Universidad de las Ciencias Informáticas

Facultad 3

Título: Herramienta para la instalación y configuración de clústeres del gestor de bases de datos PostgreSQL.

Trabajo de Diploma para optar por el título de:

INGENIERO EN CIENCIAS INFORMÁTICAS

Autores:

Yanet Rodriguez López.

Joan Jon Iglesia.

Tutores:

Ing. Abel Arias Hernández.

Ing. Alberto Jesús La Rosa Agramonte.

Ciudad de la Habana, Cuba

Junio, 2010.

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 _____.

Yanet Rodriguez López.

_____
Autor

Joan Jon Iglesia.

_____
Autor

Abel Arias Hernández.

_____
Tutor

Alberto Jesús La Rosa Agramonte.

_____
Tutor

Agradecimientos

Agradecemos a nuestros padres que son la guía de nuestra formación personal y la principal fuente de inspiración para la realización de este trabajo.

A nuestros tutores Abel Arias Hernández y Alberto Jesús la Rosa, gracias por su apoyo y su paciencia, así como la seriedad prestada en este trabajo de tanta importancia para nosotros.

A la Universidad de las Ciencias Informáticas por brindarnos los medios para nuestra formación profesional durante estos cinco años, así como nuestros profesores, a todos ellos, que de una forma u otra han contribuido a formarnos como personas de principios e ideales.

A nuestros amigos, los que comenzamos juntos, los que llegaron luego, aquellos que quedaron en el camino, los de siempre, gracias por estar ahí.

A todos aquellos que han contribuido de una forma u otra a la realización de este trabajo, gracias, y si no hemos querido decir nombres es para que nadie se sienta herido, pero ustedes saben quiénes son.

A todos, muchas gracias.

Dedicatoria:

Dedico este trabajo a mi madre, por ser mi mayor fuente de inspiración, más que madre, mi amiga incondicional, mi razón de ser.

A Ricardo, por ser mi guía durante toda mi vida, el aliento para seguir adelante, mi mayor ejemplo en cada momento de superación.

A esa personita que hace que cada día trate de convertirme en una mejor persona para servirle de ejemplo, mi hermanita del alma, y a mi hermano que aunque un poco alocado, es de mis personas favoritas en este mundo.

A mi abuelita y mis tíos, todos ellos, porque sin excepciones, son un apoyo incondicional sin el cual no habría llegado hasta aquí.

A mi segunda familia, porque ha sido mi más grande apoyo emocional durante más de tres años.

Y por último, pero no menos especial, a esa persona que mas que mi compañero y amigo, se ha convertido en lo más indispensable en mi vida. A ti mi amor.

Yanet Rodríguez López.

A mis padres Patrisia y Martín por ser mis guías durante todos estos años de vida y exigirme cada día más para lograr ser la persona que soy.

A tata por ser mi ejemplo a seguir y ser el amigo que siempre está a mi disposición.

A mi cuñada por regalarnos la mayor alegría de la casa (Birulito).

A mi otra familia por apoyarme y acogerme como uno más de ellos.

Y a mi nene por regalarme estos años juntos, que han sido de los mejores de mi vida.

Joan Jon Iglesia.

Resumen:

La Universidad de la Ciencias Informáticas, como uno de los principales centros de producción de software en nuestro país, se encuentra en un proceso de cambio, para que el desarrollo de todos los productos sea realizado con herramientas de software libre. Entre los gestores de bases de datos que más se utilizan en los proyectos que se llevan a cabo en la Universidad, se encuentra PostgreSQL, esto se debe a sus grandes ventajas, pero tiene como consecuencias el consumo de recursos y la velocidad de respuesta. Es por ello que para sistemas de gran envergadura se estila a utilizar las tecnologías clúster para ganar en rendimiento, disponibilidad y capacidad de las conexiones.

El presente trabajo propone el estudio e investigación de la tecnología clúster en bases de datos de alta concurrencia y desarrollar una herramienta que implemente las funcionalidades necesarias para garantizar la instalación y configuración de estos clústeres de manera más ágil y amigable. Un estudio realizado acerca de las tendencias tecnológicas actuales sustenta la decisión de optar por desarrollar dicha herramienta con software libre, utilizando para ello Netbeans como IDE y Java 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. Lo anteriormente planteado dio como resultado una aplicación de escritorio que responde a la problemática actual sobre la instalación y configuración de los clústeres, en este caso específico, para los clústeres del Gestor de bases de datos PostgreSQL.

Índice

Introducción

Uno de los centros más importantes en el proceso de desarrollo de software en nuestro país es sin dudas la Universidad de la Ciencias Informáticas (UCI), la misma cuenta con numerosos proyectos que son los encargados de desarrollar los productos de software tanto para las empresas nacionales como algunos convenios con empresas extranjeras. En estos momentos la UCI se encuentra en un proceso de cambio, para que el desarrollo de todos los productos sea realizado con herramientas de software libre.

En la actualidad, se hace necesaria la migración hacia el software libre por parte de las pequeñas y medianas empresas del software, principalmente aquellas pertenecientes a los países menos desarrollados, entre ellos Cuba. La utilización del software libre trae consigo grandes ventajas, entre las principales se puede mencionar que es menos costoso que el software privativo. La Licencia Pública General de GNU (GPL), está orientada principalmente a proteger la libre distribución, modificación y uso de software, su propósito es declarar que el software cubierto por esta licencia sea software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios. Además, el libre acceso al código fuente permite a los desarrolladores modificar las funcionalidades del software según sus necesidades.

Uno de los gestores de bases de datos más utilizados en estos momentos, tanto en el mundo del código abierto como en la UCI, es PostgreSQL, esto se debe a que está bajo licencia BSD, lo que proporciona más libertades que, por ejemplo, la GPL. PostgreSQL incluye características de la orientación a objetos, como son: la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. Es compatible con casi todos los sistemas operativos. Contiene soporte nativo para los lenguajes más populares del medio, por ejemplo: PHP, C, C++, Perl, Python, etc, así como de todas las características de una base de datos profesional (triggers, store procedures, funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas, etc.).

Otras de sus principales características son:

- Atomicidad: Asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
- Consistencia: Asegura que sólo se empiece aquello que se puede acabar. Se ejecutan las operaciones que no van a romper la reglas y directrices de integridad de la base de datos.

- Aislamiento: Asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generará ningún tipo de error.
- Durabilidad: Asegura que una vez realizada la operación, esta persistirá y no se podrá deshacer.

Entre las desventajas de utilizar este sistema están el consumo de recursos y la velocidad de respuesta, pues es uno de los sistemas que más carga el servidor y es mucho más lento que la mayoría de los sistemas gestores de bases de datos, es por ello que para sistemas de gran envergadura se estila a utilizar las tecnologías clúster para ganar en rendimiento, seguridad y capacidad de las 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.

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

En cuanto a este tema, la UCI cuenta con muy poca experiencia, primeramente porque hace muy poco tiempo se está migrando a Software Libre y no existe un conocimiento profundo acerca de estas tecnologías, además de los diferentes métodos que se pueden utilizar para configurar un clúster para el gestor de bases de datos PostgreSQL.

Situación Problemática.

Hoy día realizar la instalación y configuración de estos clústeres es algo complicado y engorroso para el personal que se encarga del despliegue y soporte dentro de los proyectos productivos, debido a que hay que hacerlo a través de consola, pudiendo contribuir a errores de configuración que afectaría el funcionamiento del clúster y de los sistemas que consuman sus servicios.

Problema

¿Cómo mejorar el proceso de instalación y configuración de clústeres del gestor de bases de datos PostgreSQL?

Objeto de Estudio.

Clústeres de Servidores.

Campo de Acción.

Proceso de instalación y configuración de clústeres del gestor de bases de datos PostgreSQL.

Objetivo.

Desarrollar una herramienta que facilite la instalación y configuración de clústeres del gestor de bases de datos PostgreSQL.

Hipótesis.

Si se desarrolla una herramienta que facilite el proceso de instalación y configuración de clústeres del gestor de bases de datos PostgreSQL, entonces se logrará agilizar dicho proceso, de forma más amigable.

Tareas de la Investigación.

1. Análisis del estado del arte de clústeres de servidores de bases de datos.
2. Estudio sobre las tecnologías de réplicas y failover existentes.
3. Estudio de las metodologías y herramientas para el desarrollo de software.
4. Definición de las herramientas y lenguajes de desarrollo, para el logro de una mejor integración con los servidores Linux.
5. Definición de una arquitectura que permita darle solución al problema.
6. Estudio del lenguaje BashScript para sistemas GNU Linux.
7. Desarrollo de la herramienta que se utilizará en la instalación y configuración de los clústeres del gestor de bases de datos PostgreSQL.

8. Establecimiento de la seguridad de todas las conexiones que se establezcan entre la herramienta y el servidor.
9. Validación de la herramienta en el despliegue de sistemas desarrollados en la facultad.

Métodos Científicos.

Con el objetivo de validar metodológicamente la investigación se utilizaron los siguientes métodos científicos:

Teóricos:

Analítico – Sintético: sirvió como punto de partida en la investigación, debido a que ayudó a identificar dentro del grupo de necesidades de los proyectos, el desarrollo de una herramienta capaz de facilitar el trabajo de los mismos en la instalación y configuración de los clústeres del gestor de bases de datos PostgreSQL, que hasta el momento se hacía a través de código en consola.

Inductivo – Deductivo: permitió identificar la necesidad de una herramienta para la automatización del proceso de instalación y configuración de los clústeres del gestor de bases de datos PostgreSQL a partir de los errores que se cometían en la instalación y la configuración de los mismos en la forma tradicional.

Histórico – Lógico: permitió definir los principales errores en la instalación y configuración de los clústeres del gestor de bases de datos PostgreSQL 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 del gestor de bases de datos PostgreSQL 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, donde se realiza un estudio profundo y detallado de las tecnologías, herramientas y lenguajes de programación a emplear durante la construcción de la aplicación. Se fundamenta la metodología para el desarrollo de software utilizada como guía en la investigación.

En el capítulo 2 se explican los procesos del negocio a través de un modelo de negocio, y a partir de esto se comienza a 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, así como un diagrama de despliegue que identifica la estructura física con la que contará el subsistema, una vez llegado el proceso de instalación del producto.

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.

Capítulo 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 de su utilización así como los diferentes ambientes en que pueden funcionar. Se profundiza en la tecnología de clústeres para replicación y balanceo de carga de servidores de Bases de Datos, específicamente para el Gestor de Base de Datos PostgreSQL que es la base del problema planteado.

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, cuando los pioneros de la supercomputación intentaban difundir diferentes procesos entre varias computadoras, para luego poder recoger los resultados que dichos procesos debían producir. Con un hardware más barato y fácil de conseguir se pensó, como se ha demostrado en la actualidad, que se podrían conseguir resultados similares a los de esos ordenadores.

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[1] 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 [1].

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

[1] 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.

de un clúster de computadoras. Las redes de conmutación de paquetes fueron conceptualmente inventados por la corporación RAND en 1962 [1].

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 cluster" como hoy en día lo entendemos.

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, 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 que 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 [1].

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. VAXclúster, VMSclúster 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 no 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 1994, bajo el patrocinio del proyecto ESS (Earth and Space Sciences), un grupo de investigadores del CESDIS (Center of Excellence in Space Data and Information Sciences), que desarrolla proyectos para la NASA, construyeron un clúster consistente en 16 equipos con procesadores Intel DX4, interconectados por una red tipo Ethernet de canal múltiple con el objetivo específico de "ser un superordenador" capaz de realizar firmemente cálculos paralelos HPC.

En su honor, a los clústeres de este tipo, se les conoce genéricamente como clústeres tipo Beowulf, o, simplemente, Beowulf.

Beowulf fue un gran éxito, y los clústeres de este tipo pronto se popularizaron dentro de la NASA, y más allá, convirtiéndose en una técnica extremadamente popular para obtener cómputo de alto rendimiento a bajo costo.

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.

En sentido genérico, un clúster es un conjunto de ordenadores funcionando como unidad y trabajando juntas para tratar una única tarea. El término clúster se aplica a los conjuntos o conglomerados de computadoras construidos mediante la utilización de componentes de hardware comunes y que se comportan como si fuesen una única computadora.

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 sistema distribuido de alto rendimiento, así como la creciente necesidad de potencia computacional para aplicaciones que la requieran.

Simplemente, 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.

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: está diseñado para dar altas prestaciones en cuanto a capacidad de cálculo. Por medio del mismo, se pueden conseguir capacidades de cálculo superiores a las de un ordenador más caro que el costo conjunto de los ordenadores del clúster.

Alta disponibilidad: su función es la de esperar listos para entrar inmediatamente en funcionamiento en el caso de que falle algún otro servidor. La característica de flexibilidad que proporciona este tipo de tecnología de Clúster, lo hacen necesario en ambientes de intercambio intensivo de información.

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.

Los clústeres pueden también clasificarse como Clústeres de IT 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. Además, un clúster de determinado tipo, puede también presentar características de los otros.

En el caso que se plantea se decidió implementar un clúster de alta disponibilidad, debido a que no puede haber fallas de ningún tipo en el funcionamiento, ya sea por carga de peticiones 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 multiprocesadores 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 [26].

En redes de computadoras cada uno de los ordenadores 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: 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 del ordenador desde los niveles más básicos, permitiendo también la interacción con el usuario.

Debe ser multiproceso, multiusuario. Otras características deseables son la facilidad de uso y acceso y permitir además múltiples procesos y usuarios.

Un sistema operativo se puede encontrar normalmente en la mayoría de los aparatos electrónicos que utilicen microprocesadores para funcionar, ya que gracias a estos podemos 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 µs, 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 clústeres 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 arreglos 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).

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 clúster para proceder a su utilización.

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

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.

Su principal ventaja es su habilidad de llevar a cabo tareas de una escala que no sería realista a costo-efectivo para otros sistemas. Sin embargo, en general, la programación paralela representa una labor sumamente compleja.

1.4 Elementos necesarios para implementar un clúster del Gestor de Bases de Datos PostgreSQL.

1.4.1 Balanceo de carga

Cuando un servidor de Internet se vuelve lento debido a la congestión de información, la solución más obvia es ampliar la memoria, ampliar el disco duro o actualizar el procesador. Pero el tráfico de Internet está en constante crecimiento y lo anteriormente expuesto es sólo una solución temporal. La opción más razonable es, a largo plazo, configurar más servidores y repartir las peticiones de los clientes entre ellos. Esto incrementa la velocidad de acceso del usuario al servidor, mejora la fiabilidad del sistema y la tolerancia a fallos, permitiendo reparar o mantener cualquier servidor en línea sin que afecte al resto del servicio.

De forma sencilla, el balanceo de carga es la manera en que las peticiones de Internet son distribuidas sobre una fila de servidores. Existen varios métodos para realizar el balanceo de carga. Desde el simple "Round Robin" (repartiendo todas las peticiones que llegan de Internet entre el número de servidores disponibles para dicho servicio) hasta los equipos que reciben las peticiones, recogen información, en tiempo real, de la capacidad operativa de los equipos y la utilizan para enrrutar dichas peticiones individualmente al servidor que se encuentre en mejor disposición de prestar el servicio adecuado.

Los balanceadores de carga pueden ser soluciones hardware, tales como routers y switches que incluyen software de balanceo de carga preparado para ello, y soluciones software que se instalan en el back-end de los servidores.

Dos características importantes de los balanceadores de carga son:

- *Evitan la saturación de servidores:* De esta forma, podemos evitar que picos de acceso a los ordenadores (como por ejemplo los generados por campañas publicitarias), afecten al normal funcionamiento del aplicativo.

- *Gestiona los recursos de manera inteligente:* Permite gestionar y optimizar todos los recursos disponibles dando como resultado un acceso más rápido y estable a nuestros aplicativos.

Ventajas que proporcionan:

- Mayor capacidad de ampliación
- Mayor disponibilidad de los servicios
- Mayor capacidad de gestión

Método Round Robin

El método más simple de todos, es la solución Round Robin. Las peticiones clientes son distribuidas equitativamente entre todos los servidores existentes. Este método cíclico no tiene en cuenta las condiciones y carga de cada servidor. Esto puede llevarnos a tener servidores que reciben peticiones de carga mucho mayor, mientras tenemos servidores que apenas se encuentran utilizando recursos.

Otra limitación es que los problemas de los servidores no son recogidos inmediatamente. Esto puede llevarnos a estar enviando peticiones a un servidor que se encuentra fuera de servicio o que responde lentamente. Finalmente, el método Round Robin no aprovecha las diferentes prestaciones de los servidores (un PC Pentium normal puede estar obteniendo tantas peticiones como un multiprocesador Sun situado junto a él).

HeartBeat

La traducción directa de heartbeat sería: “latidos de corazón”. Funciona enviando periódicamente un paquete, que si no llegara, indicaría que un servidor no está disponible, por lo tanto se sabe que el servidor ha caído y se toman las medidas necesarias.

Dichos latidos se pueden enviar por una línea serie, por UDP o por PPP/UDP. De hecho los desarrolladores de HeartBeat recomiendan el uso de puertos serie por varias razones, entre las que destacan que están aislados de las tarjetas de red.

También incluye la toma de una dirección IP y un modelo de recursos, incluyendo grupos de recursos. Soporta múltiples direcciones IP y un modelo servidor primario/secundario. Se ha probado satisfactoriamente en varias aplicaciones, como son: servidores DNS, servidores proxy de caché, servidores web y servidores directores de LVS. El proyecto LVS recomienda HeartBeat para aumentar la disponibilidad de su solución, pero no es parte de LVS.

Cuando un ordenador deja de «latir» y se considera muerto, se hace una transición en el clúster. La mayoría de los mensajes de manejo del clúster que no son HeartBeats se realizan durante estas transiciones.

Heartbeat también se preocupa por la seguridad, permitiendo firmar los paquetes con CRC de 32 bits, MD5 y SHA1. Esto puede evitar el desastre que podría provocarse si un nodo no miembro se enmascarase como nodo miembro del clúster. El problema es que el entorno donde se ejecuta HeartBeat no debe parar nunca y con suerte ese entorno se mantendrá comunicado y funcionando durante años.

Hay varias operaciones de mantenimiento de seguridad que necesitan ser efectuadas en ese tiempo, como pueden ser cambio de claves y de protocolos de autentificación. HeartBeat está preparado para esos cambios disponiendo de ficheros para la configuración.

Para el desarrollo de la solución se utilizará como balanceador de carga HeartBeat debido a que puede llevar a cabo la detección de la caída de nodos, las comunicaciones y la gestión del clúster en un solo proceso. Es fácilmente portable, corre en todos los Linux conocidos, así como en FreeBSD y Solaris. Actualmente soporta un modelo de dependencias muy sofisticado para clústeres de N nodos, y es muy útil y estable.

1.4.2 Herramienta para la replicación

Existen en el mundo múltiples soluciones que permiten llevar a cabo la réplica de datos enfocadas al gestor de base de datos PostgresSQL. Pueden citarse entre ellos Postgres-R, PgReplicator, Usogres, ErServer, Slony-I, PgCluster, Pgpool-II y CyberCluster.

Pgpool-II

Pgpool-II es un mediador que trabaja entre servidores y clientes de base de datos PostgreSQL.

Refiere los protocolos de frontend y backend de PostgreSQL, y pasa las conexiones entre ellos. De ese modo, una aplicación de base de datos (frontend) cree que pgpool-II es el verdadero servidor de PostgreSQL, y el servidor (backend) ve a pgpool-II como uno de sus clientes. Debido a que pgpool-II es transparente tanto para el servidor como para el cliente, una aplicación de base de datos existente puede empezar a usarse con pgpool-II casi sin ningún cambio en su código fuente.

Funciona sobre Linux, Solaris, FreeBSD y la mayoría de las arquitecturas UNIX. Windows no está soportado. Las versiones de PostgreSQL soportadas son de la 6.4 o superior. Para usar la paralelización de consultas es necesaria la versión 7.4 o superior.

Pgpool-II proporciona las siguientes características:

- *Limita el excedente de conexiones***.** PostgreSQL soporta un cierto número de conexiones concurrentes y rechaza las que superen dicha cifra. Aumenta el límite máximo de conexiones, incrementa el consumo de recursos y afecta al rendimiento del sistema. Pgpool-II tiene también un límite máximo de conexiones, pero las conexiones extras se mantienen en una cola en lugar de devolver un error inmediatamente.

- *Pool de conexiones.* Mantiene abiertas las conexiones a los servidores PostgreSQL y las reutiliza siempre que se solicita una nueva conexión con las mismas propiedades (nombre de usuario, base de datos y versión del protocolo). Ello reduce la sobrecarga en las conexiones y mejora la productividad global del sistema.

- *Replicación.* Puede gestionar múltiples servidores PostgreSQL. El uso de la función de replicación permite crear una copia en dos o más discos físicos, de modo que el servicio puede continuar sin parar los servidores en caso de fallo en algún disco.

- *Balanceo de carga.* Si se replica una base de datos, la ejecución de una consulta SELECT en cualquiera de los servidores devolverá el mismo resultado. pgpool-II se aprovecha de la característica de replicación para reducir la carga en cada uno de los servidores PostgreSQL distribuyendo las consultas SELECT entre los múltiples servidores, mejorando así la productividad global del sistema. En el mejor caso, el rendimiento mejora proporcionalmente al número de servidores PostgreSQL. El balanceo de carga funciona mejor en la situación en la cual hay muchos usuarios ejecutando muchas consultas al mismo tiempo.

- *Paralelización de consultas.* Al usar la función de paralelización de consultas, los datos pueden dividirse entre varios servidores, de modo que la consulta puede ejecutarse en todos los servidores de manera concurrente para reducir

el tiempo total de ejecución. La paralelización de consultas es una solución adecuada para búsquedas de datos a gran escala.

Slony- I

Slony-I es una herramienta que implementa una replicación maestro-esclavo. Se usa para la replicación en cascada, por ejemplo un nodo puede alimentar otro nodo que alimenta a otro nodo, también presenta tolerancia a fallos. [21]

Es un sistema diseñado para su uso en centros de datos y para hacer copias de seguridad. Slony es el plural de elefante en el idioma ruso.

Permite indicar que tablas se van a replicar de un servidor a otro, implementa la réplica sincrónica, usando disparadores para determinar las actualizaciones de las tablas.

Actualiza los datos exactamente igual al origen de los mismos, presenta la desventaja que no se pueden actualizar los datos cuando están ocurriendo cambios en ellos.

Se propone como solución la utilización de Pgpool-II como herramienta de replicación debido a que se hace necesario un sistema asincrónico capaz de hacer replicas en tiempo de ejecución.

1.5 Metodologías de Desarrollo

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se aplica una metodología por medio, lo que se obtiene es clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos. Sin embargo, muchas veces no se toma en cuenta el utilizar una metodología adecuada, sobre todo cuando se trata de proyectos pequeños de dos o tres meses. Cuando los proyectos que se van a desarrollar son de mayor envergadura, sí toma sentido basarse en una metodología de desarrollo, y se comienza a buscar la que más se adecúe al proceso que se quiera establecer. La mayoría de las veces no se encuentra la más adecuada y se termina por adaptar una metodología existente o hacer una propia, lo que para nada es incorrecto, siempre y cuando se cumpla con el objetivo.

Las metodologías de desarrollo de software se han categorizado en dos grandes grupos: las ágiles y las pesadas. Entre las metodologías ágiles están XP (Extreme Programming), FDD (Feature Driven Development), DSDM (Dynamic Systems Development Method), AUP (Agile Unified Process), Scrum, Crystal, Adaptive Software Development y otras. Así mismo se puede mencionar RUP (Rational Unified Process) como metodología pesada. [2] [3]

1.5.1 RUP

El Proceso Unificado Racional (Rational Unified Process), habitualmente resumido como RUP, es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

RUP es un proceso de realización o de evolución de software enteramente basado en UML. Está constituido por un conjunto de directivas que permiten producir software a partir del pliego de condiciones (requisitos). Cada directiva define quién hace qué y en qué momento. Un proceso permite, por tanto, estructurar las diferentes etapas de un proyecto informático. Los principios de RUP proceden del Proceso Unificado. [9]

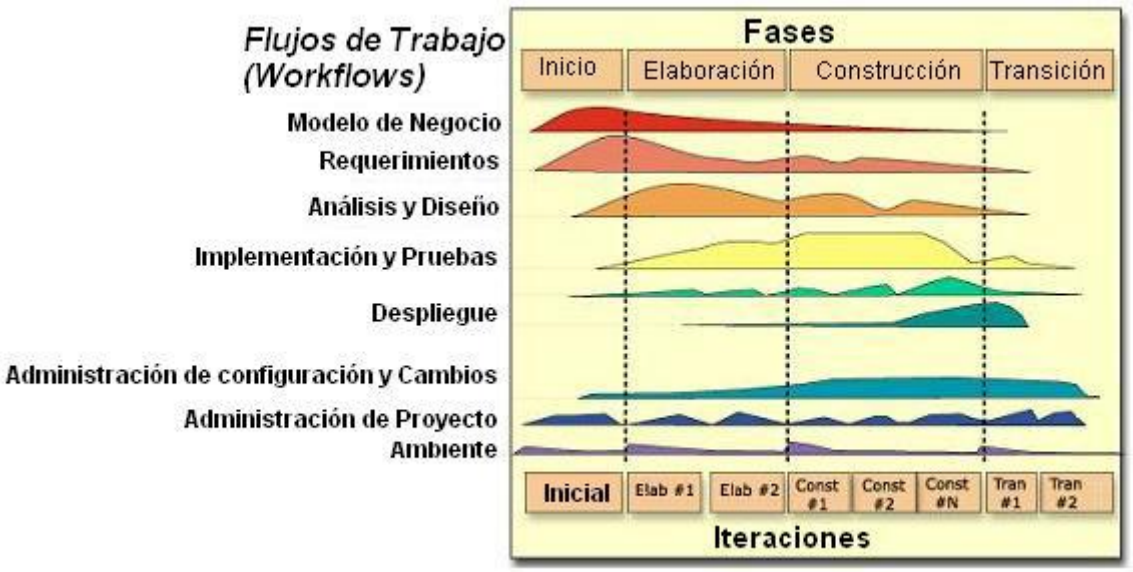
RUP puede ampliarse para adaptarlo a necesidades específicas. El Proceso Unificado de Desarrollo se caracteriza por:

- Ser conducido por los casos de uso.
- Ser iterativo e incremental, los procesos se dividen en una serie de sub proyectos y todos los sub proyectos se efectúan con las mismas actividades.
- Centrado en la Arquitectura.

El ciclo de vida RUP es una implementación del desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones. RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al culminar cada una de ellos, estos a la vez se dividen en fases que finalizan con un hito donde se debe tomar una decisión importante:

Concepción: Se hace un plan de fases, se identifican los principales casos de uso y los riesgos.
Elaboración: Se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.
Construcción: Se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.
Transición: Se Instala el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados. [10]

En RUP cada fase esta detallada por un conjunto de actividades (conjunto de acciones descrito por un diagrama de actividades) como se muestra en la siguiente figura.

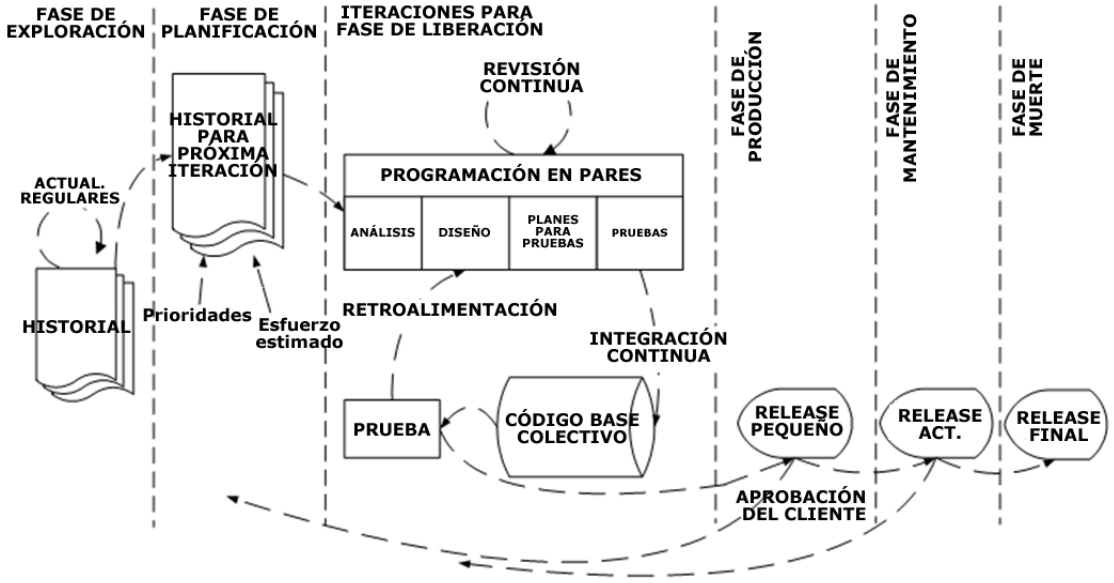


1.5.2 XP (Extreme Programming)

XP es una de las metodologías ágiles para el desarrollo de software más exitosas de la actualidad. Se utiliza en proyectos con pequeños equipos de desarrollo y con corto plazo de entrega. Se basa en la retroalimentación entre el cliente y el equipo de desarrollo, buena comunicación entre todos los participantes y simplicidad en las soluciones implementadas. Consiste en una programación rápida, cuya particularidad es que tiene como miembro del equipo al usuario final. Es adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. [4]

En XP se sigue la idea de la programación en pares, dado las ventajas que ofrece respecto a la creación del código, pues se pueden evitar errores y malos diseños al controlar cada línea de código y decisión de diseño instantáneamente. La interacción entre ambos desarrolladores puede generar discusiones que lleven a mejores estructuras y algoritmos, aumentando la calidad del software. [5] [6]

El ciclo de vida ideal de XP está compuesto por seis fases: Exploración, Planificación de la entrega, Iteraciones, Producción, Mantenimiento y Muerte del proyecto. [7] [8]



Luego de realizar un estudio de las metodologías más utilizadas, se llegó a la conclusión de que RUP es la más indicada para guiar el desarrollo de la herramienta que se desea implementar. RUP constituye uno de los estándares internacionales que más aceptación ha tenido. Además varias herramientas CASE soportan dicha metodología, permitiendo el trabajo en equipo y con la capacidad de generar código en distintos lenguajes de programación a partir de un diseño UML.

A parte de lo anteriormente planteado, RUP es una metodología que se encarga de: [22]

- Asegurar la producción de un software de alta calidad que reúna las necesidades de los usuarios finales dentro de un plan y un presupuesto predecible.
- Proveer un enfoque disciplinado para asignar tareas y responsabilidades dentro del desarrollo del sistema.
- Proveer un camino metódico, sistemático para desarrollar, diseñar y validar una arquitectura.
- Reducir en gran medida el riesgo que representa la construcción de sistemas complejos, porque evoluciona de forma incremental partiendo de sistemas más pequeños.

1.6 Paradigmas de Programación

Un paradigma de programación es un modelo básico de diseño y desarrollo de programas, que permite producir programas con unas directrices específicas, tales como: estructura modular, fuerte cohesión, alta rentabilidad, etc. [11]

Un paradigma de programación representa un enfoque particular o filosofía para la construcción del software. No es mejor uno que otro sino que cada uno tiene ventajas y desventajas. También hay situaciones donde un paradigma resulta más apropiado que otro. Si bien puede seleccionarse la forma pura de estos paradigmas al momento de programar, en la práctica es habitual que se mezclen, dando lugar a la programación multiparadigma.

1.6.1 POO (Programación Orientada a Objetos)

La Programación Orientada a Objetos (POO) es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo y encapsulamiento. Su uso se popularizó a principios de la década de 1990. Actualmente son muchos los lenguajes de programación que soportan la orientación a objetos.

La POO es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.

Con la POO tenemos que aprender a pensar las cosas de una manera distinta, para escribir nuestros programas en términos de objetos, propiedades, métodos y otras cosas que veremos rápidamente para aclarar conceptos y dar una pequeña base que permita soltarnos un poco con este tipo de programación.[12]

El elemento fundamental de la POO es, como su nombre lo indica, el objeto. Podemos definir un objeto como un conjunto complejo de datos y programas que poseen estructura y forman parte de una organización. [13]

Los objetos son entidades que combinan estado, comportamiento e identidad:

- El estado está compuesto de datos, será uno o varios atributos a los que se habrán asignado unos valores concretos (datos).
- El comportamiento está definido por los procedimientos o métodos con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él.

- La identidad es una propiedad de un objeto que lo diferencia del resto, dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).

La POO expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

De esta forma, un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados métodos que favorecen la comunicación entre ellos. La comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan ni deben separarse el estado y el comportamiento.

Los métodos (comportamiento) y atributos (estado) están estrechamente relacionados por la propiedad de conjunto. Esta propiedad destaca que una clase requiere de métodos para poder tratar los atributos con los que cuenta. El programador debe pensar indistintamente en ambos conceptos, sin separar ni darle mayor importancia a ninguno de ellos, hacerlo podría producir el hábito erróneo de crear clases contenedoras de información por un lado y clases con métodos que manejen a las primeras por el otro. De esta manera se estaría realizando una programación estructurada camuflada en un lenguaje de POO.

Esto difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada sólo se escriben funciones que procesan datos. Los programadores que emplean éste nuevo paradigma, en cambio, primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.

En comparación con un lenguaje imperativo, una "variable", no es más que un contenedor interno del atributo del objeto o de un estado interno, así como la "función" es un procedimiento interno del método del objeto.

Hay un cierto desacuerdo sobre exactamente qué características de un método de programación o lenguaje le definen como "orientado a objetos", pero hay un consenso general en que las características siguientes son las más importantes.

- Abstracción: Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos y cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción.
- Encapsulamiento: Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema. Algunos autores confunden este concepto con el principio de ocultación, principalmente porque se suelen emplear conjuntamente.
- Principio de ocultación: Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que específica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas, solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no pueden cambiar el estado interno de un objeto de maneras inesperadas, eliminando efectos secundarios e interacciones inesperadas. Algunos lenguajes relajan esto, permitiendo un acceso directo a los datos internos del objeto de una manera controlada y limitando el grado de abstracción. La aplicación entera se reduce a un agregado o rompecabezas de objetos.
- Polimorfismo: comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia

producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama asignación tardía o asignación dinámica. Algunos lenguajes proporcionan medios más estáticos (en "tiempo de compilación") de polimorfismo, tales como las plantillas y la sobrecarga de operadores de C++.

- Herencia: las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los objetos en clases y estas en árboles o enrejados que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase se dice que hay herencia múltiple.

1.6.2 POA (Programación Orientada a Aspectos)

La POA es un nuevo paradigma de programación, creado por Gregor Kiczales. Entre los objetivos que persigue se encuentran encapsular, capturar, aquellos conceptos que se escapan a los métodos actuales de descomposición de sistemas. Estos conceptos reciben el nombre de aspectos. [14]

El principal objetivo de la POA es la separación de las funcionalidades dentro del sistema:

- Por un lado funcionalidades comunes utilizadas a lo largo de la aplicación.
- Por otro lado, las funcionalidades propias de cada módulo.

Cada funcionalidad común se encapsulará en una entidad.

Muchas veces nos encontramos, a la hora de programar, con problemas que no podemos resolver de una manera adecuada con las técnicas habituales usadas en la programación imperativa o en la programación orientada a objetos. Con éstas, nos vemos forzados a tomar decisiones de diseño que repercuten de manera importante en el desarrollo de la aplicación y que nos alejan con frecuencia de otras posibilidades.

A menudo, hace falta escribir líneas de código que están distribuidas por toda o gran parte de la aplicación, para definir la lógica de cierta propiedad o comportamiento del

sistema, con las consecuentes dificultades de mantenimiento y desarrollo. En inglés este problema se conoce como scattered code, que podríamos traducir como código disperso. Otro problema que puede aparecer, es que un mismo módulo implemente múltiples comportamientos o aspectos del sistema de forma simultánea. En inglés este problema se conoce como tangled code, que podríamos traducir como código enmarañado. El hecho es que hay ciertas decisiones de diseño que son difíciles de capturar, debido a que determinados problemas no se pueden encapsular claramente de igual forma que los que habitualmente se resuelven con funciones u objetos.

Existen conceptos que no pueden encapsularse dentro de una unidad funcional, debido a que atraviesan todo el sistema o varias partes de él, como lo son la sincronización, el manejo de memoria, el manejo de errores, perfiles, seguridad o redes.

El desarrollo orientado a aspectos requiere de tres elementos básicos:

- Un lenguaje para definir la funcionalidad básica, conocido como lenguaje base o componente. Podría ser un lenguaje como C#, C++, Java o Lisp.
- Uno o varios lenguajes de aspectos, para especificar el comportamiento de los aspectos. Como podrían ser COOL para sincronización o RIDL para distribución.
- Un tejedor de aspectos (aspect weaver) que produce una aplicación que integra las funcionalidades de las clases y los aspectos. Tal proceso se puede llevar a cabo en tiempo de ejecución o en tiempo de compilación.

La POA define entonces una nueva forma de interacción, provista a través de los puntos de enlace (join points). Los puntos de enlace brindan la interfaz entre aspectos y componentes; son lugares dentro del código donde es posible agregar el comportamiento adicional especificado en los aspectos.

Un aspecto se compone de dos partes principales, el punto de corte (pointcut) y el consejo (advice code). Este último contiene el código por ser ejecutado, mientras que el punto de corte define un punto en el programa donde este código debe ser implementado. Como se puede ver las definiciones de punto de corte y punto de enlace están estrechamente relacionadas, no obstante los conceptos son algo diferentes. Los puntos de enlace están bien definidos en tiempo de ejecución de las entidades, mientras que los puntos de corte se definen por un conjunto de puntos de

enlace y no pueden atribuirse a una estructura particular o espacio de tiempo durante la ejecución del programa.

El consejo está asociado a un punto de corte a implementar. A diferencia de un método, el consejo nunca se llama directamente sino que esta tejido dentro de los puntos de enlace que están definidos en el punto de corte en particular.

Los lenguajes de aspectos de propósito general son diseñados para describir cualquier clase de aspecto, no solo específicos, por lo que no pueden imponer restricciones al lenguaje base. El nivel de abstracción del lenguaje base y del lenguaje de aspectos de propósito general es el mismo. No garantizan la separación de conceptos, esto es, que la unidad de aspecto se utilice únicamente para programar el aspecto, sin embargo, los de propósito general, proveen un ambiente más adecuado para el desarrollo de aspectos. Un ejemplo es AspectJ, donde Java es el lenguaje base, y las instrucciones de los aspectos también se escriben en Java.

En el caso que se desarrolla se decidió utilizar como paradigma la POO, entre otras cosas porque es el paradigma de programación más usado, los sistemas se modelan como un conjunto de objetos que interactúan entre sí, enfocándose en los conceptos comunes. Por esta razón se evitarían muchas líneas de códigos y se lograría un trabajo más ágil y menos costoso.

1.7 Lenguajes de Programación

Un lenguaje de programación es una construcción mental del ser humano para expresar programas. Está constituido por un grupo de reglas gramaticales, un grupo de símbolos utilizables, un grupo de términos monosémicos (es decir, con sentido único) y una regla principal que resume las demás.

Son herramientas que nos permiten crear programas y software de una forma más flexible y portable. Estos facilitan la tarea de programación, ya que disponen de formas adecuadas que permiten ser leídas y escritas por personas, a su vez resultan independientes del modelo de computador a utilizar.[15]

Los lenguajes de programación se dividen en 3 grandes grupos principalmente. Estos son: [16]

- **Lenguajes Máquina:** es el lenguaje de programación que entiende directamente la computadora o máquina. Este lenguaje de programación utiliza el alfabeto binario, es decir, el 0 y el 1.
- **Lenguajes de programación de bajo nivel:** Son mucho más fáciles de utilizar que el lenguaje máquina, pero dependen mucho de la máquina o computadora como sucedía con el lenguaje máquina.
- **Lenguajes de programación de alto nivel:** Este tipo de lenguajes de programación son independientes del ordenador, lo podemos usar en cualquier computador con muy pocas modificaciones o sin ellas, son muy similares al lenguaje humano, pero precisan de un programa interprete o compilador que traduzca este lenguaje de programación de alto nivel a uno de bajo nivel como el lenguaje de máquina que la computadora pueda entender.

Existen otras clasificaciones de los lenguajes de programación como son por nivel de abstracción, propósito, evolución histórica, manera de ejecutarse, paradigma de programación, concurrencia, interactividad, determinismo y productividad. [17]

1.7.1 Java

Java es un lenguaje de programación diseñado e implementado por Sun Microsystem en los años 90, pensado en un inicio para programar pequeños aparatos (electrodomésticos y artefactos electrónicos). [18]

Entre sus principales características tenemos que es: [19]

- **Orientado a Objetos:** Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red.
- **Distribuido**: Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.
- **Interpretado y compilado a la vez:** Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador.

Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier ordenador al cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time).

- **Indiferente a la arquitectura:** diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows Nt, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución tan variopintos, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñada para transportar el código eficientemente a múltiples plataformas hardware y software. El resto de problemas los soluciona el intérprete de Java, llamado Java Virtual Machine (Máquina Virtual de Java).
- **Portable:** La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas.
- **Multihebra:** Hoy en día ya se ven como terriblemente limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. Java soporta sincronización de múltiples hilos de ejecución (*multithreading*) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos.
- **Dinámico:** El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la Red.
- **Produce Applets:** puede ser usado para crear dos tipos de programas: aplicaciones independientes y applets. Las aplicaciones independientes se comportan como cualquier otro programa escrito en cualquier lenguaje, como por ejemplo el navegador de Web HotJava, escrito íntegramente en Java. Por su parte, las applets son pequeños programas que aparecen embebidos en las páginas Web, como aparecen los gráficos o el texto, pero con la capacidad de ejecutar acciones muy complejas, como animar imágenes, establecer

conexiones de red, presentar menús y cuadros de diálogo para luego emprender acciones, etc.

Para poder desarrollar programas en este lenguaje, se necesita, al menos, un entorno de desarrollo mínimo que es el JDK.

JDK es el acrónimo de “Java Development Kit” (Kit de Desarrollo de Java), que se puede definir como un conjunto de herramientas, utilidades, documentación y ejemplos para desarrollar aplicaciones Java. Incluye componentes esenciales como la librería de clases de JAVA, el compilador y la máquina virtual llamada “java”. [18]

El lenguaje fue seleccionado para el desarrollo de la solución, debido a la versatilidad y eficiencia de la tecnología, la portabilidad de su plataforma y la seguridad que aporta, la han convertido en la tecnología ideal para su aplicación a redes. De portátiles a centros de datos, de consolas de juegos a súper-equipos científicos, de teléfonos móviles a Internet; además ha sido probado, mejorado, ampliado y probado por una comunidad especializada de más de 6,5 millones de desarrolladores, la mayor y más activa del mundo. [20]

Otros aspectos tenidos en cuenta para su selección fueron:

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma
- Combinar aplicaciones o servicios que usan el lenguaje Java para crear servicios o aplicaciones totalmente personalizados
- La mayoría de las tecnologías Java han sido liberadas por Sun Microsystems bajo la licencia GNU GPL (Licencia Publica General de GNU), de modo que forma parte del software libre.

1.8 IDEs (Integrated Development Enviroment)

Un IDE (Integrated Development Environment) o ambiente de desarrollo, es un programa compuesto por un conjunto de herramientas para el programador. A continuación se abordarán características de fundamentales de dos de los IDE más utilizados en el mundo del software libre para la programación en Java: Eclipse y NetBeans.

Eclipse

Eclipse provee un conjunto de herramientas para administrar espacios de trabajo; construir, correr y depurar aplicaciones; compartir artefactos con un equipo y hacia

versión de código. Es una plataforma que está diseñada para ser infinitamente extendida, cada vez con herramientas más sofisticadas. Está construido sobre un mecanismo para el descubrimiento, integración y ejecución de módulos llamados plug-ins. Muchos plug-ins, comúnmente sin relación alguna, pueden ser instalados en una misma instancia de Eclipse, y convivir y cooperar sin problemas para ejecutar una cierta tarea. La clase de producto final incluye aplicaciones IDE, también denominados rich clients (clientes ricos), que se benefician del diseño de la plataforma de Eclipse y sus componentes. [23]

La plataforma Eclipse está habilitada para afrontar las siguientes necesidades:

- Soportar la construcción de gran variedad de herramientas de desarrollo.
- Soportar las herramientas proporcionadas por diferentes fabricantes de software independientes.
- Soportar herramientas que permitan manipular diferentes contenidos (HTML, Java, C, JSP, EJB, XML, y GIF).
- Facilitar una integración transparente entre todas las herramientas y tipos de contenidos sin tener en cuenta al proveedor.
- Proporcionar entornos de desarrollo gráfico (GUI) o no gráficos.
- Ejecutarse en una gran variedad de sistemas operativos, incluyendo Windows y Linux.

NetBeans

Es un Entorno Integrado de Desarrollo gratuito, de código abierto para desarrolladores de software. Consta de un conjunto de herramientas para crear aplicaciones profesionales tanto para escritorio como la empresa, la web y equipos móviles con el lenguaje Java, C/C++, y Ruby. Es fácil de instalar y de uso instantáneo, se ejecuta en varias plataformas incluyendo Windows, Linux, Mac OS X y Solaris. [24]

La construcción del software en NetBeans se realiza de forma modular y ofrece implementados los mecanismos de descubrimiento de nuevos módulos, resolución de dependencias, activación y desactivación de los módulos, comunicación entre los mismos, etc.; permitiendo que se haga hincapié en la lógica de las aplicaciones, dado la posibilidad que se pueda ir extendiendo su funcionalidad a medida que pasa el tiempo. [25]

Otras características son:

- Los proyectos desarrollados no dejan de ser multiplataforma.

- Sistema de ventanas práctico para desarrollar las interfaces de usuario.
- Sistema de ficheros virtual en el cual se montan los diferentes módulos con los cuales se van adaptando automáticamente los menús, barra de herramientas, menús contextuales, etc.
- Soporte completo para desarrollar desde NetBeans IDE.

NetBeans es el ambiente de desarrollo a utilizar. Aunque ambos, Eclipse y NetBeans, ofrecen similares recursos para el desarrollo de aplicaciones, pero para desarrollar aplicaciones de escritorio, como es el caso que se presenta, es NetBeans la mejor opción debido a que nos proporciona otras caracterizas muy útiles como gestor de ventanas, API de acciones, API para la creación de diálogos y wizards, integración con java help o generación de distribución Java Web Start entre otras muchas. Un framework más que interesante si queremos estar preparados para el desarrollo de aplicaciones de escritorio en java.

Conclusiones del capítulo

En el presente capítulo se realizó un estudio sobre las tecnologías clúster y sus características, así como los elementos necesarios para implementar un clúster para el Gestor de Bases de Datos PostgreSQL. Quedaron claros algunos conceptos de gran utilidad para el desarrollo de la solución tales como las metodologías de desarrollo de software, paradigmas y lenguajes de programación, ambientes de desarrollo, entre otros, por lo cual se decide utilizar como metodología RUP, POO como paradigma de programación, y como lenguaje de programación Java, utilizando como IDE NetBeans.

Capítulo 2. Solución Propuesta

2.1 Introducción.

En el presente capítulo se comienza a describir el proceso de negocio, se definen los requisitos funcionales y no funcionales, las clases de análisis y diseño, incluyendo los diagramas correspondientes, así como las realizaciones de cada uno de los casos de uso que contiene la herramienta.

2.2 Negocio.

2.2.1 Flujo Actual del Proceso.

El proceso comienza una vez que el arquitecto de software orienta la instalación del clúster del servidor de base de datos. El administrador de sistema define los servidores y servicios necesarios a instalar y configurar para que la aplicación funcione de la manera más eficiente. Una vez definido esto, el administrador del sistema 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 cumple con una de sus principales funcionalidades que en este caso es la replicación.

2.2.2 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.

2.2.2.1 Definición de actores del negocio.

Actor	Descripción
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.

2.2.2.2 Definición de trabajadores del negocio.

Trabajador	Descripción
Administrador del Sistema.	Determina la arquitectura de servidores necesaria para la instalación del clúster, instala y configura cada uno de los servicios necesario en cada uno de los nodos.

Tabla 2. Trabajadores del Negocio.

2.2.2.3 Diagrama de Casos de Uso de Negocio.

Figura 1. Diagrama de Casos de Uso de Negocio

2.2.2.4 Descripción textual del Caso de Uso de Negocio.

Caso de Uso de Negocio.	Solicitar instalación del sistema.
Actores.	Arquitecto de Software.
Propósito.	Este caso de uso tiene el objetivo de solicitar la instalación, configuración y restauración en los servidores.
Resumen.	El caso de uso se inicia cuando el Arquitecto de Software le solicita al Administrador del Sistema que instale el sistema. El Administrador del Sistema determina que 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 y realiza peticiones a la base de datos. Finalmente 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 del	1.1- El Administrador del Sistema determina que arquitectura es más eficiente para la aplicación desarrollada.

Sistema instale la aplicación.	1.2- El Administrador del Sistema instala cada uno de los servicios necesarios en los servidores. 1.3- El Administrador del Sistema configura óptimamente cada uno de estos servicios. 1.4- El Administrador del Sistema realiza peticiones a la base de datos.
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 no es óptimo o no funciona.	1.1- El Administrador del Sistema 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.

2.2.2.5 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.

El 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 mismo 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.

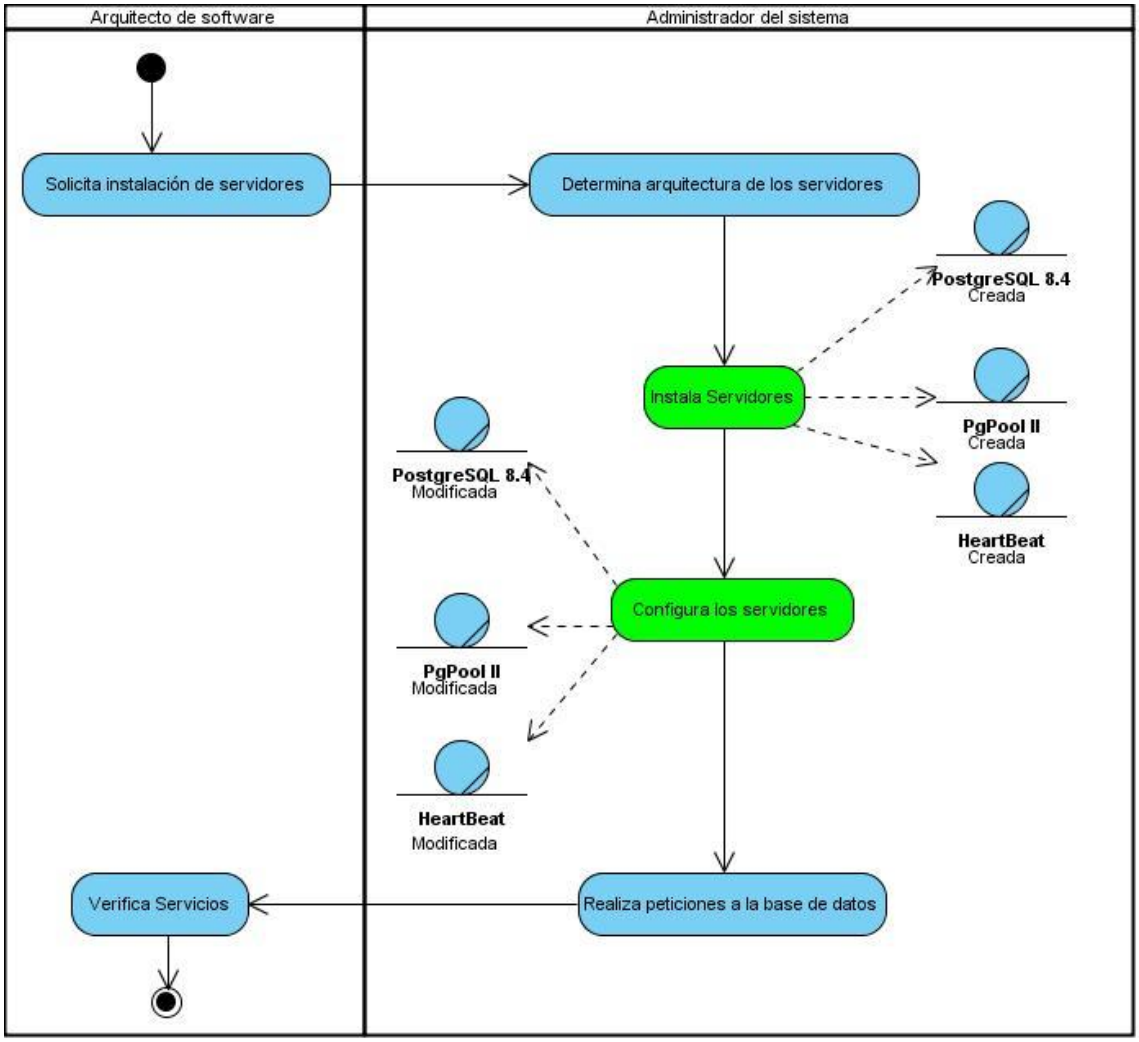


Figura 2. Diagrama Actividades Caso de Uso de Negocio Solicitar Instalación del Sistema.

2.2.2.6 Diagrama de Objetos.

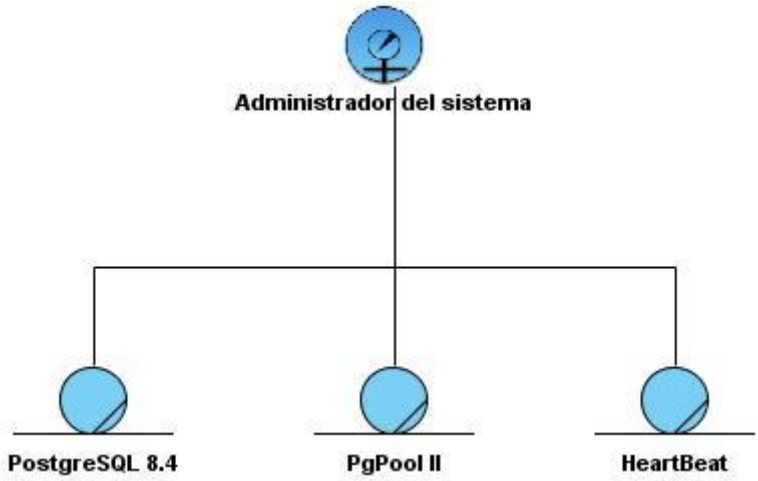


Figura 3. Diagrama Objetos Caso de Uso de Negocio Solicitar Instalación del Sistema.

2.3 Características de la Herramienta.

Debemos 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 y configuración de un clúster de servidores de bases de datos PostgreSQL.

Una vez creado o cargado un determinado perfil, la herramienta será capaz realizar una copia de cada uno de los script o ficheros. Instalar paso a paso, de manera automatizada, cada uno de los servicios necesarios. Luego se pasará al proceso de configuración de cada uno de estos servicios. Estas serían las funciones básicas que brindaría la herramienta.

2.4 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.

2.4.1 Definición de los Requisitos Funcionales.

1. Gestionar Perfil.
   1.1. Crear Perfil.
      1.1.1 Insertar Datos de los Servidores.
   1.2. Modificar Perfil.
   1.3. Eliminar Perfil.
2. Cargar Perfil.
3. Instalar Servicios.
   3.1. Copiar Scripts de Instalación a los Servidores.
   3.2. Ejecutar Scripts de Instalación en los Servidores.
4. Configurar Servicios.
   4.1. Copiar Scripts de Configuración en los Servidores.
   4.2. Ejecutar Scripts de Configuración en los Servidores.

2.4.2 Definición de los Requisitos no Funcionales.

Confidencialidad.

- La información manejada por la herramienta estará protegida.

Integridad.

- 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.

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

Portabilidad.

- La herramienta está construida para operar en sistemas operativos GNU Linux, realizado con un lenguaje libre.

Seguridad.

- Los perfiles serán guardados en ficheros binarios encriptados.

Apariencia o interfaz externa.

- Las ventanas de la herramienta tendrán claro y bien estructurados los datos, y al mismo tiempo permitirán la interpretación correcta de la información.
- 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.
- Mostrar todos los textos y mensajes en pantalla en español.
- Diseñar su funcionamiento de modo que sea intuitivo, y requiera de información mínima.

Ayuda y documentación.

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

Hardware.

- PC Administrador: Mínimo Pentium 3, 512 Mb de Ram, 2GB de espacio libre en disco duro.
- Nodos del Clúster: Mínimo Micro Dual Quad Core, 4GB de memoria RAM y 300GB de disco duro para cada uno.
- La comunicación de las terminales clientes con el servidor será a través de conexiones a una velocidad constante de 10/100 Mbps.

Software.

- Sistema Operativo Linux (distribuciones basadas en Debian).
- Máquina Virtual Java (JVM) 1.6 o superior.

2.5 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).

2.5.1 Definición de los actores del sistema.

Actor	Descripción
Administrador del Sistema.	Determina la arquitectura de servidores necesaria para la instalación del clúster, instala y configura cada uno de los servicios necesarios en cada uno de los nodos.

Tabla 4. Actores del Sistema.

2.5.2 Definición de los casos de usos principales del sistema.

Gestionar Perfil: Permite crear, modificar y eliminar perfiles.

Cargar Perfil: Permite cargar un perfil creado anteriormente.

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

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

2.5.3 Diagrama de Casos de Uso del Sistema.

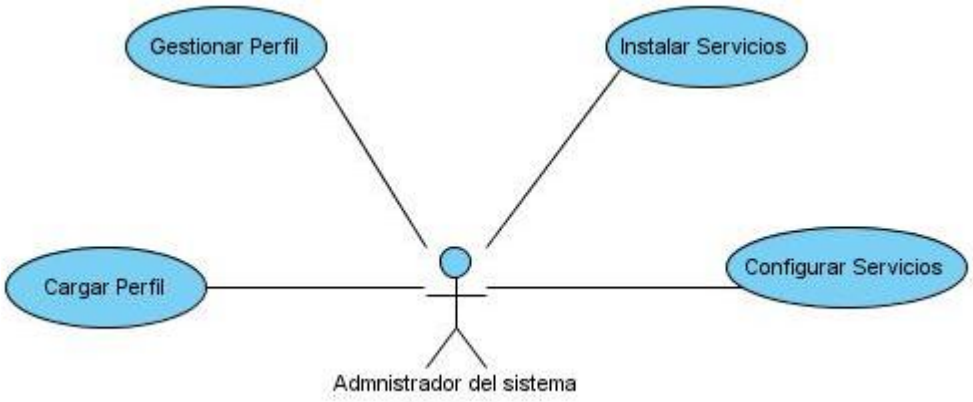


Figura 4. Diagrama de Casos de Uso de Sistema.

2.5.4 Especificación 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 especificación de casos de uso brinda los detalles de la secuencia de las acciones, las precondiciones como estado inicial, los posibles estados finales como postcondiciones, además de cuándo

comienza y termina el caso de uso. Describe explícitamente qué debe hacer el sistema, separando las responsabilidades del sistema y la de los actores.

A continuación se presenta una muestra de la descripción de los casos de uso más significativos con los que cuenta el sistema que se desarrolla, el resto de las descripciones de los casos de uso se pueden encontrar en el artefacto generado.

Caso de Uso:	Gestionar Perfil.
Actores.	Administrador del Sistema, (inicia).
Propósito.	Permite crear, modificar y eliminar un perfil.
Resumen.	El caso de uso se inicia cuando el Administrador del Sistema elige la opción de crear un perfil. La creación del perfil indica al usuario introducir el nombre del perfil y cada uno de los datos de los servidores.
Referencias.	RF 1, RF 1.1, RF 1.1.1, RF 1.2, RF 1.3
Precondiciones.	
Flujo Normal de Eventos.	
Sección "Crear Perfil"	
Acciones del Actor.	**Respuestas del sistema.**
1- El caso de uso inicia cuando el Administrador del Sistema selecciona la opción Archivo-Nuevo del menú.	2- El sistema indica la introducción del nombre del perfil.
3- El Administrador del Sistema introduce el nombre del perfil.	4- El sistema muestra el formulario de los datos correspondientes a llenar de cada uno de los servidores.
5- El Administrador del Sistema llena los campos: IP, Usuario y Tipo de servidor.	6- El sistema le permite adicionar otro servidor, aceptar para concluir o cancelar sino quiere guardar los 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 6 del flujo normal.
Postcondiciones de éxito.	Se creó el perfil.
Postcondiciones de fallo.	No se creó el perfil.
Prototipo de Interfaz	

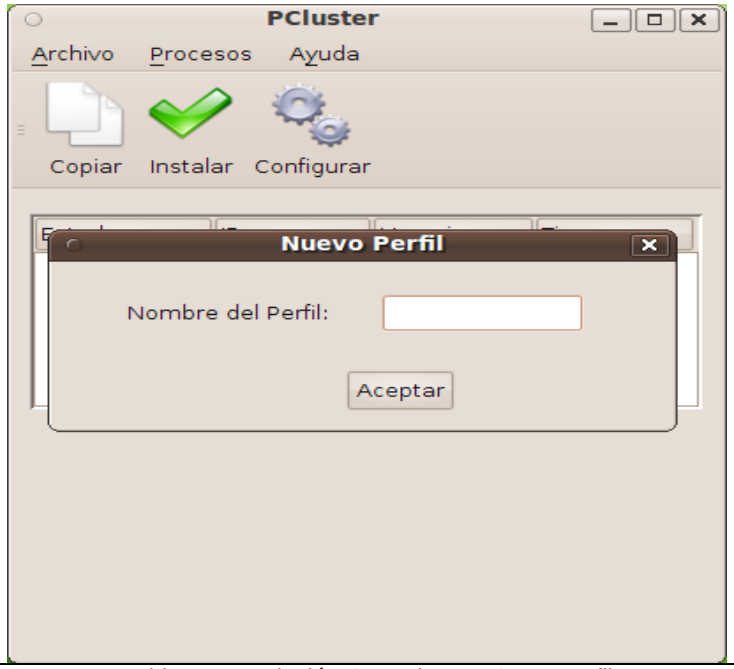


Tabla 5. Descripción Caso de Uso Crear Perfil

Caso de Uso:	Cargar Perfil.
Actores.	Administrador del Sistema, (inicia).
Propósito.	Permite cargar un perfil existente.
Resumen.	El caso de uso se inicia cuando el Administrador del Sistema elige la opción de cargar un perfil.
Referencias.	RF 2
Precondiciones.	
Flujo Normal de Eventos.	
Acciones del Actor.	**Respuestas del sistema.**
1- El caso de uso inicia cuando el Administrador del Sistema selecciona la opción Archivo-Abrir.	2- El sistema le muestra un cuadro de diálogo con los perfiles existentes.
3- El Administrador del Sistema 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.
Postcondiciones de éxito.	Se cargó el perfil.

Postcondiciones de fallo.	No se cargó el perfil.
Prototipo de Interfaz	

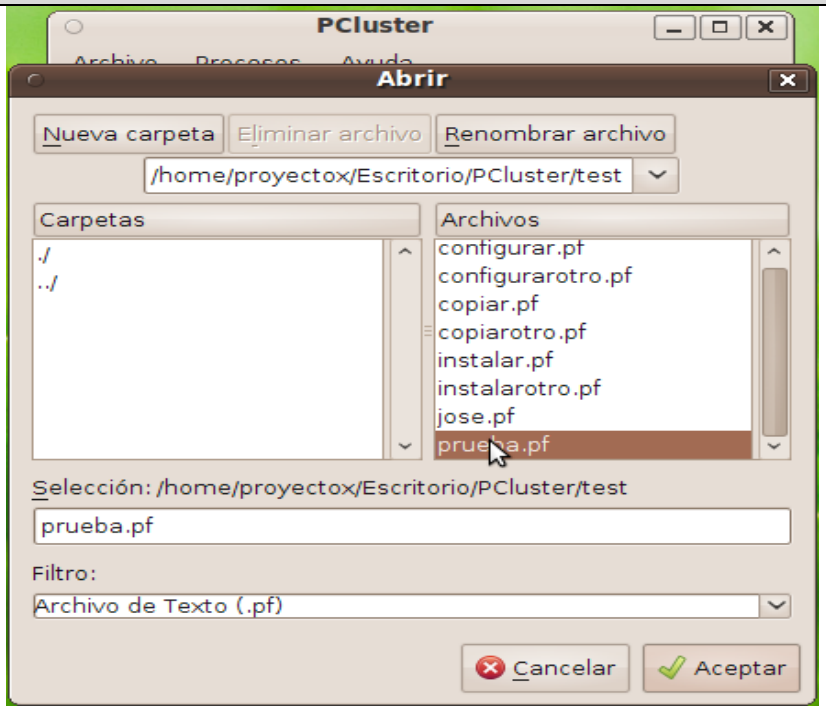


Tabla 6. Descripción Caso de Uso Cargar Perfil

Caso de Uso:	Instalar Servicios.
Actores.	Administrador del Sistema(inicia)
Propósito.	Instala los servicios en los servidores.
Resumen.	El caso de uso se inicia cuando el Administrador del Sistema elige la opción de instalar servicios. Automáticamente se instala 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.
Flujo Normal de Eventos.	
Acciones del Actor.	**Respuestas del sistema.**
1- El caso de uso inicia cuando el Administrador del Sistema selecciona la opción Instalar.	2- El sistema verifica la arquitectura del perfil.
3- El Administrador del Sistema copia los scripts para cada uno de los servidores seleccionando la opción Copiar al seleccionar un	4- El sistema copia los scripts hacia el servidor seleccionado.

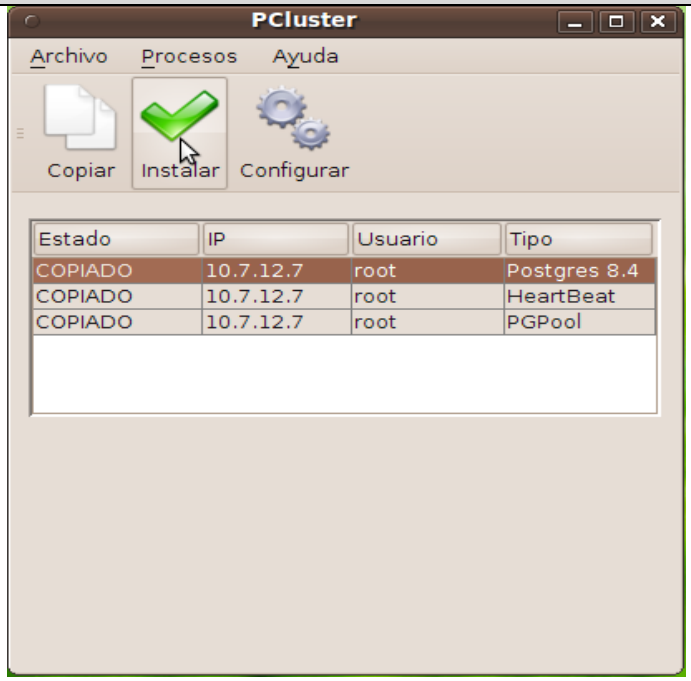
servidor.	
5- El Administrador del Sistema selecciona el servidor que desee instalar y luego selecciona la opción Instalar	6- El sistema ejecuta cada uno de los scripts correspondientes para el servidor seleccionado.
	7- Termina el caso de uso.
Flujo Alterno 1: "Problemas de instalación".	
	2- El sistema muestra un mensaje informando sobre los posibles errores que puedan haber ocasionado la falla de la instalación.
3- El Administrador del Sistema 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.
Postcondiciones de éxito.	Se instalaron los servicios.
Postcondiciones de fallo.	Problemas en la instalación de los servicios.
Prototipo de Interfaz	
	

Tabla 7. Descripción Caso de Uso Instalar Servicios

Caso de Uso:	Configurar Servicios.

Actores.	Administrador del Sistema(inicia)
Propósito.	Configura los servicios en los servidores.
Resumen.	El caso de uso se inicia cuando el Administrador del Sistema elige la opción de configurar servicios. Automáticamente se configuran cada uno de los servicios en cada uno de los servidores.
Referencias.	RF 4, RF 4.1, RF 4.2
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 del Sistema selecciona la opción Configurar.	2- El sistema verifica la arquitectura del perfil.
3- El Administrador del Sistema 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 del Sistema verifica la información que brinda el sistema y selecciona la opción Configurar para reintentar la configuración luego de verificar los problemas manualmente.	4- El sistema ejecuta el paso 4 del flujo normal.
Postcondiciones de éxito.	Se configuraron los servicios.
Postcondiciones de fallo.	Problemas en la configuración de los servicios.
Prototipo de Interfaz	

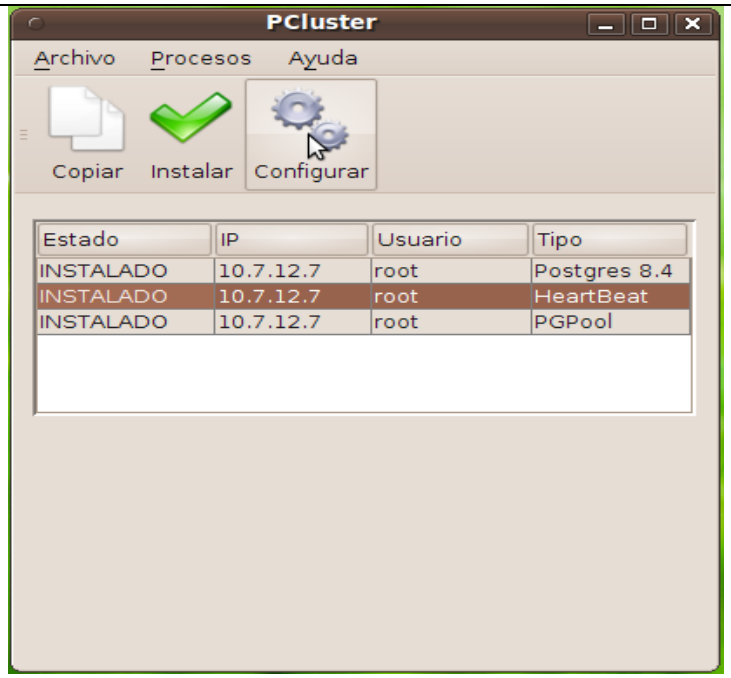


Tabla 8. Descripción Caso de Uso Configurar Servicios.

2.6 Análisis y Diseño.

El objetivo principal de esta disciplina es transformar los requerimientos a una especificación que describa cómo implementar el sistema. El análisis fundamentalmente consiste en obtener una visión del sistema que se preocupa de ver qué hace el sistema de software a desarrollar, 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, por otro lado, es un refinamiento del análisis que toma en cuenta los requisitos no funcionales, por tanto se centra en como el sistema cumple sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades. A través del mismo se modela el sistema para que soporte los requisitos, incluyendo los no funcionales y las restricciones que se le suponen.

2.6.1 Modelo de Clases de Análisis.

El modelo de clases del análisis, como su nombre indica, 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.

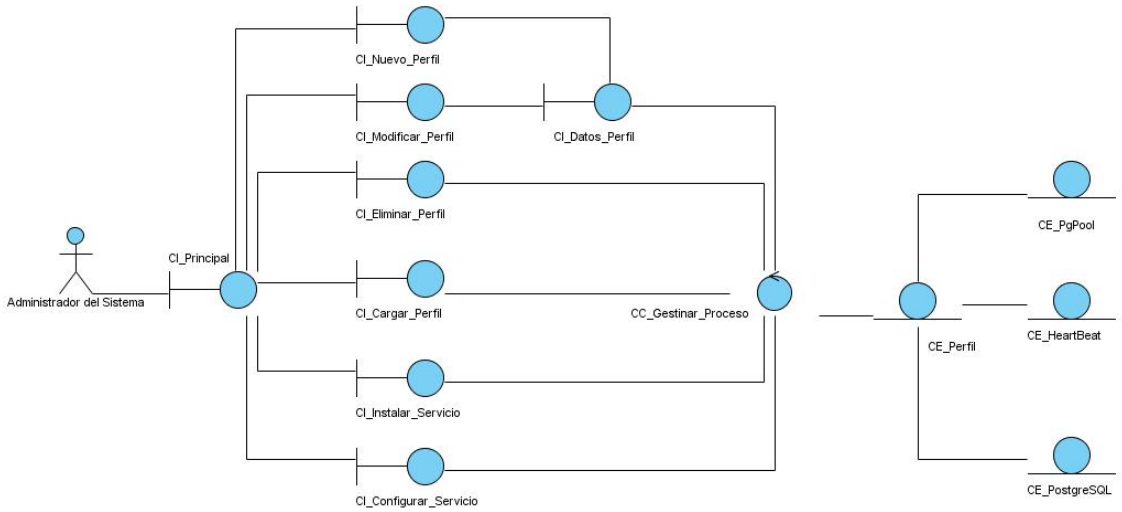


Figura 5. Diagrama de Clases de Análisis

2.6.2 Diagramas de Interacción.

Un diagrama de interacción no es más que un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre sí. 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.

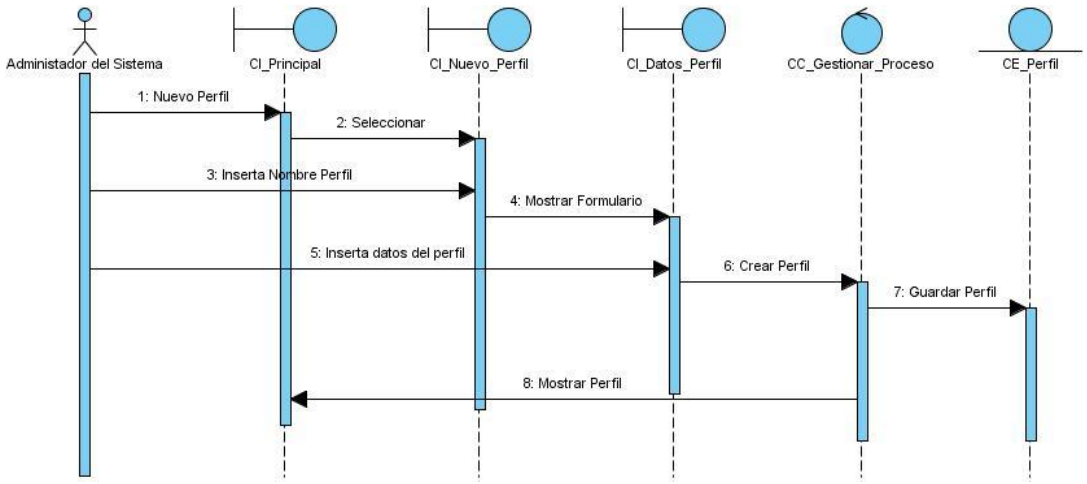


Figura 7. Diagrama de Secuencia Caso de Uso Gestionar Perfil (Crear Perfil).

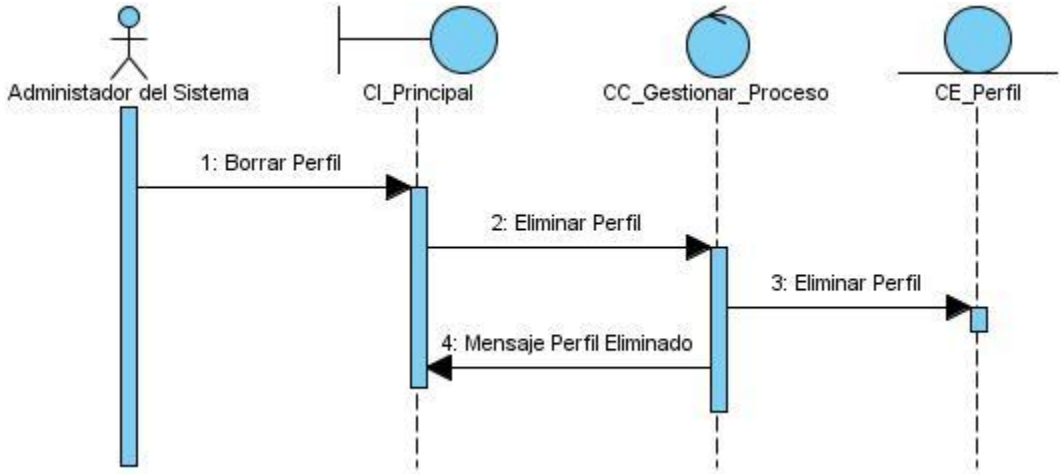


Figura 8. Diagrama de Secuencia Caso de Uso Gestionar Perfil (Eliminar Perfil).

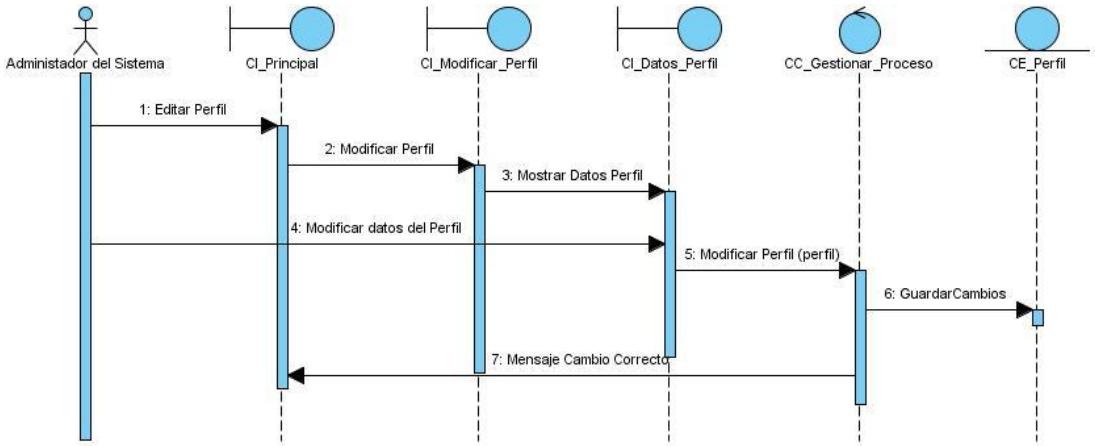


Figura 9. Diagrama de Secuencia Caso de Uso Gestionar Perfil (Modificar Perfil).

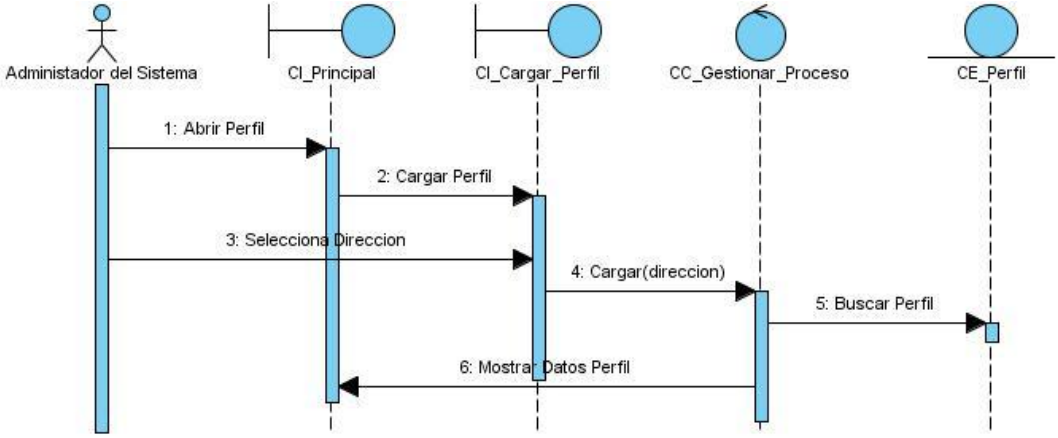


Figura 10. Diagrama de Secuencia Caso de Uso Cargar Perfil.

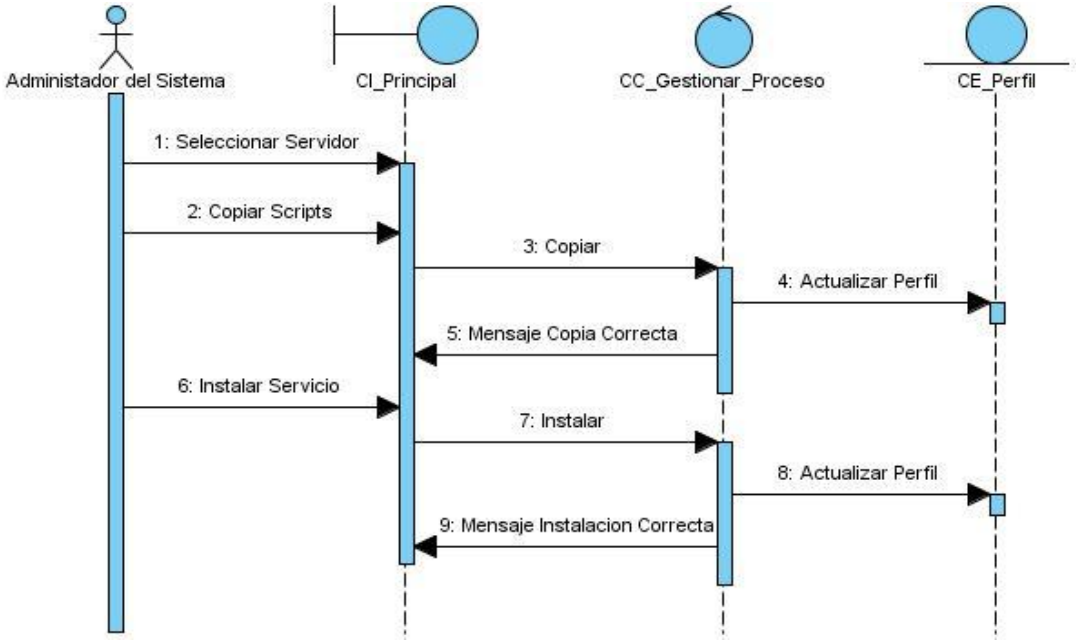


Figura 11. Diagrama de Secuencia Caso de Uso Instalar Servicios.

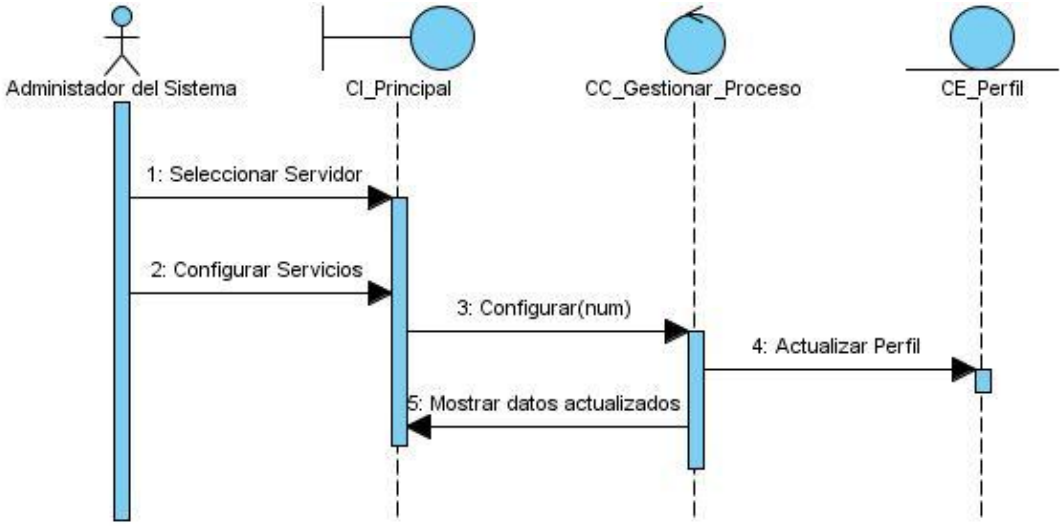


Figura 12. Diagrama de Secuencia Caso de Uso Configurar Servicios.

2.6.3 Patrones de Diseño.

Los Patrones de Diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño dentro de un contexto dado. Son soluciones basadas en la experiencia que permiten que los diseños sean mucho más flexibles, modulares y reutilizables.

En el diseño de la solución que se desarrolla se utilizaron los siguientes patrones:

- **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.

- **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.
- **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.
- **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.
- **Cadena de responsabilidad:** toma ventaja de las capacidades de Java que incluyen búsquedas y adiciones de atributos en tiempo de ejecución. De forma que las instancias de una misma clase no necesariamente requieren tener estructura idéntica y el acceso a los atributos puede ser monitoreado y atrapado. Estos mecanismos proveen una forma bastante elegante de implementar varias clases genéricas.
- **Singleton:** los patrones GoF y los de Java pueden beneficiarse mutuamente. Mientras que a Java le faltan algunas características que el GoF asume, no es imposible construir implementaciones de los patrones que funcionen como sus contrapartes del GoF. La naturaleza flexible y dinámica del lenguaje provee una buena base para una variedad de distintas y elegantes soluciones. Este patrón forma parte de los patrones GoF, específicamente al grupo de patrones creacionales.

En la solución 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.

2.6.4 Diagramas de Clases de Diseño.

Gráficamente, un diagrama de clases es una colección de nodos y arcos. Los mismos muestran un conjunto de clases, interfaces y colaboraciones, así como las relaciones entre sí. Las clases del diseño, y por ende 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 principalmente para modelar la vista de diseño estática de un sistema, 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.

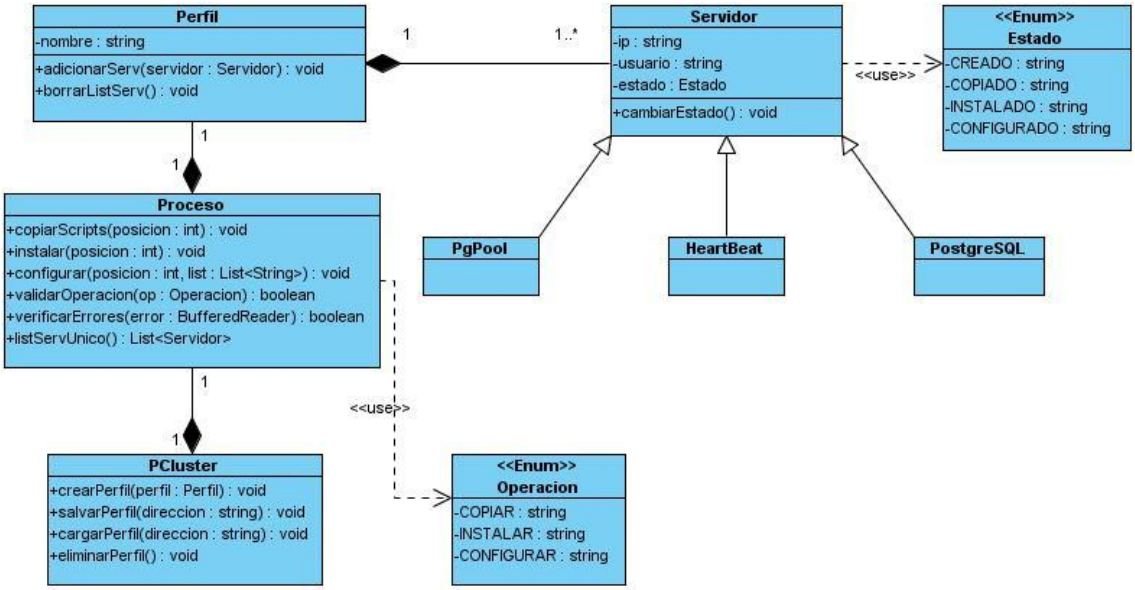


Figura 6. Diagrama de Clases del Diseño.

2.7 Conclusiones.

En este capítulo se hizo una descripción detallada del flujo actual de los procesos. Se realizaron los diagramas de casos de uso del negocio y los diagramas de actividades definiendo de esta forma el actor y el trabajador que intervienen en su ejecución. Se capturaron los requisitos funcionales y no funcionales necesarios para el desarrollo de la herramienta. Se identificó además el actor del sistema dando a su vez, una descripción del mismo. Se realizaron los diagramas de casos de uso y las especificaciones correspondientes, 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.

Capítulo 3. Implementación, Pruebas y Resultados

3.1 Introducción

En el presente capítulo se realizara una descripción, a través de los diagramas de componentes, de la implementación del software que se está llevando a cabo. Además se realizará el diagrama de despliegue, para mostrar cómo se realiza la distribución de los nodos necesarios para el despliegue de dicha aplicación. Y por último se realizarán las distintas pruebas al software para obtener una mayor seguridad del sistema y probar que todas sus funcionalidades se corresponden con los requisitos establecidos. De esta forma se garantizará la posterior validación de la herramienta en cuestión.

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. A continuación se presenta el diagrama de despliegue de la aplicación que se está implementando.

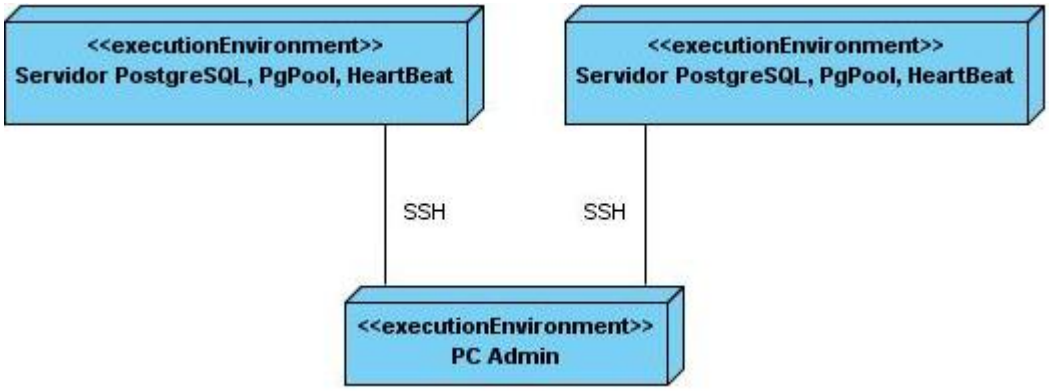


Figura 13. Diagrama de Despliegue.

3.3 Estándares de Codificación

Lo aquí descrito es una adaptación del ensayo original de Scott Hommel “Convenciones de Código para el lenguaje de programación Java.” Según el autor las convenciones de código son importantes pues mejoran la lectura del software, permitiendo entender código nuevo mucho más rápidamente y más a fondo. Además si se distribuye el código fuente como un producto, es necesario asegurarse de que esté bien hecho y presentado como cualquier otro producto.

En este epígrafe se reflejan algunos de los estándares de codificación del lenguaje java, presentados en “Java Language Specification”, de Sun Microsystems.

3.3.1 Extensiones de los ficheros.

Se utilizaron las siguientes extensiones para los ficheros:

Tipo de fichero y Extensión

Fuente Java .java

Bytecode de Java .class

3.3.2 Organización de los ficheros

3.3.2.1 Ficheros fuente Java

Cada fichero fuente Java contiene una única clase o interface pública. Algunas clases o interfaces privadas asociadas a una clase pública, se colocaron en el mismo fichero que la clase pública. La clase o interfaz pública es la primera clase o interface del fichero.

Los ficheros fuentes Java tienen la siguiente ordenación:

- Sentencias package e import.
- Declaraciones de clases e interfaces.

3.3.2.1.1Sentencias package e import

La primera línea no-comentario de los ficheros fuente Java es la sentencia package. Después de esta, siguen varias sentencias import. Por ejemplo:

package clases;

packages formularios;

import javax.swing.*;

import org.jdesktop.application.Action;

Nota: El primer componente del nombre de un paquete único se escribe siempre en minúsculas con caracteres ASCII.

3.3.2.1.2 Declaraciones de clases e interfaces

A continuación se describen las partes de la declaración de una clase o interface, en el orden en que aparecen.

Partes de la declaración de una clase o interface.

- Sentencia class o interface: class Servidor
- Comentario de implementación de la clase o interface si fuera necesario (/*...*/ o //…)
- Variables de clase (static)

- Variables de instancia
- Constructores
- Métodos: Estos métodos se agrupan por funcionalidad más que por visión o accesibilidad. El objetivo es hacer el código más legible y comprensible.

3.3.3 Indentación

Se emplearon cuatro espacios como unidad de indentación.

3.3.3.1 Longitud de la línea.

Se evitaron las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas.

3.3.4 Comentarios.

Los programas Java pueden tener dos tipos de comentarios: comentarios de implementación y comentarios de documentación. Los comentarios de implementación son aquellos que también se encuentran en C++, delimitados por /*...*/, y //. Los comentarios de documentación (conocidos como "doc comments") existen sólo en Java.

3.3.4.1 Comentarios de una línea.

Aparecen comentarios cortos de una única línea al nivel del código que siguen. Si un comentario no se pudo escribir en una línea, seguirá el formato de los comentarios de bloque. Un comentario de una sola línea es precedido de una línea en blanco como se muestra a continuación.

```
if (ValidarOperacion(Enum.Operacion.COPIAR)) {
        /* Ejecutamos el comando */
        ……
        Process p = Runtime.getRuntime().exec(comando);
        p.waitFor();
        /* ERRORES */
        BufferedReader stdError = new BufferedReader(new
InputStreamReader(p.getErrorStream()));
        VerificarErrores(stdError);
        getServidor().cambiarEstado(Enum.Estado.CREADO);
    }
```

3.3.4.2 Comentarios de fin de línea.

El delimitador de comentario // convierte en comentario una línea completa o una parte de una línea. No es usado para hacer comentarios de varias líneas consecutivas; sin embargo, se usa en líneas consecutivas para comentar secciones de código.

```
if (ValidarOperacion(Enum.Operacion.INSTALAR)) {
        int op;
        if (getServidor() instanceof PostgreSQL) {
           op = 1;
        } else if(getServidor() instanceof HeartBeat){
           op = 2;
        }else{
           op = 3;
        }
        Process p = Runtime.getRuntime().exec(comando);
        p.waitFor();       // Espera que termine el proceso
        getServidor().cambiarEstado(Enum.Estado.COPIADO);
    }
```

3.3.5 Declaraciones.

3.3.5.1 Cantidad por línea.

Se utilizó una declaración por línea, ya que facilita los comentarios como puede observarse en el ejemplo a continuación.

```
private String ip;
private String usuario;
private Enum.Estado estado;
```

3.3.5.2 Declaraciones de clases e interfaces.

Al codificar clases e interfaces de Java, se siguieron las siguientes reglas de formato:

- Ningún espacio en blanco entre el nombre de un método y el paréntesis "(" que abre su lista de parámetros.
- La llave de apertura "{" aparece al final de la misma línea de la sentencia declaración.
- La llave de cierre "}" empieza una nueva línea indentada para ajustarse a su sentencia de apertura correspondiente, excepto cuando no existen sentencias entre ambas, que aparece inmediatamente después de la de apertura "{"

```
public class Servidor implements Serializable {
    private String ip;
    private String usuario;
    private Enum.Estado estado;
    public Servidor(String ip, String usuario){
        this.ip = ip;
        this.usuario = usuario;
        this.estado = Enum.Estado.CREADO;
    }
    ……..
    public void CambiarEstado(Enum.Estado est){
        if(est == Enum.Estado.CREADO)
            this.setEstado(Enum.Estado.COPIADO);
        else if(est == Enum.Estado.COPIADO)
            this.setEstado(Enum.Estado.INSTALADO);
        else if(est == Enum.Estado.INSTALADO)
            this.setEstado(Enum.Estado.CONFIGURADO);
    }
}
```

- Los métodos se separan con una línea en blanco.

3.3.6 Sentencias.

3.3.6.1 Sentencias if, if-else, if else-if else.

La clase de sentencias if-else tiene la siguiente forma:

```
if(confpostgresql==null){
            JFrame mainFrame = Principal.getApplication().getMainFrame();
            confpostgresql = new FormConfPostgreSQL(mainFrame,true);
            confpostgresql.setLocationRelativeTo(mainFrame);
 }
if (Tabla.getSelectedRow() != -1) {
            ((DefaultTableModel)Tabla.getModel()).removeRow(Tabla.getSelectedRow());
 } else {
            JOptionPane.showMessageDialog( "Seleccione un servidor", "ERROR");
}
```

```
if (PCluster.getInstance().getPerfil().getListserv().get(i) instanceof PgPool){
        Tabla.getModel().setValueAt("PGPool", Tabla.getRowCount()-1, 3);
 } else if(PCluster.getInstance().getPerfil().getListserv().get(i) instanceof PostgreSQL) {
          Tabla.getModel().setValueAt("Postgres 8.4", Tabla.getRowCount()-1, 3);
 } else{
          Tabla.getModel().setValueAt("HeartBeat", Tabla.getRowCount()-1, 3);
 }
```

Nota: Las sentencias if usan siempre llaves {}.

3.3.6.1 Sentencias for.

Las sentencias for tienen la siguiente forma:

```
int aux =Tabla.getRowCount();
for (int i = 0; i < aux; i++) {
         ((DefaultTableModel)Tabla.getModel()).removeRow(0);
}
```

3.3.6.3 Sentencias while.

Las sentencias while tienen la siguiente forma:

```
int tiempo =0 ;
while (tiempo != 80) {
            tiempo = progressBar.getValue()+1;
            Thread.sleep(1000);
            progressBar.setValue(tiempo);
            progressBar.update(this.getGraphics());
}
```

3.3.6.4 Sentencias try-catch.

Las sentencias try-catch tienen la siguiente forma:

```
try {
        JFileChooser filechooser = new JFileChooser(System.getProperty("user.dir") +
"/test");
        FileFilter f = new FileNameExtensionFilter("Archivo de Texto (.pf)", "pf");
        filechooser.setFileFilter(f);
        int opcion = filechooser.showOpenDialog(null);
        if (opcion == JFileChooser.APPROVE_OPTION) {
                String dir = filechooser.getSelectedFile().toString();
                PCluster.getInstance().CargarPerfil(dir);
```

```
                actualizarVisual();
        }
}catch (Exception ex) {
        JOptionPane.showMessageDialog(null, ex.getMessage(), "ERROR",
JOptionPane.ERROR_MESSAGE);
}
```

3.3.7 Espacios en blanco

3.3.7.1 Líneas en blanco

Las líneas en blanco mejoran la facilidad de lectura, separando secciones de código que están lógicamente relacionadas.

Se usa siempre una línea en blanco en las siguientes circunstancias:

- Entre métodos.
- Entre las variables locales de un método y su primera sentencia.
- Antes de un comentario de bloque o de un comentario de una línea.

3.3.7.2 Espacios en blanco.

Se usan espacios en blanco en las siguientes circunstancias:

- Una palabra clave del lenguaje seguida por un paréntesis debe separarse por un espacio.

Ejemplo:

```
while (tiempo != 80) {
        .....
}
```

- Después de cada coma en las listas de argumentos.
- Las expresiones en una sentencia for se separan con espacios en blanco.

Ejemplo:

```
for (int i = 0; i < Table.getRowCount(); i++){
        …….
}
```

3.3.8 Convenciones de nombres.

Las convenciones de nombres hacen los programas más entendibles haciéndolos más fácil de leer. También dan información sobre la función de un identificador, por ejemplo, cuando es una constante, un paquete, o una clase, que puede ser útil para entender el código.

Paquetes: El prefijo del nombre de un paquete se escribe siempre con letras ASCII en minúsculas.

Clases: Los nombres de las clases son sustantivos, cuando son compuestos tienen la primera letra de cada palabra que lo forma en mayúsculas. Se mantienen los nombres de las clases simples y descriptivas. Se usan palabras completas y se evitan acrónimos y abreviaturas.

Métodos: Los métodos son verbos, cuando son compuestos tienen la primera letra en minúscula, y la primera letra de las siguientes palabras que lo forma en mayúscula.

Variables: Excepto las constantes, todas las instancias y variables de clase o método empiezan con minúscula. Las palabras internas que lo forman (si son compuestas) empiezan con su primera letra en mayúsculas. Los nombres de variables no empiezan con los caracteres sub-guión "_" o signo del dólar "$", aunque ambos están permitidos por el lenguaje.

3.3.9 Hábitos de programación.

3.3.9.1 Hábitos varios.

3.3.9.1.1 Paréntesis.

En general es una buena idea usar paréntesis en expresiones que implican distintos operadores para evitar problemas con el orden de precedencia de los mismos. Incluso si parece claro el orden de precedencia de los operadores, podría no ser así para otros, no se debe asumir que otros programadores conozcan el orden de precedencia.

3.3.9.1.2 Valores de retorno.

Se hace que la estructura del programa se ajuste a la intención. Ejemplo:

```
if (expresionBooleana) {
        return true;
} else {
        return false;
}
```

En su lugar se debe escribir:

```
return expressionBooleana;
```

[27]

3.4 Modelo de Pruebas.

Como a todo buen software, a la herramienta implementada se le hicieron varias pruebas para probar su funcionamiento y la disponibilidad de la misma, 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. A través de la prueba de caja blanca, se comprueba los caminos lógicos del software proponiendo casos de prueba donde 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.

JUnit es un conjunto de bibliotecas que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java. Es un conjunto de clases (framework) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

El propio framework incluye formas de ver los resultados que pueden ser en modo texto o gráfico (AWT o Swing).

En la actualidad las herramientas de desarrollo como NetBeans y Eclipse cuentan con plug-ins que permiten que la generación de las plantillas necesarias para la creación de las pruebas de una clase Java se realice de manera automática, facilitando al programador enfocarse en la prueba y el resultado esperado, y dejando a la herramienta la creación de las clases que permiten coordinar las pruebas.

Debido a todo esto, y a que se trata de la herramienta estándar de Java, además de que tiene mucha difusión y existe mucha documentación sobre esta librería en la red, es la que se ha elegido para realizar las pruebas unitarias de la herramienta desarrollada.

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 se introduce el resto de los datos necesarios así como la arquitectura que utilizará para el clúster: - PostgreSQL 8.4 - PGPool-II - HeartBeat. Por último, se muestra nuevamente el formulario inicial permitiendo la realización del resto de las funcionalidades.	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 9. Prueba de Caja Negra al Caso de Uso Crear Perfil

Entrada.	Resultados.	Condiciones.
Una vez que el administrador escoge la opción de Abrir en el menú de Archivo, la herramienta le muestra una ventana de diálogo 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 seleccionar un fichero que contiene la información del perfil.

Tabla 10. Prueba de Caja Negra al Caso de Uso Cargar Perfil.

Entrada	Resultados	Condiciones
El administrador escoge editar el perfil, la herramienta le muestra un formulario con los datos recogidos anteriormente en la creación del perfil. El administrador realiza las modificaciones necesarias,	La herramienta muestra en el formulario inicial, los cambios realizados.	El perfil se encuentre creado o cargado.

las cuales se hacen válidas una vez que se presiona la opción Modificar Por último se presiona la opción Aceptar.		

Tabla 11. Prueba de Caja Negra al Caso de Uso Modificar Perfil.

Entrada	Resultados	Condiciones
El administrador selecciona el servidor a instalar y presiona la opción con dicho nombre. Se muestra una ventana de instalación en dependencia del servidor escogido. El administrador sigue los pasos necesarios para la instalación.	La herramienta muestra un mensaje anunciando que la instalación se ha realizado con éxito.	El perfil se encuentre creado o cargado. Seleccionar un servidor. El servidor seleccionado para instalar se encuentre en estado Copiado.

Tabla 12. Prueba de Caja Negra al Caso de Uso Instalar Servicios.

Entrada	Resultados	Condiciones
El administrador selecciona el servidor y presiona la opción Configurar. Se muestra una ventana de configuración en dependencia del servidor escogido. El administrador sigue los pasos necesarios para la configuración.	La herramienta muestra un mensaje anunciando que la configuración se ha realizado con éxito.	El perfil se encuentre creado o cargado. Seleccionar un servidor. El servidor seleccionado para instalar se encuentre en estado Instalado.

Tabla 13. Prueba de Caja Negra al Caso de Uso Configurar Servicios.

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, o sea, es una forma de probar el correcto funcionamiento de un módulo de código.

A través de esta prueba se puede asegurar que cada uno de los módulos funcione correctamente por separado. Aunque no descubrirán todos los errores del código, debido a que sólo prueban las unidades por sí solas, 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 utilizó el framework JUnit, 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 JUnit es necesario hacer una “clase de test”, que se llama igual a la clase original pero añadiendo el sufijo “Test” al final, además se puede elegir sobre qué método específico se desea realizar dicha prueba o si es en toda la clase en general. Para realizar una prueba el test utiliza el método assertEquals para comprobar si el resultado de llamar al método coincide con el esperado. Otras posibles funciones son assertTrue, assertFalse, etc. Si este comando falla cuando se ejecuta el caso de prueba, JUnit dará el caso por fallido, de lo contrario el resultado es correcto.

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

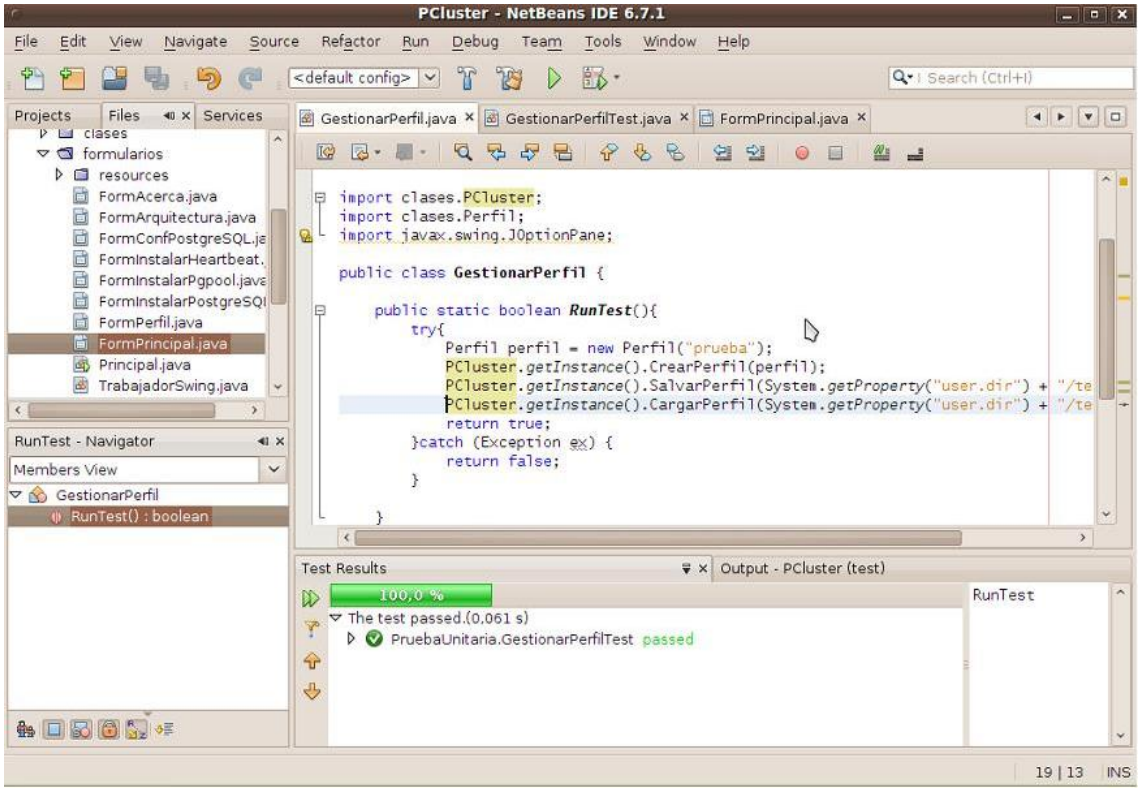


Figura 14. JUnit (Resultado de la prueba al método Gestionar Perfil)

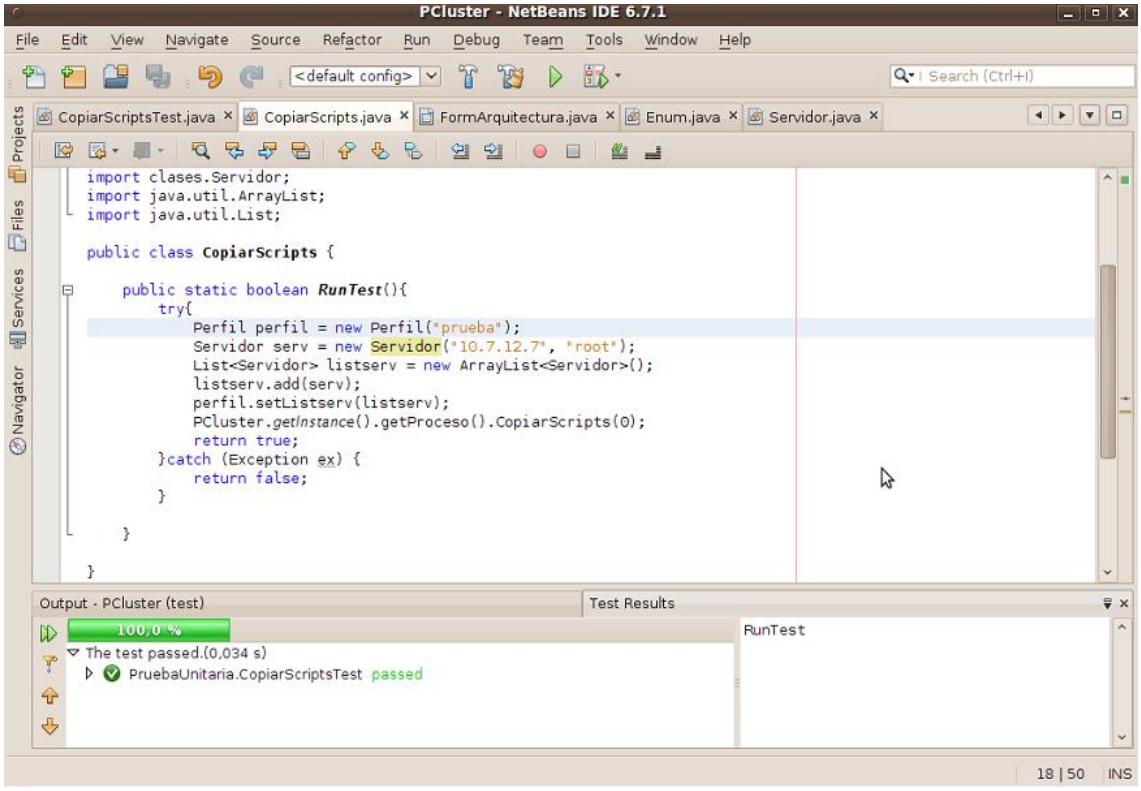


Figura 15. JUnit (Resultado de la prueba al método Copiar Perfil)

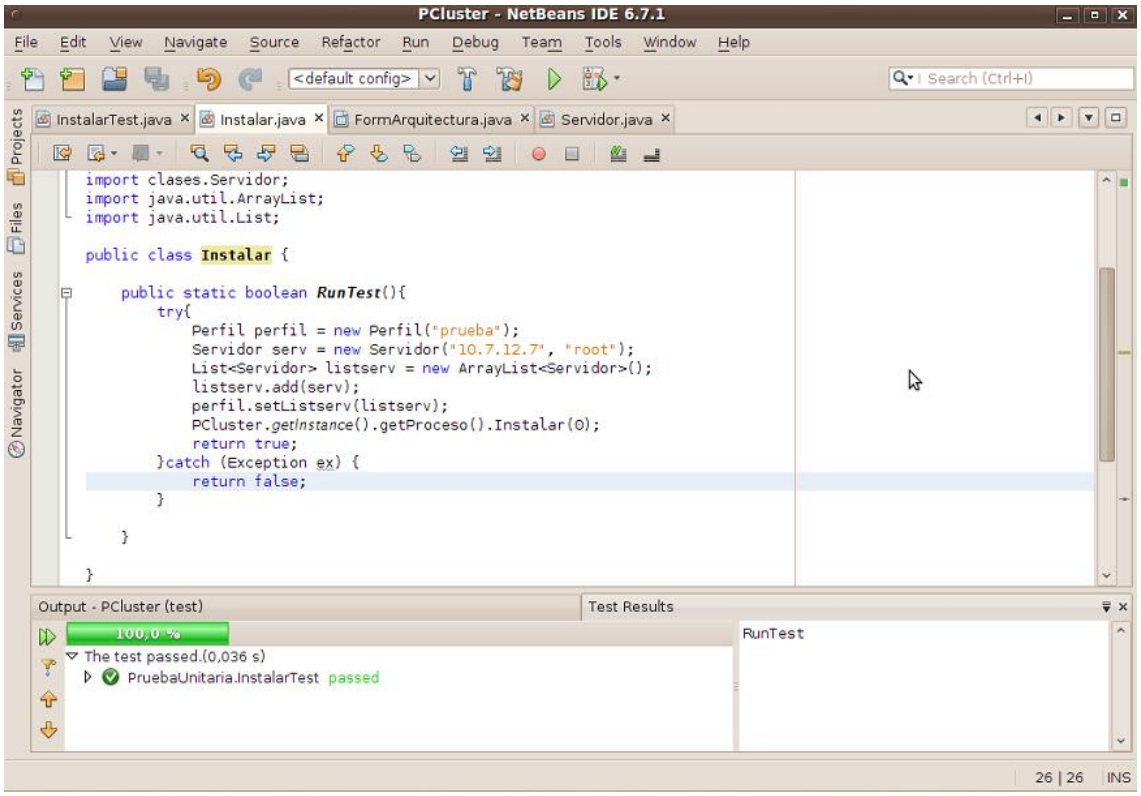


Figura 16. JUnit (Resultado de la prueba al método Instalar Perfil)

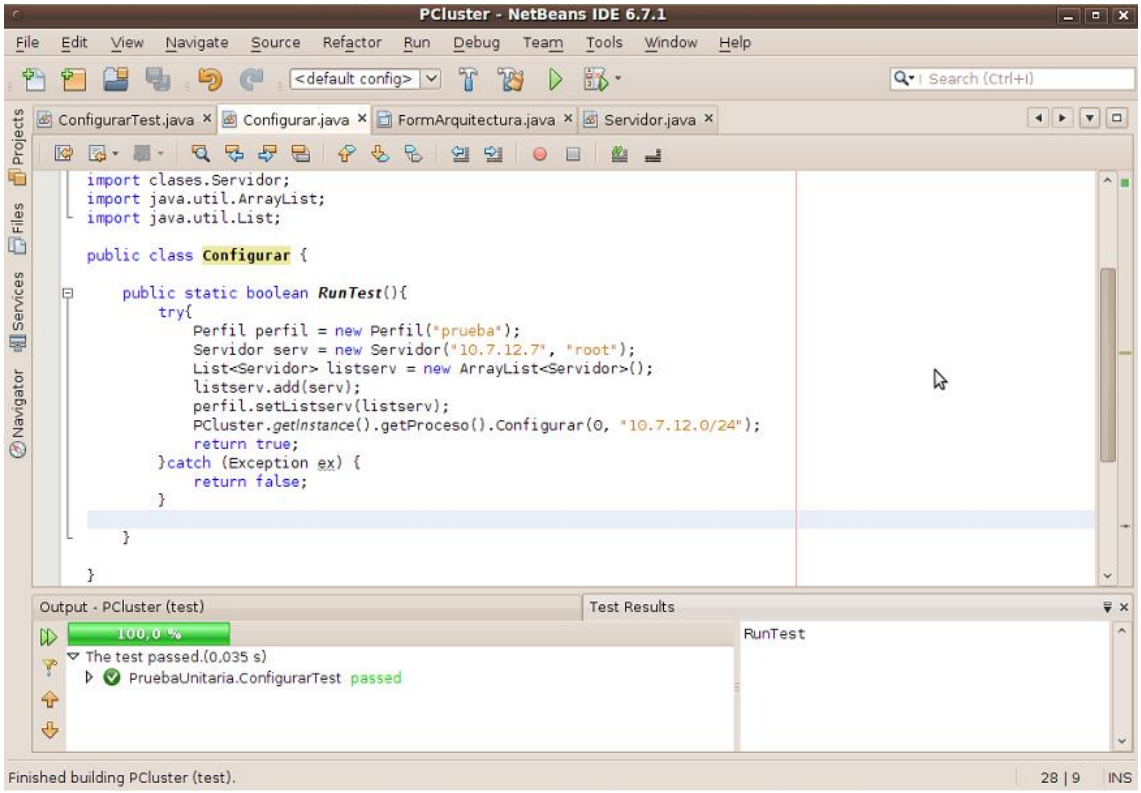


Figura 17. JUnit (Resultado de la prueba al método Configurar Perfil)

3.5 Conclusiones.

En este capítulo se obtuvo el diagrama de despliegue de la herramienta, mostrando las conexiones que permitirán enlazar los servidores a través de los protocolos que se utilizarán. Se analizaron los estilos de codificación utilizados, así como se hicieron las pruebas correspondientes al software implementado, tales como las pruebas de Caja Blanca y Caja Negra, mostrando los distintos resultados arrojados por cada una de las mismas.

Conclusiones Generales.

Con el desarrollo del presente trabajo se concluye afirmando que:

- La construcción de los artefactos del modelo del análisis y el de diseño, permitió obtener los elementos que debían ser implementados para obtener un sistema que cumpliera con los requerimientos especificados por el equipo de trabajo y aprobados por el cliente.
- El estudio de los diferentes métodos de implementación de clústeres permitió identificar que el cluster que se necesitaba implementar era precisamente uno de alta disponibilidad.
- El estudio de las principales herramientas de instalación y configuración de clústeres, permitió concluir que todas requerían de procesos de instalación y configuración engorrosos, por lo que se decidió implementar una herramienta capaz de hacer este proceso de una manera más amigable y específicamente para clústeres de PostgreSQL.
- Diseñar e implementar basados en la metodología RUP, permitió obtener una herramienta de escritorio capaz de instalar y configurar clústeres del gestor de bases de datos PostgreSQL, sin la necesidad de recurrir a la consola de cada uno de los elementos que componen el cluster.

Recomendaciones

- Continuar el desarrollo de la herramienta en una segunda fase para mejorar e incrementar las funcionalidades.

- Desarrollar esta herramienta para sistemas operativos propietarios como Microsoft Windows, ya que la actual versión es solo para sistemas Linux.

Bibliografía

1. **Clusters Sistemas Operativos.** [En línea] 2008. http://clusters.blogcindario.com/2008/10/00002-historia.html.
2. **Molpeceres, Alberto.** Procesos de desarrollo: RUP, XP y FDD. [En línea] 15 de Diciembre de 2002. http://www.javahispano.org/contenidos.downloadatt.action?id=71.
3. **Palacio, Juan.** Gestión y Modelos para la Eficiencia en Empresas de Desarrollo de Software. [En línea] http://www.baquia.com/marketing/Gestion_y_modelos_eficiencia_software.pdf.
4. **Aguilar, Catherine.** Aplicación de Conceptos de Gestión de Proyectos y Gestión de Riesgos en el Desarrollo de Productos Nuevos en el Campo de la Tecnología de Información. Universidad de Puerto Rico : s.n., Diciembre 2005.
5. **Jeffries, Ron, Anderson, Ann y Hendrickson, Chet.** Extreme Programming Installed. s.l. : Addison Wesley, Octubre 2000. ISBN 0-201-70842-6.
6. **Marches, Michele,** y otros. *Extreme Programming Perspectives.* s.l. : Addison Wesley, Agosto 2002. ISBN 0-201-77005-9.
7. **Beck, Kent.** *Extreme Programming Explained.* s.l. : Addison Wesley, Septiembre 1999. ISBN 0201616416.
8. **Smith, John.** A comparison of RUP and XP. [En línea] 2001. ftp://ftp.software.ibm.com/software/rational/web/whitepapers/2003/TP167.pdf.
9. **Jacobson, I., Booch, G y Rumbaugh, J.** El Proceso Unificado de Desarrollo de Software. , Vol. I. s.l. : Addison Wesley, 2000.
10. **Martínez, Alejandro y Martínez, Raúl.** Guía a Rational Unified Process – Universidad de Castilla la Mancha. 11. “Metodologías usadas en ingeniería del software - Paradigmas de programación y sus tipos”. [En línea] http://www.wikilearning.com/curso_gratis/metodologias_usadas_en_ingenieria_del_software-paradigmas_de_programacion_y_sus_tipos/3618-3
12. **Álvarez, Miguel Ángel.** “Qué es la programación orientada a objetos”. [En línea] http://www.desarrolloweb.com/articulos/499.php
13. **“Programación orientada a objetos. Sistemas de Procesamiento de Datos”.** [En línea] http://www.monografias.com/trabajos/objetos/objetos.shtml
14. **Asteasuain, Fernando y Contreras, Ezequiel, Bernardo.** “Programación Orientada a Aspectos”. [En línea] http://www.angelfire.com/ri2/aspectos/

15. **“Lenguajes de programación”.** [En línea] 2009 http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml
16. **“Clasificación de los lenguajes de programación”.** [En línea] 2006 http://www.larevistainformatica.com/clasificacion-de-los-lenguajes-de-programacion.html
17. **“Clasificación de lenguajes de programación”** [En línea] 2007 http://qbitacora.wordpress.com/2007/09/21/clasificacion-de-lenguajes-de-programacion/
18. **Ledesma, Claudia R. y Alí, Claudia A.** “Trabajo de Investigación: J2EE”
19. **Álvarez, Gonzalo.** “Características del lenguaje Java”. [En línea] 1997 http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html
20. **“Conozca más sobre la tecnología Java”** http://www.java.com/es/about/
21. **Browne, Christopher.** Slony-I, 1.2.13. Documentación. El grupo de desarrollo global de PostgreSQL. [En línea] 2004-2006. http://slony.info/.
22. **Kruchten, Philippe.** *The Rational Unified Process An Introduction, Second Edition.* s.l. : Addison Wesley, Marzo 2000. ISBN 0-201-70710-1.
23. **The Eclipse Foundation.** Eclipsepedia. *The Official Eclipse FAQs.* [En línea] 2009. http://wiki.eclipse.org/The_Official_Eclipse_FAQs.
24. **NetBeans IDE.** *Características.* [En línea] http://www.netbeans.org/features/index_es.html.
25. **Funes, Luis Enrique.** NetBeans Wiki. *Conociendo a NetBeans Platform: Introducción.* [En línea] 2008. http://wiki.netbeans.org/ConociendoNetbeansPlatformIntroduccion.
26. **Solingest, Hosting.** “Cluster de servidores, ¿qué es y cómo funciona?”. [En línea] 2007. http://hosting.solingest.com/cluster-de-servidores.html.
27. **Hommel, Scott.** “Convenciones de Código para el lenguaje de programación Java”. Sun Microsystems Inc.

Anexos

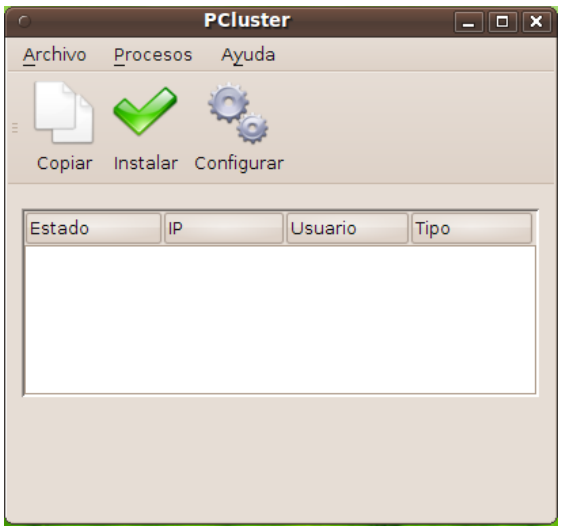


Figura 18. Ventana principal de la herramienta.

Figura 19. Ventana para añadir servidores a la arquitectura.

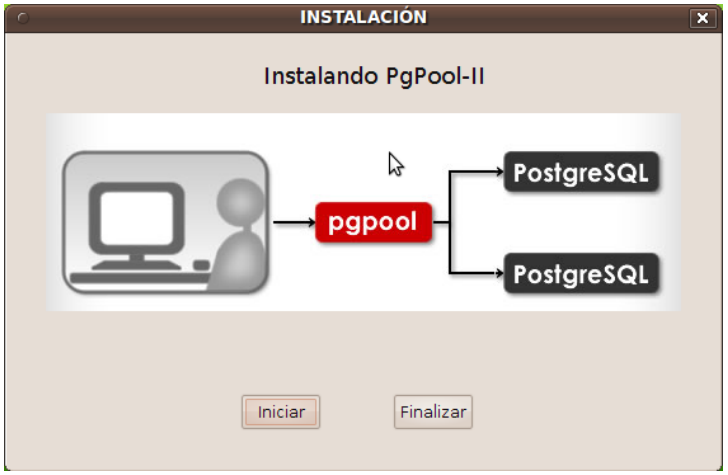


Figura 20. Ventana del proceso de instalación de pgpool-II.

UCI
Universidad de las Ciencias Informáticas

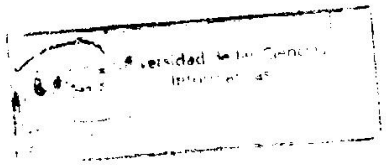
Acta de Liberación de Artefactos, Grupo de Calidad Centro CEGEL de la Facultad 15 de la Universidad de las Ciencias Informáticas.

Martes, 25 de mayo de 2010.

Luego de haber efectuado 2 iteraciones de revisiones a los artefactos: Especificación de Requisitos, Modelo de Sistema, Modelo de diseño, Modelo de Despliegue y Diseños de Casos de Prueba de la Herramienta para la Instalación y Configuración de clústeres del Gestor de Bases de Datos PostgreeSQL del proyecto SINAPSIS del Centro CEGEL de la Facultad 15 y haberse detectado un promedio de 15 No Conformidades, se puede afirmar que se han corregido los defectos encontrados, por lo que queda liberada la aplicación.

Firma del Asesor y Jefe del Grupo de Calidad Centro CEGEL

Ing. Raúl Velázquez Alvarez



Glosario de Términos

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

ARPANET: La red de computadoras ARPANET (Advanced Research Projects Agency Network) fue creada por encargo del Departamento de Defensa de los Estados Unidos como medio de comunicación para los diferentes organismos del país.

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 o activos.

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”.

Alpha: DEC Alpha es una arquitectura diseñada por DEC e introducida en 1992 bajo el nombre AXP, como reemplazo a la serie VAX.

Beowulf: Clúster 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.

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 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 es una base de datos distribuida y jerárquica que almacena información asociada a nombres de dominio en redes como Internet.

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, es el acrónimo de 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.

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.

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.

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.

Itanium: También conocido por su nombre en código Merced, fue el primer microprocesador de la arquitectura Intel Itanium que Intel lanzó al mercado. Aunque su lanzamiento inicialmente se planeó para 1998, no se produjo hasta mayo de 2001.

IBM: International Business Machines es una empresa que fabrica y comercializa herramientas, programas y servicios relacionados con la informática.

JVM: Una Máquina virtual es un programa nativo, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial, el cual es generado por el compilador del lenguaje Java.

Parallel Virtual Machine (PVM): La Máquina Virtual Paralela 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 con el fin de aprovechar esto para disminuir el tiempo de ejecución de un programa al distribuir la carga de trabajo en varias computadoras.

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

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 y Protocolo de Internet.

Xerox PARC: Centro de investigación de Palo Alto era una división de investigación de Xerox Corporation. Fue fundado en 1970 es famoso por crear esencialmente el paradigma moderno de interfaz gráfica de usuario (GUI) para ordenador personal.