo de ejecución como ciclos, hilos, carga dinámica o un sistema de objetos.
● **GTK.** Biblioteca que contiene los objetos y funciones para crear la interfaz de usuario. Maneja todos los controles (widgets) visuales, ventanas, botones etc.
● **GDK.** Biblioteca que actúa como intermediario entre gráficos de bajo nivel y gráficos de alto nivel.
● **ATK.** Biblioteca para crear interfaces de gran accesibilidad importante para personas discapacitadas o minusválidas. Pueden usarse útiles como lupas de aumento, lectores de pantalla, o entradas de datos alternativas al clásico teclado o mouse.
● **Pango.** Biblioteca para el diseño y renderizado de texto, se enfoca especialmente en la internacionalización del código. Es el núcleo para manejar las fuentes y el texto de GTK+2.
● **Cairo.** Biblioteca de renderizado avanzado de controles de aplicación.

2.2.4 SSH.

La utilización de SSH para las conexiones remotas tuvo como motivo principal la falta de seguridad de los sistemas de acceso remoto tipo Telnet. Esta falta de seguridad radica en el envío de información en texto plano, es decir sin ningún tipo de codificación que evite el acceso a la información por alguien indebido, además de esto los sistemas de log remoto anteriores no hacían verificación de ningún tipo sobre la maquina que estaba intentando conectarse con el servidor, abriendo la puerta a una serie de problemas informáticos relacionados con intrusos o personas indeseables tratando de acceder al sistema. El esquema de seguridad usado por este sistema es el de llave publica/privada, de tipo RSA para de esta manera hacer el doble proceso de verificación y de codificación de la información.

2.3 Metodología utilizada para el desarrollo de la herramienta.

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Las técnicas indican cómo debe ser realizada una actividad técnica determinada, identificada en la metodología. Se debe tener en consideración que una técnica determinada puede ser utilizada en una o más actividades de la metodología de desarrollo de software. Todo desarrollo de software es riesgoso y difícil de controlar, pero si no lleva una metodología de por medio, lo que se obtiene es clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos .Una metodología puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo. Indica cómo hay que obtener los distintos productos parciales y finales.
A continuación se describen las principales características de una de las más famosas y conocidas metodologías de desarrollo de software: Proceso Unificado de Racional (Rational Unified Process, RUP), la cual se va a utilizar para el desarrollo de la herramienta debido a que se encuentra dentro de las metodologías tradicionales pero puede ser vista como una metodología ágil además, pues es tan configurable como se desee.

2.3.1 Rational Unified Process (RUP).

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, divide en 4 fases el desarrollo del software:

Inicio: El Objetivo en esta etapa es determinar la visión del proyecto.
Elaboración: En esta etapa el objetivo es determinar la arquitectura óptima.
Construcción: En esta etapa el objetivo es obtener la capacidad operacional inicial.
Transisión: El objetivo es llegar a obtener la versión lista para su instalación en las condiciones reales.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, el cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. Vale mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevada bajo dos disciplinas:

Disciplina de Desarrollo.

- Ingeniería de Negocios: Entendiendo las necesidades del negocio.
- Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.
- Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.
- Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo lo solicitado está presente.
- Disciplina de Soporte
- Configuración y administración del cambio: Guardando todas las versiones del proyecto.
- Administrando el proyecto: Administrando horarios y recursos.
- Ambiente: Administrando el ambiente de desarrollo.

Distribución: Hacer todo lo necesario para la salida del proyecto.

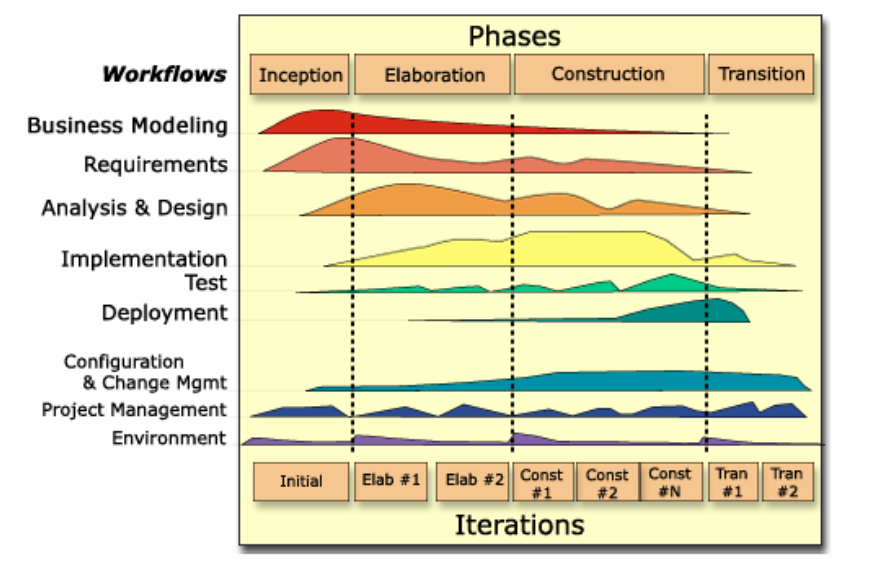


Figura 1: Fases e Iteraciones de la Metodología RUP.

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierta luego en un entregable al cliente. Esto trae como beneficio la retroalimentación que se tendrá en cada entregable o en cada iteración.

Los elementos del RUP son:

- Actividades: son los procesos que se llegan a determinar en cada iteración.
- Trabajadores: son las personas o entes involucrados en cada proceso.
- Artefactos: un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace necesario el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

2.4 Arquitectura y herramientas utilizadas para el desarrollo de la herramienta.

2.4.1 Arquitectura.

Para el diseño de la herramienta se hizo uso del patrón arquitectónico tres niveles o capas, el cual define que:

La aplicación se divide en tres capas lógicas distintas, cada una de ellas con un grupo de interfaces perfectamente definido. La primera capa se denomina capa de presentación y normalmente consiste en una interfaz gráfica de usuario de algún tipo. La capa intermedia, o capa de negocio, consiste en la aplicación o lógica del negocio, y la tercera capa, la capa de datos, contiene los datos necesarios para la aplicación.

La capa intermedia (lógica de aplicación) es básicamente el código al que recurre la capa de presentación para recuperar los datos deseados. La capa de presentación recibe entonces los datos y los formatea para su presentación. Esta separación entre la lógica de aplicación de la interfaz de usuario añade una enorme flexibilidad al diseño de la aplicación. Pueden construirse y desplegarse múltiples interfaces de usuario sin cambiar en absoluto la lógica de aplicación siempre que está presente una interfaz claramente definida a la capa de presentación.

La tercera capa contiene los datos necesarios para la aplicación. Estos datos consisten en cualquier fuente de información, incluido una base de datos de empresa como Oracle o Sybase, un conjunto de documentos XML o incluso un servicio de directorio como el servidor LDAP. Además del tradicional mecanismo de almacenamiento relacional de bases de datos, existen muchas fuentes diferentes de datos de empresa a las que pueden acceder las aplicaciones.

2.4.2 Herramientas Utilizadas.

2.4.2.1 IDE de Python.

Existen diversos IDE para el lenguaje seleccionado, entre los más destacados figuran: Python Scripter, Wing IDE 3.0, Komodo, y finalmente Eclipse que es un IDE escrito en Java. En un principio fue pensado para el desarrollo de programas Java, pero mediante sus plugins, se puede extender su ámbito a otros lenguajes. Precisamente, PyDev es un plugin que permite programar en Python usando este IDE. Eclipse es una herramienta Open-Source, y PyDev se distribuye bajo la Eclipse Public License. Es un IDE completo que integra editor de código, depurador e intérprete. Al estar escrito en Java es multiplataforma, por lo que no hay problemas con el sistema operativo en el que se instale, incorpora un depurador gráfico de Python, resaltado de sintaxis, autocompletado, definiendo las rutas del intérprete instalado (soporta varias versiones), integra también el intérprete. Aunque de momento no es necesario, PyDev también permite desarrollar módulos en Jython, que es la implementación en Java de Python y cuyo principal fuerte es el uso de la librería estándar de Java en módulos Python. Eclipse estructura los proyectos en directorios, paquetes, módulos, recursos, lo que ayuda a organizarlos y establecer una jerarquía.

2.4.2.2 Glade.

Entorno de diseño de interfaz de usuario para GNOME y GTK+, el escogido en este caso es Glade-3 su homologo Gaspacho es un proyecto con mejor interfaz, pero presenta errores de compatibilidad con Python además de ser más joven, Glade permite generar la interfaz de usuario mediante formularios, los cuales son almacenados en formato XML. Se puede especificar las propiedades de cada widget, sus señales y manejadores, los cuales posteriormente son implementados usando Python, además permite separar la interfaz de usuario de la lógica de aplicación lo cual es de vital importancia en el cumplimiento de la arquitectura en capas propuesta.

2.4.2.3 GEdit.

GEdit es un editor de textos nativo en los sistemas gnome de Linux, el cual permitió la implementación de cada uno de los scripts en Bash que instalarán y configurarán cada uno de los servicios necesarios para la gestión de los clústeres de Apache Tomcat debido a la posibilidad que brinda de identificar las sintaxis de varios lenguajes dentro de ellos el Bash

Script, permitiendo una mejor compresión del código script que se iba desarrollando. Existen otros editores que pueden ser utilizados como el Geanny y el Notepad++.

2.5 Negocio.

2.5.1 Flujo Actual del Proceso.

El proceso se inicia al arquitecto de software orientar la instalación del clúster del servidor de aplicaciones al administrador de sistema, el cual comienza a definir los servidores y servicios necesarios a instalar y configurar para que la aplicación funcione de la manera más eficiente.
Una vez definida la estructura de los servidores, el administrador de sistemas pasa a la instalación de cada uno de los servicios, para luego pasar al proceso de configuración de cada uno de ellos. El paso final sería comprobar que el clúster esté funcionando de manera correcta, esto se comprueba instalando la aplicación Java. Cuando la aplicación se encuentra desplegada en cada uno de los nodos Apache Tomcat del clúster, se comienza a hacer pruebas de estrés al balanceador de carga, se detiene el servicio del servidor de aplicaciones en un clúster determinado para comprobar que el sistema siga funcionando y la parte más importante es verificar el funcionamiento de la réplica de sesiones, la cual consiste en que los usuarios que hayan estado conectados al clúster que falló, sigan trabajando normalmente sin perder ninguna información que hayan estado modificando o introduciendo dentro del sistema. Si no se detectan problemas, ya los servidores y en este caso el clúster, están listos para su explotación.

Actualmente la forma en que se realiza el proceso de instalación es algo engorroso debido a que la instalación de todos los servicios se hace manualmente, y lo que más ha afectado en la configuración de los mismos, pues es un proceso muy detallado en el cual se pueden cometer errores que son vitales para el funcionamiento correcto y eficiente del clúster.

2.5.2 Modelo de Negocio.

El modelo del negocio describe el negocio en términos de casos de usos del negocio, que corresponde a lo que generalmente se le llama procesos. La descripción del negocio propuesto en detalle tendrá entre sus actividades principales la identificación de los procesos de negocio, delimitación del modelo de casos de uso del negocio, la especificación de los casos de uso del negocio, la identificación de trabajadores y entidades del negocio que ejecutan las

realizaciones de los casos de uso del negocio y detallar la definición de las entidades del negocio y las responsabilidades de los trabajadores del negocio.

2.5.2.1 Definición de actores del negocio.

Actor	Descripción
Arquitecto de Software	El Arquitecto de Software es el que inicia las acciones que generan los procesos de negocio y es el beneficiado con el resultado de estos procesos.

Tabla 1. Actores del Negocio.

2.5.2.2 Definición de trabajadores del negocio.

Trabajador	Descripción
Administrador de Sistemas.	Determina la arquitectura de servidores necesaria para la instalación de las aplicaciones Java, instala y configura cada uno de los servicios necesario en los servidores, instala los servicios de seguridad, necesarios tanto para la aplicación como para los servidores.

Tabla 2. Trabajadores del Negocio.

2.5.2.3 Diagrama de Casos de Uso de Negocio.

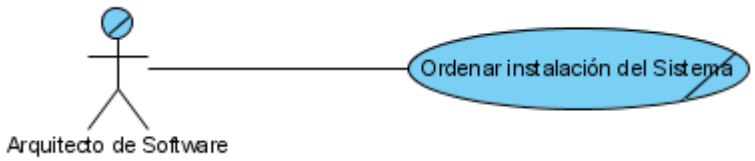


Figura 1. Diagrama de Casos de Uso de Negocio

2.5.2.4 Descripción textual del Caso de Uso de Negocio.

Caso de Uso de Negocio.	Ordenar instalación del Sistema.
Actores.	Arquitecto de Software.
Propósito.	Este caso de uso tiene el objetivo de ordenar la instalación de la aplicación Java en los servidores.
Resumen.	El caso de uso se inicia cuando el Arquitecto de Software le solicita al Administrador de Sistemas que instale la aplicación

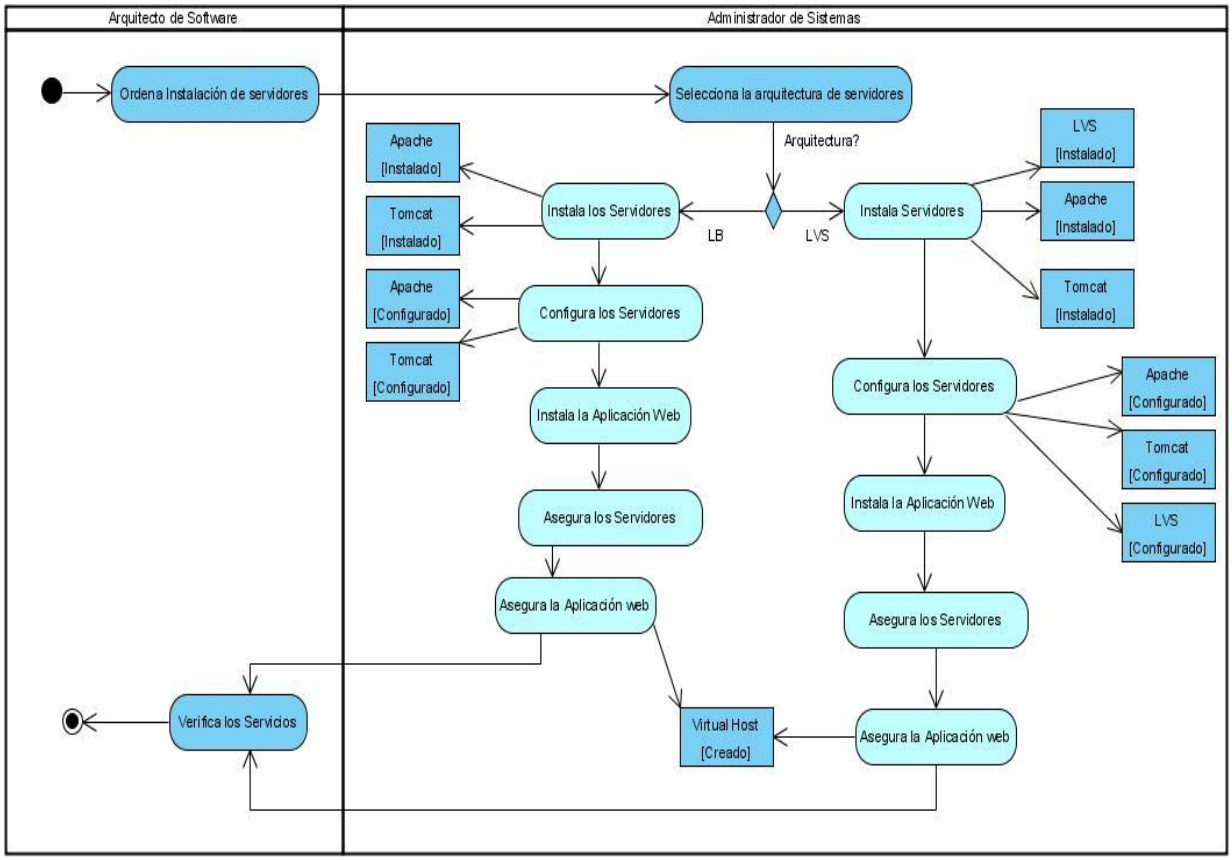
	Java. El Administrador de Sistemas determina qué arquitectura es más eficiente para la aplicación desarrollada, luego instala cada uno de los servicios necesarios en los servidores, luego configura óptimamente cada uno de estos servicios, instala los módulos de seguridad necesarios en cada uno de los servidores y finalmente instala la aplicación Java. Luego el Arquitecto de Software verifica el cumplimiento de la tarea y el funcionamiento de la aplicación.
Curso Normal de Eventos.	
Acciones del Actor	**Respuestas del proceso de negocio.**
1- El Arquitecto de Software le solicita al Administrador de Sistemas instale la aplicación Java.	1.1- El Administrador de Sistemas determina qué arquitectura es más eficiente para la aplicación desarrollada. 1.2- El Administrador de Sistemas instala cada uno de los servicios necesarios en los servidores. 1.3- El Administrador de Sistemas configura óptimamente cada uno de estos servicios. 1.4- El Administrador de Sistemas instala los módulos de seguridad necesarios en cada uno de los servidores. 1.5- El Administrador de Sistemas instala la aplicación Java.
2- El Arquitecto de Software verifica el cumplimiento de la tarea y el funcionamiento de la aplicación.	
Curso Alternativo de los Eventos	
1- El Arquitecto de Software le solicita al Administrador de Sistema la corrección de los problemas, debido a que el funcionamiento de la aplicación Java no es óptimo o no funciona.	1.1- El Administrador de Sistemas verifica cada uno de los servicios instalados y cada una de las configuraciones para detectar los problemas.
Prioridad	Crítico

Tabla 3. Descripción Caso de Uso de Negocio Ordenar instalación del Sistema.

2.5.2.5 Diagrama de Actividades.

Los casos de uso del negocio consisten en secuencias de actividades, que en conjunto producen algún resultado importante para el actor del negocio. El proceso consiste en un flujo básico de una o más alternativas de flujos. La estructura del flujo se describe gráficamente con la ayuda de un diagrama de actividad.

Un Diagrama de actividades ha sido diseñado para mostrar una visión simplificada de lo que ocurre durante una operación o proceso. Es un caso especial de un diagrama de estados en el cual casi todos los estados, son estados de acción (identifican que acción se ejecuta al estar en él) y casi todas las transiciones son enviadas al terminar la acción ejecutada en el estado anterior.



El diagrama de actividades es un grafo de acciones que contiene los estados en que puede hallarse una actividad, puede contener bifurcaciones y describe un proceso que explora el orden de las tareas o actividades que logran los objetivos del negocio.

Figura 2. Diagrama Actividades Caso de Uso de Negocio Ordenar Instalación de Servidores.

2.5.2.6 Diagrama de Objetos.

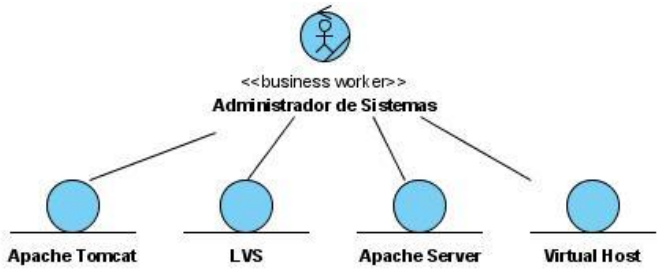


Figura 3. Diagrama Objetos Caso de Uso de Negocio Ordenar Instalación de Servidores

2.6 Características de la Herramienta.

Se debe contar con una herramienta de escritorio portable basada en software libre y para sistemas GNU Linux inicialmente a través de la cual se podrán crear perfiles los cuales contendrán la información necesaria para realizar el proceso de instalación de un clúster de servidores de aplicaciones Apache Tomcat.

Esta herramienta una vez creado o cargado un determinado perfil será capaz de instalar paso a paso de manera automatizada cada uno de los servicios necesarios, una vez instalados, pasará al proceso de configuración de cada uno de estos servicios para luego indicarle cual es la aplicación que se quiere desplegar en cada uno de los clústeres de Apache Tomcat. Estas serían las funciones básicas que brindaría la herramienta, pero además, permitirá en él o los balanceadores de carga activar un virtual host el cual tendrá configurado el uso de un certificado digital para la seguridad de la aplicación desarrollada en tecnología Java.

La herramienta le permitirá al usuario elegir dos tipos de arquitectura para instalar el clúster: un balanceador y **n** nodos de Apache Tomcat o **n** balanceadores controlados por un LVS y **n** nodos de Apache Tomcat.

2.7 Especificación de Requisitos de Software.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Mientras que los requerimientos no funcionales son las propiedades o cualidades que el producto debe tener, debe pensarse en estos atributos como las características que hacen al producto atractivo, usable, rápido o confiable.

2.7.1 Definición de los Requisitos Funcionales.

1. Gestionar Perfil.
    1.1. Crear Perfiles.
        1.1.1. Seleccionar Arquitectura.
        1.1.2. Insertar Datos de los Servidores.
    1.2. Modificar Perfil.

1.3. Eliminar Perfil.

1.4. Cargar Perfil

2. Instalar Servicios.
   2.1. Copiar Scripts de Instalación a los Servidores.
   2.2. Ejecutar Scripts de Instalación en los Servidores.
3. Configurar Servicios.
   3.1. Copiar Scripts de Configuración en los Servidores.
   3.2. Ejecutar Scripts de Configuración en los Servidores.
4. Gestionar seguridad de la Aplicación Web.
   4.1. Insertar datos del Virtual Host.
   4.2. Copiar Script de Generación de Certificados Digitales en el o los Servidores.
   4.3. Copiar Script de Generación del Virtual Host en el o los Servidores.
   4.4. Ejecutar Script de Generación de Certificados Digitales en el o los Servidores.
   4.5. Ejecutar Script de Generación del Virtual Host en el o los Servidores.
5. Gestionar instalación de la Aplicación Java.
   5.1. Copiar fichero .war en los Servidores.
   5.2. Copiar Script para desplegar el fichero .war en los Servidores.
   5.3. Ejecutar Scritp de despliegue en los Servidores.

2.7.2 Definición de los Requisitos no Funcionales.

Apariencia o interfaz externa.

1. Las ventanas de la herramienta contendrán claro y bien estructurados los datos, y al mismo tiempo permitirán la interpretación correcta e inequívoca de la información.
2. Mostrar mensajes de errores en la introducción de datos de una forma sencilla y explicativa, la entrada de datos incorrecta será detectada claramente por la herramienta.
3. Mostrar todos los textos y mensajes en pantalla en español.
4. Diseñar su funcionamiento de modo que sea intuitivo, y requiera de información mínima.

Portabilidad.

1. La herramienta está construida para operar en sistemas operativos GNU Linux, realizada con un lenguaje libre y sin necesidad de instalación.

Seguridad.

1. Los perfiles serán guardados en ficheros binarios, las conexiones a los servidores serán vía ssh y las contraseñas no serán almacenadas.

Confidencialidad.

1. La información manejada por la herramienta estará protegida.

Integridad.

1. La información manejada por la herramienta será objeto de cuidadosa protección contra la corrupción y estados inconsistentes, de la misma forma será considerada igual a la fuente o autoridad de los datos.

Confiabilidad.

1. Se garantizará la consistencia de los datos, se realizarán comprobaciones y validaciones cada vez que sea necesaria la validación de los datos.

Ayuda y documentación.

1. Entregar documentos técnicos y las guías de usuario, que incluyen presentaciones realizadas en cada tema.
2. Entregar carpeta del proyecto, con la documentación técnica generada en el desarrollo para la especificación del sistema.
3. Se deberá entregar el código fuente al propietario del negocio.

Hardware.

1. Mínimo PC Pentium 3, 256 Mb de Ram, 2GB de espacio libre en disco duro.
2. La comunicación de las terminales clientes con el servidor será a través de conexiones a una velocidad constante de 10/100 Mbps.

Software.

1. SO Debian Etch o superior.
2. SO Ubuntu 7.10 o superior.
3. Python 2.5 o superior.

2.8 Modelo del Sistema.

Es una representación gráfica del entorno del sistema (actores) y su funcionalidad principal (casos de uso). Un diagrama de casos de uso muestra, por tanto, los distintos requisitos funcionales que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuarios u otras aplicaciones).

2.8.1 Definición de los actores del sistema.

Actor	Descripción
Administrador de Sistemas.	Determina la arquitectura de servidores necesaria para la instalación de las aplicaciones Java, instala y configura cada uno de los servicios necesario en los servidores, instala los servicios de seguridad necesarios tanto para la aplicación como para los servidores.

Tabla 4. Actores del Sistema.

2.8.2 Definición de los casos de usos principales del sistema.

Gestionar Perfil: Permite crear, cargar, modificar y eliminar perfiles.

Instalar servidores: Permite instalar los servicios correspondientes a cada servidor.

Configurar servidores: Permite configurar los servicios instalados en cada servidor.

Instalar aplicación: Permite desplegar la aplicación en los nodos Apache Tomcat.

Asegurar aplicación: Permite instalar un virtual host con un certificado digital.

2.8.3 Diagrama de Casos de Uso del Sistema.

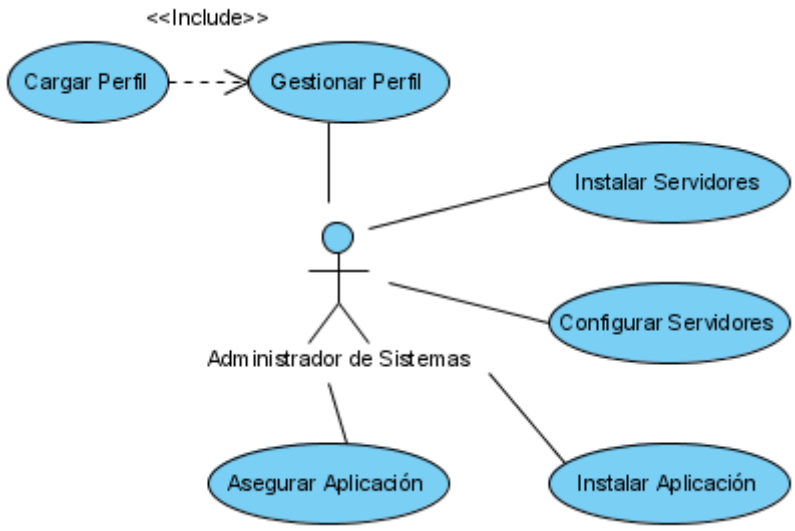


Figura 4. Diagrama de Casos de Uso de Sistema.

2.8.4 Descripción expandida de los Casos de Uso del Sistema.

Para entender la funcionalidad asociada a los casos de uso no es suficiente con la representación gráfica del Diagrama de casos de uso. La descripción extendida brinda los detalles de la secuencia de las acciones, las precondiciones como estado inicial, los posibles estados finales como poscondiciones, además de cuando comienza y termina el caso de uso. Describe explícitamente que debe hacer el sistema, separando las responsabilidades del sistema y la de los actores.

A continuación se muestra la descripción expandida de dos casos de uso arquitectónicamente significativos el resto se encuentra en el **Anexo 1***: Descripción expandida de los Casos de Uso del Sistema.*

Caso de Uso:	Gestionar Perfil.
Actores.	Administrador de Sistemas, (inicia).
Propósito.	Permite la creación, modificación y eliminación de un perfil.
Resumen.	El caso de uso se inicia cuando el Arquitecto de Software elige la opción de crear un perfil. La creación del perfil indica la selección de la arquitectura de clústeres a instalar, y cada uno de los datos de los servidores.
Referencias.	RF 1, RF 1.1, RF 1.1.1, RF 1.1.2, RF 1.2, RF 1.3, RF 1.4
Precondiciones.	
Flujo Normal de Eventos.	
Sección "Crear Perfil"	
Acciones del Actor.	**Respuestas del sistema.**

1- El caso de uso inicia cuando el Administrador de Sistemas selecciona la opción Nuevo del menú.	2- El sistema permite elegir el tipo de arquitectura de los clústeres: - LB. - LVS.
3- El Administrador de Sistemas selecciona la arquitectura que desea instalar.	4- El sistema muestra el formulario de los datos correspondientes a llenar de cada uno de los servidores en dependencia al tipo de arquitectura: -LB: permite la entrada de los datos de un solo balanceador y la de varios nodos del clúster. De cada uno se deben llenar los siguientes campos: ip, usuario root y contraseña. -LVS: permite la entrada de los datos de varios balanceadores, de varios nodos clúster y la de un LVS que controlara y balanceara mediante Round Robin a los balanceadores.
5- El Administrador de Sistemas llena los campos: IP, Usuario y Tipo de servidor.	6- El sistema le permite adicionar otro servidor, guardar para concluir o cancelar sino quiere guardar los cambios.
	7- Finaliza el caso de uso.
Flujo Alterno 1: "Perfil Existente".	
	2- El sistema muestra un mensaje informando sobre la existencia del perfil y si desea cargarlo o no.
3- El Administrador de Sistemas selecciona la opción deseada.	4- El sistema verifica la respuesta: si el Administrador de Sistemas desea cargarlo le muestra los formularios con los datos, sino va al paso 7 del flujo normal.
Flujo Alterno 2: "Validar Datos".	
	6- El sistema muestra mensajes de error de validación de los campos dando sugerencias y luego permite ir al paso 6 del flujo normal.
Poscondiciones de éxito.	Se creó el perfil.
Poscondiciones de fallo.	No se creó el perfil.
Prototipo de Interfaz	

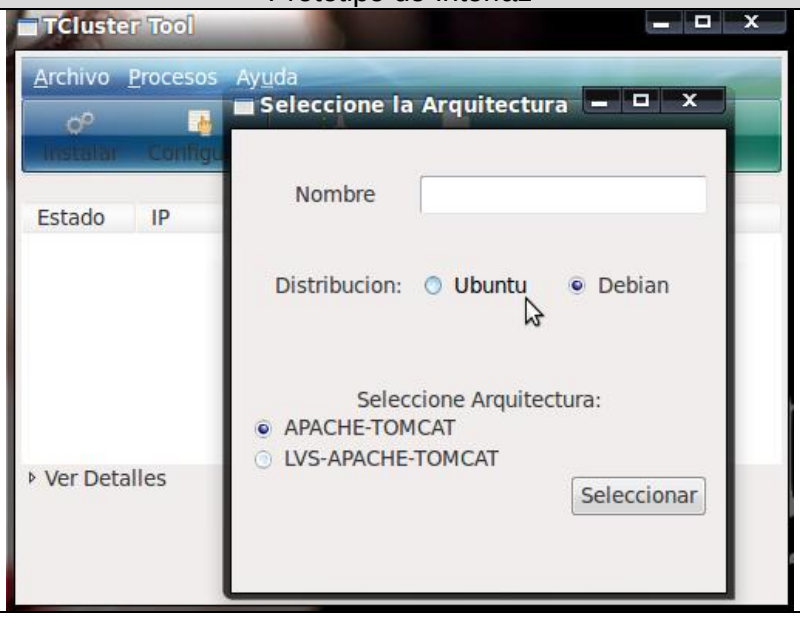


Tabla 5. Descripción Caso de Uso Crear Perfil

Caso de Uso:	Instalar Servidores.
Actores.	Administrador de Sistemas(inicia)
Propósito.	Instala los servicios en los servidores.
Resumen.	El caso de uso se inicia cuando el Arquitecto de Software elige la opción de instalar servidores. Automáticamente según la arquitectura de perfil instala cada uno de los servicios en cada uno de los servidores.
Referencias.	RF 2, RF 2.1, RF 2.2
Precondiciones.	El perfil debe estar cargado.
Flujo Normal de Eventos.	
Acciones del Actor.	**Respuestas del sistema.**
1- El caso de uso inicia cuando el Administrador de Sistemas selecciona la opción Instalar.	2- El sistema verifica la arquitectura del perfil.
3- El Administrador de Sistemas copia los scripts para cada uno de los servidores seleccionando la opción Copiar al seleccionar un servidor.	4- El sistema copia los scripts hacia el servidor seleccionado.
5- El Administrador de Sistemas selecciona el servidor que desee instalar y luego selecciona la opción Instalar	6- El sistema ejecuta cada uno de los scripts correspondientes para el servidor seleccionado.
	7- Termina el caso de uso.
Flujo Alterno 1: "Problemas de instalación".	
	2- El sistema muestra un mensaje informando sobre los posibles errores que puedan haber ocasionado la falla de la instalación.
3- El Administrador de Sistemas verifica la información que brinda el sistema y selecciona la opción para reintentar la instalación luego de verificar los problemas manualmente.	4- El sistema ejecuta el paso 5 del flujo normal.
Poscondiciones de éxito.	Se instalaron los servicios.
Poscondiciones de fallo.	Problemas en la instalación de los servicios.
Prototipo de Interfaz	

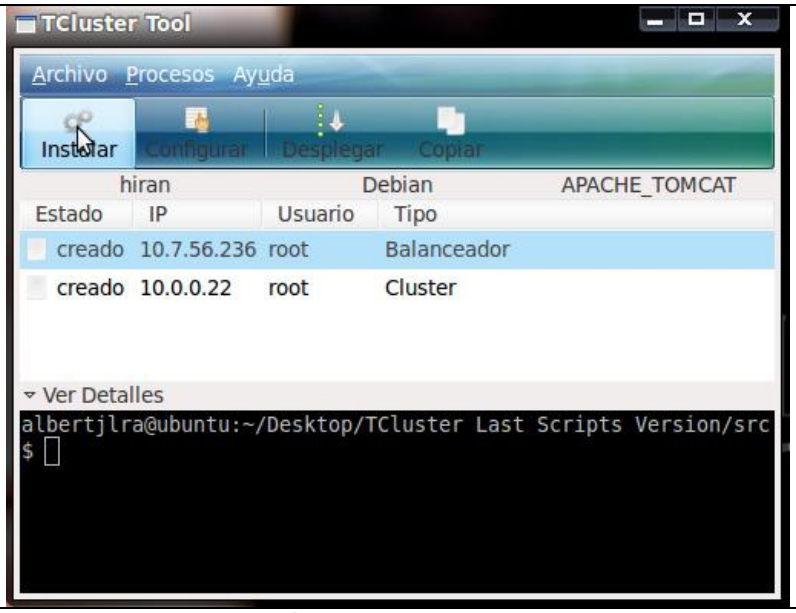


Tabla 6. Descripción Caso de Uso Instalar Servidores

Caso de Uso:	Configurar Servidores.
Actores.	Administrador de Sistemas(inicia)
Propósito.	Configura los servicios en los servidores.
Resumen.	El caso de uso se inicia cuando el Arquitecto de Software elige la opción de configurar servidores. Automáticamente según la arquitectura de perfil configuran cada uno de los servicios en cada uno de los servidores.
Referencias.	RF 3, RF 3.1, RF 3.2
Precondiciones.	El perfil debe estar cargado y los servidores instalados.
Flujo Normal de Eventos.	
Acciones del Actor.	**Respuestas del sistema.**
1- El caso de uso inicia cuando el Administrador de Sistemas selecciona la opción Configurar.	2- El sistema verifica la arquitectura del perfil.
3- El Administrador de Sistemas selecciona el servidor que desee configurar y luego selecciona la opción Configurar.	4- El sistema ejecuta cada uno de los scripts correspondientes para el servidor seleccionado.
	5- Termina el caso de uso.
Flujo Alterno 1: "Problemas de configuración"	
	2- El sistema muestra un mensaje informando sobre los posibles errores que puedan haber ocasionado la falla de la configuración.
3- El Administrador de Sistemas verifica la información que brinda el sistema y selecciona la	4- El sistema ejecuta el paso 4 del flujo normal.

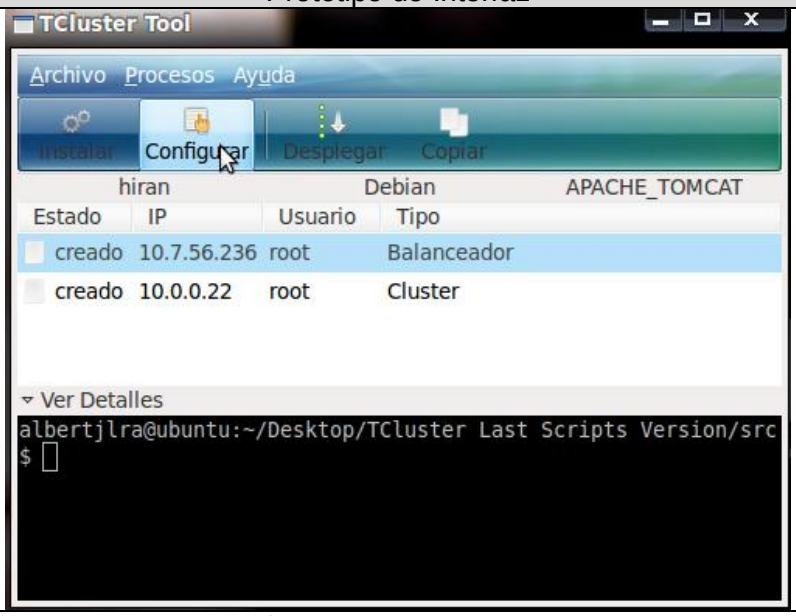
opción Configurar para reintentar la configuración luego de verificar los problemas manualmente.	
Poscondiciones de éxito.	Se configuraron los servicios.
Poscondiciones de fallo.	Problemas en la configuración de los servicios.
Prototipo de Interfaz	
	

Tabla 7. Descripción Caso de Uso Configurar Servidores.

2.9 Análisis y Diseño.

El análisis consiste en obtener una visión del sistema que se preocupa de ver qué hace, de modo que sólo se interesa por los requisitos funcionales. Es descrito en el lenguaje de los desarrolladores y analiza con profundidad los requisitos funcionales. Esboza de forma clara cómo llevar a cabo la funcionalidad dentro del sistema, incluida la funcionalidad significativa para la arquitectura; sirve como una primera aproximación al diseño.

El diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva cómo cumple el sistema sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades. En el diseño se modela el sistema, para que soporte los requisitos, incluyendo los no funcionales y las restricciones que se le suponen.

2.9.1 Modelo de Clases de Análisis.

El modelo de clases del análisis está estructurado por clases y paquetes estereotipados que proporcionan la estructura de la vista interna del sistema. Es utilizado fundamentalmente por los desarrolladores para comprender cómo debería darse forma al sistema, es decir, cómo debería ser diseñado e implementado. Este modelo define realizaciones de casos de uso, y cada una de ellas representa el análisis de un caso de uso del modelo de casos de uso.

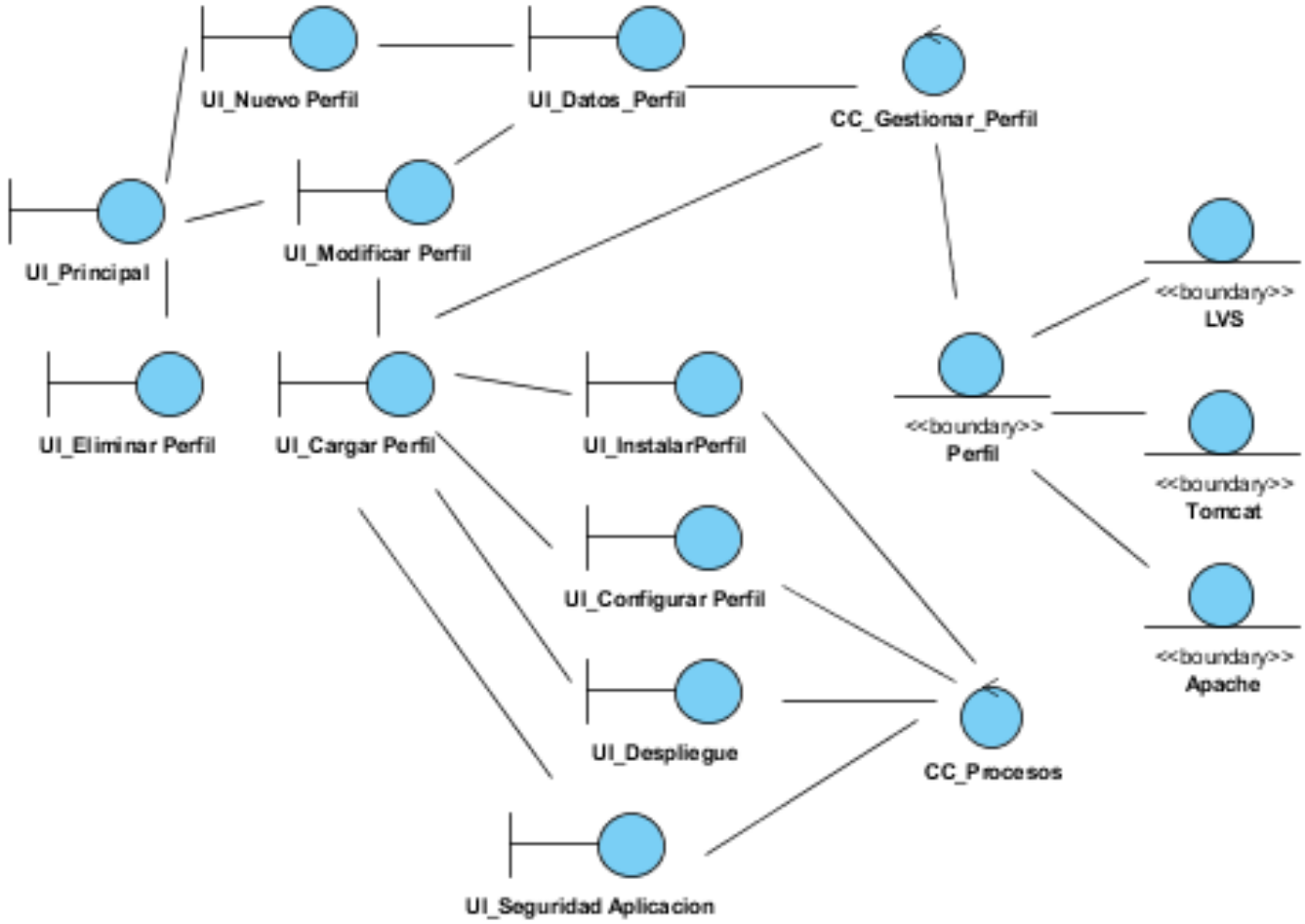


Figura 5. Diagrama de Clases de Análisis

2.9.2 Patrones de Diseño.

Son una solución estándar para un problema común de diseño dentro de un contexto dado y constituyen una manera más práctica de describir ciertos aspectos de la organización de un programa. Estos permitieron dar flexibilidad y extensibilidad al diseño en cuestión.

Patrones Gof:

Los patrones de diseño como son descritos en el libro de los conocidos como GANG OF FOUR están divididos en tres partes y cada una describe los patrones según el propósito. Aplicados al ambiente de programación de Python y las necesidades de la solución propuesta se uso:

Patrones de creación, se relacionan con problemas de instanciación de objetos:

0. **Singleton:** los patrones GoF y los de Python pueden beneficiarse mutuamente. Mientras que a Python le faltan algunas características que el GoF asume, no es imposible construir implementaciones de los patrones que funcionen como sus contrapartes del GoF. En el caso de la implementación del Singleton sustituyó los constructores privados con un mecanismo de excepción y un método de clase con una función regular pero el patrón subyacente es claramente el Singleton. La naturaleza flexible y dinámica del lenguaje provee una buena base para una variedad de distintas y elegantes soluciones.

Patrones de Comportamiento, se enfocan en la dinámica interna e interacción de los objetos en el sistema*:*

1. **Cadena de responsabilidad:** es a menudo utilizado en el contexto de interfaces gráficas. Como el ambiente de ventanas genera eventos los objetos o los manejan o los pasan a su objeto padre. El uso de Python puede reforzar el patrón, la naturaleza flexible y dinámica del lenguaje provee una base para una variedad de distintas y elegantes soluciones. El uso del patrón despacho de comandos [5], incluyen búsquedas y adiciones de atributos en tiempo de ejecución.

Patrones GRASP:

Los patrones generales de software para asignar responsabilidades, GRASP por sus siglas en ingles (General Responsability Asignment Software Paterns), describen en su totalidad los principios fundamentales sobre la asignación de responsabilidades a objetos, todo en forma de patrones.

2. **Experto:** se usó para que cada objeto realizara la funcionalidad de acuerdo a la

información que domina, por lo que aseguró que se le asigne determinada responsabilidad a la clase que mayor información posee para cumplir dicha tarea.

3. **Creador:** se usó para guiar la asignación de responsabilidades con la creación de objetos, con propósito fundamental de encontrar un creador que se deba conectar con el objeto producido en cualquier evento.
4. **Bajo Acoplamiento:** se usó para asignar una responsabilidad de modo que su colocación no incremente el acoplamiento, con el objetivo de diseñar clases más independientes, que reduzcan el impacto de los cambios y sean más reutilizables.
5. **Alta Cohesión:** se usó cuando el objeto tenía delimitadas sus responsabilidades, es decir, para no asignar responsabilidades que no estuvieran limitadas dentro de sus funciones.

En la propuesta, se diseñó asignando responsabilidades de modo que la cohesión sea alta, los métodos tienen bien delimitadas sus responsabilidades, no recargando en ningún caso sus funcionalidades, permitiendo mayor eficiencia y que el tiempo de respuesta y ejecución no exceda lo estimado, generando por ende bajo acoplamiento.

2.9.3 Diagramas de Clases de Diseño.

Los diagramas de clases del diseño muestran un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Gráficamente, un diagrama de clases es una colección de nodos y arcos. Una clase de diseño y de ese modo, los subsistemas que contienen las clases de diseño a menudo participan en la realización de varios casos de uso. Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema principalmente, esto incluye modelar el vocabulario del sistema, modelar las colaboraciones o modelar esquemas. Los diagramas de clases son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa.

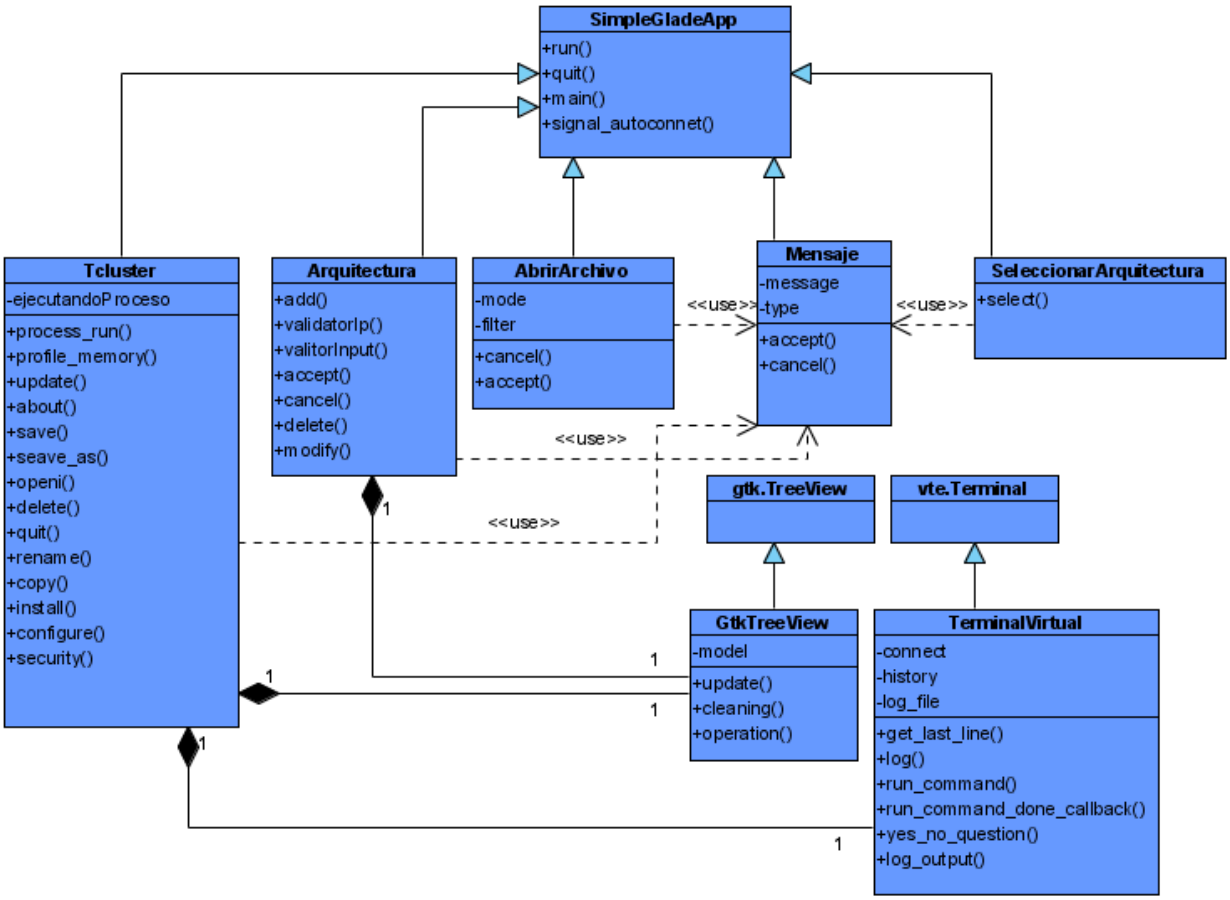


Figura 6. Diagrama Paquete tcgtk.

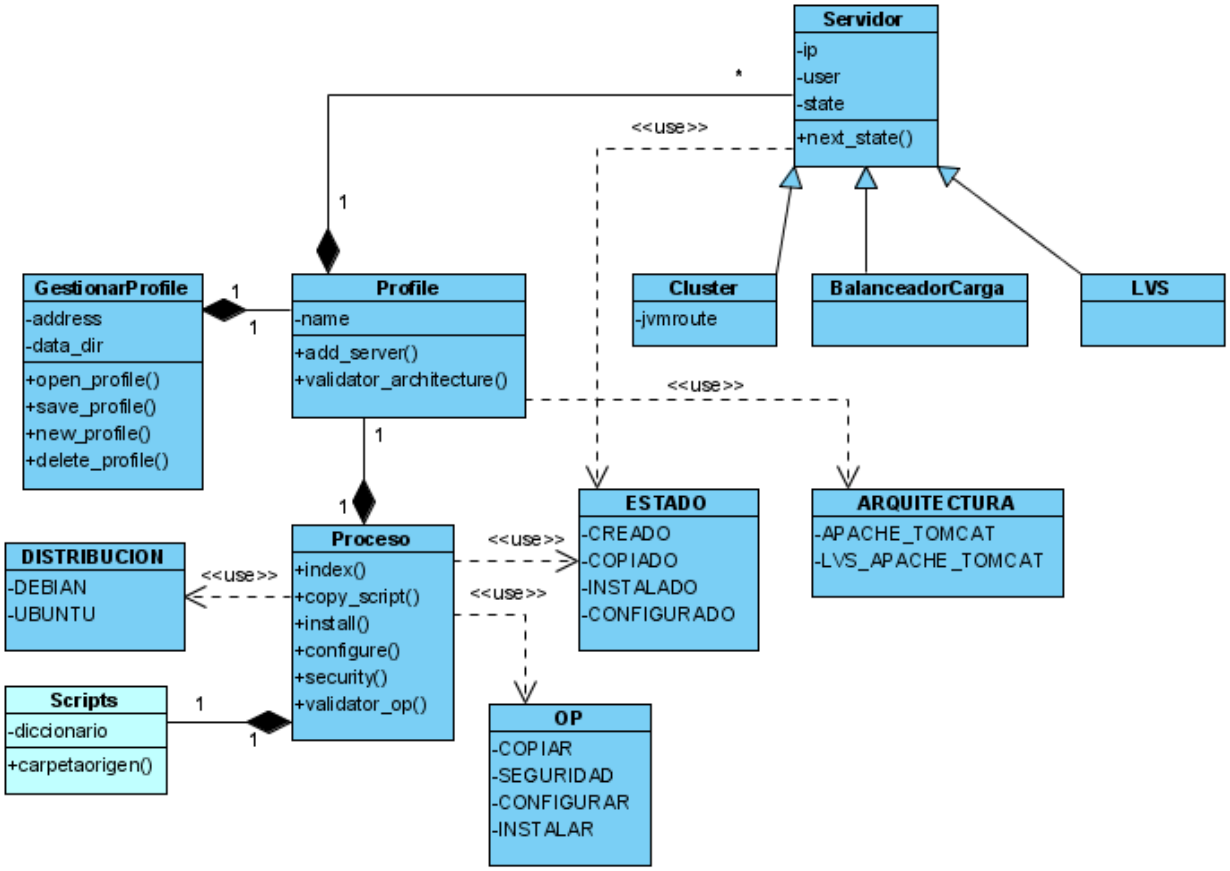


Figura 7. Diagrama Paquete tcluster.

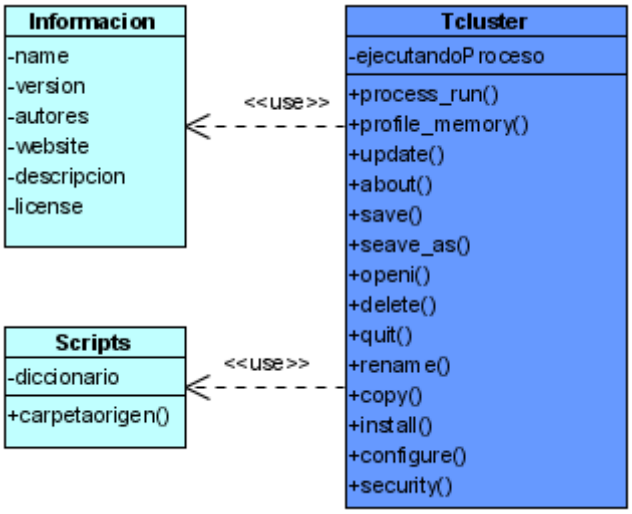


Figura 8. Diagrama Paquete conf.

2.9.4 Métricas

Con el objetivo de determinar el grado de calidad y fiabilidad del diseño que se propone se establecieron algunas métricas de diseño basadas en clases para medir categorías tales como tamaño, herencia, nivel de profundidad, ya que son aspectos orientados al código, a la cohesión, al acoplamiento y la reutilización. A continuación estarán algunas métricas aplicadas al paquete tcluster que controla el negocio de la aplicación debido a que este es el más importante en todo el sistema.

Métrica Tamaño de Clase (TC).

El tamaño general de una clase se puede determinar empleando métricas para saber el número total de operaciones (tanto operaciones heredadas como privadas de la instancia) que están encapsuladas dentro de la clase y encontrando el número de atributos (tanto atributos heredados como atributos privados de la Instancia) que están encapsulados en la clase.

Las medidas o umbrales para los parámetros de calidad han sido una polémica a nivel mundial en el diseño de sistemas. Algunos especialistas plantean umbrales para estas métricas según como se muestra a continuación, los cuales fueron aplicados al diseño.

Numero de Operaciones y/o Atributos	
Métricas TC	Umbral
Pequeño	<=20
Medio	>20 y <=30
Grande	>30

Tabla 8. Umbrales para la Métrica TC.

Los valores grandes de TC demuestran que una clase puede tener demasiada responsabilidad, lo cual reducirá su reutilización y complicará la implementación y la comprobación, por otra parte cuanto menor sea el valor medio para el tamaño, más probable es que las clases existentes dentro del sistema se puedan reutilizar ampliamente.

Resultados

Al aplicar la métrica TC en las clases presentes en el paquete tcluster quedaron recogidos los siguientes datos:

Clases	No. Atributos	No. Operaciones
GestionarProfile	4	4
Perfil	4	4
Servidor	3	1
Cluster	4	1
BalanceadorCarga	3	1
LVS	3	1
Proceso	7	6
Distribucion	2	0
OP	4	0
Estado	4	0
Arquitectura	2	0
Scripts	9	4

Tabla 9. Clases de Negocio.

La totalidad de las clases que conforman el paquete están dentro de la categoría de pequeñas, lo que demuestra que el sistema no es complejo, quedando demostrada la calidad del diseño.
Los resultados obtenidos son positivos según esta métrica. Así se concluye que el paquete cuenta con 12 clases, para un promedio de cantidad de atributos de 4,08 y un promedio de cantidad de operaciones de 1,83. Para más información sobre la descripción de estas clases ver **Anexo 3***: Diagramas de Interacción.***.**

Árbol de Profundidad de Herencia (APH).

Esta métrica se define como la longitud máxima desde el nodo hasta la raíz del árbol. A medida que crece el APH, es más probable que las clases de niveles inferiores hereden muchos métodos. Esto da lugar a posibles dificultades cuando se intenta predecir el comportamiento de una clase.

Una jerarquía de clases profunda con un valor grande de APH, lleva también a una mayor complejidad del diseño. Por el lado positivo, los valores grandes de APH implican que se pueden reutilizar muchos métodos.

Por su parte, algunos autores como Lorenz y Kidd sugieren que un umbral de 6 niveles como indicador es un abuso en la herencia en distintos lenguajes de programación.

Resultados.

A partir de los datos obtenidos al aplicar la métrica APH se obtuvo que el nivel más alto de herencia entre las clases del diseño es de: 1 nivel de profundidad, por lo que queda demostrado que el diseño no es complejo, ya que existe un bajo acoplamiento entre las clases y no es difícil su mantenimiento, pues el nivel de profundidad de la herencia es el mínimo que puede existir en la relación entre clases, este resultado conlleva a que el diseño realizado tiene calidad.

Número de Operaciones Redefinidas para una Sub-Clase (NOR).

Según Pressman[3] gran conocedor del tema, los valores grandes para el NOR, generalmente indica un problema en el diseño, o sea si el NOR es grande el diseñador ha violado la abstracción representada por la superclase.

Esto provoca una débil jerarquía de clases y un software orientado a objetos, que puede ser difícil de probar y modificar. Esta es una de las métricas aplicadas en el sistema para medir la calidad del diseño propuesto.

Resultados.

A partir de los datos obtenidos después de aplicarle al sistema la métrica NOR se obtuvieron que de 12 subclases con las que cuenta el sistema en el paquete de negocio no existen subclases que reemplazan operaciones de las superclases de la que heredan, estas solo reutilizan los métodos de la clase padre, en caso necesario definen operaciones nuevas, por lo que el diseño propuesto posee gran calidad de acuerdo a esta métrica.

2.9.5 Diagramas de Interacción.

Un diagrama de interacción consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Un diagrama de secuencia es un diagrama de interacción que destaca la disposición temporal de los mensajes. Por cada realización de caso de uso se ha realizado un diagrama de interacción (específicamente diagrama de secuencia), donde se expone el flujo principal de información entre los objetos del diseño, con sus métodos y parámetros. En el **Anexo 3***: Diagramas de Interacción*. se muestran los diagramas de secuencia del diseño de los casos de uso fundamentales del sistema.

[3] Ingeniero de Software americano, autor y consultante y presidente de R.S. Pressman & Associates.

2.10 Conclusiones.

En este capítulo se logró detallar el flujo actual de los procesos. Se realizaron los diagramas de casos de uso del negocio y los diagramas de actividades quedando definidos de esta forma el actor y el trabajador que interviene en su ejecución. Se capturaron los requisitos funcionales y no funcionales de la herramienta. Se identificó además el actor del sistema y se dio una descripción del mismo, los diagramas y especificaciones de sus casos de usos, posibilitando la descripción de la solución propuesta a partir de los diagramas de clases del análisis y diseño por casos de uso, junto con los de secuencia y finalmente se mostró el prototipo de interfaz de usuario.

3. Implementación, Pruebas y Resultados

3.1 Introducción.

En el presente capítulo se describe la implementación del software a través de los diagramas de componentes. Además con el diagrama de despliegue se muestra cómo se realiza la distribución de los nodos necesarios para el despliegue de la aplicación. También se realiza pruebas de “Caja blanca” y “Caja negra” para obtener una mayor seguridad del sistema.

3.2 Diagrama de Despliegue.

El diagrama de despliegue representa la arquitectura en tiempo de ejecución de un sistema. Muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se relacionan en esos nodos.

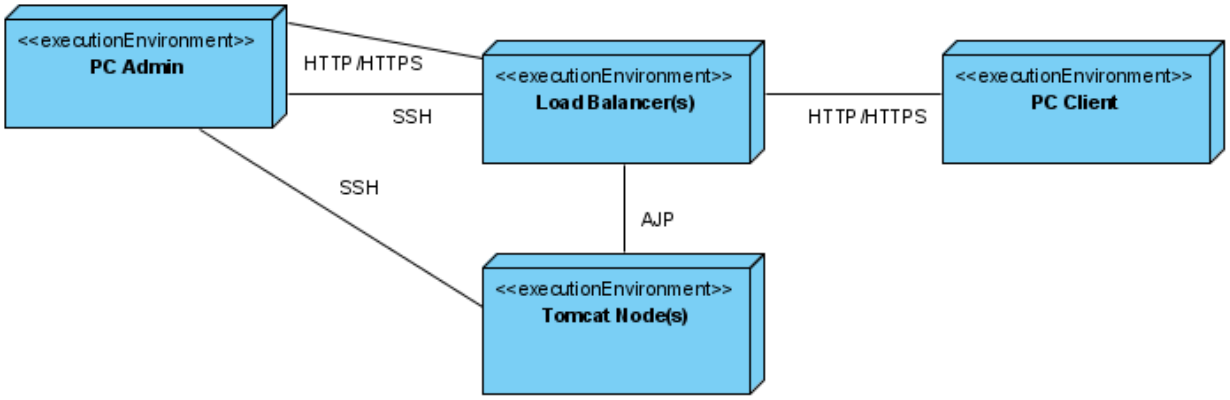


Figura 9. Diagrama de Despliegue

3.3 Estándares de Codificación

Se listan distintas convenciones utilizadas en el código Python comprendido en la librería estándar de la distribución principal de Python para el desarrollo de este proyecto. Lo aquí descrito es una adaptación del ensayo original de Guido Guía de Estilo de Python, con algunos añadidos de la guía de estilo de Barry. Una de las ideas clave de Guido es que el código se suele leer mucho más de lo que se escribe. Las guías de estilo están dirigidas a mejorar la legibilidad del código y hacerlo consistente a lo largo del amplio espectro del código Python.

Como dice el PEP, "La legibilidad cuenta" [4].

3.3.1 Formateo del código.

Indentación

Usa 4 espacios por cada nivel de indentación.

¿Tabuladores o espacios?

Se utiliza la forma más popular de indentar en Python sólo espacios. El código indentado con una mezcla de tabuladores y espacios no es aconsejable y se puede corregir invocando el intérprete de línea de comandos de Python con la opción -t, este muestra avisos sobre código que mezcla tabuladores y espacios. Cuando se utiliza -tt estos avisos se convierten en errores. ¡Estas opciones son altamente recomendables! La mayoría de los editores tienen características que hacen esto bastante sencillo.

Tamaño máximo de línea

Todas las líneas están limitadas a un máximo de 79 caracteres.

Todavía existen muchos dispositivos que están limitados a 80 caracteres por línea; además limitando el ancho de las ventanas a 80 caracteres posibilitas el tener varias ventanas una al lado de otra. El ajuste de línea por defecto en este tipo de dispositivos no da buenos resultados. Por lo tanto, se limitan todas las líneas a un máximo de 79 caracteres. Para cadenas de texto largas (cadenas de documentación o comentarios), se limitan a 72 caracteres [4].

La forma preferida de dividir líneas largas es utilizar la característica de Python de continuar las líneas de forma implícita dentro de paréntesis, corchetes y llaves. En los casos necesarios, se añaden un par de paréntesis extra alrededor de una expresión, en otros casos se usa una barra invertida. Se ha indentado la línea siguiente de forma apropiada.

Ejemplo tomado de la clase CProceso:

```
def configurar(self,numero_de_servidor):
    '''Ejecuta los script para configurar un servidor'''
    self.__servidor=self.__profile.getservidores()[numero_de_servidor]
    if(self.operacion_valida(OP.CONFIGURAR)):
        origen=self.__scripts.carpetaOrigen(self.indice())
        nombres=self.__scripts.configuracion(self.indice())
        for nombre in nombres:
```

```
            comando='ssh ' + self.__servidor.getusuario() + '@' + \
            self.__servidor.getip() + ' bash /home/' + origen[1]+ \
            '/configure/' + nombre
            self.__terminal.run_command(comando)
            error = self.__terminal.log_output()
            if error!=None:
               return error
         servidor.subirEstado()
      else:
         return 'No es posible realizar la configuracion'
```

Líneas en blanco

Separa las funciones no anidadas y las definiciones de clases con dos líneas en blanco.

Las definiciones de métodos dentro de una misma clase se separan con una línea en blanco.

Se usan líneas en blanco extras (de forma reservada) para separar grupos de funciones relacionadas. Las líneas en blanco se omiten entre un grupo de funciones con una sola línea (por ejemplo, con un conjunto de funciones sin implementación).

Se usan líneas en blanco en las funciones, de forma limitada, para indicar secciones lógicas [4].

3.3.2 Codificación de caracteres (PEP 263)

El código de la distribución base de Python debería utilizar las codificaciones ASCII o Latin-1 (también conocida como ISO-8859-1). Para Python 3.0 y superiores, se recomienda UTF-8 en lugar de Latin-1, ver PEP 3120. Todos los módulos descritos has sido usando la codificación UTF-8 lo que es especificado al intérprete en la segunda línea de todos los módulos.

```
#!/usr/bin/env python
# -*- coding: UTF8 -*-
# Python modulo CProcesos.py
```

3.3.3 Imports

6. Los imports han sido colocados en distintas líneas, por ejemplo:

   Sí:

   ```
   import os
   import sys
   ```

No:

```
import sys, os
```

En otros casos también la forma:

```
from subprocess import Popen, PIPE
```

7. Los imports se han colocado siempre en la parte superior del archivo, justo después de cualquier comentario o cadena de documentación del módulo, y antes de las variables globales y las constantes del módulo [4].

   Los imports se han agrupado siguiendo el siguiente orden:

   1. imports de la librería estándar
   2. imports internos del paquete
   3. imports de externos de paquetes relacionados se añade una línea en blanco después de cada grupo de imports.

8. Nunca se usan imports relativos para importar código de un paquete. Se utiliza siempre la ruta absoluta del paquete para todos los imports. Incluso ahora que el PEP 328 está totalmente implementado en Python 2.5, el usar imports relativos se desaconseja seriamente; los imports absolutos son más portables y normalmente más legibles.

Ejemplo tomado del modulo gtk_tcluster.py

```
try:
  import pygtk
  pygtk.require('2.0') # Intenta usar la versión2
except:
  print "Algunas distribuciones vienen con GTK2, pero no con pyGTK (o pyGTKv2)"


try:
   import gtk
   import gtk.glade
   import os
   import time
except:
   print "Necesita instalar GTKv2"
   raw_input("Presione una tecla para terminar")
```

```
    sys.exit(1)

#Internos
from simple_glade_app import SimpleGladeApp
from gtk_new_profile import SeleccionarArquitectura
from gtk_file_chooser_dialog import AbrirArchivo
from gtk_dialog import Mensaje
from gtk_tree_view import GtkTreeView

#Externos
from paquetes.tcluster.tcluster import CTCluster
from paquetes.conf.configuracion import GRAFIC_DIR
from paquetes.conf.informacion import Informacion
from paquetes.tcluster.proceso import CProcesos
from paquetes.tcgtk.gtk_terminal import TerminalVirtual
```

3.3.4 Espacios en blanco en expresiones y sentencias

Se usan espacios en blanco extra de la siguiente forma:

- Inmediatamente después de entrar en un paréntesis o antes de salir de un paréntesis, corchete o llave.

  Sí:

  ```
  spam(ham[1], {eggs: 2})
  ```

  No:

  ```
  spam( ham[ 1 ], { eggs: 2 } )
  ```

- Inmediatamente antes de una coma, punto y coma, o dos puntos:

  Sí:

  ```
  if x == 4: print x, y; x, y = y, x
  ```

  No:

  ```
  if x == 4 : print x , y ; x , y = y , x
  ```

- Inmediatamente antes de abrir un paréntesis para una lista de argumentos de una llamada a una función:

  Sí:

  ```
  spam(1)
  ```

  No:

  ```
  spam (1)
  ```

- Inmediatamente antes de abrir un paréntesis usado como índice o para particionar (slicing):

  Sí:

  ```
  dict['key'] = list[index]
  ```

  No:

  ```
  dict ['key'] = list [index]
  ```

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

  Sí:

  ```
  x = 1
  y = 2
  long_variable = 3
  ```

  No:

  ```
  x             = 1
  y             = 2
  long_variable = 3
  ```

3.3.5 Otras Recomendaciones Usadas

- Se redondea siempre los siguientes operadores binarios con un espacio en cada lado: asignación (=), asignación aumentada (+=, -= etc.), comparación (==, <, >, !=, <>, <=, >=, in, not in, is, is not), booleanos (and, or, not).

- Se usa espacios alrededor de los operadores aritméticos:

  Sí:

  ```
  i = i + 1
  submitted += 1
  x = x * 2 - 1
  hypot2 = x * x + y * y
  c = (a + b) * (a - b)
  ```

  No:

  ```
  i=i+1
  submitted +=1
  x = x*2 - 1
  hypot2 = x*x + y*y
  c = (a+b) * (a-b)
  ```

- No se usan espacios alrededor del signo '=' cuando se use para indicar el nombre de un argumento o el valor de un parámetro por defecto.

  Sí:

  ```
  def complex(real, imag=0.0):
      return magic(r=real, i=imag)
  ```

  No:

  ```
  def complex(real, imag = 0.0):
      return magic(r = real, i = imag)
  ```

- Aunque a veces es adecuado colocar un if/for/while con un cuerpo pequeño en la misma línea, nunca se hace para sentencias multi-clausula.

  Preferiblemente no:

  ```
  if foo == 'blah': do_blah_thing()
  for x in lst: total += x
  while t < 10: t = delay()
  ```

Definitivamente no:

```
if foo == 'blah': do_blah_thing()
else: do_non_blah_thing()

try: something()
finally: cleanup()

do_one(); do_two(); do_three(long, argument,
                             list, like, this)

if foo == 'blah': one(); two(); three()
```

3.3.6 Comentarios

Los comentarios que contradicen el código son peores que no tener ningún comentario. Por lo cual se actualizan conjuntamente con el código las líneas de comentarios.

Los comentarios son frases completas. Si un comentario es una frase o sentencia, la primera palabra esta en mayúsculas, a menos que sea un identificador que comience con una letra en minúsculas.

Si un comentario es corto, se omite el punto al final. Los comentarios de bloque generalmente consisten en uno o más párrafos construidos con frases completas, y cada frase termina con un punto.

Se usan dos espacios después de un punto de final de línea.

Comentarios de bloque

Los comentarios de bloque generalmente se aplican al código que se encuentra a continuación, y se indentan al mismo nivel que dicho código. Cada línea de un comentario de bloque comienza con un # y un único espacio (a menos que haya texto indentado dentro del comentario).

Los párrafos dentro de un comentario de bloque se separan por una línea conteniendo un único #.

Comentarios en línea

Se usan comentarios en línea de forma moderada.

Un comentario en línea es un comentario que se encuentra en la misma línea que una sentencia. Los comentarios en línea se separan por al menos dos espacios de la sentencia que comentan. Comienzan con un # y un espacio.

Los comentarios en línea son innecesarios y de hecho sólo sirven para distraer si indican cosas obvias:

No:

```
x = x + 1                 # Incrementa x
```

Aunque a veces, son de utilidad:

```
x = x + 1                 # Compensa por el borde
```

3.3.7 Cadenas de Documentación

Las convenciones para escribir buenas cadenas de documentación (también llamados "docstrings") se inmortalizaron en el PEP 257.

- Se escriben docstrings para todos los módulos, funciones, clases, y métodos públicos. Los docstrings no son necesarios para métodos no públicos, pero tienen un comentario que describe lo que hace el método. Este comentario aparece antes de la línea "def".
- El PEP 257 describe las convenciones para buenas cadenas de documentación. Es importante notar, que el """ que finaliza un docstring de varias líneas debería situarse en una línea separada, y preferiblemente precederse por una línea en blanco, por ejemplo:

```
"""Return a foobang

Optional plotz says to frobnicate the bizbaz first.

"""
```

- Para cadenas de documentación de una línea, es adecuado mantener el """ de cierre en la misma línea.

3.3.8 Convenciones de nombres

Descriptivo: Estilos de Nombrado

- minusculas_con_guiones_bajos

- PalabrasEnMayusculas (CapWords o CamelCase -- llamado así por el parecido de las mayúsculas con las jorobas de los camellos). Algunas veces también se llaman StudlyCaps.

  Nota: Cuando se usan abreviaturas en CapWords, se mantiene en mayúsculas todas las letras de la abreviatura. Por lo tanto HTTPServerError es mejor que HttpServerError.

- capitalizacionMezclada (¡se diferencia de PalabrasEnMayusculas porque la primera letra está en minúsculas!)

Nombres de Paquetes y Módulos

Los módulos tienen nombres cortos formados en su totalidad por letras minúsculas. Se utilizan guiones bajos en el nombre del módulo si mejora la legibilidad. Los paquetes Python también tienen nombres cortos formados de letras minúsculas, aunque no se usan guiones bajos por estar desaconsejado.

Dado que los nombres de los módulos se mapean a nombres de archivos, y algunos sistemas de ficheros no diferencian entre mayúsculas y minúsculas y truncan los nombres largos, es importante que los nombres de los módulos que se elijan sean suficientemente cortos -- esto no es un problema en Unix, pero puede ser un problema cuando el código se porta a versiones antiguas de Mac o Windows, o a DOS.

Nombres de Clases

Los nombres de clases usan la convención CapWords.

Nombres de Excepciones

Dado que las excepciones deberían ser clases, se aplica la convención relativa al nombrado de clases. De todas formas, usan el sufijo "Error" en los nombres de las excepciones (si la excepción es realmente un error).

Nombres de Variables Globales

Son declaradas en su totalidad por letras mayúsculas.

Nombres de Funciones

Los nombres de funciones están en letras minúsculas, con palabras separadas mediante guiones bajos según sea necesario para mejorar la legibilidad.

Argumentos de funciones y métodos

Se usa siempre 'self' como primer argumento de los métodos de instancia.

Se utiliza siempre 'cls' como primer argumento de métodos de clase.

Nombres de métodos y variables de instancia

Se utilizan las reglas de los nombres de funciones: minúsculas con palabras separadas por guiones bajos cuando es necesario para mejorar la legibilidad.

Se usan guiones bajos al inicio sólo para métodos no públicos y variables de instancia.

Para evitar colisiones de nombres con subclases, se utilizan dos guiones bajos al principio del nombre para invocar las reglas de planchado de nombres de Python.

Diseñar para la herencia

Se decide siempre si los métodos y variables de instancia de una clase (llamados de forma colectiva "atributos") deberían ser públicos o no públicos. Si se ha tenido dudas, se elige que sea no público; es más sencillo convertirla en público más tarde que hacer no público un atributo que antes era público.

Los atributos públicos son aquellos que se espera que sean utilizados por los clientes de la clase, y para los cuales se compromete a evitar cambios que provoquen incompatibilidades hacia atrás. Los atributos no públicos son aquellos que no están destinados a que sean utilizados por terceras personas; no ofreces ninguna garantía de que los atributos no públicos no vayan a cambiar o a eliminarse.

No se usa el término "privado" aquí, dado que ningún atributo es realmente privado en Python (sin un trabajo añadido generalmente innecesario).

Otra categoría de atributos son aquellos que son parte de la "API de las subclases" (a menudo llamado "protected" en otros lenguajes). Algunas clases se diseñan para que se herede de ellas, bien para extender o bien para modificar aspectos del comportamiento de la clase. Cuando se diseña una clase de esta forma, es necesario tomar algunas decisiones explícitas sobre qué atributos van a ser públicos, cuáles son parte de la API de la subclase, y cuáles están pensados para ser utilizados exclusivamente dentro de la clase actual.

Teniendo esto en cuenta, aquí están las líneas a que se han seguido:

- Los atributos públicos no contienen guiones bajos al inicio del nombre.
- Si el nombre del atributo público colisiona con el de una palabra reservada, se añade un único guión bajo al final del nombre del atributo. Esto es preferible a abreviar o cambiar la grafía de la palabra.

- Para campos públicos simples, se expone sólo el nombre del atributo, sin complicaciones de métodos para accederlos y modificarlos (get y set). Teniendo en cuenta que Python proporciona una forma sencilla de modificarlos, en caso de que dentro de un tiempo se necesite. Se utilizan propiedades para ocultar la implementación funcional con una sintaxis simple de acceso a atributos de datos.

  Nota 1: Las propiedades sólo funcionan en las nuevas clases.

  Nota 2: No usar propiedades para realizar operaciones que requieran de grandes cálculos; el utilizar la notación de atributos hace que la persona que accede al atributo piense que el acceso es (relativamente) barato en tiempo computacional.

3.3.9 Recomendaciones para la programación

- Las comparaciones con singletons como None se llevan a cabo con 'is' o 'is not', nunca con operadores de igualdad.

- Se utilizan excepciones basadas en clases.

  Las excepciones en forma de cadenas están prohibidas en el código nuevo, dado que esta característica del lenguaje se eliminará en Python 2.6.

  Los módulos o paquetes deberían definir sus propias clases excepción específica para el dominio del problema, las cuales deberían heredar de la clase Exception Se incluye siempre una cadena de documentación para la clase. Por ejemplo:

```
class MessageError(Exception):
    """Base class for errors in the email package."""
```

  En este caso se aplican las convenciones de nombrado de las clases, aunque deberías añadir el sufijo "Error" a las clases excepción, si la excepción se trata de un error. Las excepciones que no sean errores, no necesitan ningún sufijo especial.

- Cuando se lanza una excepción, se utiliza "raise ValueError('mensaje')" en lugar de la forma antigua "raise ValueError, 'mensaje'".

  Se prefiere la forma con paréntesis porque cuando los argumentos de la excepción son largos o incluyen formateo de cadenas, no necesitas usar caracteres para indicar que continúa en la siguiente línea gracias a los paréntesis. La forma antigua se eliminará en Python 3000.

- Cuando se capturan excepciones, se menciona las excepciones específicas cuando es posible en lugar de utilizar una clausula 'except:' genérica.

  Por ejemplo, utiliza:

```
try:
    import platform_specific_module
except ImportError:
    platform_specific_module = None
```

  Una cláusula 'except:' genérica capturaría las excepciones SystemExit y KeyboardInterrupt, complicando el poder interrumpir el programa usando Control-C, y pudiendo ocultar otros problemas. Si se quiere capturar todas las excepciones relativas a errores en el programa, se utiliza 'except Exception:'.

  Una buena regla a seguir es limitar el uso de las cláusulas 'except' genéricas a dos casos:

  - Si el manejador de la excepción imprimirá o registrará el traceback; de esta forma al menos el usuario sabrá que ha ocurrido un error.
  - Si el código tiene que realizar algún trabajo de limpieza, pero en ese caso se tiene que hacer que la excepción se propague hacia arriba usando 'raise'. La construcción 'try...finally' es una mejor forma de tratar con este caso.

- Adicionalmente, para todas las clausulas try/except, es necesario limitar la clausula 'try' a la mínima expresión de código necesaria. De nuevo, esto se hace para evitar ocultar bugs.

  Sí:

```
try:
    value = collection[key]
except KeyError:
    return key_not_found(key)
else:
    return handle_value(value)
```

  No:

```
try:
```

```
        # ¡Demasiado amplio!
        return handle_value(collection[key])
    except KeyError:
        # También capturará la excepción KeyError lanzada por handle_value()
        return key_not_found(key)
```

- Se utiliza métodos de cadenas en lugar del módulo string.

  Los métodos de las cadenas son siempre mucho más rápidos y comparten la misma API con las cadenas unicode. Se puede aplicar esta regla porque no es necesaria la compatibilidad con versiones de Python anteriores a la 2.0.

- Utiliza ''.startswith() y ''.endswith() en lugar del particionado de cadenas para comprobar prefijos y sufijos.

  startswith() y endswith() son más limpios y menos propensos a errores. Por ejemplo:

  Sí:

  ```
  if foo.startswith('bar'):
  ```

  No:

  ```
  if foo[:3] == 'bar':
  ```

  La excepción a esta norma se da si es necesario que el código funcione con Python 1.5.2

- Las comparaciones del tipo de objeto se utiliza siempre isinstance() en lugar de comparar tipos directamente.

  Sí:

  ```
  if isinstance(obj, int):
  ```

  No:

  ```
  if type(obj) is type(1):
  ```

  Cuando se comprueba si un objeto es una cadena, ¡hay que tener en cuenta que puede que se trate de una cadena unicode! En Python 2.3, str y unicode tienen una clase padre común, basestring, por lo que puedes usar:

```
if isinstance(obj, basestring):
```

En Python 2.2, el módulo types define el tipo StringTypes para este propósito en concreto, por ejemplo:

```
from types import StringTypes
if isinstance(obj, StringTypes):
```

En Python 2.0 y 2.1, se tendría que hacer:

```
from types import StringType, UnicodeType
if isinstance(obj, StringType) or \
   isinstance(obj, UnicodeType) :
```

- Para secuencias, (cadenas, listas, tuplas), se aprovecha el que las secuencias vacías son equivalentes al falso booleano.

  Sí:

  ```
  if not seq:
  if seq:
  ```

  No:

  ```
  if len(seq)
  if not len(seq)
  ```

- No se comparan valores booleanos con True o False usando == :

  Sí:

  ```
  if greeting:
  ```

  No:

  ```
  if greeting == True:
  ```

  Peor:

  ```
  if greeting is True:
  ```

3.4 Modelo de Pruebas.

A la herramienta implementada se le hicieron varias pruebas, para ello se utilizaron los métodos de pruebas de caja blanca y de caja negra. La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. En las pruebas de caja blanca del software se comprueban los caminos lógicos del software proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado.

Debido a la propia filosofía de Python, es difícil encontrar tanto herramientas de prueba como depuradores. Al ser un lenguaje interpretado que no necesita compilarse ni enlazarse, se supone que el propio código (usando el intérprete) es todo lo necesario para probar y depurar un programa.

Dentro de la gama de este tipo de herramientas se encuentran varias de ellas como: PyUnit, Py.Test, TestOOB, NOSE, TRIAL y QUNITTEST, de las cuales las más factible fue PyUnit.

PyUnit es la herramienta o framework oficial para hacer pruebas unitarias en Python. Se incluye en la librería estándar (en el módulo unittest) y ha sido creado por los desarrolladores de JUnit para facilitar la creación y gestión de tests de prueba en módulos Python. Este framework es soportado perfectamente por el IDE que estoy usando en el desarrollo del proyecto, que se trata del Eclipse con la extensión PyDev. Permite separar el código de pruebas del código del propio proyecto para mejorar la comprensión, refactorización y la facilidad para hacer cambios en el programa. Por otra parte, se pueden realizar los tests por línea de comandos.

PyUnit también hace pruebas sobre interfaces gráficas escritas en TKinter (el toolkit estándar para creación de interfaces), así que puede ser útil en el futuro cuando avance en el proyecto. Las pruebas que es capaz de realizar son pruebas de igualdad, de excepciones, de fallo, pruebas de cordura (comprobar que no se producen situaciones imposibles) y pruebas de mayúsculas (relacionadas con las cadenas de caracteres).

Debido a todo esto, y a que se trata de la herramienta estándar de Python, tiene mucha difusión y existe mucha documentación sobre esta librería en la red. Es la que he elegido para realizar las pruebas unitarias en mi proyecto.

3.4.1 Modelo de Prueba de Caja Negra.

Entrada	Resultados	Condiciones
Una vez que el administrador escoge crear un nuevo perfil la herramienta le muestra un formulario para la captura de los datos necesarios para el nuevo perfil: El administrador teclea el nombre del perfil, luego el sistema operativo del clúster: - Ubuntu Linux - Debian GNU/Linux Luego como último paso la arquitectura que utilizará para su clúster: - APACHE-TOMCAT - LVS-APACHE-TOMCAT Luego le muestra nuevamente el formulario inicial permitiéndole la creación de los servidores necesarios en dependencia de la arquitectura seleccionada.	La herramienta crea el nuevo perfil almacenándolo en un fichero binario.	El administrador debe llenar y seleccionar todos los campos disponibles. El administrador no debe crear un perfil con un mismo nombre.

Tabla 8. Prueba de Caja Negra al Caso de Uso Crear Perfil

Entrada.	Resultados.	Condiciones.
Una vez que el administrador escoge la opción de cargar un perfil la herramienta le muestra una ventana de dialogo mostrándole los perfiles creados, selecciona uno y selecciona la opción abrir.	La herramienta muestra en el formulario principal la información del perfil y cada uno de los servidores del mismo.	El administrador debe llenar y seleccionar un fichero que contiene la información del perfil.

Tabla 9. Prueba de Caja Negra al Caso de Uso Cargar Perfil.

Entrada.	Resultados.	Condiciones.
Una vez cargado el perfil, el administrador selecciona uno de los servidores y selecciona la opción Instalar.	La herramienta muestra en el formulario principal la consola de detalles de la instalación, permitiéndole al administrador ver el proceso de instalación en el servidor remoto, dándole la posibilidad de interactuar con la misma en caso de alguna toma de decisiones en cuanto a la instalación.	El administrador seleccionar un servidor.

Tabla 10. Prueba de Caja Negra al Caso de Uso Instalar Servidores

Entrada	Resultados	Condiciones
Una vez cargado el perfil el administrador selecciona uno de los servidores y selecciona la opción Configurar.	La herramienta muestra en el formulario principal la consola de detalles de la configuración permitiéndole al administrador ver el proceso de configuración en el servidor remoto.	El administrador seleccionar un servidor.

Tabla 11. Prueba de Caja Negra al Caso de Uso Configurar Servidores

3.4.2 Modelo de Prueba de Caja Blanca

Las pruebas de unidad o unitarias, son una técnica que permite determinar si un programa cumple con los requisitos solicitados. Es una forma de probar el correcto funcionamiento de un módulo de código.

Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Aunque no descubrirán todos los errores del código, ya que sólo prueban las unidades por sí solas, y por lo tanto, no descubrirán errores de integración, problemas de rendimiento y otros problemas que afectan a todo el sistema en su conjunto; las pruebas de unidad, por definición, cubren la funcionalidad propia del módulo tanto con una perspectiva de caja blanca como de caja negra; pero prestando poca o ninguna atención a la integración con otros módulos. Como se mencionó anteriormente para la realización de las pruebas se utilizo el PyUnit, framework estándar para desarrollar los casos de prueba de unidad, para programas desarrollados en el lenguaje utilizado. Para poder hacer un caso de prueba propio en PyUnit es necesario hacer una subclase de "TestCase", que es un objeto que puede correr un único método de prueba.

Para realizar una prueba de algo en Python se usa el comando "assert", si este comando falla cuando se ejecuta el caso de prueba PyUnit dará el caso por fallido, de lo contrario el caso el resultado es correcto.

```
#!/usr/bin/env python
import unittest
from paquetes.tcluster.tcluster import TCluster
from paquetes.tcluster.profile import Profile
from paquetes.tcluster.enum import ARQUITECTURA, DISTRIBUCION

class GestionarProfile(unittest.TestCase):

    def runTest(self):
        self.__controlador=TCluster()
        self.__profile=Profile(nombre='TClusterTestCase',
                               distribucion=DISTRIBUCION.DEBIAN,
                               arquitectura=ARQUITECTURA.APACHE_TOMCAT)
        self.__controlador.crearProfile(self.__profile)
        self.__controlador.salvarProfile('/home/islamico/test')
        self.__controlador.cargarProfile('/home/islamico/test.tc')
        self.assertEqual(self.__profile.getarquitectura(),
                         self.__controlador.getprofile().getarquitectura())
        self.__controlador.eliminar_profile()


testCase = GestionarProfile()
runner = unittest.TextTestRunner()
runner.run(testCase)
```

Se realizaron pruebas al módulo y el resultado fue satisfactorio como se muestra en las siguientes figuras:

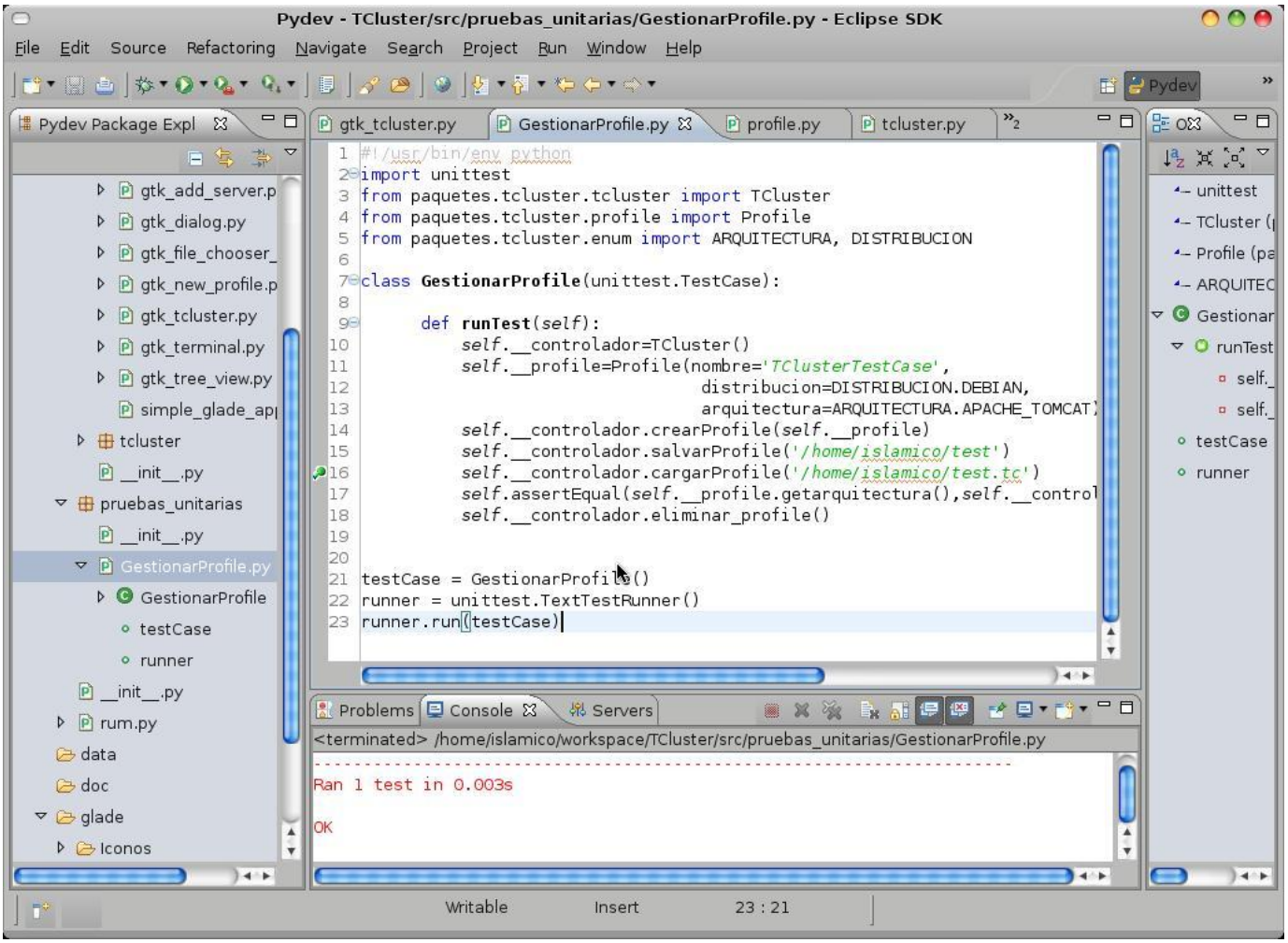


Figura 10. PyUnit (SubClase testCase)

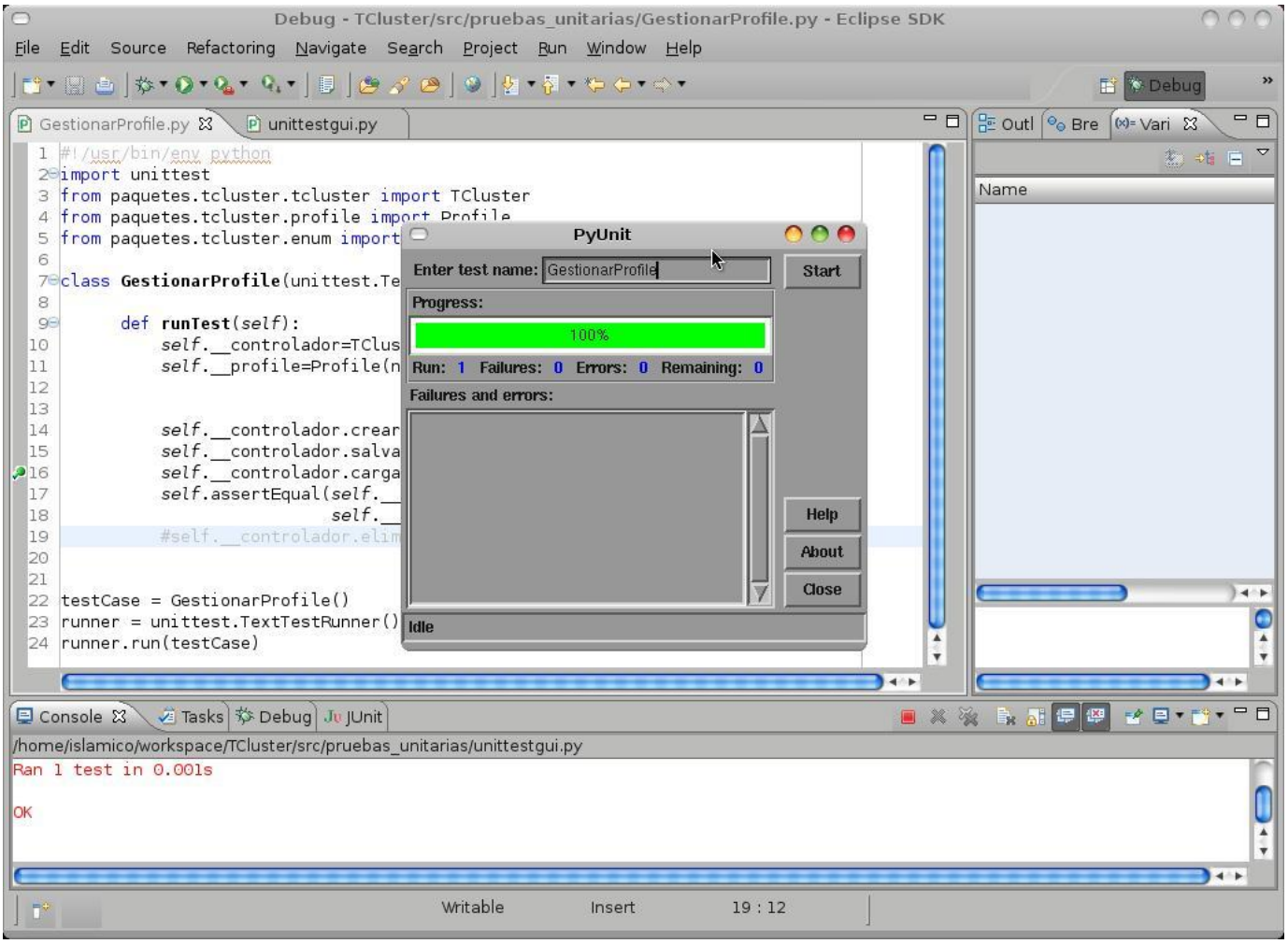


Figura 11. PyUnit (Resultado de la Prueba)

3.5 Conclusiones.

En este capítulo se abordó los resultados de las pruebas de caja blanca y de caja negra realizadas, los estilos de codificación utilizados y el diagrama de despliegue de la herramienta mostrando las conexiones que permitirá hacia los servidores y los protocolos que se utilizarán.

Conclusiones.

Con el desarrollo del presente trabajo y con la implementación de la herramienta para la instalación y configuración de clústeres de Apache Tomcat se concluye afirmando que:

- Se capturaron los requisitos funcionales y no funcionales necesarios, para que mediante una herramienta se pudiera brindar una solución más óptima y eficiente al problema planteado, logrando la capacidad de instalar y configurar clústeres de Apache Tomcat sin la necesidad de recurrir al engorroso proceso de instalación y configuración manual de cada uno de los elementos que componen estos clústeres.
- Se realizó un estudio de las principales herramientas de instalación y configuración de clústeres, y permitió concluir que todas requerían de procesos engorrosos, por lo que se decidió implementar una herramienta de escritorio, capaz de hacer este proceso de una manera más amigable, y específicamente para clústeres de Apache Tomcat.
- La herramienta fue modelada con Visual Paradigm y UML e implementada con el lenguaje Python empleando el IDE Eclipse con el módulo PyDev integrado.
- Se estudiaron los diferentes métodos de implementación de clústeres, específicamente los clústeres de alta disponibilidad. Se eligió el de nivel de software por su bajo costo y poca complejidad.
- Se logró diseñar e implementar basados en la metodología RUP, una herramienta de escritorio capaz de instalar y configurar clústeres de Apache Tomcat, proporcionando además funcionalidades para garantizar la seguridad de los servidores y de la aplicación que será hosteada en el mismo.
- Se logró una herramienta capaz de instalar dos variantes de clústeres en dependencia de los recursos con los que se cuentan, una primera con un solo balanceador de carga y una segunda con varios balanceadores de carga.

Recomendaciones.

- Continuar el desarrollo de esta herramienta en una segunda fase para mejorar e incrementar las funcionalidades.
- Desarrollar esta herramienta para sistemas operativos propietarios como Microsoft Windows, debido a que la actual versión es solo para sistemas Linux.

Referencias Bibliográficas.

1. **Manuel, Irwin.** [En línea] 18 de 01 de 2008. http://www.nativosdigitales.blogspot.com/2008/01/frases-de-stallman.html.
2. Cluster (Sistema Operativo). [En línea] 2009. http://clusters.blogcindario.com/2008/10/00002-historia.html.
3. Hosting Solingest. [En línea] 2007. http://hosting.solingest.com/cluster-de-servidores.html.
4. **Duque, Raúl González.** Guía de estilo del código Python. [En línea] 2007. http://mundogeek.net/traducciones/guia-estilo-python.htm.
5. **Van Rossum, G.,** "Command Dispatch Pattern", Mayo 1997. http://www.python.org.

Bibliografía

1. **J. Kabir, Mohammed**, “Apache Server 2 Bible”, Hungry Minds, Inc, 2002.
2. **J. Barret, Daniel**, “SSH, the Secure Shell: The Definitive Guide”, O’Reilly & Associates, Inc., 2001.
3. **Jacobson, I., Booch, G., Rumbaugh, J.**, “El Proceso Unificado de Software”, Pearson Educación, S.A., 2000.
4. **Pressman, Roger S.**, “Ingeniería de Software, Un enfoque práctico”, Quinta Edición, McGraw Hill.
5. **Gamma, E., Helm. R., Johnson, R., Vlissides, J.**, “Desing Patterns CD: Elements of Reusable Object-Oriented Software”, Addison Wesley Longman, Inc., 1998.
6. **Colectivo de Autores**, “Beginning Python: From Novice to Professional”, Magnus Lie Hetland, 2005.
7. **Albing, C., Vossen, J.P., Newham, C.**, “Bash Cookbook”, O’Reilly Media, 2007.
8. **Burtch, K.O.**, “Linus Shell: Scripting with Bash”, Sams Publishing, 2004.
9. **Ricart, M.A.**, “Apache Server Survival Guide”, New Riders, 1996.
10. **Laurie, B., Laurie, P.**, “Apache: The Definitive Guide, Second Edition”, O’Reilly & Associates, Inc., 1999.
11. **Chopra, V., Bakore, A., Eaves, J., Galbrait, B., Li, S., Wiggers, C.**, “Professional Apache Tomcat 5”, Wiley Publishing, Inc., 2004.
12. **Brittain, J., Darwin, I.F.**, “Tomcat: The Definitive Guide, Second Edition”, O’Reilly Media, 2008.
13. **Ristic, I.**, “Apache Security”, O’Reilly, 2005.
14. **Appu, A.**, “Administering and Securing the Apache Server”, Premier Press, 2002.
15. **Sobell, M.G.**, “A Practical Guide to Linux® Commands, Editors, and Shell Programming”, Prentice Hall PTR, 2005.
16. **Smith, R.W.**, “Advanced Linux Networking”, Addison Wesley, 2002.
17. “Ultra Monkey: High Availability and Load Balancing Solution for Linux”, Horms, 2005, http://www.ultramonkey.org/sl-ha-lb-eg.html.
18. **Gamma, E., Helm, R., Johnson, R., Vlissides, J.**, “Design Patterns -- Elements of Reusable Object-Oriented Software”, Addison-Wesley**,** 1995**.**

Glosario de Términos.

A continuación, en orden alfabético, se muestra el significado de algunos términos usados en este documento cuyo uso no es común y que pueden dificultar la compresión del mismo:

- Apache Tomcat: Tomcat (también llamado Jakarta Tomcat o Apache Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

- API: Una interfaz de programación de aplicaciones o API (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

- Apache 2: El servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo.

- ARPANET: La red de computadoras ARPANET (Advanced Research Projects Agency Network) fue creada por encargo del Departamento de Defensa de los Estados Unidos (“DoD” por sus siglas en inglés) como medio de comunicación para los diferentes organismos del país. El primer nodo se crea en la Universidad de California, Los Angeles y fue la espina dorsal de Internet hasta 1990, tras finalizar la transición al protocolo TCP/IP en 1983.

- ARCnet: Arquitectura de red de área local desarrollado por Datapoint Corporation que utiliza una técnica de acceso de paso de testigo como el Token Ring. La topología física es en forma de anillo, utilizando cable coaxial y hubs pasivos (hasta 4 conexiones) o activos.

- Aix: AIX (Advanced Interactive eXecutive) es un sistema operativo UNIX System V propietario de IBM. Inicialmente significaba “Advanced IBM Unix” pero probablemente el nombre no fue aprobado por el departamento legal y fue cambiado a “Advanced Interactive eXecutive”. AIX corre en los servidores IBM eServers pSeries, utilizando procesadores de la familia IBM POWER de 32 y 64bits.

- Alpha: DEC Alpha es una arquitectura diseñada por DEC e introducida en 1992 bajo el nombre AXP, como reemplazo a la serie VAX. Cuenta con un set de instrucciones RISC de 64 bits especialmente orientada a cálculo de punto flotante.

- Ant: Ant es una herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de compilación y construcción (build). Es similar a Make pero sin las engorrosas dependencias del sistema operativo.

- Beowulf: cluster construido por Donald Becker y Thomas Sterling en 1994. Fue construido con 16 computadores personales con procesadores Intel DX4 de 200 MHz, que estaban conectados a través de un switch Ethernet. El rendimiento teórico era de 3.2 Gflops.

- Bash: Bash es un - 80 -aíz- 80 - de Unix (intérprete de órdenes de Unix) escrito para el proyecto GNU. Su nombre es un acrónimo de bourne-again - 80 -aíz- 80 - (otro - 80 -aíz- 80 - bourne) haciendo un juego de palabras (born-again significa renacimiento) sobre el Bourne - 80 -aíz- 80 - (sh), que fue uno de los primeros intérpretes importantes de Unix.

- Clúster: El término clúster se aplica a los conjuntos o conglomerados de computadoras construidos mediante la utilización de componentes de hardware y software comunes y que se comportan como si fuesen una única computadora.

Hoy en día juegan un papel importante en la solución de problemas de las ciencias, las ingenierías y del comercio moderno.

- Corporación RAND: La Corporación RAND (Research And Development) es un think tank norteamericano formado, en un primer momento, para ofrecer investigación y análisis a las fuerzas armadas norteamericanas.

- Debian: Debian o Proyecto Debian (en inglés Debian Project) es una comunidad conformada por desarrolladores y usuarios, que mantiene un sistema operativo GNU basado en software libre precompilado y empaquetado, en un formato sencillo en múltiples arquitecturas de computador y en varios núcleos.

- DNS: El Domain Name System (DNS) es una base de datos distribuida y jerárquica que almacena información asociada a nombres de dominio en redes como Internet. Aunque como base de datos el DNS es capaz de asociar diferentes tipos de información a cada nombre, los usos más comunes son la asignación de nombres de dominio a direcciones IP y la localización de los servidores de correo electrónico de cada dominio.

- EAServer: EAServer es un servidor de aplicaciones desarrollado por la empresa Sybase, el cual incluye un conjunto integrado de herramientas que se usan para crear y ejecutar aplicaciones Web con soporte a altos niveles de tráfico, contenido dinámico y procesamiento intensivo de transacciones en línea.

- Estilo Grid: La computación en grid o en malla es un nuevo paradigma de computación distribuida, en el cual todos los recursos de un número indeterminado de computadoras son englobados para ser tratados como un único superordenador de manera transparente.

- FLOPS: En informática, FLOPS es el acrónimo de Floating point Operations Per Second (operaciones en coma flotante por segundo). Se usa como una medida del rendimiento de una computadora, especialmente en cálculos científicos que requieren un gran uso de operaciones de punto flotante. FLOPS, al ser un

acrónimo, no debe nombrarse en singular como FLOP, ya que la S final alude a second (o segundo) y no al plural.

- FreeBSD: FreeBSD es un sistema operativo libre para computadoras basado en las CPU de arquitectura Intel, incluyendo procesadores 386, 486 (versiones SX y DX), y Pentium. También funciona en procesadores compatibles con Intel como AMD y Cyrix. Actualmente también es posible utilizarlo hasta en once arquitecturas distintas como Alpha, AMD64, IA-64, MIPS, PowerPC y UltraSPARC.

- GUI: En el contexto del proceso de interacción persona – ordenador, la interfaz gráfica de usuario es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático.

- GTK: GTK+ o The GIMP Toolkit es un conjunto de bibliotecas multiplataforma para desarrollar interfaces gráficas de usuario (GUI), principalmente para los entornos gráficos GNOME, XFCE y ROX aunque también se puede usar en el escritorio de Windows, MacOS y otros.

- GNU/Linux: GNU/Linux es el término empleado para referirse al sistema operativo Unix-like que utiliza como base las herramientas de sistema de GNU y el núcleo Linux. Su desarrollo es uno de los ejemplos más prominentes de software libre; todo el código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera bajo los términos de la GPL de GNU (Licencia Pública General de GNU) y otras licencias libres.

- GlassFish: GlassFish es un servidor de aplicaciones desarrollado por Sun Microsystems que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación. La versión comercial es denominada Sun GlassFish Enterprise Server. Es gratuito y de

código libre, se distribuye bajo un licenciamiento dual a través de la licencia CDDL y la GNU GPL.

- HTTP: El protocolo de transferencia de hipertexto (HTTP, HyperText Transfer Protocol) es el protocolo usado en cada transacción de la Web (WWW). HTTP fue desarrollado por el consorcio W3C y la IETF, colaboración que culminó en 1999 con la publicación de una serie de RFC, siendo el más importante de ellos el RFC 2616, que especifica la versión 1.1.

- HP OpenVMS: El sistema operativo OpenVMS (Sistema de Memoria Virtual) es un sistema multiusuario y multiproceso diseñado por Digital, ahora parte de Hewlett-Packard, para su utilización en entornos de tiempo compartido, tiempo real, procesamiento por lotes y procesamiento de transacciones. Conocido inicialmente como VMS, se ejecutaba sobre sistemas VAX, el nombre cambió a OpenVMS en 1990. Fue posteriormente portado a DEC Alpha (1992) e Intel Itanium.

- HP-Ux: HP-UX es la versión de Unix desarrollada y mantenida por Hewlett-Packard desde 1983, ejecutable típicamente sobre procesadores HP PA RISC y en sus últimas versiones sobre Intel Itanium (arquitectura Intel de 64 bits); a pesar de estar basada ampliamente en System V incorpora importantes características BSD. En la actualidad la última versión de este sistema operativo es la 11.23, también conocido como 11iv2 (2003), aunque existen numerosas instalaciones de sistemas más antiguos, especialmente HP-UX 10.x (1995-97) o incluso 9.x. (1992-95). Apartir de la versión 11.11 (2000) se usa un sistema de numeración doble, así la 11.11 es también conocida como 11i, la 11.20 es 11iv1.5 y así sucesivamente. Está previsto el lanzamiento de 11.31 (11iv3) a finales de 2006.

- Itanium: El Itanium, también conocido por su nombre en código Merced, fue el primer microprocesador de la arquitectura Intel Itanium (antes llamada IA64, creada por Hewlett-Packard y desarrollada conjuntamente por HP e Intel) que

Intel lanzó al mercado. Aunque su lanzamiento inicialmente se planeó para 1998, no se produjo hasta mayo de 2001.

- IBM: International Business Machines o IBM (NYSE: IBM) (conocida coloquialmente como el Gigante Azul) es una empresa que fabrica y comercializa herramientas, programas y servicios relacionados con la informática. Tiene su sede en Armonk (Estados Unidos) y está constituida como tal desde el 15 de junio de 1911.

- Java: Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

- Jboss: Jboss es un servidor de aplicaciones J2EE de código abierto implementado en Java puro. Al estar basado en Java, Jboss puede ser utilizado en cualquier sistema operativo que lo soporte.

- JSP: JavaServer Pages (JSP) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

- J2EE: Java Platform, Enterprise Edition o Java EE (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de programación parte de la Plataforma Java para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de N niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process, Java EE es también considerada informalmente como un estándar debido a que los suministradores deben

cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE; estandarizado por The Java Community Process / JCP.

- JVM: Una Máquina virtual Java (en inglés Java Virtual Machine, JVM) es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java.

- JonAS: JonAS es un servidor de aplicaciones J2EE de código abierto implementado en Java. JonAS forma parte de la iniciativa de código abierto de ObjectWeb.

- Kernel: En informática, el núcleo (también conocido en español con el anglicismo kernel, de la - 85 -aíz germánica Kern) es la parte fundamental de un sistema operativo. Es el software responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora o en forma más básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema.

- LVS: Linux Virtual Server (LVS) es una solución para gestionar balance de carga en sistemas Linux. Es un proyecto de código abierto iniciado por Wensong Zhang en mayo de 1998. El objetivo es desarrollar un servidor Linux de alto rendimiento que proporcione buena escalabilidad, confiabilidad y robustez usando tecnología clustering.

- MPI: MPI (“Message Passing Interface”, Interfaz de Paso de Mensajes) es un estándar que define la sintaxis y la semántica de las funciones contenidas en una biblioteca de paso de mensajes diseñada para ser usada en programas que exploten la existencia de múltiples procesadores.

- Mac OS X: Mac OS X (pronunciado Mac O-Ese Diez) es una línea de sistemas operativos computacionales desarrollada, comercializada y vendida por Apple Inc. Se basa en Unix y usa una interfaz gráfica desarrollada por Apple llamada

Aqua, que se inspira libremente en la interfaz de Mac OS Classic. El gestor de ventanas X11, característico en la familia de sistemas Unix, y Java se usan sólo para compatibilidad con software no nativo de Mac.

- NASA: NASA son las siglas, en inglés, para la Administración Nacional de Aeronáutica y del Espacio (inglés: National Aeronautics and Space Administration) de los Estados Unidos, que es la agencia gubernamental responsable de los programas espaciales.

- NIC: Una tarjeta de red permite la comunicación entre diferentes aparatos conectados entre sí y también permite compartir recursos entre dos o más equipos (discos duros, CD-ROM, impresoras, etc). A las tarjetas de red también se les llama adaptador de red o NIC (Network Interface Card, Tarjeta de Interfaz de Red en español). Hay diversos tipos de adaptadores en función del tipo de cableado o arquitectura que se utilice en la red (coaxial fino, coaxial grueso, Token Ring, etc.), pero actualmente el más común es del tipo Ethernet utilizando un interfaz o conector RJ-45.

- OpenMosix: OpenMosix es un sistema de cluster para Linux que permite a varias máquinas actuar como un único sistema multiprocesador (denominado en inglés SSI). Esto permite que no se tenga que reprogramar nuestras aplicaciones para aprovechen el clúster. Los procesos no saben en qué nodo del clúster se ejecutan, y es el propio openMosix el responsable de “engañarlos”, y redirigir las llamadas al sistema al nodo del cluster en el que se lanzó el proceso. openMosix implementa un algoritmo balanceador que permite repartir de forma óptima la carga, si esta el cluster bien calibrado.

- Oracle: Oracle es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), desarrollado por Oracle Corporation.

- Python: Python es un lenguaje de programación interpretado creado por Guido van Rossum en el año 1991. Se compara habitualmente con TCL, Perl, Scheme, Java y Ruby. En la actualidad Python se desarrolla como un proyecto de código abierto, administrado por la Python Software Foundation. La última versión estable del lenguaje es la 3.0.1.

- Parallel Virtual Machine (PVM): La Máquina Virtual Paralela (conocida como PVM por sus siglas en inglés de Parallel Virtual Machine) es una biblioteca para el cómputo paralelo en un sistema distribuido de computadoras. Está diseñado para permitir que una red de computadoras heterogénea comparta sus recursos de cómputo (como el procesador y la memoria RAM) con el fin de aprovechar esto para disminuir el tiempo de ejecución de un programa al distribuir la carga de trabajo en varias computadoras.

- Rocks: (originalmente llamado NPACI Rocks) es una distribución de Linux para clústers de computadores de alto rendimiento. Fue inciada por la NPACI y la SDSC in 2000, y fue financiada inicialmente en parte por una subvención de la NSF (2000-2007) pero actualmente está financiada por la siguiente subvención de la NSF. Rocks se basó inicialmente en la distribución Red Hat Linux, sin embargo las versiones más modernas de Rocks están basadas en CentOS, con un instalador anaconda modificado, que simplifica la instalación 'en masa' en muchos computadores. Rocks incluye muchas herramientas (tales como MPI) que no forman parte de CentOS pero son los componentes integrales que hacen un grupo de ordenadores en un clúster.

- Round Robin: Uno de los algoritmos de planificación de procesos más simples dentro de un sistema operativo que asigna a cada proceso una porción de tiempo equitativa y ordenada.

- SSH: SSH (Secure Shell, en español: intérprete de órdenes seguro) es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo la computadora mediante un intérprete de comandos, y también puede redirigir el

tráfico de X para poder ejecutar programas gráficos si se tiene un Servidor X (en sistemas Unix) corriendo.

- SSL: Secure Sockets Layer –Protocolo de Capa de Conexión Segura- (SSL).

- Servlet: Los servlets son objetos que corren dentro del contexto de un contenedor de servlets (ej: Tomcat) y extienden su funcionalidad. También podrían correr dentro de un servidor de aplicaciones (ej: OC4J Oracle), que, además de contenedor para servlet, tendrá contenedor para objetos más avanzados, como son los EJB (Tomcat sólo es un contenedor de servlets).

- Sockets: Socket designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiarse cualquier flujo de datos, generalmente de manera fiable y ordenada.

- Solaris: Solaris es un sistema operativo de tipo Unix desarrollado por Sun Microsystems desde 1992 como sucesor de SunOS. Es un sistema certificado oficialmente como versión de Unix. Funciona en arquitecturas SPARC y x86 para servidores y estaciones de trabajo.

- Servidores Web: Un servidor web es un programa que implementa el protocolo HTTP (HyperText Transfer Protocol). Este protocolo pertenece a la capa de aplicación del modelo OSI y está diseñado para transferir lo que se llama hipertextos, páginas web o páginas HTML (HyperText Markup Language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

- Sun Microsystems: Sun Microsystems es una empresa informática de Silicon Valley, fabricante de semiconductores y software.

- Software Libre: Software libre (en inglés free software) es la denominación del software que brinda libertad a los usuarios sobre su producto adquirido y por

tanto, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

- TCP/IP: familia de protocolos de Internet, en ocasiones se le denomina conjunto de protocolos TCP/IP, en referencia a los dos protocolos más importantes que la componen: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP).

- TLS: Transport Layer Security –Seguridad de la Capa de Transporte- (TLS).

- Unix: Unix (registrado oficialmente como UNIX) es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.

- Windows Server: Windows Server es una marca que abarca una línea de productos servidor de Microsoft Corporation.

- WebLogic: Oracle WebLogic es un servidor de aplicaciones Java EE y también un servidor web HTTP desarrollado por BEA Systems posteriormente adquirida por Oracle Corporation. Se ejecuta en Unix, Linux, Microsoft Windows, y otras plataformas.

- Xerox PARC: Xerox PARC (Palo Alto Research Center, (centro de investigación de Palo Alto) era una división de investigación de Xerox Corporation, con sede en Palo Alto (California, EE.UU.). Fue fundado en 1970 como consecuencia directa del fenomenal éxito de la empresa y ampliado como compañía independiente (aunque propiedad de Xerox) en 2002. Es famoso por crear esencialmente el paradigma moderno de interfaz gráfica de usuario (GUI) para ordenador personal (PC).

Anexos

Anexo 1: Descripción expandida de los Casos de Uso del Sistema.

Caso de Uso:	Gestionar Perfil.
Actores.	Administrador de Sistemas, (inicia).
Propósito.	Permite la creación, modificación y eliminación de un perfil.
Resumen.	El caso de uso se inicia cuando el Arquitecto de Software elige la opción de crear un perfil. La creación del perfil inidica la selección de la arquitectura de clústeres a instalar, y cada uno de los datos de los servidores.
Referencias.	RF 1.1, RF 1.1.1, RF 1.1.2, RF 1.1.3 RF 1.1.4
Precondiciones.	Perfil cargado
Flujo Normal de Eventos.	
Sección "Modificar Perfil"	
Acciones del Actor.	**Respuestas del sistema.**
1- El caso de uso inicia cuando el Administrador de Sistemas selecciona la opción Redefinir del menú.	2- El sistema le muestra una ventana para cambiar el nombre y la arquitectura.
3- El Administrador de sistema selecciona la opción Aceptar.	4- El sistema muestra un formulario con los servidores.
5- El Administrador de Sistemas selecciona un servidor y cambia los datos de los campos y selecciona la opción Modificar.	6- El sistema modifica los cambios y la da opción de guardarlos cambios.
	7- Finaliza el caso de uso.
Flujo Alterno 1: "Validar Datos".	
	6- El sistema muestra mensajes de error de validación de los campos dando sugerencias y luego permite ir al paso 5 del flujo normal.
Poscondiciones de éxito.	Se modificó el perfil.
Poscondiciones de fallo.	No se modificó el perfil.
Prototipo de Interfaz	

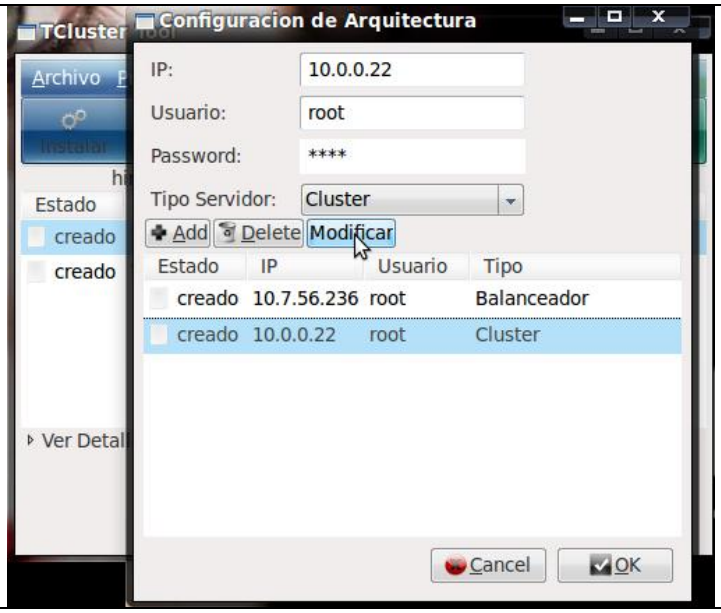


Tabla 12. Descripción Caso de Uso Modificar Perfil

Caso de Uso:	Cargar Perfil.
Actores.	Administrador de Sistemas, (inicia).
Propósito.	Permite la carga de un perfil.
Resumen.	El caso de uso se inicia cuando el Arquitecto de Software elige la opción de cargar un perfil.
Referencias.	RF 1, RF 1.1, RF 1.1.1, RF 1.1.2, RF 1.2, RF 1.3, RF 1.4
Precondiciones.	
Flujo Normal de Eventos.	
Acciones del Actor.	**Respuestas del sistema.**
1- El caso de uso inicia cuando el Administrador de Sistemas selecciona la opción Cargar.	2- El sistema le muestra un cuadro de dialogo con los perfiles existentes.
3- El Administrador de Sistemas selecciona de la lista el perfil que desea cargar.	4- El sistema muestra los datos del perfil en el formulario principal.
	5- Finaliza el caso de uso.
Flujo Alterno 1: "Validar Datos".	
	3- El sistema muestra un mensaje de error al no poder cargar el perfil y pasa al paso 2 del Flujo Normal de Eventos.
Poscondiciones de éxito.	Se cargo el perfil.
Poscondiciones de fallo.	No se cargo el perfil.
Prototipo de Interfaz	

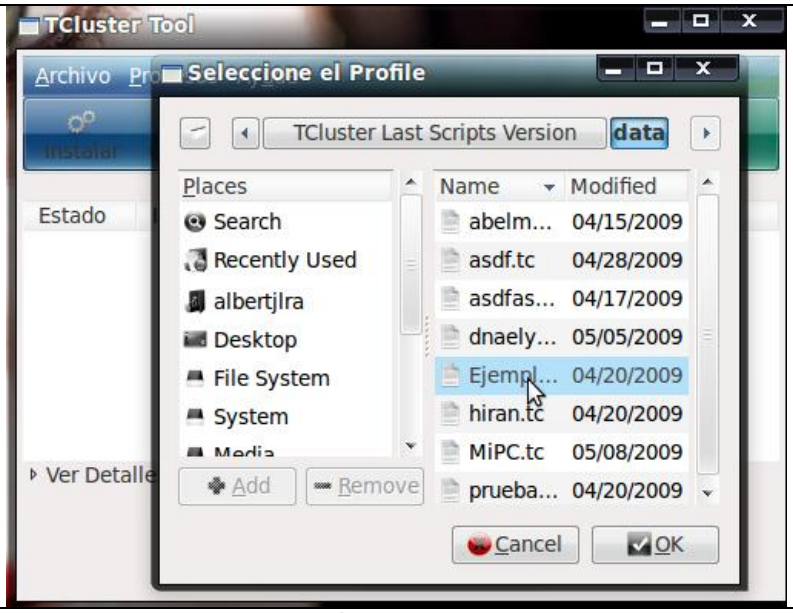


Tabla 13. Descripción Caso de Uso Cargar Perfil

Caso de Uso:	Asegurar Aplicación.
Actores.	Administrador de Sistemas, (inicia).
Propósito.	Permite asegurar la aplicación Java con certificados digitales.
Resumen.	El caso de uso se inicia cuando el Arquitecto de Software elige la opción de cargar un perfil.
Referencias.	RF 4, RF 4.1, RF 4.2, RF 4.3, RF 4.4, RF 4.5
Precondiciones.	Perfil cargado
Flujo Normal de Eventos.	
Acciones del Actor.	**Respuestas del sistema.**
1- El caso de uso inicia cuando el Administrador de Sistemas selecciona la opción Asegurar.	2- El sistema le muestra un cuadro de dialogo para introducir el nombre del virtual host que se creará en el servidor web Apache 2 el cual contendrá la configuración para cargar el certificado digital.
3- El Administrador de Sistemas coloca el nombre del virtual host y selecciona la opción aceptar.	4- El sistema genera un certificado digital de 1024 bits de encriptación pidiéndole al Administrador del Sistema los datos necesarios para la creación del mismo.
5- El Administrador de Sistema entra los datos necesarios para la creación del certificado.	6- El sistema genera el certificado digital y configura el virtual host en el servidor web Apache 2.
	7- Finaliza el caso de uso.
Flujo Alterno 1: "Error".	
	5- El sistema muestra un mensaje de error al no poder generar el certificado o reiniciar el servidor web Apache 2.
6- El Administrador de Sistemas verifica los errores y vuelve al paso 1 del Flujo Normal de	

Eventos.	
Poscondiciones de éxito.	Se aseguró la Aplicación Java.
Poscondiciones de fallo.	No se aseguró la Aplicación Java.
Prototipo de Interfaz	

Tabla 14. Descripción Caso de Uso Asegurar Aplicación.

Caso de Uso:	Desplegar Aplicación.
Actores.	Administrador de Sistemas, (inicia).
Propósito.	Permite desplegar la aplicación Java en cada uno de los nodos Apache Tomcat.
Resumen.	El caso de uso se inicia cuando el Arquitecto de Software elige la opción de desplegar aplicación.
Referencias.	RF 5, RF 5.1, RF 5.2, RF 5.3
Precondiciones.	Perfil cargado
Flujo Normal de Eventos.	
Acciones del Actor.	**Respuestas del sistema.**
1- El caso de uso inicia cuando el Administrador de Sistemas selecciona la opción Desplegar.	2- El sistema le muestra un cuadro de dialogo para seleccionar el fichero .war.
3- El Administrador de Sistemas selecciona el nodo que desea desplegar la aplicación.	4- El sistema copia el fichero .war para el nodo Apache Tomcat, y reinicia el mismo.
	5- Finaliza el caso de uso.
Flujo Alterno 1: "Error".	
	3- El sistema muestra un mensaje de error al no poder cargar el fichero .war y para a paso 1 del Flujo Normal de Eventos.
Poscondiciones de éxito.	Se desplegó la Aplicación Java.
Poscondiciones de fallo.	No se desplegó la Aplicación Java.
Prototipo de Interfaz	

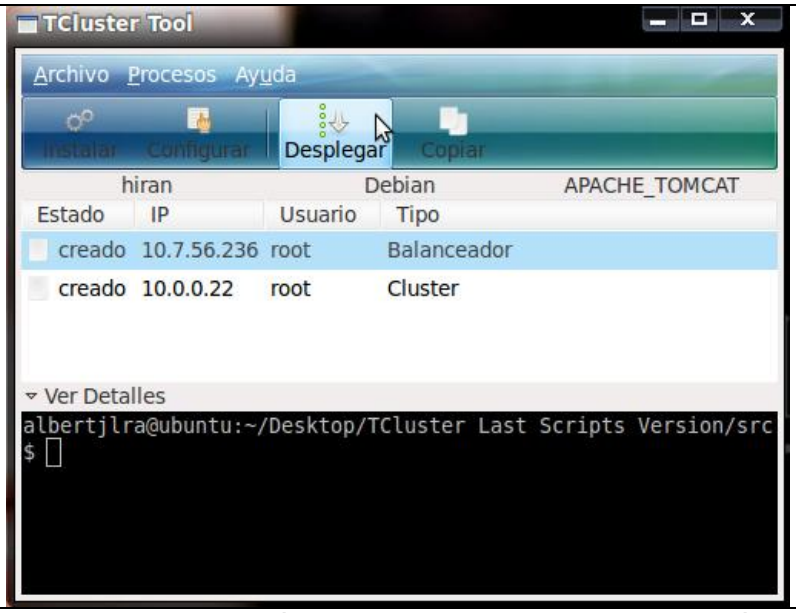


Tabla 15. Descripción Caso de Uso Desplegar Aplicación.

Anexo 2: Descripción de las principales Clases de Diseño.

Nombre:	Perfil
Tipo de Clase:	Entidad
Atributo	**Tipo**
nombre	str
distribucion	DISTRIBUCION
arquitectura	ARQUITECTURA
servidores	list
Para cada responsabilidad:	
Nombre:	**Descripción:**
agregar_servidor	Adiciona un servidor a la lista de servidores que contiene el perfil
limpiar_servidores	Resetea la lista dejándola vacía, sin ningún servidor
cantidad_de_cluster	Retorna la cantidad de servidores del tipo cluster que hay en la lista de servidores del perfil
validar_arquitectura	Confirma que la arquitectura del perfil definida por el usuario es valida

Tabla 16. Descripción Clase Perfil.

Nombre:	**GestionarPerfil**
Tipo de Clase:	Controladora
Atributo	**Tipo**
dataDir	str
perfil	Perfil
direccion	str
Para cada responsabilidad:	
Nombre:	**Descripción:**
crear_perfil()	Crea un nuevo perfil definido por el usuario y lo almacena en un fichero pera ser utilizado posteriormente
salvar_perfil()	Guarda el perfil en un fichero o cualquier cambio que se realice sobre el
cargar_perfil()	Carga el objeto perfil de un fichero previamente guardado y lo pone a dispocion del usuario
eliminar_perfil()	Elimina el perfil que está cargado del disco duro

Tabla 17. Descripción Clase Gestionar Perfil.

Nombre:	Procesos
Tipo de Clase:	Controladora
Atributo	**Tipo**
scripts	Scripts
profile	Perfil
terminal	Terminal
Para cada responsabilidad:	
Nombre:	**Descripción:**
indice()	Conforma la llave para el diccionario donde se encuentran las direcciones de los scripts de instalación y configuración
copiar_scripts()	Copia los scripts de instalación y configuración para un determinado servidor seleccionado por el usuario
instalar()	Instala el servidor ejecutando los scripts pertinentes para cada tarea
configurar()	Configura el servidor ejecutando los scripts necesarios
seguridad()	Configura la seguridad mediante los certificados digitales para ello ejecuta los scripts necesarios y solicita datos a través de la Terminal
operacion_valida()	Valida si la operación a realizar sobre el servidor es validad partiendo del estado en que se encuentra este servidor

Tabla 18. Descripción Clase Procesos.

Anexo 3: Diagramas de Interacción.

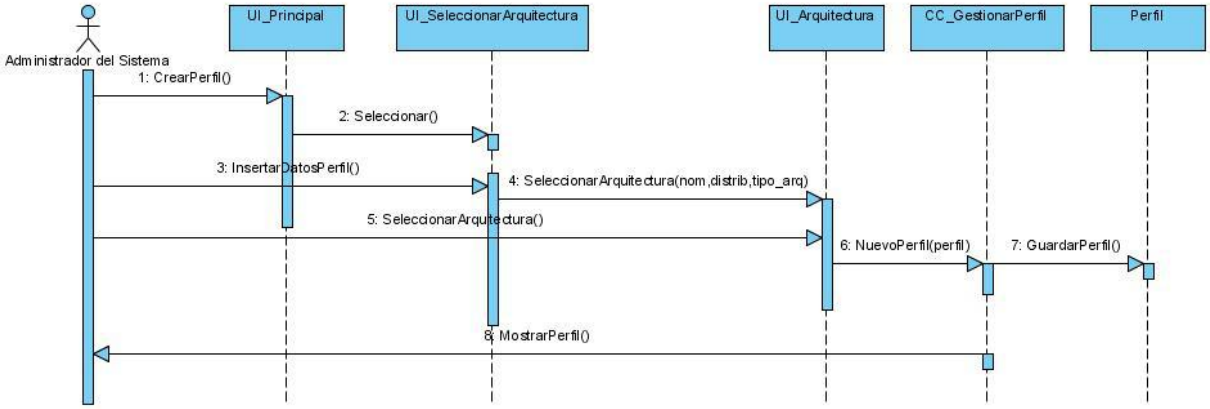


Figura 12. Diagrama de Secuencia Caso de Uso Crear Perfil.

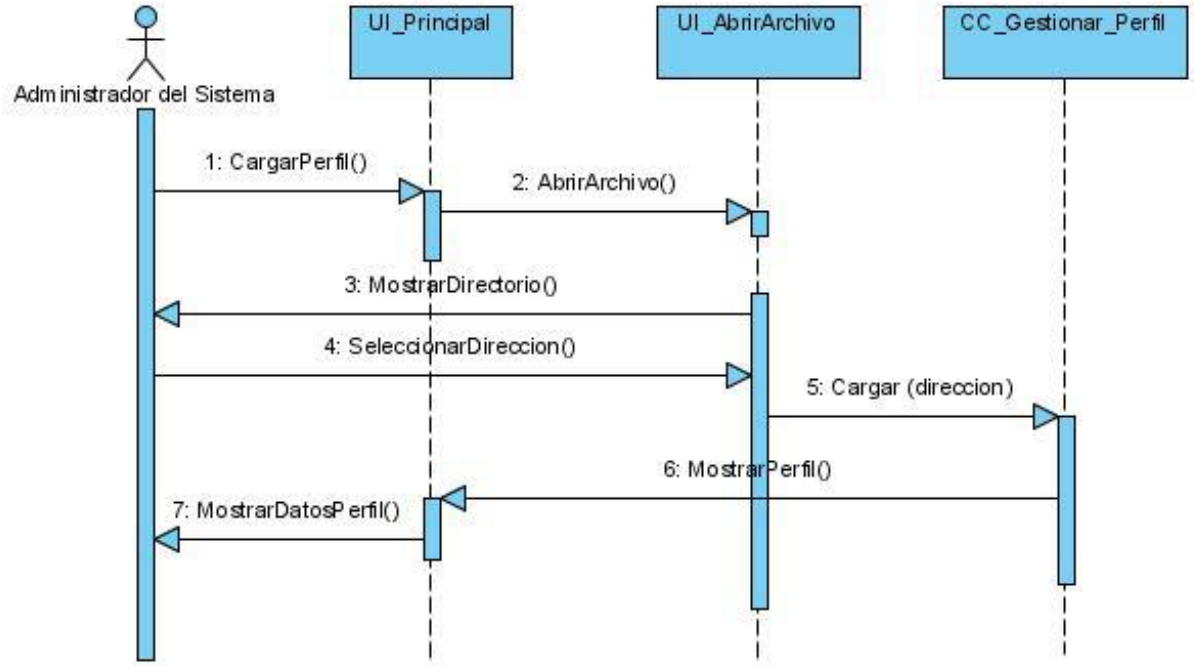


Figura 13. Diagrama de Secuencia Caso de Uso Cargar Perfil.

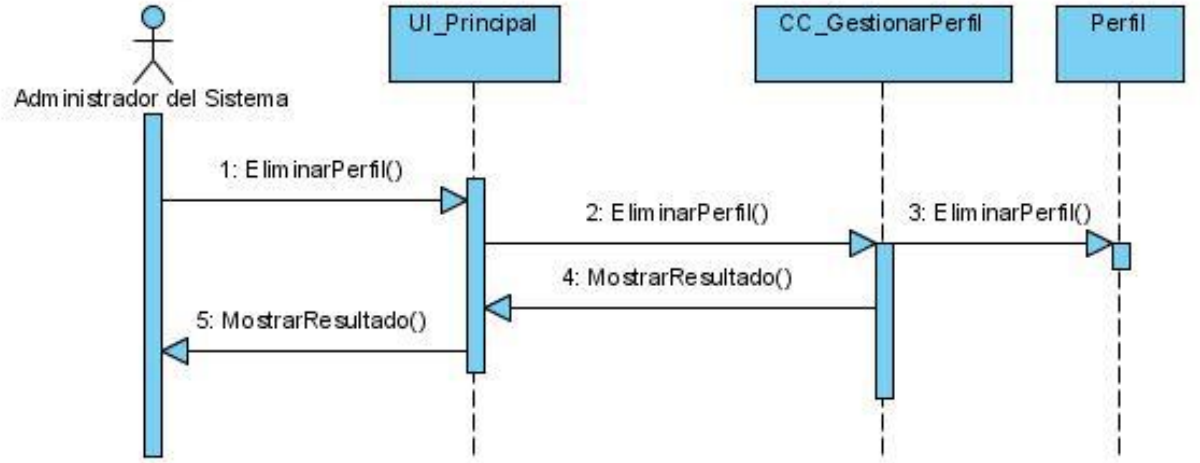


Figura 14. Diagrama de Secuencia Caso de Uso Eliminar Perfil.

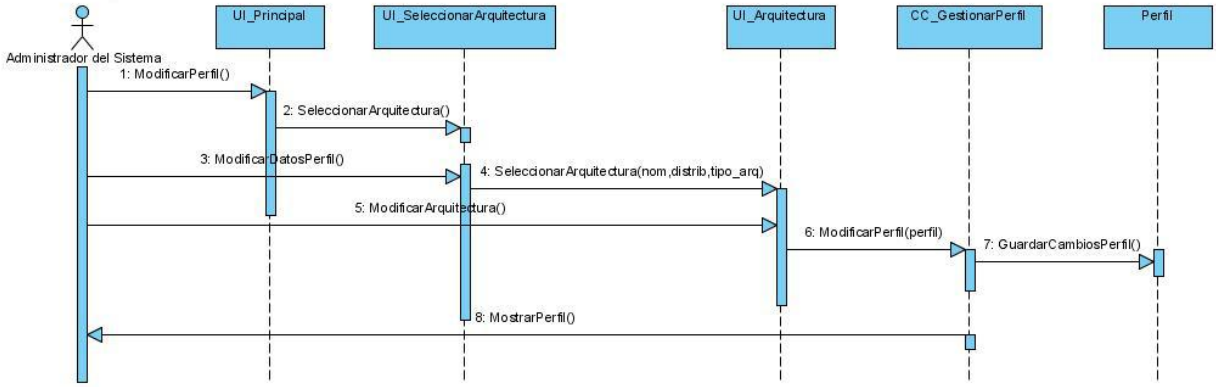


Figura 15. Diagrama de Secuencia Caso de Uso Modificar Perfil.

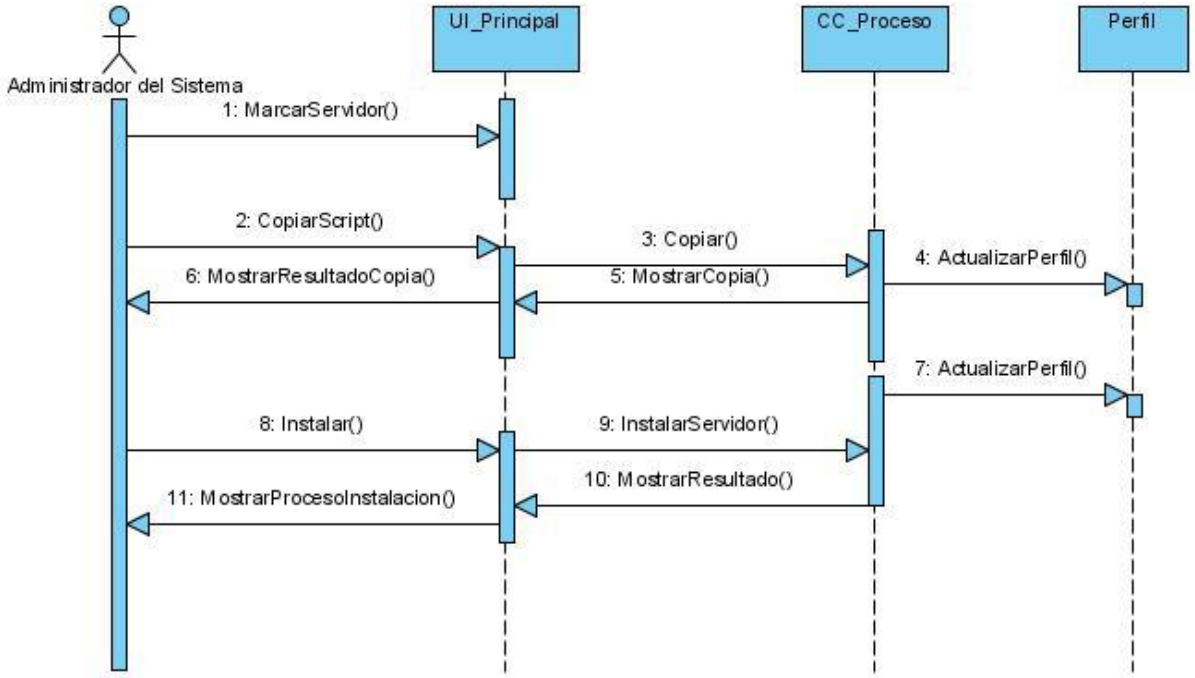


Figura 16. Diagrama de Secuencia Caso de Uso Instalar Servidor.

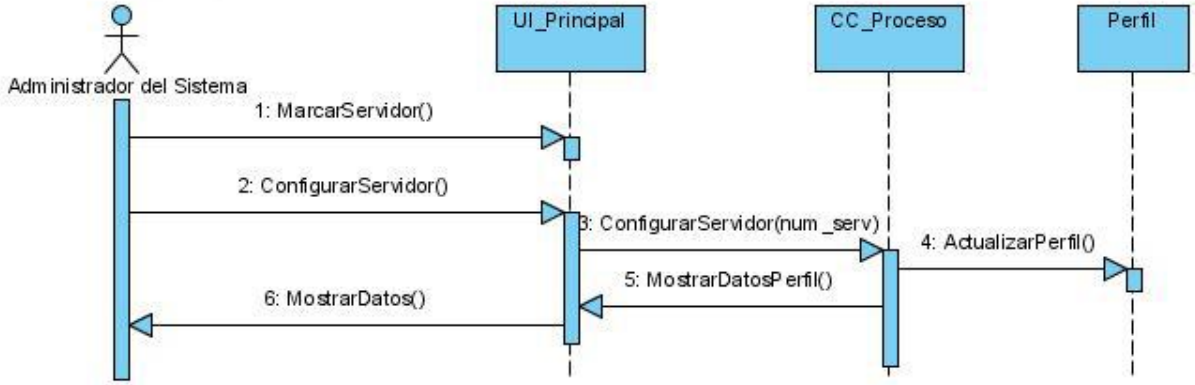


Figura 17. Diagrama de Secuencia Caso de Uso Configurar Servidor.

Anexo 4: Ficheros de configuración (Variante 1 Apache - n Apache Tomcats).

1. Fichero tomcat5.5

JAVA_HOME=¨URL de la maquina virtual¨ Ejemplo: /usr/jvm/sun-java6-xx

```
CATALINA_OPTS="-Djava.awt.headless=true -Xmx1024M -server \
        -XX:MaxPermSize=512m \
        -XX:+UseConcMarkSweepGC \
        -XX:+CMSPermGenSweepingEnabled \
        -XX:+CMSClassUnloadingEnabled"

TOMCAT5_SECURITY=no
```

2. Fichero context.xml

```
<!-- The contents of this file will be loaded for each web application -->
<Context distributable="true">
   <!-- Default set of monitored resources -->
   <WatchedResource>WEB-INF/web.xml</WatchedResource>

   <!-- Uncomment this to disable session persistence across Tomcat restarts -->
   <!--
   <Manager pathname="" />
   -->
</Context>
```

3. Fichero server.xml

#se cambia parámetros por defecto para mayor seguridad

```
<Server port="8025" shutdown="ApagarTomcatServer">
```

#se deshabilita el conector http para que no haya acceso por esta vía, así solamente el #Apache tendrá acceso, y los usuarios a las aplicaciones que el Apache de acceso.

```
<!--
   <Connector port="8180" maxHttpHeaderSize="8192"
        maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
        enableLookups="false" redirectPort="8443" acceptCount="100"
        connectionTimeout="20000" disableUploadTimeout="true" />
        -->
```

#se habilita el conector AJP por el cual el Apache tendrá acceso.

```
 <Connector port="8009"
            enableLookups="false" redirectPort="8443" protocol="AJP/1.3" />
```

#se definine el identificador del clúster, puede ser el nombre que se desee, por ejemplo ¨c1¨ en el caso del clúster 1.

```
<Engine name="Catalina" defaultHost="localhost" jvmRoute="c1">
```

#se habilita las configuraciones para que el Tomcat funciones como clúster.

```
<Cluster className="org.apache.catalina.cluster.tcp.SimpleTcpCluster"
             managerClassName="org.apache.catalina.cluster.session.DeltaManager"
             expireSessionsOnShutdown="false"
             useDirtyFlag="true"
             notifyListenersOnReplication="true">

          <Membership
             className="org.apache.catalina.cluster.mcast.McastService"
             mcastAddr="228.0.0.4"
             mcastPort="45564"
             mcastFrequency="500"
             mcastDropTime="3000"/>

          <Receiver
             className="org.apache.catalina.cluster.tcp.ReplicationListener"
             tcpListenAddress="ip de la PC local" o "auto"
             tcpListenPort="4001"
             tcpSelectorTimeout="100"
             tcpThreadCount="6"/>

          <Sender
             className="org.apache.catalina.cluster.tcp.ReplicationTransmitter"
             replicationMode="pooled"
             ackTimeout="15000"/>
```

```
        <Valve className="org.apache.catalina.cluster.tcp.ReplicationValve"
            filter=".*\.gif;.*\.js;.*\.jpg;.*\.png;.*\.htm;.*\.html;.*\.css;.*\.txt;"/>

        <Deployer className="org.apache.catalina.cluster.deploy.FarmWarDeployer"
            tempDir="/tmp/war-temp/"
            deployDir="/tmp/war-deploy/"
            watchDir="/tmp/war-listen/"
            watchEnabled="false"/>

        <ClusterListener
className="org.apache.catalina.cluster.session.ClusterSessionListener"/>
      </Cluster>
```

4. Fichero workers.properties

```
workers.tomcat_home=/var/lib/tomcat5.5

workers.java_home=/usr/lib/jvm/

worker.list=balancer, jkstatus

worker.c1.port=8009

worker.c1.host=ipTomcat1

worker.c1.type=ajp13

worker.c1.lbfactor=1

worker.c1.redirect=c2        //nodo al que redireccionará en caso de fallo.

worker.c2.port=8009

worker.c2.host= ipTomcat2

worker.c2.type=ajp13

worker.c2.lbfactor=1

worker.c2.redirect=c1 //nodo al que redireccionará en caso de fallo
```

worker.balancer.type=lb

worker.balancer.host= ipBalanceador

worker.balancer.balance_workers=c1,c2 //lista de nodos

worker.balancer.method.B

worker.balancer.sticky_session=True

worker.jkstatus.type=status

5. Fichero httpd.conf

JkWorkersFile /etc/apache2/workers.properties

JkLogFile /var/log/apache2/mod_jk.log

JkLogLevel info

JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "

JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories

JkRequestLogFormat "%w %V %T"

JkMount /jkmanager/* jkstatus //administrador de balanceador

JkMount /"Nombre Aplicación" balancer //para montar una aplicación desplegada en el Tomcat

JkMount /"Nombre Aplicación" /* balancer

<Location /jkmanager/>

JkMount jkstatus

</Location>

6. Fichero del Virtual Host

NameVirtualHost *:80

<VirtualHost *:80>

ServerName dominio.com

Redirect 301 / https://dominio.com/

CustomLog /var/log/apache2/access_site.log combined

ErrorLog /var/log/apache2/error_site.log

```
</VirtualHost>

NameVirtualHost *:443
<VirtualHost *:443>
        ServerName dominio.com
        HostnameLookups Double
        CustomLog /var/log/apache2/access_site.log combined env=!dontlog
        SetEnvIf Request_URI "^/u" dontlog
        ErrorLog /var/log/apache2/error_convenio.log
        Loglevel warn
        SSLEngine On
        SSLCertificateFile /etc/apache2/ssl/cert.pem

</VirtualHost>
```

7. Fichero sysctl.conf

```
#
# /etc/sysctl.conf - Configuration file for setting system variables
# See sysctl.conf (5) for information.
#
#kernel.domainname = example.com
#net/ipv4/icmp_echo_ignore_broadcasts=1
# Uncomment the following to stop low-level messages on console
#kernel.printk = 4 4 1 7
##############################################################################3
# Functions previously found in netbase
#
# Uncomment the next line to enable Spoof protection (reverse-path filter)
```

```
#net.ipv4.conf.default.rp_filter=1


# Uncomment the next line to enable TCP/IP SYN cookies
#net.ipv4.tcp_syncookies=1
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.conf.default.forwarding=1
# Uncomment the next line to enable packet forwarding for IPv6
#net.ipv6.conf.default.forwarding=1
net.ipv4.conf.all.arp_ignore=1
net.ipv4.conf.eth0.arp_ignore=1
net.ipv4.conf.all.arp_announce=2
net.ipv4.conf.eth0.arp_announce=2
```

8. Fichero interfaces

```
auto lo:0
iface lo:0 inet static
  address ip virtual
  netmask 255.255.255.255
pre-up sysctl -p > /dev/null
```

9. Fichero ha.cf

```
debugfile /var/log/heartbet_debug
logfile        /var/log/heartbet_log
logfacility        local7
bcast        eth0
```

mcast eth0 225.0.0.1 694 1 0

auto_failback on

#nodos balanceadores

node loadb1

node loadb2

respawn hacluster /usr/lib/heartbeat/ipfail

apiauth ipfail gid=haclient uid=hacluster

10. Fichero haresourses

loadb1 ldirectord::ldirectord.cf LVSSyncDaemonSwap::master IPaddr2::192.168.0.105/24/eth0/192.168.0.255

11. Fichero authkeys

auth 2

#1 crc

2 sha1 ultramonkey

#3 md5 Hello!

12. Fichero ldirectord

checktimeout=10

checkinterval=2

autoreload=no

logfile="/var/log/heartbet_log"

quiescent=yes

virtual=ip virtual:80

real=ip nodo 1:80 gate

real=ip nodo 2:80 gate

fallback=127.0.0.1:80 gate

service=http

scheduler=rr

protocol=tcp

checktype=connect