Universidad de las Ciencias Informáticas

Facultad 1

Título: Diseño e implementación del módulo de réplica de datos para "GeForza".

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

Autores: Ramón Ernesto De Ávila León.

Lorián Rodríguez Barani.

Tutores: Ing. Alexander Rodríguez Mompié.

Ing. Carlos Tonet Groero Carmona.

Ciudad de La Habana, Junio 2010.

"Año del 52 de la Revolución"

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 _____.

_____

Ramón Ernesto de Ávila León

Autor

_____

Lorián Rodríguez Barani

Autor

_____

Ing. Carlos Tonet Groero Carmona

Tutor

_____

Ing. Alexander Rodríguez Mompié

Tutor

"Andar, es un modo de llegar".

José Martí

Ing. Carlos Tonet Groero Carmona:

Graduado de Ingeniería en Ciencias Informáticas, cursa el diplomado de Líder de Proyecto, opta por la Maestría de Gestión de Información de Organizaciones, en estos momentos es el líder del proyecto Fuerza de Trabajo Calificada, además de arquitecto de base de datos de ese mismo proyecto. Ha participado en diferentes eventos nacionales.

cgroero@uci.cu

Ing. Alexander Rodríguez Mompie:

Graduado de Ingeniería en Ciencias Informáticas, en estos momentos es el jefe del polo de Gestión Universitaria de la Facultad 1, ha participado en varios eventos nacionales.

arodriguezm@uci.cu

AGRADECIMIENTOS

Lorián Rodríguez Barani

A la Revolución por la maravillosa oportunidad que me dio de estudiar lo que me gusta y en una universidad de excelencia.

Al Comandante en Jefe Fidel por ser ejemplo de cubano, de revolucionario y de hombre.

A los héroes y heroínas de nuestro pueblo por dar su sangre y sus vidas para que personas como yo puedan ser ingenieros.

A los maravillosos profesores que he tenido en esta universidad que con buenos y malos ejemplos y clases me han trasmitido sus conocimientos, gracias a los cuales hoy me puedo graduar.

A mis tutores por estar ahí dispuestos en cada momento a dar su mano y me han apoyado en todo lo que está a su alcance.

A Víctor Luis Borroto, Lázaro Rubén García, Ronny de Informatización por ser de los pocos que estuvieron dispuestos cuando los necesitamos.

A Félix Iván Romero que acabado de llegar de Venezuela y con una Prueba de nivel estuvo al tanto del progreso de la tesis hasta el último momento.

A Marcos Luis Ortiz Valmaseda y Yoemir Orduñez Santana por dedicarnos su reducido tiempo, sin ellos esto no hubiese sido posible, mi familia y yo le estaremos agradecidos la vida entera por la ayuda que me han brindado.

A las chicas del 5309 y 5310 en especial a Yunia por acogerme y compartir conmigo durante todo este último año, espero que si algún día regreso me acojan con el mismo cariño que lo hicieron antes y que no pongan a más ninguno en mi lugar.

A Ramón Ernesto y todas sus amistades por compartir conmigo durante todo el año.

A mis amigos del Pre Wilder, Juan Denis, Yandy, Daylis y Lourdiana que luchamos tanto entre todos para llegar a ser profesionales viniendo de una ESPA y lo hemos logrado. A mi primer entrenador Ronald Grant Rodríguez por enseñarme la mayor parte de las cosas que aprendí en mi adolescencia y por ser mi segundo padre.

A mis amigos de primer año del grupo 1108 sin dudas el mejor de todos los tiempos, en el que están mis mejores amigos y los que me han acompañado en cada segundo de mi trayectoria universitaria.

A los amigos que hoy estuviesen graduándose junto conmigo y que no están, Rubén "El Pelú", Víctor Manuel Infante Leiro, Yilena Oviedo "La Riquitimami" a Anelkis Cobas López mi gran amiga.

A los maravillosos amigos y amigas que hice en esta universidad a mi hermano Iván por estar siempre dispuesto a todo en todo momento y sin miedo, a Marcel, Mario, Adrian, Los Locol Ramón y el Cuza, Raicel "El Racho", Reyter, mi otro gran amigo Elien "Elier". A Yisel y Darianna, por ayudarme tanto y ser mucho más que amigas durante 5 años con ellas siempre he podido contar para lo que sea, Katia Dolné, Vivi, Yanis, Yudi, Adis, Yalina, Yeny, Dayli, Dani y Arnelys por haber sido ante todo amigas incondicionales y saberse ganar mi respeto y admiración a lo largo de estos años.

A mi amiga virtual por estar pendiente de mí a cada momento a pesar de la distancia y los obstáculos.

A mis amigos de Matanzas y los de mi barrio que están aquí en la universidad, Abel, Yenismara, Guillermo, Alfredo, Esteban, Yailet, Dairon.

A toda la gente de la Facultad 1 la facultad más gloriosa que tiene y tendrá la UCI.

A mis tres suegros Miladys, Richard y Víctor por acogerme como un hijo más por abrirme las puertas de su casa, por su apoyo y por haber creado y criado a su maravillosa hija.

A mi hermana, mi sobrina, mis primos Elien y Dariel, mi tía y madre Layda Gloria por ser parte de la familia tan linda que tengo. A mi tío Héctor y mi abuelo Evangelio Barani que aunque no están físicamente siempre están conmigo, a los cuales quise y aún quiero.

A mi Nené por compartir conmigo durante 4 largos años por estar conmigo cuando la familia no está, por secar mis lágrimas, por ser mi bastón cuando estaba cojo, mi luz cuando todo estaba oscuro, por ser mi vida, mi alma, mi corazón, mi ángel, mi ser.

A mi hermano, mi doble, mi original, por enseñarme a golpear y a ser golpeado a levantarme cuando me caigo y sobre todas las cosas por enseñarme siempre a ganar y por ser la persona que me dio la noticia de que comenzaría mi vida en esta universidad aunque él no estaría conmigo.

A Dania por estar con nosotros desde pequeños por ser como una madre por querernos y tratarnos como si fuésemos sus hijos.

A mi padre por enseñarme a luchar, a trabajar por la familia a ver las cosas desde otro punto de vista, por dejarme hacer lo que quiero y como quiero, y aunque no es el típico padre que todos esperamos tener siempre ha estado al tanto de mi desempeño como persona y como estudiante.

A la persona más importante en mi vida. A mi madre que ha sido madre, padre, hermana, amiga, confidente, árbitro en las peleas con mis hermanos, porque nunca ha sido juez para juzgar, por luchar tanto con nosotros y por nosotros, por respetar mis decisiones y hacer que yo respete las de ella, por estar siempre alerta, por enseñarme a vivir, a respetar, a amar, por poner en mis manos el primer teclado de una computadora de esas de pantalla verde y disquetes de 5 y ¼ y por hacer tantos sacrificios desde hace 24 años para poderme ver aquí hoy graduado como Ingeniero en Ciencias Informáticas.

Ramón Ernesto de Ávila León

A mi compañero de tesis Lorián, que a lo largo de esta etapa hemos sabido echar para adelante contra nuestros mismos pronósticos.

A Marcos Luis Ortiz Valmaseda y Yoemir Ordúñez Santana por toda la ayuda que nos dieron, aún sacrificando su escaso tiempo, sin su participación esto hubiera sido muchísimo más difícil.

A mis tutores Carlos y Alexander, por estar ahí todo este tiempo, por ayudarnos en todo lo que pudieran, por guiarnos en este difícil proceso de terminar nuestra carrera.

A la profe Aneyty que aún sin ser mi tutora y sacrificando su tiempo me ayudó muchísimo.

A mis compañeros y amigos de mi grupo de primer año el 1106.

A mis amigos de la vocacional Yankiel y Javier por la amistad tan larga que tenemos.

A los amigos que hice aquí en la universidad Ismel, Frank, Juan, Carlos y Roly, por todos los buenos momentos que pasamos juntos sobre todo en estos últimos tiempos, por toda la ayuda que me dieron para la realización de mi tesis.

A mi hermano y verdadero amigo Stanco, por esta amistad que tenemos desde la primera fiesta que hicimos en 1er año, porque hemos estado juntos en los mejores y peores momentos que hemos vivido en esta universidad, por haberme aguantado estos 5 años, por enseñarme tantas cosas de la vida.

A mis amigas que vienen conmigo desde 1er año Luzbel, Katia e Irelys, si algún día fueran a escribir un libro sobre la amistad deberían dedicarnos al menos un capítulo pues hemos sabido mantener esta relación bajo

cualquier condición y circunstancia, a cada una de ellas mi mayor agradecimiento, a Katia por esa inteligencia que tiene, que con su tamaño no sé donde guarda tanta capacidad de deducción, a Irelys por esa ingenuidad típica de los pinareños que la hace una persona muy sincera, por ese cariño puro que me ha dado desde que nos conocemos, a Luzba por su incondicionalidad, su entrega, su afán de protección hacia cada uno de nosotros, por lo más lindo que posee, esa risa natural que siempre nos alegra el día.

A una de las personas más importantes y que más quiero en mi vida, a Yadira, que sobre todas las cosas siempre ha estado ahí para lo que me hiciera falta, nos conocimos el primer día de curso pero creo que ya teníamos una amistad en otra vida si es que eso es posible, le agradezco todo el apoyo y ayuda que me ha dado siempre y más con la realización de mi tesis, le deseo lo mejor del mundo porque se lo merece

A Julia y Víctor, por haberme acogido como un hijo más en su casa, por el cariño incondicional que me tienen.

A mi familia por darme siempre todo su apoyo y ayuda, por estar pendiente de mí en cada cosa que hacía, por todo lo que me han enseñado desde pequeño, a mis abuelos, mis tías y tíos, mis primos, en fin, a todos ellos gracias.

A mis hermanas por su preocupación, por todo su amor incondicional, por aguantarme todos estos años y los que faltan, por estar ahí siempre, las quiero mucho y a mis sobrinas preciosas que son mi orgullo, les agradezco por sacarme de los malos momentos viéndolas reírse y por alegrar a mi familia.

Le agradezco eternamente a mi madre, por haberme dado la posibilidad de vivir, por la impecable educación que siempre me dio, por todo el tiempo que me ha dedicado y que aún me dedica, por ser esa mujer fuerte que lucha contra todo por verme bien y feliz, por su preocupación cada vez que tenía algún problema, por ser el ejemplo de mujer que siempre buscaré, por demostrarme cada día que en esta vida no hay nada más importante que los hijos, por ser esa fuente inagotable de enseñanza, por tener ese carácter de hierro con manos suaves, por vivir para nosotros, por ser simplemente mi mamá.

Un agradecimiento muy especial a mi papá, que hace 4 años que no está con nosotros pues la vida decidió quitármelo cuando más falta me hacía, se que estaría muy orgulloso y emocionado porque hoy me he convertido en ingeniero, por darme todo ese amor incondicional de padre, por enseñarme que la vida es una sola y no debemos preocuparnos si en algún momento nos va mal o bien, que debemos solamente vivirla pues es muy poquita y no se sabe cuando se te puede acabar, por enseñarme a querer queriéndome, por ser un ejemplo, por dármelo todo en el momento que lo necesitaba, por regalarme además a mis dos hermanitas, no importa el lugar donde te encuentres, sé que estas aquí.

DEDICATORIA

Lorián Rodríguez Barani

A mi madre por ser la luz de mi vida, por enseñarme y amarme tanto y por ser mi guardián en todo momento.

A mi padre, mis hermanos, mis primos y mis tías por estar presentes en todo momento en el transcurso de mi carrera.

A mi novia por ser todo para mí durante 4 años.

A mis grandes amigos de ayer y de hoy.

Ramón Ernesto de Ávila León

A mis amigos de todos los tiempos, los que hice aquí y los que vinieron conmigo.

A mi familia por todo su apoyo y confianza.

A mi madre por ser un ejemplo de fortaleza, por ser la mejor mujer de este mundo, por darme lo que más necesita un ser humano su eterno amor y su confianza.

A mi padre por ser el guía de este sueño, mi mayor deseo es haberle dado el placer de verme hecho un ingeniero y sé que estuviera muy orgulloso de las cosas que he logrado en estos últimos años en que me ha faltado, donde quiera que esté debe saber que he estado pensando mucho en él.

RESUMEN

El Ministerio de Economía y Planificación es el órgano en Cuba encargado de aplicar y controlar la política de estado y de gobierno en materia de economía, planificación, metrología, estadística, normalización, entre otras. Dentro de este se encuentra el departamento de Fuerza de Trabajo Calificada cuya principal función es la realización de los planes de ingreso a la educación superior y ubicación laboral. Esta tarea se realiza con la participación de los Organismos de Administración Central del Estado, por lo que se necesita que estas entidades manejen la misma información.

En la actualidad esta información no se encuentra actualizada en su totalidad, ni es la misma en todas las entidades, por lo que se dificulta el proceso de planificación, lo que trae consigo problemas para el país, pues esta tarea se realiza con 5 años de antelación.

La presente investigación propone darle solución a lo antes planteado a través de la implementación de un módulo de réplica de datos para el Sistema Unificado de Gestión de Fuerza de Trabajo Calificada (GeForza), este funcionará en un entorno Maestro-Esclavo, además debe permitir la configuración de las fuentes de datos, monitorear el proceso en el momento en que ocurre y obtener reportes acerca de las réplicas hechas, Para ello se emplearán las herramientas más convenientes para la universidad, relacionadas con el almacenamiento de la información y las últimas tendencias en el mundo de la informática.

Palabras clave: entorno maestro-esclavo, réplica de datos, fuente de datos.

Índice de Contenido

Índice de Tablas

Índice de Figuras

Introducción

La información almacenada es uno de los bienes más preciados con que cuenta la humanidad y el desarrollo alcanzado en la actualidad por los sistemas de información, ha cambiado la forma de acceder, procesar, y guardar esta riqueza. Con la evolución de la informática se ha facilitado el proceso de encontrar los datos necesitados por los usuarios, que buscan una alta eficiencia y un gran número de servicios en las aplicaciones que sirven de puente entre estos y su objetivo final, el conocimiento almacenado. Esta es la razón fundamental, por la cual se debe tener en cuenta a la hora de construir un sistema de información, que las herramientas y las formas que se utilizan para su fabricación sean las más adecuadas y transparentes para las personas que van a interactuar con este.

Un aspecto importante es que la información no debe estar centralizada en un solo lugar pues puede provocar que en algún momento deje de prestarse el servicio, por lo que se hace necesario tenerla compartida en locaciones diferentes. Además lo más importante de esta distribución es que hace posible que el sistema sea más fiable, flexible, esté siempre disponible, brinde la posibilidad de compartir datos y por su puesto ofrezca un mejor rendimiento. Estos datos distribuidos que están unidos o conectados por una red de comunicación son los llamados Sistemas de Información Distribuidos y dentro de estos están los Sistemas de Bases de Datos Distribuidas.

El Ministerio de Economía y Planificación (MEP) es el organismo encargado de dirigir y controlar la aplicación de la política del Estado y el Gobierno de Cuba en materia de economía, planificación, estadística, normalización, metrología, control de la calidad, servicios comunales, planificación física y diseño industrial. Uno de los procesos más importantes que dirige es la elaboración de los planes anuales de ingreso a la educación superior y los planes anuales de ubicación laboral. Para la elaboración de estos planes se hace un trabajo en conjunto con todos los Organismos de Administración Central del Estado (OACE).

La información almacenada con la cual se realizan las planificaciones de los recursos no está completamente actualizada por lo que se hace compleja la realización de esta función debido a que depende en un gran porciento de la actualidad de esta. Una planificación realizada con una información desactualizada podría traer problemas para el país, pues este proceso se realiza con 5 años de antelación. Además, como se trabaja en conjunto para elaborar todos estos planes se necesita que la información con que cuenten todas las instituciones involucradas sea la misma y actualmente la información es trasmitida en dispositivos de almacenamientos para ser trasladada por un personal,

funciona igual tanto para la transmisión como para la recepción, lo que provoca un gasto en transporte, poca seguridad y pérdida de tiempo en este proceso, por lo que se plantea como ***problema a resolver***: ¿Cómo mantener la integridad, seguridad y la actualización de la información gestionada por el Sistema Unificado de Gestión de Fuerza de Trabajo Calificada "GeForza" utilizado en el departamento de Fuerza de Trabajo Calificada del Ministerio de Economía y Planificación y los Organismos de Administración Central del Estado?

El ***objeto de estudio*** es el proceso de replicación de datos y se limita al ***campo de acción*** de la réplica de datos del Sistema Unificado de Gestión de Fuerza de Trabajo Calificada.

El ***objetivo general*** de la investigación es: diseñar e implementar un módulo de réplica de datos para el Sistema Unificado de Gestión de Fuerza de Trabajo Calificada, a partir de este se desglosan los siguientes ***objetivos específicos*****:**

- Proponer un módulo de réplica de datos que cumpla con las necesidades del Sistema Unificado de Gestión de Fuerza de Trabajo Calificada.
- Implementar mediante herramientas libres un módulo de réplica para GeForza.

Para darle sustento a la investigación se plantea como ***hipótesis:*** con la implementación de un módulo de replicación de datos se mantendrá segura, íntegra y actualizada la información gestionada por el Sistema Unificado de Gestión de Fuerza de Trabajo Calificada.

Para cumplir con los objetivos planteados se proponen las siguientes ***tareas de investigación:***

- Realización de un estudio del estado del arte de sistemas de réplica de datos.
- Realización de un estudio de los requerimientos no funcionales del Sistema Unificado de Gestión de Fuerza de Trabajo Calificada.
- Realización de un estudio de la base de datos del Sistema Unificado de Gestión de Fuerza de Trabajo Calificada.
- Aplicación de técnicas de recopilación de información a los proveedores de requisitos.
- Ejecución del análisis y diseño del Módulo de Réplica de datos a partir de los requerimientos funcionales determinados.
- Implementación del módulo de réplica de datos.
- Evaluación de los resultados obtenidos en la utilización del módulo implementado a través de las pruebas.

A lo largo de la investigación se utilizaron varios ***métodos teóricos*** que ayudaron a guiar exitosamente el curso de la solución, algunos de los utilizados son el histórico-lógico utilizado en el análisis de la trayectoria y las etapas principales dentro de la investigación expresando de forma lógica el desarrollo y los conocimientos, el análisis-sintético para la recopilación de información a través de la relación entre lo general y lo particular que posibilita mejor organización de la solución planteada. Entre los ***métodos empíricos*** se utiliza la observación y la experimentación que actúan como guía metodológica para la investigación y medición de los resultados.

El ***resultado*** que se espera con la realización de la investigación es obtener un módulo de réplica de datos del Sistema Unificado de Gestión de Fuerza de Trabajo Calificada que permita la actualización de la información entre el Ministerio de Economía y Planificación y los Organismos de Administración Central del Estado de forma rápida y segura.

El contenido de la investigación está dividido en 4 capítulos organizados de la siguiente forma:

Capítulo 1: Fundamentación teórica: incluye un estudio del estado del arte de las herramientas de réplica de datos, las tendencias actuales y las técnicas y metodologías utilizadas para el desarrollo luego de haberse estudiado cómo se comportan estas en Cuba y el resto del mundo.

Capítulo 2: Características del sistema: se realiza el modelo de dominio analizando a través de este como se comporta el negocio actual y la situación a la cual se enfrenta el cliente. Se describen los procesos que dan lugar a la situación y los elementos que se van a automatizar especificándose los mismos en requisitos funcionales y no funcionales.

Capítulo 3: Análisis y Diseño: se exponen los diagramas de clases del análisis y diagrama de clases del diseño que ayudan a la construcción del software así como los diagramas de interacción que representan el comportamiento del sistema propuesto.

Capítulo 4: Implementación y Prueba: se analizan los elementos fundamentales que intervienen en la solución del sistema y la relación de los componentes técnicos que lo integran a través de los diagramas de componentes. Muestra el diagrama de despliegue y la validación del software según los métodos establecidos.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

Actualmente muchos sistemas en el mundo y en Cuba necesitan mantener íntegra y actualizada la información que se gestiona en diferentes lugares a la vez, para poder lograrlo una de las técnicas más usadas es la réplica de datos sobre las bases de datos distribuidas. En el presente capítulo se describen las técnicas, herramientas y procedimientos que se llevaron a cabo para cumplir el objetivo de la investigación. Se incluye además tendencias actuales y las técnicas usadas para el desarrollo de la aplicación.

1.2 Informatización de la sociedad cubana

La Informatización de la Sociedad es el proceso de utilización ordenada y masiva de las Tecnologías de la Información y las Comunicaciones en la vida cotidiana para satisfacer las necesidades de todas las esferas de la sociedad, y lograr cada vez con más eficiencia todos los procesos y por consiguiente mayor generación de riquezas y aumento en la calidad de vida de los ciudadanos.

Una sociedad que aplique la informatización en todas sus esferas y procesos será más eficaz y competitiva. Es evidente que para los países subdesarrollados resulta un reto el logro de este propósito, pues su problemática fundamental está en lograr la supervivencia de sus pueblos. Cuba ha identificado desde muy temprano la conveniencia y necesidad de dominar e introducir en la práctica social las Tecnologías de la Información y las Comunicaciones (TIC); y lograr una cultura digital como una de las características imprescindibles del hombre nuevo, lo que facilitaría a la sociedad acercarse más hacia el objetivo de un desarrollo sostenido.(1)

1.2.1 Ministerio de Economía y Planificación. Departamento de Fuerza de Trabajo Calificada

El Ministerio de Economía y Planificación (MEP) es el organismo encargado de dirigir, ejecutar y controlar la aplicación de la política del Estado y el Gobierno. Cuenta con un departamento que se encarga de administrar y gestionar la Fuerza de Trabajo Calificada (FTC) del país, para esto existen tres sistemas informáticos que tratan de agilizar esta tarea; resolviendo el problema, pero no de manera eficiente, además, presentan una serie de dificultades que los hacen una solución poco eficaz, para ser utilizado en tareas de tan alta importancia.

La dirección del departamento de FTC, tras los constantes inconvenientes en la realización de su trabajo, causados por los sistemas que utilizan, decidió, realizar un convenio de colaboración con la

Universidad de las Ciencias Informáticas (UCI) para realizar un nuevo sistema, el cual brinde una gama de servicios (definidos por el departamento de FTC) que faciliten y agilicen su trabajo.

El objetivo concreto de la investigación es realizar un estudio sobre los sistemas ya implantados en el Ministerio de Economía y Planificación (MEP) y proponer un modelo para la Gestión de Alcance del Proyecto "Fuerza de Trabajo Calificada" para desarrollar un nuevo sistema más eficiente y capaz de mejorar todo los procesos mencionados.(2)

1.3 Sistemas de gestión de base de datos

Los Sistemas de Gestión de Bases de Datos (SGBD) son un tipo de *software* muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, un lenguaje de manipulación de datos y un lenguaje de consulta. (3)

Los SGBD deben cumplir los siguientes objetivos:

- **Abstracción de la información:** los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos.

- **Independencia:** consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

- **Consistencia:** que todos los datos repetidos se actualicen de forma simultánea.

- **Seguridad:** los SGBD deben garantizar que la información se encuentre segura de permisos a usuarios y grupos de usuarios, permitiendo otorgar diversas categorías de permisos.

- **Manejo de transacciones:** los SGBD proveen mecanismos para programar las modificaciones de los datos de una forma mucho más simple que si no se dispusiera de ellos.

- **Tiempo de respuesta:** es deseable minimizar el tiempo que el SGBD tarda en devolver la información solicitada y en almacenar los cambios realizados.(4)

1.3.1 Bases de datos

Una base de datos es un conjunto no redundante de datos estructurados, organizados independientemente de su utilización y su implementación, pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.(5)

Características de los sistemas de bases de datos:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación.(6)

1.3.1.1 Bases de datos distribuidas

Una base de datos distribuida (BDD) es un conjunto de múltiples bases de datos lógicamente relacionadas que se encuentran distribuidas entre diferentes sitios interconectados por una red de comunicaciones, poseen además, la capacidad de procesamiento autónomo lo cual indica que puede realizar operaciones locales o distribuidas. En un Sistema de Bases de Datos Distribuida (SBDD) se encuentran múltiples sitios de bases de datos que están ligados por un sistema de comunicaciones de tal forma, que un usuario en cualquier sitio puede acceder a los datos en cualquier parte de la red exactamente como si los datos estuvieran siendo accedidos de forma local.(7)

Ventajas

Refleja una estructura organizacional: los fragmentos de la base de datos se ubican en los departamentos a los que tienen relación.

- **Autonomía local:** un departamento puede controlar los datos que le pertenecen.
- **Disponibilidad:** un fallo en una parte del sistema solo afectará a un fragmento, en lugar de a toda la base de datos.
- **Rendimiento:** los datos generalmente se ubican cerca del sitio con mayor demanda, también los sistemas trabajan en paralelo, lo cual permite balancear la carga en los servidores.
- **Economía:** es más barato crear una red de muchas computadoras pequeñas, que tener una sola computadora muy poderosa.

- **Modularidad:** se pueden modificar, agregar o quitar sistemas de la base de datos distribuida sin afectar a los demás sistemas (módulos).

Desventajas

- **Complejidad:** se debe asegurar que la base de datos sea transparente, se debe lidiar con varios sistemas diferentes que pueden presentar dificultades únicas. El diseño de la base de datos se tiene que trabajar tomando en cuenta su naturaleza distribuida.

- **Economía:** la complejidad y la infraestructura necesaria implica que se necesitará una mayor mano de obra.

- **Seguridad:** se debe trabajar en la seguridad de la infraestructura así como cada uno de los sistemas.

- **Integridad:** se vuelve difícil mantener la integridad, aplicar las reglas de integridad a través de la red puede ser muy caro en términos de transmisión de datos.

- **Falta de experiencia:** las bases de datos distribuidas es un campo relativamente nuevo y poco común por lo cual no existe mucho personal con experiencia o conocimientos adecuados.

- **Carencia de estándares:** aún no existen herramientas o metodologías que ayuden a los usuarios a convertir un SGBD centralizado en un SGBD distribuido.

- **Diseño de la base de datos se vuelve más complejo:** además de las dificultades que generalmente se encuentran al diseñar una base de datos, el diseño de una base de datos distribuida debe considerar la fragmentación, replicación y ubicación de los fragmentos en sitios específicos.(8; 9)

1.4 La réplica de datos

La réplica de datos consiste en el transporte de datos entre dos o más servidores, permitiendo que ciertos datos de la base de datos estén almacenados en más de un sitio, y así aumentar la disponibilidad de los mismos y mejorar el rendimiento de las consultas globales. El modelo de réplica está formado por: publicador, distribuidor, suscriptor, publicación, artículo y suscripción; y varios agentes responsabilizados de copiar los datos entre el publicador y el suscriptor. A los tipos básicos de replicación (de instantáneas, transaccional y de mezcla), se le incorporan opciones para ajustarse aún más a los requerimientos del usuario. El publicador es un servidor que pone los datos a disposición de otros servidores para poder replicarlos. El distribuidor es un servidor que aloja la base de datos de

distribución y almacena los datos históricos, transacciones y metadatos. Los suscriptores reciben los datos replicados.(10)

Tipos Básicos de Replicación de Datos

Replicación de Instantáneas

En la replicación de instantáneas los datos se copian tal y como aparecen exactamente en un momento determinado. Por consiguiente, no requiere un control continuo de los cambios. Las publicaciones de instantáneas se suelen replicar con menos frecuencia que otros tipos de publicaciones. Puede llevar más tiempo propagar las modificaciones de datos a los suscriptores. Se recomienda utilizar: cuando la mayoría de los datos no cambian con frecuencia; se replican pequeñas cantidades de datos; los sitios con frecuencia están desconectados y es aceptable un período de latencia largo.(10; 11)

Replicación transaccional

En este caso se propaga una instantánea inicial de datos a los suscriptores, y después, cuando se efectúan las modificaciones en el publicador, las transacciones individuales se propagan a los suscriptores. Al finalizar la propagación de los cambios, todos los suscriptores tendrán los mismos valores que el publicador. Suele utilizarse cuando: se desea que las modificaciones de datos se propaguen a los suscriptores, normalmente pocos segundos después de producirse; se necesita que las transacciones sean atómicas, que se apliquen todas o ninguna al suscriptor; los suscriptores se conectan en su mayoría al publicador; su aplicación no puede permitir un período de latencia largo para los suscriptores que reciban cambios.

Replicación de mezcla

Permite que varios sitios funcionen en línea o desconectados de manera autónoma, y mezclar más adelante las modificaciones de datos realizadas en un resultado único y uniforme. La instantánea inicial se aplica a los suscriptores. Los datos se sincronizan entre los servidores a una hora programada o a petición. Las actualizaciones se realizan de manera independiente, sin protocolo de confirmación, en más de un servidor, así el publicador o más de un suscriptor pueden haber actualizado los mismos datos. Por lo tanto, pueden producirse conflictos al mezclar las modificaciones de datos. Cuando se produce un conflicto, el agente de mezcla invoca una resolución para determinar qué datos se aceptarán y se propagarán a otros sitios. (10)

Factores para elegir el método de replicación a utilizar

En la elección de un método adecuado para la distribución de los datos en una organización influyen varios factores. Los cuales se separan en dos grupos: factores relacionados con los requerimientos de la aplicación y factores relacionados con el entorno de red.

Entre los factores relacionados con los requerimientos de la aplicación están la autonomía de un sitio que da la medida de cuanto puede operar el sitio desconectado de la base de datos publicadora. La consistencia transaccional de un sitio viene dado por la necesidad de ejecutar o no inmediatamente todas las transacciones que se han ejecutado en el servidor, o si es suficiente con respetar el orden de las mismas. La latencia de un sitio se refiere al momento en que se deben de sincronizar las copias de los datos. ¿Necesitan los datos estar el 100% en sincronía? o si es admisible determinada latencia ¿de qué tamaño es aceptable el rezago?

Entre los factores relacionados con el entorno de red está la velocidad de transmisión de datos de la red, considerando preguntas como ¿Cómo luce la red? ¿Es rápida? Debe analizarse además la confiabilidad de la red y responder preguntas como ¿Cuán confiable es la red? Por otra parte en el caso que los servidores no permanezcan todos los días encendidos, como pudiera suceder en algunas organizaciones, deben considerarse los horarios de disponibilidad de cada servidor.

La consideración de estos factores sirve de guía en la configuración del ambiente de replicación. Considerando las siguientes preguntas: ¿Qué datos se van a publicar? ¿Reciben todos los suscriptores todos los datos o sólo subconjuntos de ellos? ¿Se deben particionar los datos por sitio? ¿Se debe permitir que los suscriptores envíen actualizaciones de los datos? Y en caso de permitirlas ¿Cómo deben implementarse? ¿Quiénes pueden tener acceso a los datos? ¿Se encuentran estos usuarios en línea? ¿Se encuentran conectados mediante enlaces caros?

Fases generales para implementar y supervisar la replicación

A pesar de que existen varias formas de implementar y supervisar la réplica, y el proceso de replicación es diferente según el tipo y las opciones elegidas, en general, la replicación se compone de las siguientes fases:

- Configuración de la replicación.
- Generación y aplicación de la instantánea inicial.
- Modificación de los datos replicados.

- Sincronización y propagación de los datos.

Cuando se hace una réplica de datos se debe definir quién será el publicador y quién el suscriptor, desde este punto de vista existen dos tipos de réplica Maestro-Maestro y la otra es Maestro-Esclavo.

Maestro-Maestro: es un método de replicación que permite tener datos almacenados en un grupo de computadoras y estos pueden ser actualizados por cualquier miembro de este grupo, el sistema de replicación Maestro-Maestro es el encargado de la propagación de los datos modificados por cualquier miembro del grupo de computadoras hacia el resto y resolver cualquier conflicto que esto pueda traer como consecuencia. Este modelo tiene como ventaja que si un servidor Maestro se cae o deja de funcionar, otro asume esa función y continúa con la actualización de la información, estando diseminados por toda la red.

Maestro-Esclavo: Permite tener almacenada la información en un grupo de computadoras pero solo puede ser actualizada en un solo sentido desde el servidor Maestro hacia los Esclavos, o sea es solo lectura, pues los servidores esclavos no tienen la propiedad de modificar la información, cuando el servidor Maestro registra un cambio en la información que posee se inicia el proceso de distribuir hacia los servidores esclavos, con este tipo de réplica se evita muchos conflictos pues a diferencia del Maestro-Maestro la misma información no puede ser actualizada al mismo tiempo por varios servidores diferentes, además este tipo de réplica tiene un menor tiempo de latencia.(1012)

1.5 Descripción actual del dominio del problema

1.5.1 Situación problémica

El departamento de Fuerza de Trabajo Calificada (FTC) del Ministerio de Economía y Planificación (MEP), tiene como uno de los procesos más importantes la elaboración de los planes anuales de ingreso a la educación superior y los planes anuales de ubicación laboral. Para la elaboración de estos planes se hace un trabajo en conjunto con todos los Organismos de Administración Central del Estado (OACE).

Esta información almacenada con la cual se realizan las planificaciones de los recursos no está completamente actualizada por lo que se hace compleja la realización de esta función debido a que depende en un gran porciento de la actualidad de la misma. Una planificación realizada con una información desactualizada podría traer problemas para el país, pues este proceso se realiza con 5 años de antelación. Además como se trabaja en conjunto para elaborar todos estos planes se necesita que la información con que cuenten todas las instituciones involucradas sea la misma y actualmente la

información es trasmitida en dispositivos de almacenamientos para ser trasladada por un personal, funcionando igual tanto para la transmisión como para la recepción, lo que provoca un gasto en transporte, poca seguridad y pérdida de tiempo en este proceso.

1.6 Estado del arte

La réplica de datos a nivel mundial para el SGBD *PostgreSQL*, se caracteriza por el uso de múltiples herramientas que facilitan este proceso, entre las más usadas se encuentran, *PgClúster*, *PgPool-II*, *Slony-I*, *PyRéplica*, estas son de fácil instalación, administración y mantenimiento, además de su capacidad para funcionar sobre varias plataformas.

1.6.1 Herramientas para la réplica de datos para PostgreSQL

PgClúster: Replicador de múltiples maestros y sincrónico (servidor "especial").

Características

- Funcionalidad Avanzada.
- No posee un único punto de falla.
- Sincronismo de alto rendimiento.
- Balance de carga y monitoreo.
- Consta de tres tipos de servidores, un equilibrador de carga, un clúster, y un servidor de replicación.

Desventajas

- Requiere una versión "parcheada" de *PostgreSQL*.
- Instalación y configuración compleja.
- Puede requerir configuraciones avanzadas de *hardware*.(13)

PgPool-II: es una herramienta que funciona entre servidores PostgreSQL y un cliente de base de datos *PostgreSQL*.

Características

– Agrupación de conexiones: guarda las conexiones con los servidores *PostgreSQL*, para volver a usarlas cada vez que se crea una nueva conexión con las mismas propiedades (es decir, nombre de usuario, bases de datos, versión del protocolo), reduce sobrecarga de la conexión y mejora el rendimiento global del sistema.

– Replicación: administra varios servidores *PostgreSQL*, permite crear una copia de seguridad en tiempo real sobre 2 o más discos físicos, por lo que el servicio puede continuar sin detención de servidores en caso de un fallo de disco.

– Balance de carga: posibilita reducir la carga sobre cada servidor mediante la distribución de las consultas entre los demás servidores que conforman un sistema distribuido, el rendimiento mejora proporcionalmente de acuerdo a la cantidad de servidores que existan.

– Conexiones limitadas: existe un límite en el número máximo de conexiones simultáneas al servidor, aunque puede ser configurado, sin embargo, aumenta el consumo de recursos y el rendimiento del sistema, las conexiones adicionales que no puedan ser atendidas se pondrán en cola en lugar de devolver un error inmediatamente.

– Paralelización de consultas: utilizando la función de consulta en paralelo, los datos pueden ser divididos entre los múltiples servidores, de modo que una consulta se puede ejecutar en todos los servidores al mismo tiempo para reducir el tiempo de ejecución global, esta propiedad funciona mejor en la búsqueda de datos a gran escala.

Desventajas

- Único punto de falla.
- No garantiza la replicación de funciones correctamente.
- Soporte limitado de autenticación.
- Se detiene si se pierde el sincronismo. (14; 15)

Slony-I: es un sistema de replicación Maestro-Esclavo, que incluye todas las características y capacidades necesarias para replicar bases de datos grandes a un número razonablemente limitado de servidores esclavos. *Slony-I* está diseñado para ser usado en los centros de datos y sitios de copia de seguridad, donde el modo normal de funcionamiento es que todos los nodos estén disponibles.

Características

- Diseñado para centros de datos y para realizar copias de seguridad.
- Esquema avanzado de configuración.
- Pequeño lenguaje de administración.

Desventajas

- Instalación, configuración y administración compleja.
- Solo un maestro (para escritura).
- Inviable para conexiones inestables o configuraciones variables.(16)

PyRéplica: es un sistema de replicación simple para *PostgreSQL* programado en *Python* con un código compacto, simple y claro, es bastante eficiente, de fácil instalación, administración y mantenimiento, además de su capacidad para funcionar en cualquier plataforma *Windows* o *Linux*.

Características

- Asincrónico.
- Maestro / Esclavo y Multi-Maestro limitado.
- Detección de conflictos.
- Notificaciones vía Email.
- Monitoreo de las conexiones (*KeepAlive*).
- Conexiones directas a los *backends.*
- Sin protocolos especiales (Lenguaje de Consultas Estructurado textual).
- Protegido con transacciones en dos fases.

Ventajas

- Puede usarse para propagar órdenes (Lenguaje de Definición de Datos) DDL a varios servidores.
- Soporta replicación de valores devueltos por funciones de fecha, aleatorias, secuencias, etc.

Desventajas

No soporta:

- Replicación de DDL automática.
- Replicación de secuencias.
- Replicación sincrónica.
- Resolución de conflictos (se pueden evitar con reglas/disparadores y/o detectarlos).(11)

1.6.2 Soluciones de réplica de datos en Cuba

La mayoría de las soluciones de replicaciones de datos en Cuba, se han realizado en la universidad entre las más importantes se encuentran:

– **Sistema de sincronización y gestión de nodos aislados para la replicación de bases de datos en *PostgreSQL.***

Para la realización del sistema el lenguaje de programación que se uso fue *Python* con la librería grafica *GTK* (biblioteca que contiene los objetos y funciones para crear la interfaz gráfica de usuario) y la herramienta de desarrollo de interfaces *Glade* (Diseñador de interfaces). Los reportes necesarios se muestran en formato HTML (Lenguaje de Marcado de Hipertexto) a través del navegador *Mozilla Firefox*, el entorno de réplica de este sistema es Maestro – Maestro.

– **Implementación del componente réplica de bases de datos para Akademos v2.0.**

Este es un componente de réplica hecho para escritorio, se utilizó como lenguaje de programación el *Python*, la herramienta de replicación que usaron fue PyRéplica que también está programada en *Python* y que además de permitir replicar datos en los dos entornos Maestro–Maestro y Maestro–Esclavo, es de fácil instalación y administración. Para el desarrollo de la interfaz de usuario se usó el *PythonCard*, porque además de estar programado en *Python* es de fácil y sencillo diseño.

– **Solución para la réplica de datos del proyecto Prisiones.**

En esta solución la replicación ocurre a través de disparadores, se puede aplicar a los SGBD, *Oracle*, PostgreSQL, *MySQL*, *Microsoft SQL Server.* Utiliza los protocolos *TCP/IP*, *FTP* y *HTTP* para la transferencia de datos de replicación, los archivos de gran tamaño son enviados por *FTP* permitiendo resumir la transmisión en caso de interrupciones en la red. Los conflictos se detectan y pueden ser solucionados interactivamente o automáticamente mediante la definición de reglas para la resolución de estos. Presenta interfaz visual Web, por lo que se puede administrar y configurar de forma remota solo con emplear un navegador Web.

– **Réplica bidireccional basada en control de cambios.**

Esta solución se caracteriza por ser bidireccional y basado en el control de cambios y no en colas por lo que evita las transmisiones redundantes. Trabaja en cualquier gestor de bases de datos y garantiza la integridad de los datos pues provee de mecanismos de detección y solución de conflictos, fue hecha en la plataforma *.Net*, el entorno de replicación que soporta es Maestro – Maestro.

1.7 Proceso de desarrollo de *software*

Un proceso define “quién” está haciendo “qué”, “cuándo” y “cómo” para alcanzar un determinado objetivo. Un Proceso de Desarrollo de *Software* es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto. Tiene la misión de transformar los requerimientos del usuario en un producto de *software*; de manera que los integrantes del equipo y todo aquel que pueda estar interesado en el producto final, tenga la misma visión.(17)

1.7.1 Metodología de desarrollo de *software*

Proceso Unificado del Software (RUP)

RUP es una metodología para el desarrollo del *software* potente. Está centrada en la arquitectura basada en componentes, es iterativo e incremental y dirigido por casos de uso. Con respecto a la verificación de la calidad, RUP ayuda a planificar, diseñar, implementar, ejecutar y evaluar pruebas que verifican estas cualidades. Crea pruebas para cada escenario o caso de uso para asegurar que todos los requerimientos están propiamente implementados. Verifica la calidad del *software* con respecto a los requerimientos basados en la confiabilidad, funcionalidad, desempeño de la aplicación y del sistema. Prueba cada una de las iteraciones, algo vital ya que los problemas del *software* son de 100 a 1000 veces más costosos de encontrar y reparar después del desarrollo. Con respecto al control de cambios, RUP establece espacios de trabajo seguros para cada desarrollador, permite controlar, llevar un riesgo y monitorear cambios para permitir un desarrollo iterativo. La metodología RUP es la más aceptable para proyectos de largo plazo. RUP es una metodología robusta aunque si se configura llegaría a ser una metodología ligera, como lo es XP. (18)

1.7.2 Lenguaje de modelado

Lenguaje Unificado de Modelado (UML) es el lenguaje de modelado de sistemas de *software* más conocido en la actualidad, son un grupo de especificaciones de notación orientadas a objetos, las cuales están compuestas por distintos diagramas, que representan las diferentes etapas del desarrollo de un proyecto de *software*. Permite la especificación, visualización, construcción y documentación de elementos de la Ingeniería del *Software*.

Cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de lo que se quiere representar.

Los Diagramas de Estructura enfatizan en los elementos que deben existir en el sistema modelado:

- Diagrama de clases.
- Diagrama de componentes.
- Diagrama de objetos.
- Diagrama de estructura compuesta.
- Diagrama de despliegue.
- Diagrama de paquetes.
- Diagrama de casos de uso.

Los Diagramas de Comportamiento enfatizan en lo que debe suceder en el sistema modelado:

- Diagrama de actividades.
- Diagrama de estados.

Los Diagramas de Interacción son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:

- Diagrama de secuencia.
- Diagrama de comunicación, que es una versión simplificada del Diagrama de colaboración.
- Diagrama de tiempos.
- Diagrama global de interacciones o Diagrama de vista de interacción.(17)

1.7.3 Lenguaje de programación Web.

PHP

PHP es el acrónimo de Preprocesador de Hipertexto. Se trata de un lenguaje interpretado de alto nivel embebido en páginas *HTML* y ejecutado en el servidor. Está muy orientado al desarrollo de aplicaciones Web y permite insertar contenidos dinámicos en las páginas.(19)

Algunas características:

- Rapidez de ejecución.
- Es un lenguaje específicamente diseñado para realizar aplicaciones Web.
- El software necesario para ejecutar aplicaciones es libre.
- Mantiene un bajo consumo de recursos de máquina.
- Gran seguridad, muy poca probabilidad de corromper los datos.

- Trabaja con una diversidad de bases de datos, y es quizás su característica más fuerte.
- Rico en funciones predefinidas.
- Puede ser instalado en servidores *Windows*.
- Fácil aprendizaje.
- Es un lenguaje libre.
- Trabaja en combinación con otras tecnologías: *Perl*, *JavaScript*, *Python* y *HTML* dinámico.
- Permite embeber sus pequeños fragmentos de código dentro de la página *HTML.*
- Las tareas fundamentales que puede realizar directamente el lenguaje son definidas en el mismo lenguaje como funciones.
- Combina excelentemente con otras inmejorables herramientas, como el servidor apache y la base de datos *MySQL* (o *MSQL,* o *PostgreSQL*)
- Dispone de librerías para la programación de entorno de escritorio, como GTK.
- Buena documentación.

Principales Ventajas:

- Alto rendimiento, sencillo y fácil de aprender.
- Bibliotecas incorporadas para muchas tareas Web habituales.
- Acceso al código abierto.
- Similar en sintaxis a *C* y a *PERL.*
- Soporta en cierta medida la orientación a objeto.
- El análisis léxico para recoger las variables que se pasan en la dirección lo hace PHP de forma automática. Librándose el usuario de tener que separar las variables y sus valores.
- Excelente soporte de acceso a base de datos.
- Viene equipado con un conjunto de funciones de seguridad que previenen la inserción de órdenes dentro de una solicitud de datos.
- Se puede hacer de todo lo que se pueda transmitir por vía *HTTP*.

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas Web dinámicas. Una página Web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso

al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con *JavaScript* se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (20)

Ventajas:

- Fácil de aprender, barato, sencillo, moderno, rápido y potente: permite la programación orientada a objetos.
- Usabilidad.
- Reducción de la carga del servidor.
- Es visual: permite la "programación visual" (ventanas, botones, colores, formularios, etc.).
- Interactividad.
- Realimentación inmediata.
- Validaciones sencillas antes de enviar al servidor.

Desventajas

- JavaScript fue diseñado de forma que se ejecutara en un entorno muy limitado que permitiera a los usuarios confiar en la ejecución de los *scripts*.

- Los *scripts* no pueden comunicarse con recursos que no pertenezcan al mismo dominio desde el que se descargó.

- Los *scripts* no pueden cerrar ventanas que no hayan abierto ellos mismos.

1.8 Software libre

El *software* libre no se puede ver desde el punto de vista del costo de su compra, se dice que es *software* libre porque es de libre expresión. Es una cuestión de la libertad de los usuarios de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el *software*, precisamente, significa que los usuarios de programas tienen las cuatro libertades esenciales.

- La libertad de ejecutar el programa, para cualquier propósito.
- La libertad de estudiar cómo trabaja el programa, y cambiarlo para que haga lo que usted quiera. El acceso al código fuente es una condición necesaria para ello.
- La libertad de redistribuir copias para que pueda ayudar al prójimo.

- La libertad de distribuir copias de sus versiones modificadas a terceros. Si lo hace, puede dar a toda la comunidad una oportunidad de beneficiarse de sus cambios. El acceso al código fuente es una condición necesaria para ello.

Un programa es *software* libre si los usuarios tienen todas esas libertades. Entonces, debería ser libre de redistribuir copias, tanto con o sin modificaciones, ya sea gratis o cobrando una tarifa por distribución, a cualquiera en cualquier parte. El ser libre de hacer estas cosas significa, que no tiene que pedir o pagar el permiso.

La libertad de ejecutar el programa significa la libertad para cualquier tipo de persona u organización de usarlo en cualquier tipo de sistema de computación, para cualquier tipo de trabajo y propósito, sin estar obligado a comunicarlo a su programador, o alguna otra entidad específica. En esta libertad, el propósito de los usuarios es el que importa, no el propósito de los programadores. Como usuario es libre de ejecutar un programa para sus propósitos; y si lo distribuye a otra persona, también es libre para ejecutarlo para sus propósitos, pero no tiene derecho a imponerle sus propios propósitos.(21; 22)

1.9 Herramientas y tecnologías de desarrollo

1.9.1 Visual Paradigm como herramienta de modelado

Visual Paradigm for UML, es una herramienta diseñada para desarrollar *software* con Programación Orientada a Objetos, busca reducir la duración del ciclo de desarrollo brindando ayuda tanto a arquitectos, analistas, diseñadores y desarrolladores.

La herramienta ayuda al equipo de desarrollo de sistemas a agilizar el modelado del *software*, aumentando al máximo y acelerando el trabajo en equipo y las contribuciones individuales. Además posee una buena cantidad de productos o módulos para facilitar el trabajo durante la confección de un *software* así como garantizar la calidad del producto final.

Permite dibujar todos los tipos de diagramas, código inverso, generar código desde diagramas y generar documentación.

Facilita a las organizaciones y al diagrama de diseño visual, integrar y desplegar sus aplicaciones empresariales de misión crítica y sus bases de datos. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios realizados por sus compañeros.

Entre sus principales características se encuentra que es un entorno de creación de diagramas para *UML*, diseño centrado en casos de uso y enfocado al negocio que genera un *software* de mayor calidad, uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación, es amigable, multiplataforma, generación de documentos además de la integración con los distintos Ambientes de Desarrollo Integrados.(23)

1.9.2 NetBeans como Entorno de Desarrollo Integrado

NetBeans IDE (Entorno Integrado de Desarrollo) es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en *Java* pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el *NetBeans IDE*. Es un producto libre y gratuito sin restricciones de uso.

También está disponible la plataforma *NetBeans*, una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de *software*, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que también pueden utilizarse para desarrollar sus propias herramientas y soluciones.

Ambos productos son de código abierto y gratuito para uso tanto comercial como no comercial. El código fuente está disponible para su reutilización.

1.9.3 Sistema gestor de bases de datos

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD (Distribución de Software Berkeley) y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. PostgreSQL utiliza un modelo cliente-servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Los componentes más importantes de un sistema *PostgreSQL* se muestran en la figura 1.(24; 25)

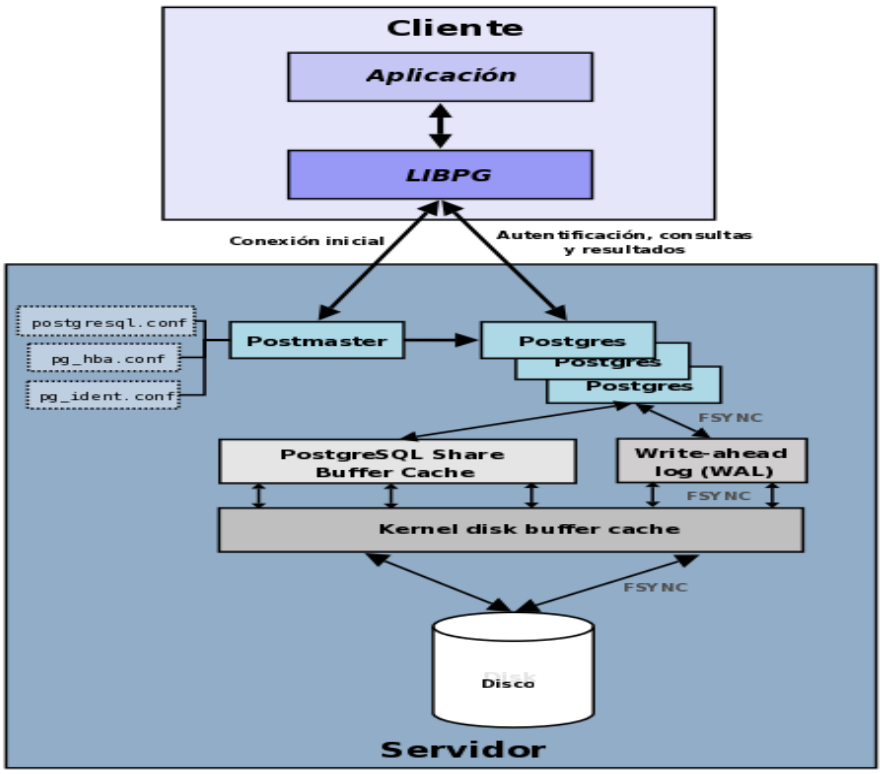


Figura 1 *Componentes de un sistema* PostgreSQL

- **Aplicación cliente**: esta es la aplicación cliente que utiliza *PostgreSQL* como administrador de bases de datos. La conexión puede ocurrir vía *TCP/IP* ó *sockets* (puertos) locales.

- **Demonio *postmaster***: este es el proceso principal de *PostgreSQL*. Es el encargado de escuchar por un puerto/*socket* por conexiones entrantes de clientes. También es el encargado de crear los procesos hijos que se encargaran de autentificar estas peticiones, gestionar las consultas y mandar los resultados a las aplicaciones clientes.

- **Ficheros de configuración**: los tres ficheros principales de configuración utilizados por *PostgreSQL*, postgresql.conf, pg_hba.conf y pg_ident.conf.

- **Procesos hijos *PostgreSQL***: procesos hijos que se encargan de autentificar a los clientes, de gestionar las consultas y mandar los resultados a las aplicaciones clientes.

- ***PostgreSQL share buffer cache***: memoria compartida usada por *PostgreSQL* para almacenar datos en caché.

- ***Write-Ahead Log (WAL)***: componente del sistema encargado de asegurar la integridad de los datos.

- ***Kernel disk buffer cache***: caché de disco del sistema operativo.

- **Disco**: disco físico donde se almacenan los datos y toda la información necesaria para que *PostgreSQL* funcione.

Características más importantes y soportadas por *PostgreSQL.*

Sus características técnicas la hacen uno de los SGBD más potentes y robustos del mercado. Su desarrollo comenzó hace más de 15 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo.

Límites de *PostgreSQL*:

Límite	Valor
Máximo tamaño base de datos	Ilimitado (Depende de tu sistema de almacenamiento).
Máximo tamaño de tabla	32 TB.
Máximo tamaño de fila	1.6 TB.
Máximo tamaño de campo	1 GB.
Máximo número de filas por tabla	Ilimitado.
Máximo número de columnas por tabla	250-1600 (dependiendo del tipo).
Máximo número de índices por tabla	Ilimitado.

Tabla 1 *Límites del PostgreSQL*

Durante los años de existencia del proyecto *PostgreSQL*, el tamaño del mismo, tanto en número de desarrolladores, como en números de línea de código, funciones y complejidad del mismo ha ido aumentando año tras año. La siguiente tabla muestra una gráfica con la evolución del número de líneas de código en cada versión de *PostgreSQL*.

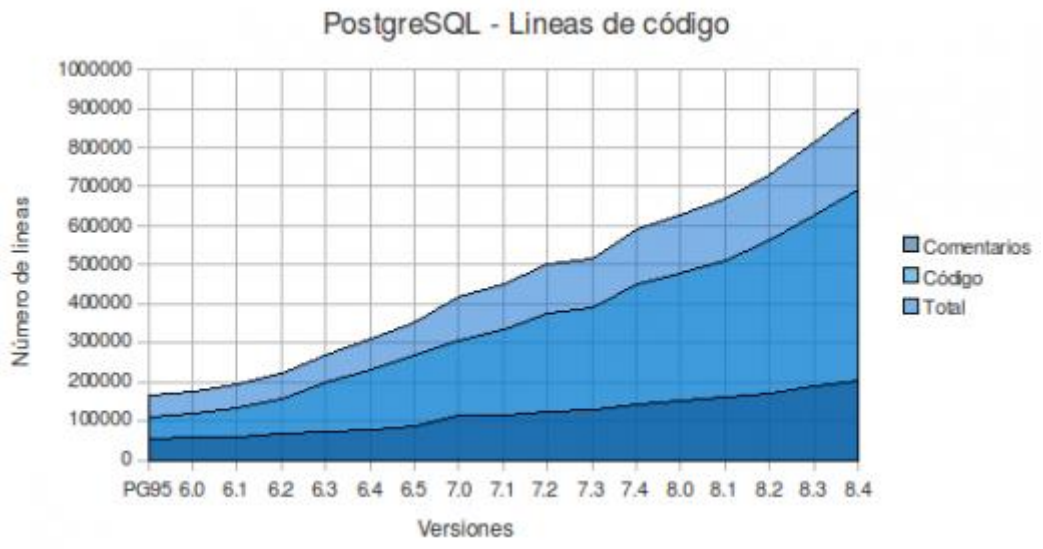


Tabla 2 *Evolución del PostgreSQL*

Ventajas

- Por su arquitectura de diseño, escala muy bien al aumentar el número de (CPUs) Unidad Central de Procesamiento y la cantidad de *RAM*.

- Soporta transacciones y desde la versión 7.0, claves ajenas (con comprobaciones de integridad referencial).

- Tiene mejor soporte para disparadores y procedimientos en el servidor.

- Soporta un subconjunto mayor que el que soporta *MySQL*. Además, tiene ciertas características orientadas a objetos.

Desventajas

- Consume bastantes recursos y carga más el sistema.
- Menos funciones en PHP.(26)

1.9.4 Framework de desarrollo

ExtJs

El núcleo de *ExtJs* es una biblioteca de *JavaScript* ligera y con muchas características disponible bajo licencia *MIT* (Instituto Tecnológico de Massachusetts). El Núcleo de *ExtJs* está lleno de características excitantes orientado a un rápido desarrollo alentando el uso de código escalable y bien diseñado. Esta biblioteca proporciona abstracciones para manipulación del *DOM* (Modelo de Objetos del Documento), *Ajax* (*JavaScript* Asíncrono y *XML*) y eventos personalizados, animaciones, plantillas, mecanismos de programación orientada a objetos y más. El núcleo de *ExtJs* fue liberado bajo una licencia *MIT* y es perfecto para la inclusión en una página Web dinámica o hasta una pequeña aplicación.(27)

Ventajas

- Alto rendimiento, de interfaz de usuario personalizable.
- Bien diseñado y modelo de componentes extensibles.
- Una interfaz intuitiva, posee una *API* fácil de usar.(28)

1.9.5 Servidor Web Apache

Un servidor web es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet. El servidor web se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo de acuerdo a los comandos solicitados.(29)

El servidor HTTP Apache es un servidor web HTTP de código abierto para cualquier plataforma, desarrollado por la Apache *Software Foundation*:

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- El hecho de ser gratuito es importante pero no tanto como que se trate de código fuente abierto, lo que le otorga transparencia.
- Es un servidor altamente configurable de diseño modular, es muy sencillo ampliar sus capacidades.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor, es posible su configuración para que ejecute un determinado script cuando ocurra un error en concreto.

- Tiene una alta configurabilidad en la creación y gestión de registros permite la creación de ficheros de registros a medida del administrador. (30; 31)

1.10 Fundamentación de las herramientas, lenguajes y metodología a utilizar

Las herramientas, lenguajes y metodología a usar fueron establecidas por la dirección del proyecto Fuerza de Trabajo Calificada en su marco de trabajo basándose en las tendencias actuales de migración a software libre.

Se construirá una aplicación Web para solucionar el problema planteado debido a las ventajas que esta posee principalmente porque no es necesaria la instalación en cada una de las estaciones de trabajo en las cuales se use la aplicación, además de que se puede configurar la réplica desde cualquier computadora sin importar su ubicación.

Como lenguaje de programación del lado del cliente se trabajará con *JavaScript* por ser un lenguaje interpretado y no es preciso compilar los programas para ejecutarlos y se pueden probar en cualquier navegador sin necesidad de procesos intermedios. Del lado del servidor se utilizará PHP que al igual que el anterior es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor, entre sus ventajas se encuentra que posee gran seguridad, muy poca probabilidad de corromper los datos, es de fácil aprendizaje y el software necesario para ejecutar sus aplicaciones es libre.

Se utilizará PostgreSQL como SGBD debido a que es el más potente del mercado en lo que respecta a *software* de código abierto, tiene mejor soporte para disparadores y procedimientos en el servidor, usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema, es capaz de manejar una enorme cantidad de datos, permitiendo gran conjunto de accesos simultáneos de los usuarios, brindando seguridad y estabilidad de los mismos.

Como metodología de desarrollo se trabajará con RUP pues establece espacios de trabajo seguros para cada desarrollador, permite controlar, llevar un riesgo y monitorear cambios para permitir un desarrollo iterativo, con respecto a la verificación de la calidad, ayuda a planificar, diseñar, implementar, ejecutar y evaluar pruebas que confirman estas cualidades. Crea pruebas para cada escenario o caso de uso para asegurar que todos los requerimientos están propiamente implementados.

La herramienta para la modelación del sistema que se utilizará será *Visual Paradigm* que aunque es propietaria la universidad posee la licencia por tanto puede usarse para este propósito, ayuda al equipo

de desarrollo de sistemas a agilizar el modelado del *software*, aumentando al máximo y acelerando el trabajo en equipo y las contribuciones individuales, permite dibujar todos los tipos de diagramas, código inverso, generar código desde diagramas y generar documentación, entre sus principales características se encuentra que es un entorno de creación de diagramas para *UML*, diseño centrado en casos de uso y enfocado al negocio que genera un *software* de mayor calidad.

Como plataforma de desarrollo se utilizará *NetBeans* es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas, es de código abierto y gratuito el código fuente está disponible para su reutilización, permite desarrollar aplicaciones a partir de un conjunto de módulos o componentes de software, además es una herramienta ideal para el desarrollo de aplicaciones web.

Se empleará el *framework ExtJs* para la interfaz de usuario, pues está lleno de características excitantes orientado a un rápido desarrollo alentando el uso de código escalable y bien diseñado.

Como servidor Web se utilizará el Apache, pues corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal, es un servidor altamente configurable de diseño modular, es muy sencillo ampliar sus capacidades, además permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor, es factible su configuración para que ejecute un determinado script cuando ocurra un error en concreto.

Como herramienta de réplica de datos se trabajará con PgPool-II, pues guarda las conexiones con los servidores *PostgreSQL* para volver a usarlas cada vez que se crea una nueva conexión con las mismas propiedades, posibilita reducir la carga sobre cada servidor mediante la distribución de las consultas entre los demás servidores que conforman un sistema distribuido, limita las conexiones, además de la paralelización de consultas que permite dividir las peticiones entre los demás servidores disminuyendo el tiempo global de respuesta .

1.11 Conclusiones

En el presente capítulo se han descrito los principales conceptos que permiten utilizar las diversas herramientas, tendencias y tecnologías actuales mediante las cuales se pueden desarrollar aplicaciones de réplica de datos.

Han sido analizadas las distintas soluciones existentes a nivel nacional e internacional llegándose a la conclusión de que aunque estas cumplen las funcionalidades para las cuales fueron desarrolladas, presentan el inconveniente de que fueron realizadas mediante herramientas propietarias y algunas

requieren de muchos pasos para su instalación y configuración además de que son aplicaciones de escritorio lo cual trae consigo la necesidad de instalación del *software* en cada uno de los lugares en los que se vaya a utilizar, además de que es necesario instalar y configurar de forma predefinida el sistema operativo para el cual fue desarrollada.

Se fundamentó la utilización del software libre por sus ventajas, la utilización de herramientas como *Visual Paradigm* para el modelado del sistema, como lenguajes de programación se trabajará con JavaScript para la programación del lado del cliente mediante el marco de trabajo ExtJs, PHP del lado del servidor y UML para el modelado. Se utilizará NetBeans como plataforma de desarrollo, PostgreSQL como sistema gestor de bases de datos, como servidor Web el Apache, como metodología de desarrollo se usará RUP por las amplias posibilidades que brinda en cuanto a documentación y la generación de artefactos.

Capítulo 2: Características del Sistema

2.1 Introducción

En el presente capítulo se hace una descripción de la propuesta de solución para el problema actual, se justifica y enuncia el Modelo de Dominio como vía eficiente para representar el negocio de la solución explicándose las entidades y las relaciones existentes entre ellas. Se enumeran los requisitos funcionales y no funcionales que tendrá el sistema que se propone, además se modela el sistema propuesto a través de los actores y los casos de uso y se incluye la descripción detallada de cada caso de uso.

2.2 Proceso de actualización de la información actual en el MEP

Debido al gran volumen de información que maneja en la actualidad en el Ministerio de Economía y Planificación y específicamente el Departamento de Fuerza de Trabajo Calificada, se ha convertido en una prioridad vital mantener íntegra, segura y actualizada la información con la cual trabajan estos organismos de la Administración Central del Estado.

El Departamento de Fuerza de Trabajo Calificada se encarga de manejar la información referente a los graduados universitarios, la disponibilidad de empleo para los mismos, también está a cargo de hacer un estudio con 5 años de antelación relacionado con los posibles graduados de la educación superior y su posterior ubicación.

El éxito de todo este proceso depende del nivel de seguridad, actualización e integridad de la información manejada, proceso que actualmente se lleva a cabo de forma ineficiente por no contar con los mecanismos o sistemas capaces de realizarlo de forma más efectiva.

La información ya actualizada por la persona correspondiente es guardada en dispositivos de almacenamiento.

En el momento en que **cualquier persona** deba viajar al lugar donde se van a actualizar los datos se le entrega el dispositivo para que haga la actualización, de igual manera esta persona trae de regreso los datos actualizados en ese lugar.

El proceso requiere de la gestión de transporte para trasladar a esa persona al lugar indicado lo cual incurre en un gran gasto de tiempo, recursos y poca seguridad de la información.

En el mejor de los casos la información se envía y recibe por correo electrónico lo cual tampoco garantiza la seguridad e integridad de los datos.

El Departamento de Fuerza de Trabajo Calificada necesita tener el control de la información referente a todos los posibles ingresos a la educación superior y los graduados de la misma, así como sus ubicaciones y desempeño laboral después de terminados sus estudios.

2.3 Propuesta de solución

Para darle solución al problema se propone un sistema Web para replicar la información entre las Bases de Datos (BD) en *PostgreSQL* de las entidades implicadas, el MEP y los OACE. En el final del capítulo 1 se propuso como herramienta para la replicación de datos *PgPool-II*, la cual guarda las conexiones con los servidores *PostgreSQL* para volver a usarlas cada vez que se crea una nueva conexión con las mismas propiedades. Reduce sobrecarga de la conexión y mejora el rendimiento global del sistema, posibilita reducir la carga sobre cada servidor mediante la distribución de las consultas entre los demás servidores que conforman en sistema distribuido, mejorando el rendimiento proporcionalmente de acuerdo a la cantidad de servidores que existan.

Utilizando la función de consulta en paralelo, los datos pueden ser divididos entre los múltiples servidores, de modo que una consulta se puede ejecutar en todos los servidores al mismo tiempo para reducir el tiempo de ejecución global, esta propiedad funciona mejor en la búsqueda de datos a gran escala. Esta solución debe permitir además configurar las fuentes de datos tanto origen como destino, además de permitir que se guarden *backups* (copias de seguridad) de la información para restaurar las bases de datos en caso de alguna falla en el proceso de réplica, así como monitorear y obtener reportes de la replicación de datos realizada.

2.3.1 Funcionamiento de la solución

En todos los servidores de BD se instalará la herramienta PgPool-II, la cual se configurará como sigue: en la fuente de datos (FD) maestra que estará en el MEP en el archivo de configuración de la herramienta se incluirán todas las direcciones IP de los servidores esclavos a los cuales se le replicará la información con lo cual se garantiza que los cambios que ocurran en la BD central se copien automáticamente hacia los servidores que se encuentran en los OACE, además se debe crear un usuario de administración para la herramienta que será **pgpool2** que tenga los permisos necesarios sobre la BD para que al detectar algún cambio lo replique hacia las FD esclavas, en los servidores de las demás entidades la herramienta se configurará de igual manera pero con la diferencia de que solamente poseerán la dirección IP de la FD maestra (servidor del MEP), posibilitando que algún

cambio ocurrido se envíe hacia el servidor central, el funcionamiento de esta herramienta es comportarse como si fuera el Sistema Gestor de Bases de Datos por lo que el usuario no tiene que interactuar de manera directa con él, sino que todo ocurre a nivel de herramientas y la comunicación que se establece entre estas. A continuación se presenta una figura para mayor entendimiento de cómo funciona el proceso de réplica de datos para GeForza:

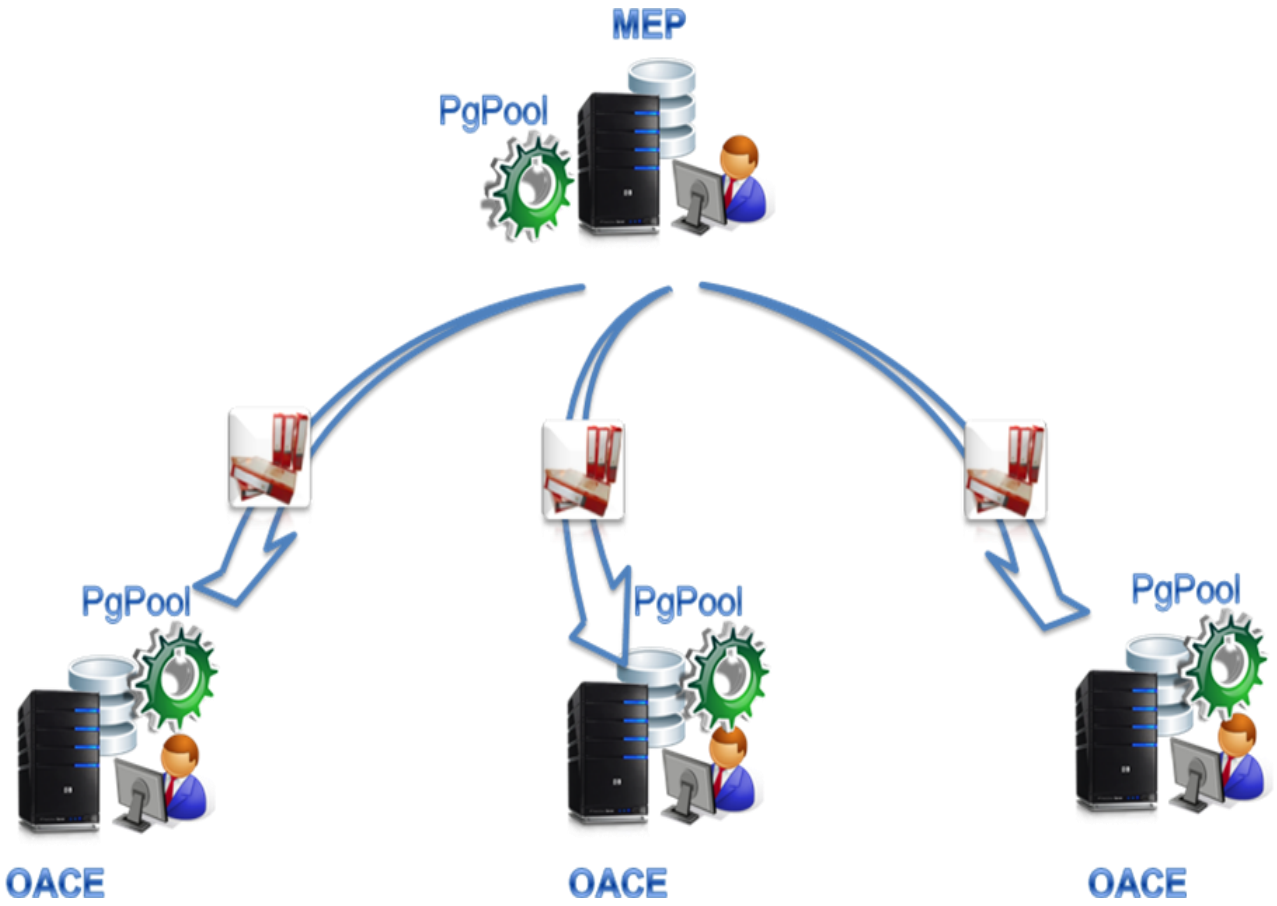


Figura 2 *Sistema de réplica para GeForza*

2.3.2 Descripción del modelo de dominio

El Modelo de Dominio es una representación visual estática de un entorno real que muestra clases conceptuales significativas en el dominio del problema, captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema, después de hacer un estudio se llega a la conclusión de que existe poca estructuración en el negocio y solo se pueden identificar objetos y conceptos dentro del área de interés (los procesos de réplica de datos en el

Departamento de Fuerza de Trabajo Calificada) por lo que lo más adecuado es realizar un Modelo de Dominio que se muestra a continuación:

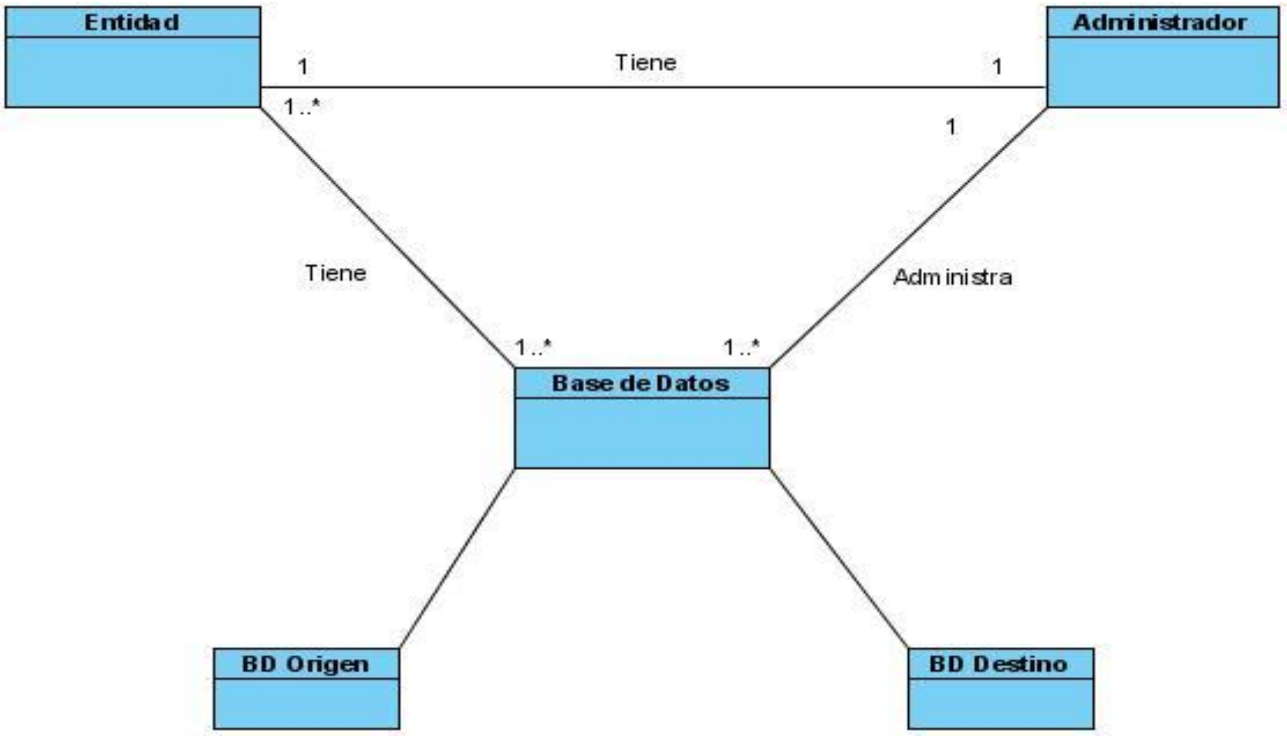


Figura 3 *Modelo del dominio*

Conceptos relacionados:

Entidad: Representa las dependencias que están relacionadas en el proceso (MEP y OACE).

Administrador: Persona encargada del mantenimiento y gestión de la información en las entidades, es quien configurará la réplica de datos.

Base de Datos: Base de datos del sistema.

BD Origen: Base de datos del sistema que funcionará como origen de la actualización de los datos.

BD Destino: Base de datos del sistema que funcionará como destino de la actualización de los datos.

2.4 Especificación de los requisitos de software

Desde el punto de vista funcional la aplicación debe contar con una serie de requisitos que son los que permitirán que la misma cumpla con el objetivo para el cual fue desarrollada, estos requisitos son los que le dan el nivel de aceptación que los clientes finales requieren. El sistema también se regirá por

una serie de aspectos desde el punto de vista no funcional, como la usabilidad, portabilidad, rendimiento y confidencialidad, aspectos muy importantes también en la aceptación de un software.

2.4.1 Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, se mantienen invariables sin importar con que propiedades o cualidades se relacionen. (32)

A continuación se muestran los requerimientos funcionales correspondientes al módulo de réplica de datos del proyecto GeForza.

RF 1. Configurar réplica.

1.1. Configurar fuente de datos.

1.2. Configurar la herramienta PgPool-II.

1.3. Guardar configuración.

RF 2. Crear copia de seguridad.

2.1. Seleccionar base de datos.

2.2. Seleccionar destino de la copia de seguridad.

2.3. Guardar copia de seguridad.

RF 3. Cargar copia de seguridad.

3.1. Seleccionar copia de seguridad.

3.2. Cargar copia de seguridad.

3.3. Restaurar fuente de datos.

2.4.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable, formando una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con la toda la funcionalidad requerida, las propiedades no funcionales, como

cuán seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación.(32)

Los requisitos no funcionales con los que debe contar la aplicación se clasifican en cuanto a:

RNF1 Interfaz externa:

1.1. La aplicación debe tener un diseño sencillo e intuitivo que permita a las personas que interactúen con ella un entorno más cómodo, con colores suaves que sean agradables para el trabajo.

RNF2 Usabilidad:

2.1. Solo un número limitado de personas podrá tener acceso a esta aplicación garantizando que solo las personas encargadas de configurar la réplica puedan hacerlo. Se les deberá dar un curso básico para adiestrarlos en el uso de esta.

2.2. Los usuarios deben tener un conocimiento básico sobre informática para el uso de esta solución a desarrollar.

RNF3 Eficiencia:

3.1. El sistema debe funcionar con un máximo rendimiento pero ajustado a bajas prestaciones de las computadoras debido que no todos los organismos poseen tecnología de punta .

3.2. El sistema requiere de un buen rendimiento que se apoya en el mínimo acceso a base de datos, y realización de consultas no redundantes.

3.3. Para un funcionamiento óptimo de la aplicación se deben seguir las diferentes técnicas de elaboración en la web, que faciliten el rápido acceso a sus páginas.

3.4. La herramienta propuesta debe ser rápida y el tiempo de respuesta debe ser el mínimo posible, adecuado a la rapidez con que el cliente requiere la respuesta a su petición.

RNF4 Seguridad:

4.1 Solo los administradores y las personas autorizadas pueden acceder a la aplicación.

4.2 La transmisión de información entre una Fuente de Datos (FD) a otra se realizará mediante una conexión segura.

4.3 Las conexiones de los clientes se pueden restringir por dirección IP y/o por nombre de usuario mediante el fichero pg_hba.conf.

RNF5 Software:

5.1 Se debe disponer de sistema operativo Windows (2000, NT, XP) Windows Server 2000 o superior, GNU/Linux.

RNF6 Ayuda:

6.1 El usuario debe tener acceso al Manual de Usuario del producto final, en éste debe documentarse la forma de utilizar el sistema.

6.2 Independientemente de donde se encuentre el operador en la interfaz, este podrá obtener ayuda referente a las operaciones que puede realizar, al igual que una propuesta de las operaciones posibles.

RNF7 Restricciones del diseño:

7.1 Lenguaje de programación: PHP.

7.2 El sistema debe de desarrollarse con la tecnología PHP 5.2.6 o superior con una arquitectura Modelo Vista Controlador.

7.3 Como IDE se empleará *NetBeans* 6.8.

7.4 Sistema Gestor de Base de Datos: PostgreSQL 8.3.

7.5 El servidor de aplicación será Lite-HTML o una versión lite de Apache.

7.6 La lógica de presentación constituirá una capa independiente de la lógica de negocio, existiendo validaciones simples de datos de entrada.

7.7 La interfaces destinadas al cliente, deben de programarse en JavaScript.

7.8 Para la modelación del sistema se utilizará *Visual Paradigm*.

7.9 Se utilizará como metodología de desarrollo de software RUP, usando el lenguaje de modelación UML.

7.10 Si el usuario no puede autenticarse o no se encuentra registrado no debe tener acceso a la aplicación.

7.11 El sistema garantizará la autenticación como primera acción para los casos en que sea necesario; esta consistirá en suministrar un nombre de usuario único y una contraseña que debe ser de conocimiento exclusivo de la persona que se autentica.

7.12 Sólo si el usuario es autenticado o se encuentra registrado se autoriza el acceso.

7.13 Si se intenta acceder al sistema desde una estación no autorizada, el acceso será denegado, y la operación será registrada en el sistema.

7.14 El sistema no permitirá que el mismo usuario esté operando desde dos estaciones de trabajo diferentes.

7.15 La versión MEP tendrá todos los privilegios y la de los organismos sólo aquellos que se relacionen con estos (se debe restringir los permisos para esta versión).

7.16 Se utilizará como framework de desarrollo ExtJs.

7.17 El servidor de aplicación será Apache 2.0.40.

7.18 Las computadoras clientes del sistema deben tener un navegador de Internet con base Netscape (Mozilla Firefox).

RNF8 Requisitos legales, de derecho de autor y otros:

8.1 La Universidad de las Ciencias Informáticas tiene el derecho de autor sobre el Sistema Unificado de la Gestión de la Fuerza de Trabajo Calificada.

8.2 El proyecto utiliza la política de *software* libre donde todas las herramientas que se utilizan son *software* libre. Para la herramienta Visual Paradigm se utiliza la licencia que la universidad compró.

RNF9 Hardware:

9.1. Las PCs clientes deben tener las siguientes características.

9.1.1Memoria RAM 256 MB o superior.

9.1.2Disco duro de 20 GB o más.

9.1.3Procesador de 1.2 GHz o más.

9.1.4Conectividad.

9.2. Se requiere de un servidor para bases de datos con las siguientes características.

9.2.1Servidor Xeon a 3.0 GHz.

9.2.2Memoria Ram de 1 GB.

9.2.3Dos discos duros de 36 y 250 GB, este último con dos particiones.

9.3. Se requiere de un servidor web con las siguientes características.

9.3.1Servidor Xeon a 3.0 GHz.

9.3.2Memoria RAM de 1 GB.

9.3.3Tres discos duros de 36 GB cada uno con RAID 5.

RNF10 Soporte:

10.1 Realizar pruebas, mantenimiento e instalaciones necesarias para lograr el mejoramiento y evolución en el tiempo.

10.2 Para garantizar el soporte de esta herramienta, se debe documentar la aplicación con un manual de ayuda para los usuarios, así como la posibilidad de emitir sus quejas y sugerencias a los desarrolladores de la herramienta, por correo o por teléfono, realizar mantenimiento al sistema y darle solución a cualquier problema que surja con la aplicación.

2.5 Actores del sistema

Cada trabajador del negocio (inclusive si fuera un sistema ya existente) que tiene actividades a automatizar es un candidato a actor del sistema. Si algún actor del negocio va a interactuar con el sistema, entonces también será un actor del sistema.

Los actores del sistema:

- No son parte de él.
- Pueden intercambiar información con él.

- Pueden ser un recipiente pasivo de información.
- Pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado. (32)

Actor	Descripción
Administrador	Especialista encargado de realizar la réplica de datos.

Tabla 3 *Descripción de los actores del sistema*

2.6 Casos de uso

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema.(32)

2.6.1 Definición de los casos de uso

Caso de uso	Nombre
CU-1	Configurar réplica.
CU-2	Crear copia de seguridad.
CU-3	Cargar copia de seguridad.

Tabla 4 *Definición de los casos de uso*

2.7 Diagrama de casos de uso

Un diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores, describe la funcionalidad propuesta del nuevo sistema a desarrollar y los procesos a automatizar, representados a través de casos de uso y que responden a los requisitos funcionales del mismo.

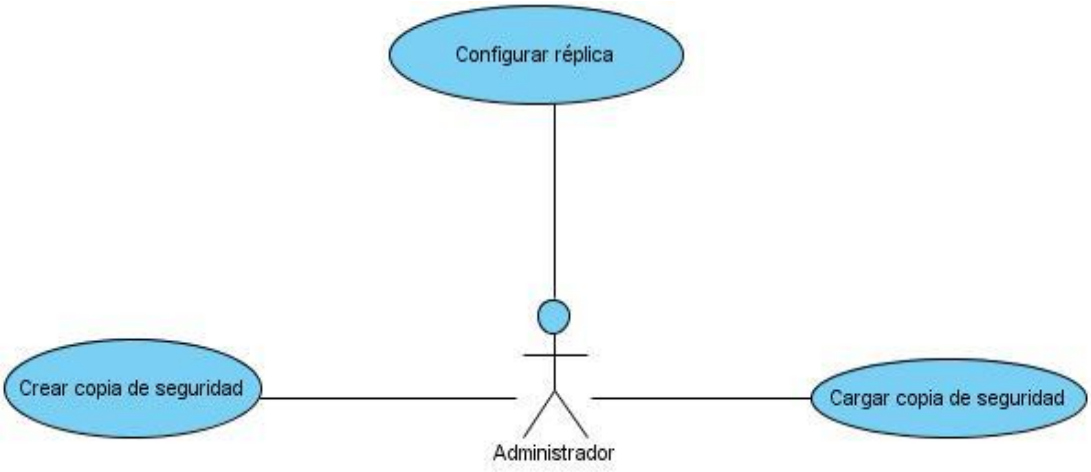
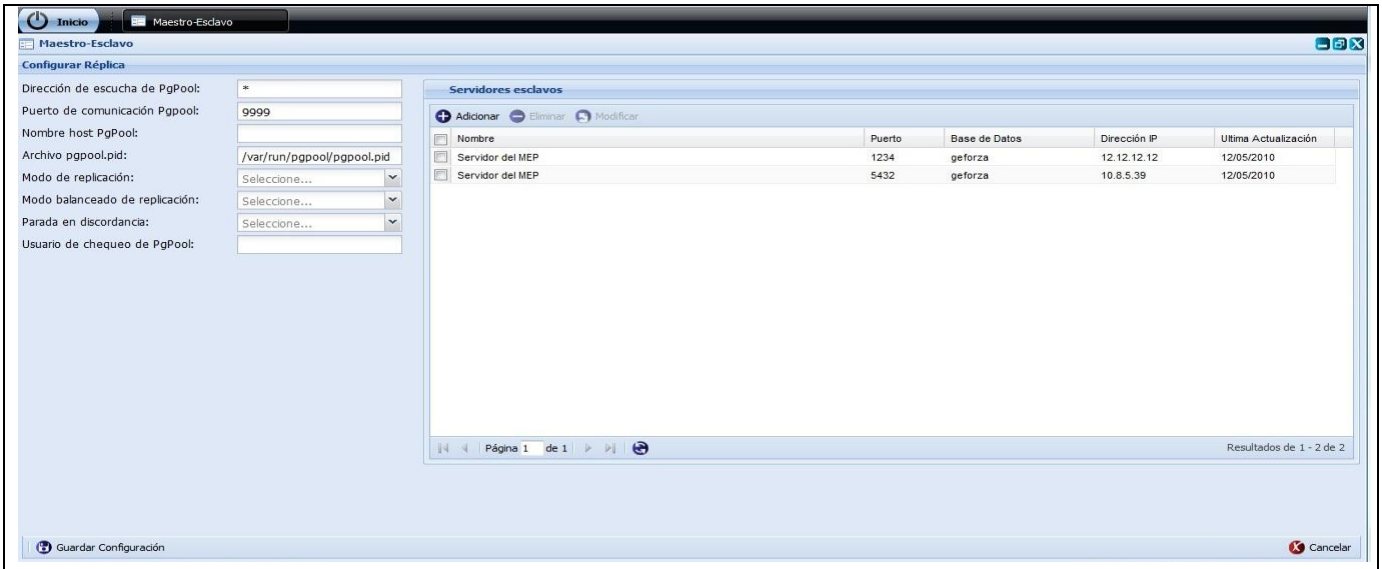


Figura 4 *Diagrama de casos de uso del sistema*

2.7.1 Casos de uso extendidos

Caso de uso configurar réplica

Caso de uso	
CU – 1	Configurar réplica.
Propósito	Que los servidores queden configurados para realizar la réplica de datos.
Actor	Administrador.
Resumen	El caso de uso comienza cuando el administrador da clic en el menú de la interfaz principal en la opción configurar réplica.
Referencias	RF1.
Precondiciones	La FD local debe haber sido configurada.
Interfaz	



Acción del actor	**Respuesta del sistema**
	1- El sistema muestra la configuración de la herramienta PgPool-II y las fuentes de datos.
2- El administrador revisa la configuración del PgPool-II y el listado de las fuentes de datos si están correctos los datos guarda la configuración, en caso de que la configuración del PgPool-II esté incorrecta ver flujo alterno 1, en caso de que la configuración de las fuentes de datos esté incorrecta ver sección Modificar FD, en caso de que no existan fuentes de datos en ver sección Adicionar FD.	3- El sistema está listo para realizar la réplica.
Flujo alterno 1	
1- El administrador introduce los datos para la configuración del PgPool-II y revisa que existan fuentes de datos, en caso contrario sigue los pasos para "Adicionar FD", en caso de que existan fuentes de datos en el listado, revisa su configuración, si está correcta, guarda la configuración, en caso contrario sigue los pasos para "Modificar FD".	2- El sistema está listo para realizar la réplica.

Tabla 5 *Descripción del caso de uso Configurar réplica*

<table>
<tr><th colspan="2">Sección “Adicionar FD”</th></tr>
<tr><td colspan="2">Interfaz
 
</td></tr>
<tr><td>Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td></td><td>1- Muestra la interfaz Adicionar fuente de datos.</td></tr>
<tr><td>2- El administrador entra los datos de la nueva fuente de datos en los campos correspondientes, posteriormente aplica los cambios hechos.

3- El administrador acepta la acción realizada.</td><td>4- El sistema pide al administrador que verifique los datos antes de guardarlo.</td></tr>
<tr><td>5- El administrador presiona “ok” en el mensaje mostrado.</td><td>6- El sistema guarda la nueva fuente de datos y cierra la interfaz.</td></tr>
</table>

Tabla 6 *Descripción de la sección Adicionar FD*

Sección "Modificar FD"	
Interfaz 	
Acción del actor	**Respuesta del sistema**
	1-Muestra la interfaz Modificar FD.
2- El administrador entra los nuevos datos de los servidores en los campos correspondientes, posteriormente da clic en aceptar.	4- El sistema pide al administrador que verifique los datos antes de guardarlo.
5- El administrador presiona "ok" en el mensaje mostrado.	6- El sistema guarda la actualización de la fuente de datos y cierra la interfaz.

Tabla 7 *Descripción de la sección Modificar FD*

Caso de uso Crear copia de seguridad

Caso de uso	
CU – 2	Crear copia de seguridad.
Propósito	Crear una salva de la FD para proteger la información.
Actor	Administrador.
Resumen	El caso de uso comienza cuando el administrador da clic en el menú de la interfaz principal en la opción crear copia de seguridad.

Referencias	RF2.
Precondiciones	Que exista una base de datos para realizarle la copia de seguridad.

Interfaz

Acción del actor	**Respuesta del sistema**
1- El administrador selecciona la opción crear copia de seguridad.	2- El sistema muestra la interfaz para crear la copia de seguridad.
3-El administrador despliega el *combobox* donde salen las bases de datos.	4- El sistema muestra un listado de todas las bases de datos.
5- El administrador selecciona una base de datos para realizar la copia de seguridad. 6- El administrador da clic en crear backup.	7- El sistema brinda la opción de seleccionar el destino donde se guardará la copia de seguridad.
8- El administrador selecciona el destino y guarda la copia de seguridad.	9- El sistema cierra la interfaz para realizar la copia de seguridad.

Tabla 8 *Descripción del caso de uso crear copia de seguridad*

Caso de uso Cargar copia de seguridad

Caso de uso	
CU – 5	Cargar copia de seguridad.
Propósito	Cargar la salva de la FD para restaurar la información de la misma.
Actor	Administrador.
Resumen	El caso de uso comienza cuando el administrador da clic en el menú de la interfaz principal en la opción cargar backup.
Referencias	RF3.
Precondiciones	Que la copia de seguridad haya sido creada.

Interfaz

Acción del actor	Respuesta del sistema
1- El administrador selecciona la opción importar copia de seguridad.	2- El sistema muestra la interfaz para importar la copia de seguridad.
3- El administrador selecciona el origen de la copia de seguridad.	4- El sistema muestra la copia de seguridad de la fuente de datos previamente guardada.
5- El administrador selecciona la copia de seguridad.	7- El sistema muestra un mensaje de

6- El administrador da clic en importar.	confirmación al administrador.
8- El administrador presiona ok	9- El sistema restaura la base de datos. 10- El sistema cierra la interfaz importar copia de seguridad.

Tabla 9 *Descripción del caso de uso Cargar copia de seguridad*

2.8 Conclusiones

Después de la detallada descripción de la propuesta de solución planteada en el presente capítulo se está en condiciones de implementar un sistema que cumpla con las especificidades planteadas a partir de las características expuestas, se precisó realizar un modelo de dominio explicando los principales conceptos involucrados en el negocio, se definieron los requisitos con que debe contar la aplicación tanto funcionales como no funcionales en su totalidad, así como los actores, los casos de uso del sistema y cada una de las funcionalidades dejando el camino preparado para realizar el análisis y diseño de la presente solución.

Capítulo 3: Análisis y Diseño de la Solución Propuesta

3.1 Introducción

En este capítulo a partir de la modelación de las clases del análisis se tratan de comprender de forma clara los requisitos del sistema, además de que se precisará como será implementada la solución propuesta en el anterior capítulo a través de los diagramas de clases del diseño, se realizará también los diagramas interacción.

3.2 Análisis de la solución

Durante el análisis, se analizan los requisitos que fueron descritos en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar todo el sistema, incluyendo su arquitectura.

Aunque en el modelo del análisis hay un refinamiento de los requisitos, no se tiene en cuenta el lenguaje de programación a usar en la construcción, la plataforma en la que se ejecutará la aplicación, los componentes prefabricados o reusables de otras aplicaciones, entre otras características que afectan al sistema, ya que el objetivo del análisis es comprender perfectamente los requisitos del *software* y no precisar cómo se implementará la solución.(33)

3.2.1 Clases del análisis

Representan un modelo conceptual temprano que describe las características y comportamiento comunes de un conjunto de cosas que existen en el sistema. Se expresa que es conceptual pues pospone todos los elementos de diseño ya que no considera posibles tecnologías a emplear en el desarrollo del software; constituyen un prototipo de las futuras clases que darán vida al mismo. Las clases del Análisis están siempre identificadas con uno de los tres estereotipos existentes, los cuales son:

- Interfaz: Se encargan de la modelación de toda la interacción que puede existir entre los actores y el sistema; constituyen las fronteras del mismo.
- Control: Representan la coordinación, secuenciación, transacciones y a veces la lógica del negocio; se emplean a menudo para encapsular el control referido a un CU.
- Entidad: representa la información de larga duración y a menudo persistente que se maneja en el sistema.(34)

3.2.2 Diagrama de clases del análisis

El diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema como parte del mundo real, no desde el punto de vista de la construcción del sistema.(35)

A continuación se presentan los diagramas del análisis:

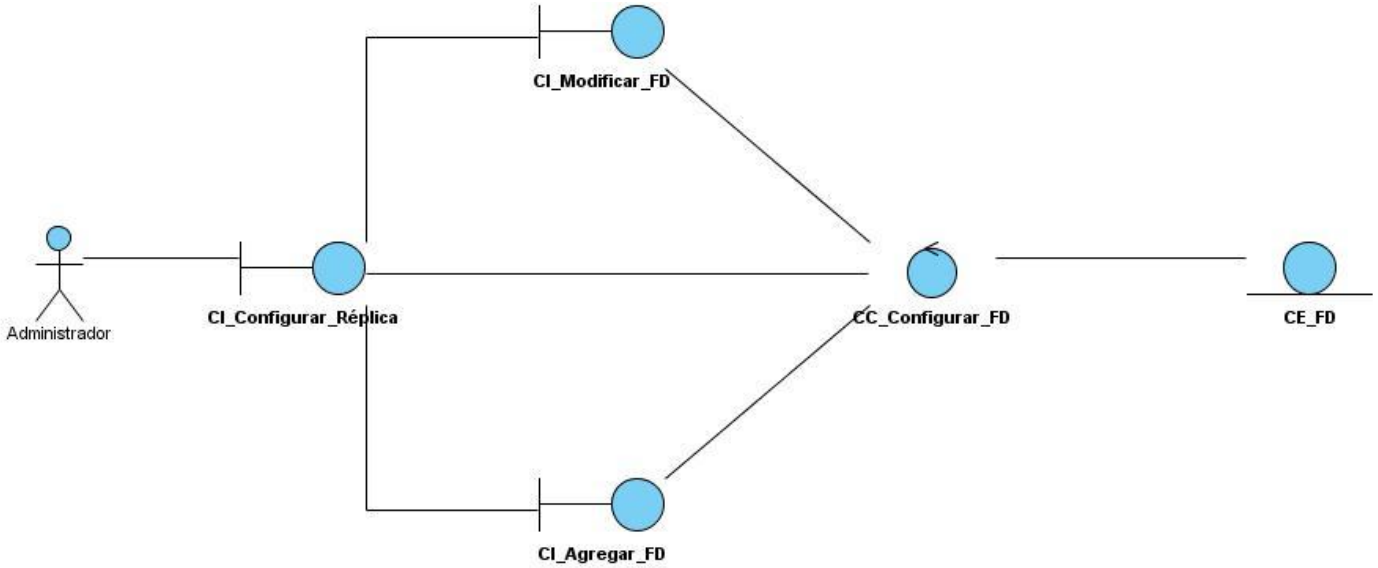


Figura 5 *Diagrama de clases del análisis CU Configurar réplica*

Figura 6 *Diagrama de clases del análisis CU Crear copia de seguridad*

Figura 7 *Diagrama de clases del análisis CU Cargar copia de seguridad*

3.3 Diseño de la solución

El diseño tiene el propósito de formular los modelos que se centran en los requisitos no funcionales y en el dominio de la solución y que prepara para la implementación y prueba del sistema. Pretende crear un plano del modelo de implementación, por lo que el grueso del esfuerzo está en las últimas iteraciones de elaboración y las primeras de construcción.

Es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Esto contribuye a una arquitectura estable y sólida, y crear un plano del modelo de implementación. Durante la fase de construcción, cuando la arquitectura es estable y los requisitos están bien entendidos, el centro de atención se desplaza a la implementación.

Se modela el sistema y se define su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requisitos. Además impone una estructura del sistema que se debe conservar lo más fielmente posible cuando se le dé forma al sistema.

3.3.1 Diagrama de clases del diseño

Un diagrama de clase del diseño es un diagrama estático que describe la estructura de un sistema, mostrando sus clases, operaciones y relaciones entre ellas.

Muestran sus clases participantes, subsistemas y sus relaciones. Se utilizan para coordinar todos los requisitos que diferentes realizaciones de caso de uso imponen a una clase, sus objetos y los subsistemas que contiene.

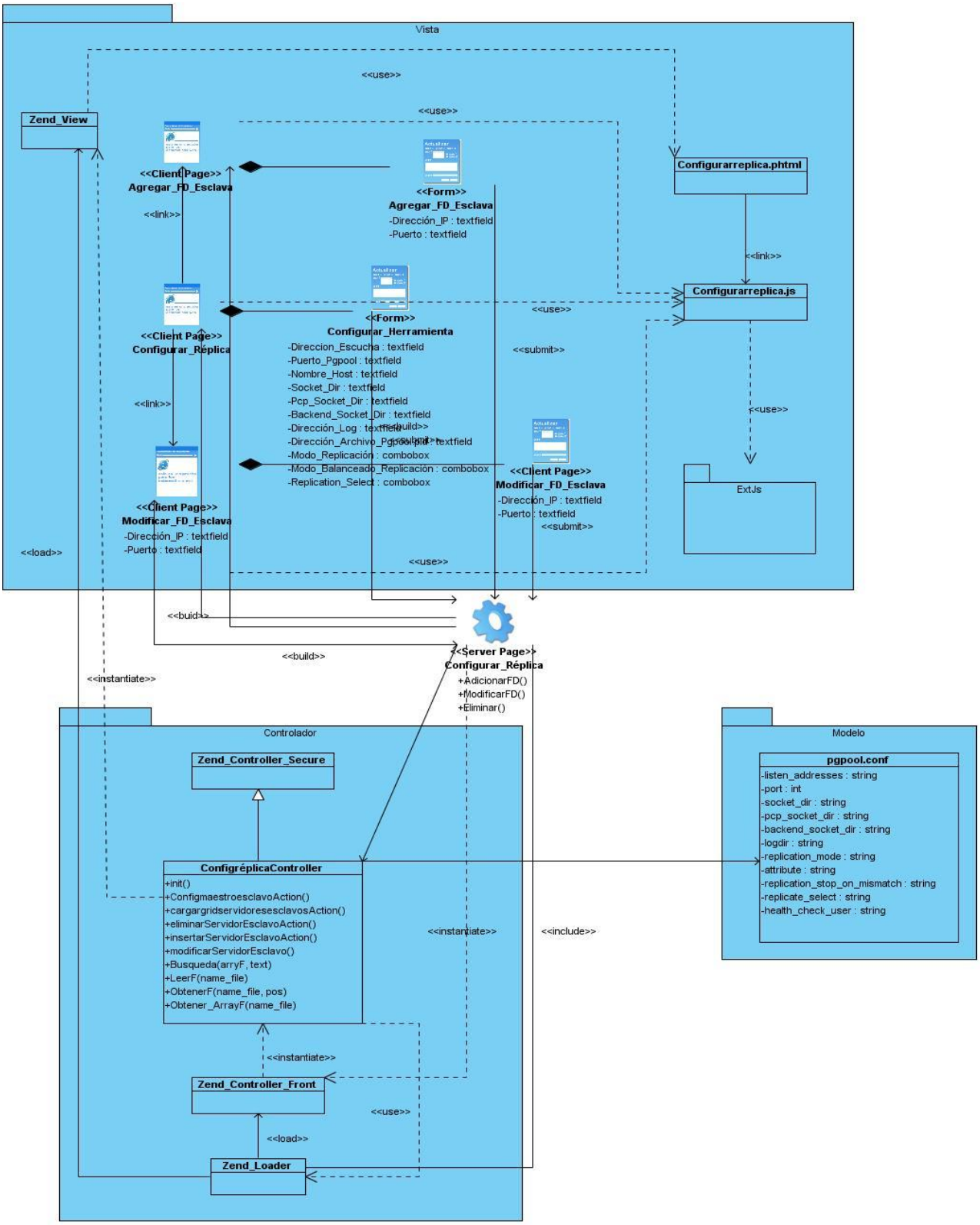


Figura 8 *Diagrama de clases del diseño CU_Configurar réplica*

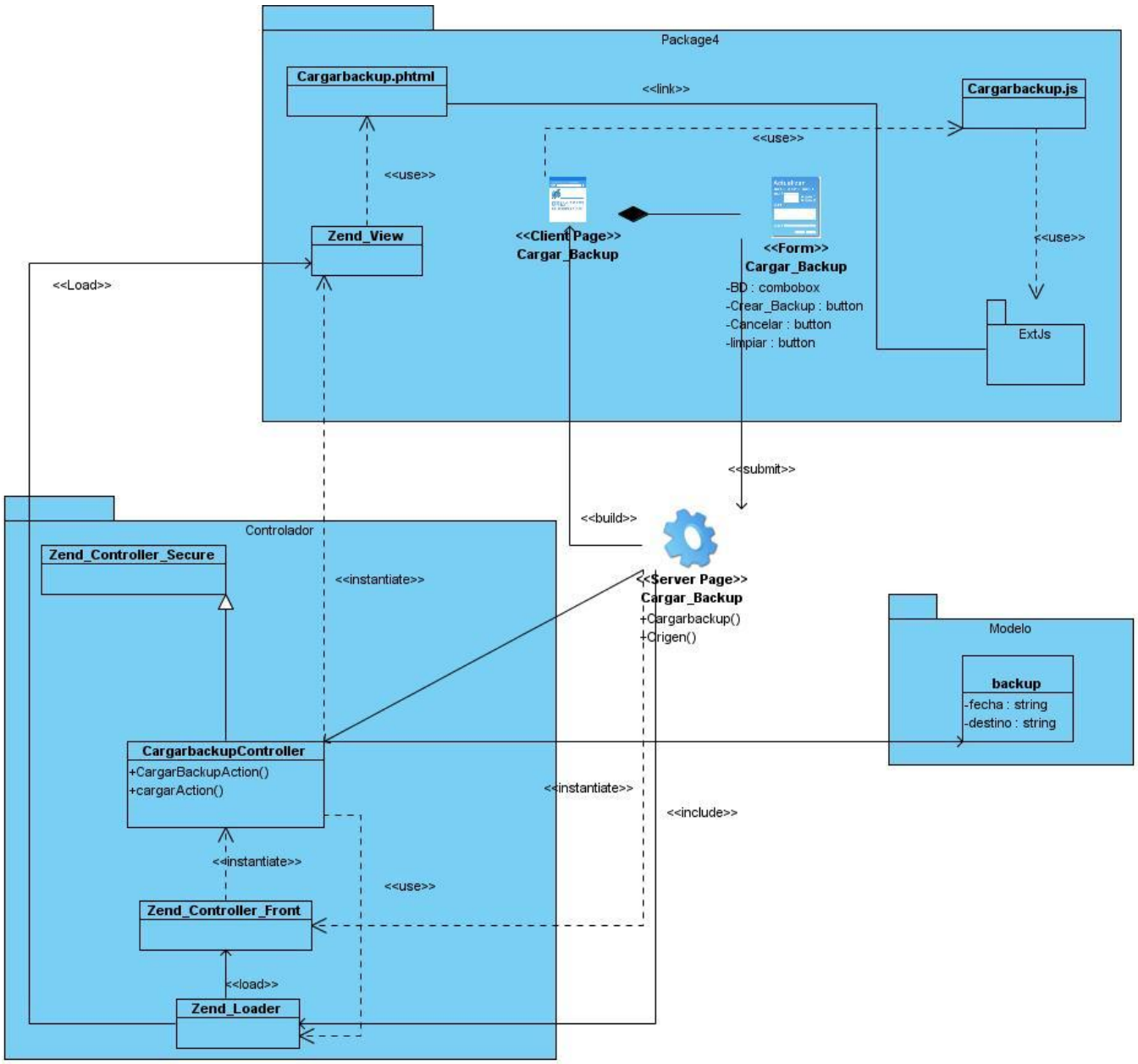


Figura 9 *Diagrama de clases del diseño CU_Cargar copia de seguridad*

3.3.2 Diagrama de interacción

Un diagrama de interacción explica gráficamente como los objetos interactúan a través de mensajes para realizar las tareas, se trata de un término genérico que se aplica a varios tipos de diagramas que hacen hincapié en las interacciones entre objetos. Los diagramas de actividades están íntimamente relacionados. El patrón de interacción entre objetos se muestra en un diagrama de interacción. Los diagramas de interacción tienen diferentes formas, basadas todas ellas en una misma información

subyacente pero resaltando cada una un punto de vista de la misma: diagramas de secuencia, diagramas de colaboración.(33)

3.3.2.1 Diagrama de colaboración

Un diagrama de colaboración modela los objetos y los enlaces significativos dentro de una interacción. Los objetos y los enlaces son significativos solamente en el contexto proporcionado por la interacción. Un rol describe un objeto, y un rol en la asociación describe un enlace dentro de una colaboración. Muestra los roles en la interacción en una disposición geométrica. Los mensajes se muestran como flechas, ligadas a las líneas de la relación, que conectan a los roles. La secuencia de mensajes, se indica con los números secuenciales que preceden a las descripciones del mensaje.

Un uso de un diagrama de colaboración es mostrar la implementación de una operación. La colaboración muestra los parámetros y las variables locales de la operación, así como asociaciones más permanentes. Cuando se implementa el comportamiento, la secuencia de los mensajes corresponde a la estructura de llamadas anidadas y el paso de señales del programa.(34)

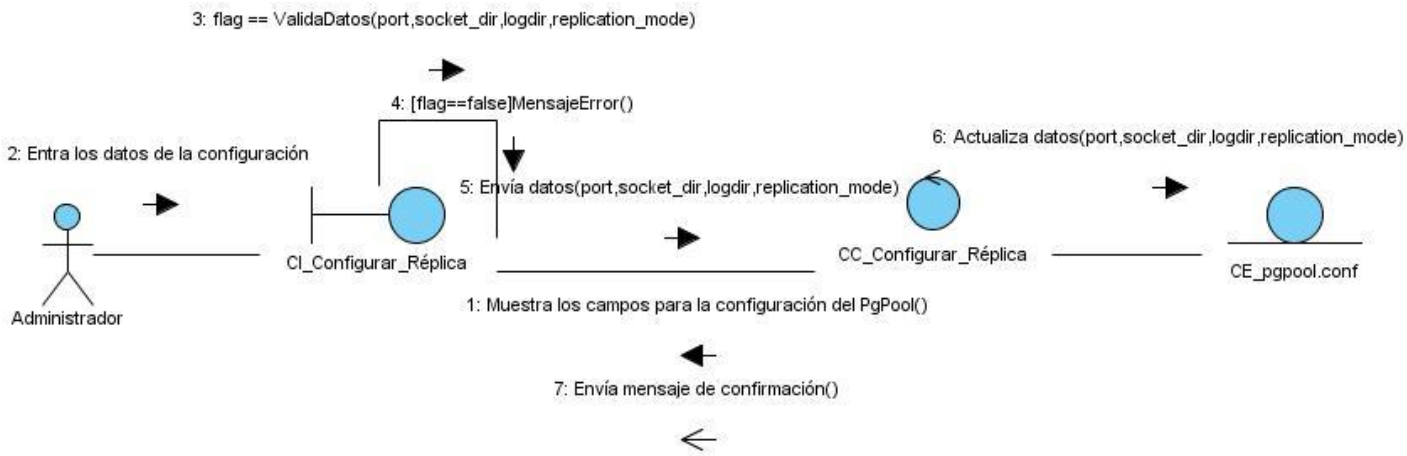


Figura 10 *Diagrama de colaboración CU Configurar réplica (escenario Configurar PgPool).*

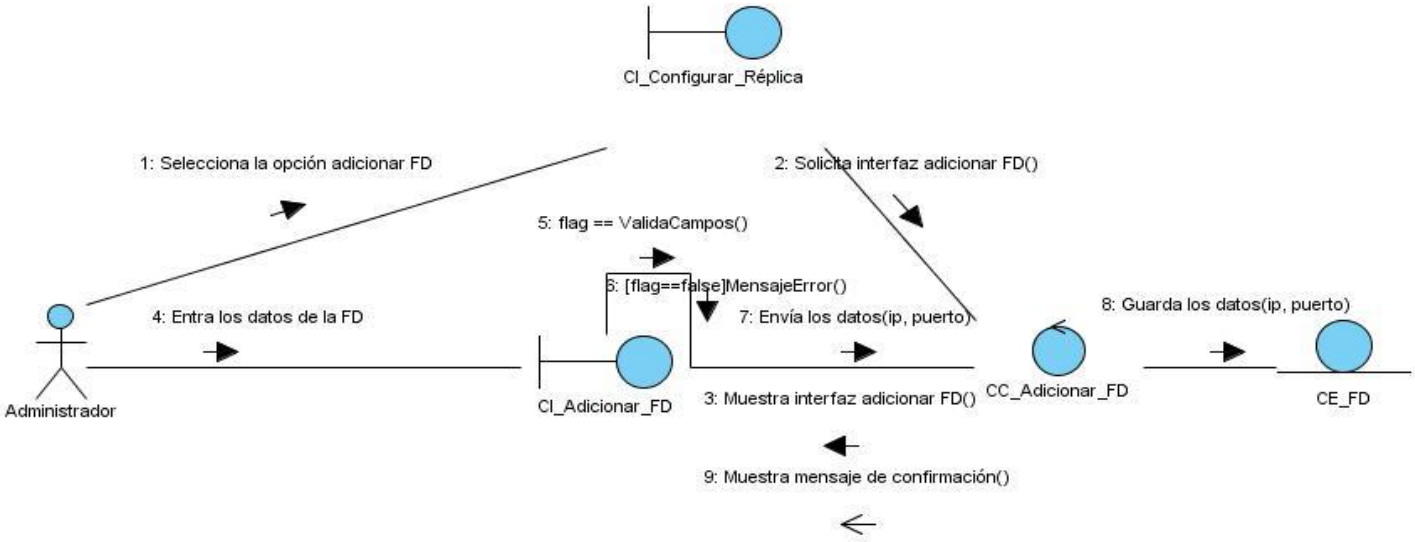


Figura 11 *Diagrama de colaboración CU Configurar réplica (escenario Adicionar FD).*

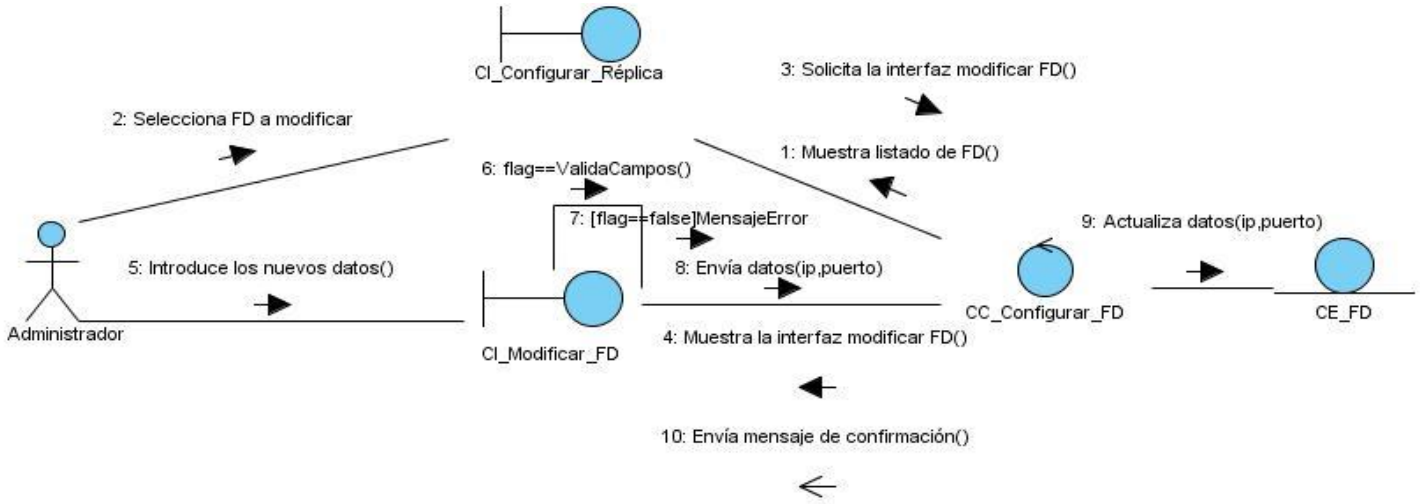


Figura 12 *Diagrama de colaboración CU Configurar réplica (escenario Modificar FD).*

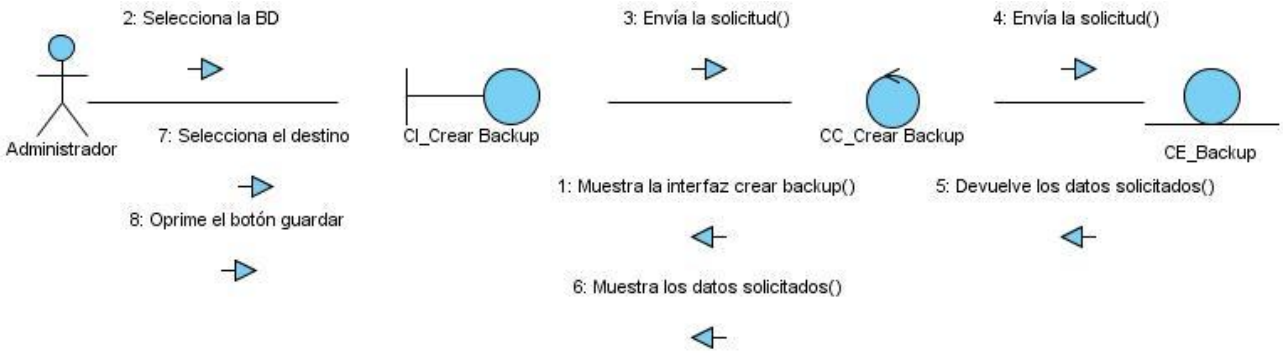


Figura 13 *Diagrama de colaboración CU Crear copia de seguridad.*

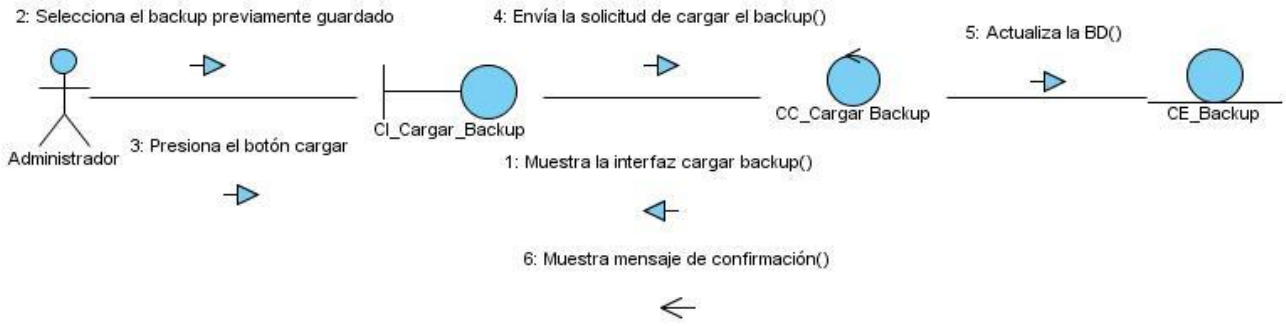


Figura 14 *Diagrama de colaboración CU Cargar copia de seguridad.*

3.3.2.2 Diagramas de secuencia

Un diagrama de secuencia representa una interacción como un gráfico bidimensional. La dimensión vertical es el eje de tiempo, que avanza hacia abajo de la página. La dimensión horizontal muestra los roles de clasificador que representan objetos individuales en la colaboración. Cada rol de clasificador se representa mediante una columna vertical - línea de vida. Durante el tiempo que existe un objeto, el rol se muestra por una línea discontinua. Durante el tiempo que dura una activación de un procedimiento en el objeto, la línea de vida se dibuja como una línea doble.

Se muestra un mensaje como una flecha desde la línea de vida de un objeto a la del otro. Las flechas se organizan en el diagrama en orden cronológico hacia abajo.(34)

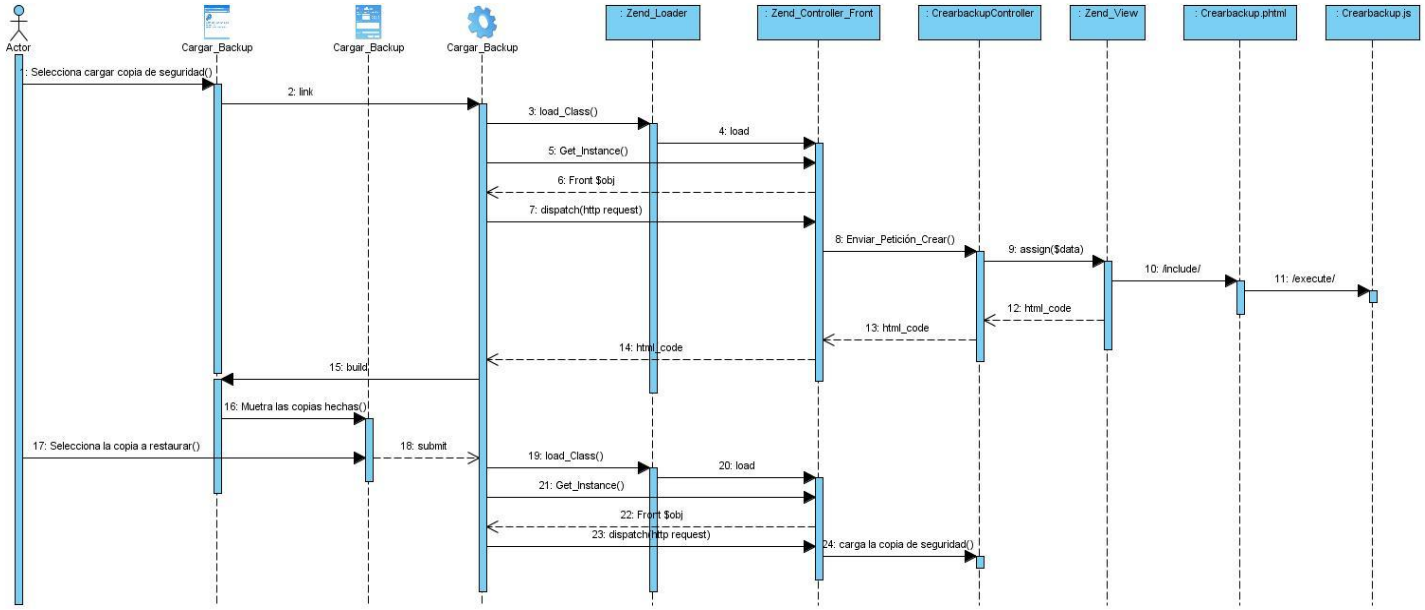


Figura 15 *Diagrama de secuencia CU_Cargar copia de seguridad*

Para más información ver otros diagramas de secuencia en los anexos.

3.3.3 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular, identifica: Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades.

Un patrón de diseño es:

- una solución estándar para un problema común de programación.
- una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.
- un proyecto o estructura de implementación que logra una finalidad determinada.
- un lenguaje de programación de alto nivel.
- una manera más práctica de describir ciertos aspectos de la organización de un programa.
- conexiones entre componentes de programas.
- la forma de un diagrama de objeto o de un modelo de objeto.

Características

- Proponen soluciones a problemas concretos, no son teorías genéricas.
- Indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO).

- En ocasiones tienen más utilidad con algunos lenguajes de programación y en otras son aplicables a cualquier lenguaje.
- Se utilizan en situaciones frecuentes pues se basan en la experiencia acumulada al resolver problemas reiterativos.
- Ayudan a construir software basado en la reutilización (a construir clases reutilizables).
- Los propios patrones se reutilizan cada vez que se vuelven a aplicar.
- Al aplicar un patrón, el código resultante no tiene por qué delatar el patrón o patrones que lo inspiró.
- Al aplicar un patrón aparecen clases concretas que solucionan un problema concreto y que no será aplicable a otros problemas que requieran el mismo patrón.

Ventajas

- Los patrones de diseño proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifica y describe formas de solucionar problemas que ocurren de forma frecuente en el desarrollo.
- Por tanto, están basados en la recopilación del conocimiento de los expertos en desarrollo de software.
- Es una experiencia real, probada y que funciona. Es Historia y ayuda a no cometer los mismos errores.(36)

3.3.3.1 Patrones Grasp

Para el diseño de clases y de los diagramas de interacción se emplearon algunos patrones *Grasp* acrónimo de Patrones de Asignación de Responsabilidad los cuales describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Los patrones *Grasp* utilizados son:

- **Experto:**

Este patrón asigna una responsabilidad al experto en información, es decir, la clase que tiene la información necesaria para cumplir con la responsabilidad. El problema que resuelve el patrón experto está referido al principio más básico mediante el cual las responsabilidades son asignadas en el diseño orientado a objetos.

- **Aplicabilidad:** El experto es usado más que cualquier otro patrón en la asignación de responsabilidades, es un principio usado continuamente en el diseño orientado a objetos. Experto no significa una oscura idea, sino que expresa la "intuición" común mediante el cual los objetos hacen cosas relacionadas con la información que ellos tienen.

Hay que notar que el cumplimiento de una responsabilidad requiere información que está esparcida entre clases diferentes de objetos. Esto implica que hay muchos "expertos " parciales que colaboran en la tarea.

El patrón experto es usado en diseños donde un objeto de *software* hace ciertas operaciones, que también son realizadas en el mundo real por el objeto que representa. Por ejemplo en el mundo real, sin el uso de ayuda electromecánica, una venta no puede decir el total; esto es un concepto inanimado. Alguien calcula el total de la venta. Pero en la tierra del diseño orientado a objetos, todos los objetos de *software* están "vivos" o "animados", y ellos pueden tomar responsabilidades y hacer cosas. Fundamentalmente, ellos hacen cosas relacionadas con la información que ellos conocen. A esto lo llaman algunos el principio de "animación", es como una tira cómica donde todo está "vivo".

El patrón experto como muchas cosas en la tecnología de objetos tiene una analogía con el mundo real. Comúnmente se les da responsabilidades a individuos que tienen la información necesaria para cumplir con la tarea.

Creador:

Permite crear instancias de otras clases en correspondencia con la responsabilidad dada. Con esto se logra conservar el encapsulamiento ya que los objetos logran valerse de su propia información para realizar lo que se les pide.

- **Bajo acoplamiento:**

Este patrón soluciona el inconveniente de dar soporte a una dependencia escasa y a un aumento de la reutilización.

- **Alta cohesión:**

Este patrón es utilizado para mantener la complejidad dentro de los límites manejables.

- **Controlador:** Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación.

3.3.4 Patrones de Arquitectura

Un estilo arquitectónico o variante arquitectónica define a una familia de sistemas informáticos en términos de su organización estructural. Un estilo arquitectónico describe componentes y las relaciones entre ellos con las relaciones de su aplicación.

3.3.4.1 Patrón MVC (Modelo-Vista-Controlador)

Es un patrón de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la lógica de negocio.

Descripción del patrón

Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.

Vista: este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador: este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos. En MVC corresponde al modelo.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

- El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace).
- El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos.

- El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
- El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón observador puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aún así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice.
- La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.(37)

3.4 Tratamiento de excepciones

En cada una de las acciones que ejecuta el sistema se tiene en cuenta el tratamiento de excepciones o errores, lo cual evita que se ejecuten acciones innecesarias propiciándole al usuario un aumento de tiempo útil. El diseño de la interfaz ha estado dirigido a evitar errores, teniendo en cuenta paralelamente la creación de interfaces útiles y amigables. Se ha buscado simplificar la validación de los datos garantizando una validación intrínseca de los mismos, procurando facilitar la corrección de errores lógicos tanto en la introducción de la información como en cualquier otro momento del tratamiento de la misma. La técnica para el manejo de los errores en el sistema se concebirá de manera que cuando ocurra un error se genere una excepción; es decir, la ejecución normal se detenga y se transfiera el control a la zona de tratamiento de excepciones. Las excepciones internas se generan automáticamente por el sistema. Los mensajes de error que emita el sistema ya sea de la base de datos o de la aplicación cliente se captarán y se traducirán a un lenguaje comprensible para el usuario. Los formularios manejan los datos en memoria y solo se actualiza en la base de datos cuando se indique salvarlos.

Estándares en la interfaz de la aplicación

En el sistema, se utilizan las tonalidades suaves y refrescantes. El vocabulario manejado es lo menos técnico posible, acercándose al utilizado por los usuarios. El fondo de las páginas es de color blanco y azul para mayor frescura de la vista. Todo esto se ha hecho con el objetivo de que el uso del sitio brinde comodidad y confort al usuario. Los mensajes de error son pequeños y en Español.

Usabilidad:

La usabilidad se refiere a la facilidad con la cual los usuarios pueden ejecutar las tareas deseadas en una aplicación. Es muy importante efectuar una revisión de usabilidad apropiada antes de comenzar con el desarrollo real. Algunas de las referencias comunes para las revisiones de usabilidad son los estándares de la Organización Internacional para la Estandarización (ISO), ISO 16982 e ISO 9241. Estos estándares se pueden usar como plantilla con un grupo de preguntas que pueden medir la facilidad y la comodidad del usuario al momento de ejecutar diferentes tareas.

Accesibilidad:

La accesibilidad es el grado con el cual la mayor cantidad de usuarios posibles puede acceder a una aplicación. Esto también ayuda a ampliar los negocios y generar más ingresos al alojar la mayor cantidad posible de usuarios. La accesibilidad se puede lograr mediante la construcción de un modelo de usuario que refleje las características del usuario y personalice las interacciones según este modelo

Adaptabilidad:

Adaptabilidad es la habilidad de una aplicación Web para personalizar sus contenidos y servicios con respecto a un usuario en particular. Distintos aspectos de una aplicación Web se pueden adaptar a los usuarios, como por ejemplo la interfaz de usuario, la presentación y la navegación. Es importante asegurarse que se realice la personalización correcta para los usuarios adecuados del sistema. Esto ayuda a los usuarios a ejecutar sus tareas fácilmente, y también mejora el interés del usuario en la aplicación.

3.5 Conclusiones

En el presente capítulo se ha descrito a través de los artefactos que define la metodología de desarrollo Proceso Unificado del Rational (RUP) para la construcción de aplicaciones la solución propuesta. Quedan detalladas todas las clases y relaciones existentes entre ellas, se muestran de forma detallada los diagramas de clases del análisis y diseño correspondientes, además de los

diagramas de interacción (diagramas de colaboración y diagramas de secuencia) y el diagrama de componentes. Fueron descritos los principios del diseño en los que se basa la solución propuesta.

Capítulo 4: Implementación y Pruebas.

4.1 Introducción

En el presente capítulo se presenta como va a ser desarrollada la aplicación, que comienza con los artefactos obtenidos del análisis y diseño y continúa con la construcción del sistema en términos de componentes. Se describen los casos de prueba de caja negra para asegurar la calidad del producto de software y se obtendrán también los diagramas de componentes.

4.2 Implementación de la solución

La implementación se desarrolla en la fase de construcción y tiene como propósito general el desarrollo de la arquitectura y el sistema como un todo.

4.2.1 Componente

Un componente es una parte física y reemplazable de un sistema que se conforma con un conjunto de interfaces y proporciona la realización de dicho conjunto. Se usan para modelar los elementos físicos que pueden hallarse en un nodo por lo que empaquetan elementos como clases, colaboraciones e interfaces.

Algunos estereotipos estándar de componentes son los siguientes:

- Ejecutable: Es un programa que se puede ejecutar en un nodo.
- Biblioteca: Es una biblioteca de objetos estática o dinámica.
- Tabla: Es una tabla de una BD.
- Archivo: Es un fichero que contiene código fuente o datos.
- Documento: Es un documento.
- Página Web: Es una página que se obtiene de la ejecución del sistema.

Los componentes tienen las siguientes características:

- Tienen relaciones de traza con los elementos del modelo que implementan.
- Pueden implementar varios elementos. Por ejemplo, varias clases; Sin embargo la forma exacta en que se crea esta traza depende de cómo van a ser estructurados y modularizados los ficheros de código fuente, dado el lenguaje de programación que se esté usando.

4.2.2 Diagrama de componentes

El diagrama de componentes se representa como un grafo de componentes de software unido por medio de las relaciones de dependencia. Entre los componentes se encuentran las clases, interfaces, librerías y componentes ejecutables. Normalmente los diagramas de componentes se utilizan para modelar código fuente, versiones ejecutables, bases de datos físicas entre otros. El diagrama de componentes muestra un conjunto de ficheros relacionados entre sí para lograr una completa funcionalidad del sistema.

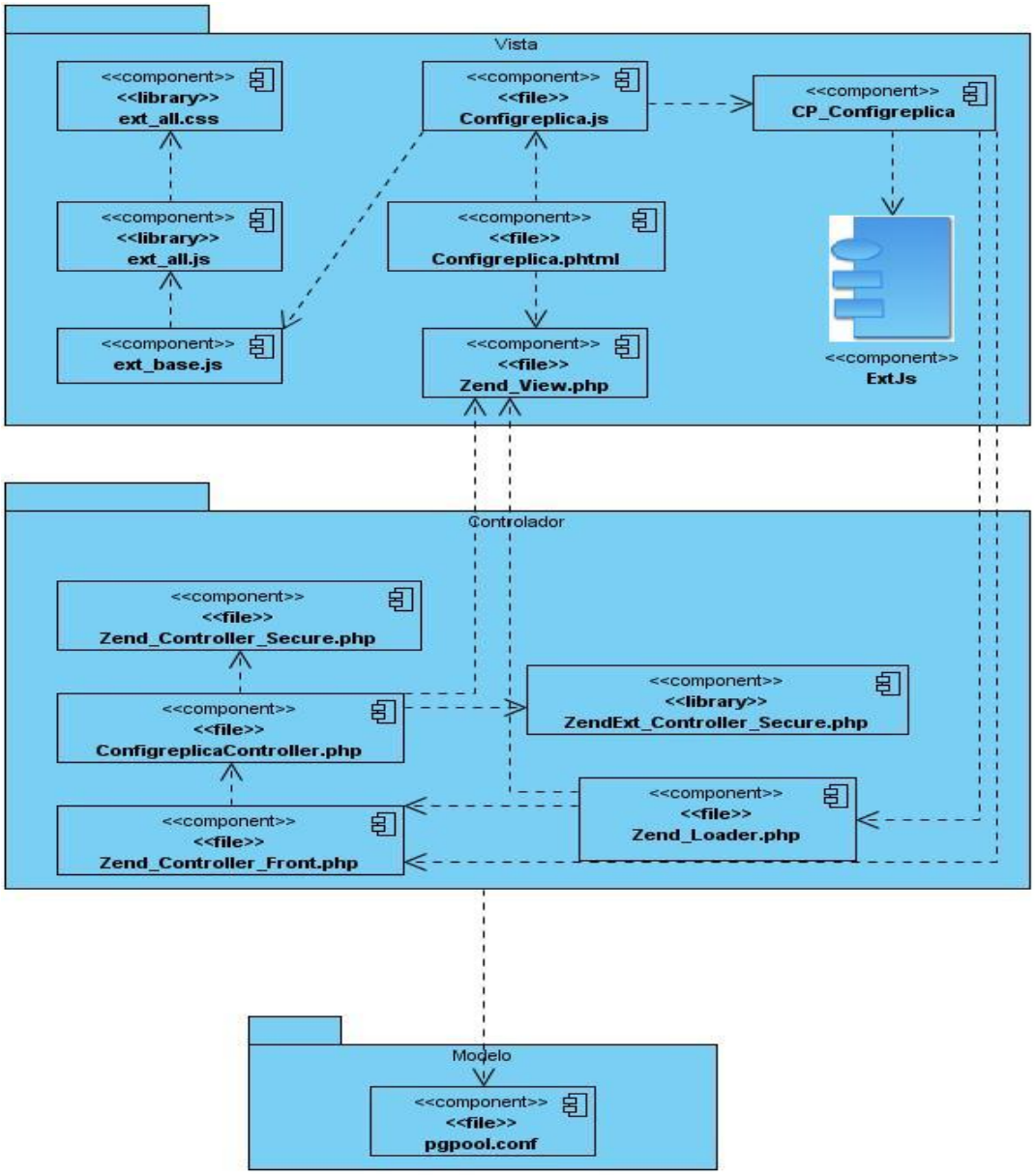


Figura 16 *Diagrama de componentes CU_Configurar réplica*

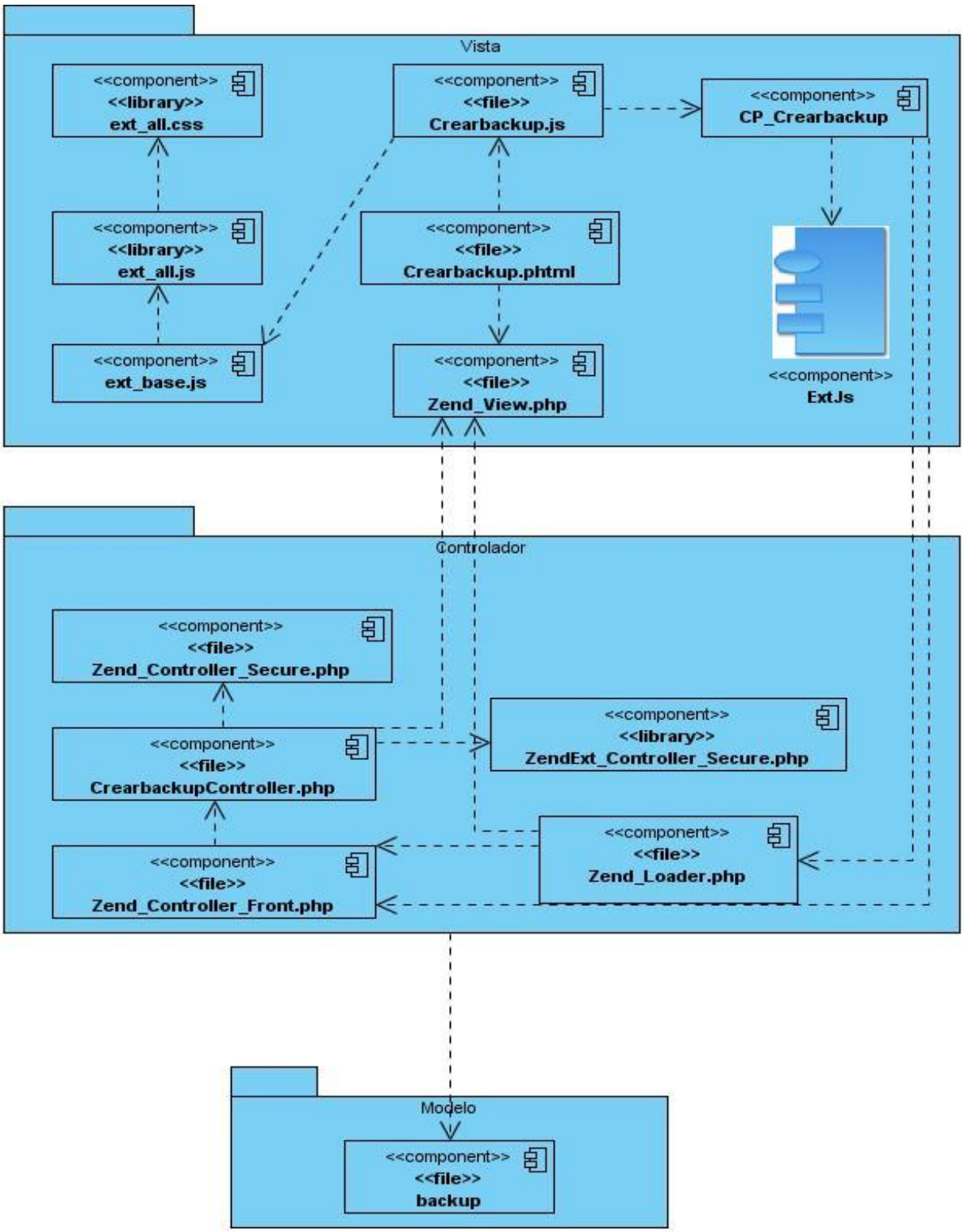


Figura 17 *Diagrama de Componentes CU_Crear Copia de Seguridad*

4.3 Diagrama de despliegue

El diagrama de despliegue muestra la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes de sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de ejecución.

El mismo está compuesto por:

Nodos: Elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos.

Dispositivos: Nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela.

Conectores: Expresan el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.(34)

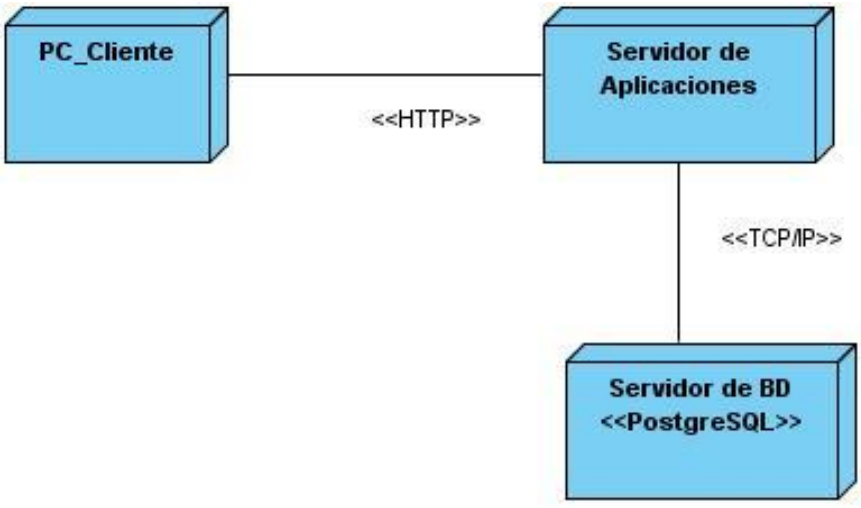


Figura 18 *Diagrama de despliegue*

4.4 Prueba

La prueba es un proceso que está en ejecución a la par del desarrollo de software y se realiza por el convencimiento de que todo el sistema debe ser revisado con el objetivo de obtener el nivel de calidad deseado. Las pruebas deben respaldarse en métricas bien definidas y debe llevarse a cabo de forma

sistemática ya que las mismas generan información valiosa sobre la madurez del producto de software lo cual permite la posterior toma de decisiones.

Cuando se aplican pruebas de software se reduce el costo y tiempo de desarrollo, ya que mediante ellas se detectan fallas en etapas tempranas del proceso.

4.4.1 Métodos de prueba

Existen dos métodos fundamentales: el método de la caja negra y de la caja blanca, 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 y las de caja blanca se comprueba los caminos lógicos del software proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones y/o bucles.

4.4.1.1 Pruebas de caja negra

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que 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 una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del *software*.

Se centran principalmente en los requisitos funcionales del *software*. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.
En este método de prueba se pueden aplicar las técnicas siguientes:

- Técnica de la Partición de Equivalencia: es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el *software*, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar.
- Técnica del Análisis de Valores Límites: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

– Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

4.4.1.2 Pruebas de caja blanca

El objetivo de las pruebas de caja blanca es garantizar que se ejerciten al menos una vez todos los caminos independientes de cada modulo o método, todos los bucles en sus límites operacionales, así como las estructuras internas de datos para asegurar su validez. El proceso de este tipo de prueba se concentra principalmente en validar que cada uno de los módulos o segmentos de código funcione correctamente.

Las pruebas de caja blanca permiten examinar la lógica interna del programa sin considerar los aspectos de rendimiento. Se diseñan casos de prueba para examinar la lógica del programa. Es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar casos de prueba que garanticen que:

- Se ejerciten todos los caminos independientes de cada módulo.
- Se ejerciten todas las decisiones lógicas.
- Se ejecuten todos los bucles.
- Se ejecuten las estructuras de datos internas.

La prueba de Caja Blanca es considerada como uno de los tipos de pruebas más importantes que se le aplican a un software, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

4.4.2 Casos de prueba

Un caso de prueba es un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular ó una función esperada. La entidad más simple que siempre es ejecutada como una unidad, desde el comienzo hasta el final.

Los casos de pruebas deben verificar:

– Si el producto satisface los requerimientos del usuario, tal y como se describe en las especificación de los requerimientos.

– Si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño.

4.4.2.1 Prueba de caja negra aplicando la técnica de particiones equivalentes

Se utilizará el método de caja negra, aplicando la técnica de particiones equivalentes que está definido en la estrategia de pruebas trazada por el proyecto Fuerza de Trabajo Calificada.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1: Configurar la herramienta PgPool-II	EC 1.1: Introducir los valores de la configuración correctamente.	Se guarda la configuracion en el fichero.	• Llena los campos. • Se selecciona la opción “Guardar Configuración ”.
	EC 1.2: Introducir los valores de la configuración dejando campos en blanco.	No se guarda la configuración.	• Llena los campos. • Seleccionar “Guardar Configuración”. • Muestra los campos que quedaron en blanco señalados en rojo.
SC2: Adicionar FD	EC 2.1: Adicionar FD correctamente	Se adiciona la FD	• Se selecciona la opción “Adicionar FD”. • Llena los campos . • Seleccionar “Aplicar”. • Muestra un mensaje “Se ha agregado satisfactoriamente”. • Selecciona “Aceptar”.
	EC 2.2: Adicionar FD con datos incorrectos.	No se adiciona la FD .	• Se selecciona la opción “Adicionar FD”. • Llena los campos . • Seleccionar “Aplicar”. • Muestra un mensaje al usuario que los datos son incorrectos. • Selecciona “Aceptar”.
	EC 2.3: Adicionar FD con campos en blanco.	No se adiciona la FD .	• Se selecciona la opción “Adicionar FD”. • Llena los campos . • Seleccionar “Aplicar”. • Muestra un mensaje al usuario que los datos son incorrectos. • Selecciona “Aceptar”.
	EC 2.4: Cancelar la operación.	No se adiciona la FD .	• Se selecciona la opción “Adicionar”. • Llena los campos. • Selecciona “Cancelar”. • Se cierra la ventana para adicionar.
SC3: Modificar	EC 3.1: Modificar FD correctamente.	Se modifica la FD.	• Se selecciona una de las FD existentes. • Se selecciona la opción “Modificar”. • Llena los campos a modificar. • Seleccionar “Aceptar”. • Muestra un mensaje “Se ha modificado

			satisfactoriamente”.
	EC 3.2: Modificar FD con datos incorrectos.	No se modifica la FD.	• Se selecciona una de las FD existentes. • Se selecciona la opción “Modificar”. • Seleccionar “Aceptar”. • Muestra un mensaje de error a usuario.
	EC 3.3: Modificar FD con campos en blanco.	No se modifica la FD.	• Se selecciona una de las FD existentes. • Se selecciona la opción “Modificar”. • Seleccionar “Aceptar”. • Muestra un mensaje de error a usuario.
	EC 3.4: Cancelar la operación.	No se modifica la FD.	• Se selecciona una de las FD existentes. • Se selecciona la opción “Modificar”. • Seleccionar “Cancelar”. • Se cierra la ventana para modificar.
SC3: Eliminar	EC 3.1: Eliminar FD correctamente.	Se elimina la FD	• Se selecciona una de las FD existentes. • Se selecciona la opción “Eliminar”. • Muestra el mensaje “Seguro que desea eliminar la FD”. • Seleccionar “Aceptar”. • Muestra el mensaje “Se ha eliminado satisfactoriamente”.
	EC 3.2: Cancelar la operación.	No se elimina la FD.	• Se selecciona una de las FD existentes. • Se selecciona la opción “Eliminar”. • Muestra el mensaje “Seguro que desea eliminar la disponibilidad”. • Seleccionar “Cancelar”. • Se cierra la ventana para eliminar.

Tabla 10 *Caso de prueba CU Configurar réplica*

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1: Crear copia de seguridad	EC 1.1: Introducir los valores correctamente.	Se guarda la copia de seguridad.	• Llena los campos. • Se selecciona la opción “Crear copia de seguridad ”. • Se selecciona el destino de dicha copia. • Se guarda exitosamente la copia de seguridad.
	EC 1.2: Introducir los valores dejando campos	No se guarda la copia de seguridad.	• Llena los campos. • Se selecciona la opción “Crear copia de seguridad ”. • Muestra un mensaje de error al usuario.

	en blanco.		
	EC 2.4: Cancelar la operación.	No se guarda la copia de seguridad.	• Llena los campos. • Se selecciona la opción “Crear copia de seguridad ”. • Se selecciona la opción “Cancelar” • Se cierra la interfaz para crear copia de seguridad.

Tabla 11 *Caso de prueba CU Crear copia de seguridad*

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1: Cargar copia de seguridad	EC 1.1: Introducir los valores correctamente.	Se carga la copia de seguridad	• Llena los campos. • Se selecciona el origen de la copia de seguridad. • Se selecciona la opción “Importar”. • Se restaura exitosamente la BD.
	EC 1.2: Introducir los valores dejando campos en blanco.	No se guarda la copia de seguridad.	• Llena los campos. • Se selecciona la opción “Importar”. • Muestra un mensaje de error al usuario.
	EC 1.3: Cancelar la operación.	No se guarda la copia de seguridad.	• Llena los campos. • Se selecciona la opción “Cargar copia de seguridad ”. • Se selecciona la opción “Cancelar” • Se cierra la interfaz para crear copia de seguridad.
	EC 1.4: Cambiar de copia de segruidad.	Se carga la copia de seguridad	• Llena los campos. • Se selecciona el origen de la copia de seguridad. • Se selecciona la opción “Limpiar”. • Se vuelve a seleccionar el origen de la copia de seguridad. • Se selecciona la opción “Importar”. • Se restaura exitosamente la BD.

Tabla 12 *Caso de prueba CU Cargar copia de seguridad*

4.4.2.2 Prueba de rendimiento

En el paquete *postgresql-contrib-8.3* se puede encontrar una utilidad llamada *pgbench*. Esta utilidad permite, en primer lugar, inicializar una base de datos con una serie de tablas sencillas y, en segundo lugar, realizar pruebas de rendimiento sobre servidores PostgreSQL mediante la ejecución de una cierta cantidad de consultas de varios tipos y con una concurrencia parametrizable.

El primer paso consiste en crear la base de datos *bench_replication:*

```
createdb -h pgsql1 -p 9999 -U pgpool2 bench_replication
createlang -h pgsql1 -p 9999 -U pgpool2 -d bench_replication plpgsql
```

Con *log_statement* y *log_connections* activados en */opt/pgpool2/etc/pgpool2.conf*, se mostrarán entradas en */var/log/syslog* similares a las siguientes:

```
LOG: pid 4365: connection received: host=192.168.0.5 port=38024
LOG: pid 4365: statement: CREATE DATABASE bench_replication;
LOG: pid 4365: statement: RESET ALL
LOG: pid 4365: statement: SET SESSION AUTHORIZATION DEFAULT
```

Con *log_statement = 'all'* en */etc/postgresql/8.3/main/postgresql.conf*, en el registro de cualquiera de los PostgreSQL aparecerán las siguientes líneas:

```
LOG: connection received: host=192.168.0.3 port=33690
LOG: connection authorized: user=pgpool2 database=postgres
LOG: statement: CREATE DATABASE bench_replication;
LOG: statement: RESET ALL
LOG: statement: SET SESSION AUTHORIZATION DEFAULT
```

Como usuario postgres, en ambos nodos se podrá usar psql para ver las bases de datos y verificar que se han creado:

```
$ su - postgres
$ psql -l
          List of databases
    Name       | Owner   | Encoding
```

```
------------------+----------+-----------
 bench_replication | pgpool2  | SQL_ASCII
 postgres          | postgres | SQL_ASCII
 template0         | postgres | SQL_ASCII
 template1         | postgres | SQL_ASCII
(4 rows)
```

Se procede a rellenar las bases de datos con tablas e información de pruebas mediante el uso de pgbench:

```
/usr/lib/postgresql/8.3/bin/pgbench -i -h pgsql1 -p 9999 -U pgpool2 -d bench_replication
```

Con un sencillo script se cuenta el número de registros insertados en cada instancia de PostgreSQL sin pasar por PgPool-II, de modo que se puede verificar que la replicación se ha realizado correctamente:

```
#!/bin/sh

PGSQL=/usr/bin/psql
HEAD=/usr/bin/head
TAIL=/usr/bin/tail
CUT=/usr/bin/cut
IP_LIST="192.168.0.3 192.168.0.4"
PORT=5432

for ip in $IP_LIST
do
    echo "ip address: $ip"
    for t in branches tellers accounts history
    do
        echo -n "table $t: "
        COUNT=`$PGSQL -h $ip -p $PORT -U pgpool2 -d bench_replication -c "SELECT count(*) FROM $t" | $HEAD -n 3 | $TAIL -n 1`
            echo $COUNT
    done
done

exit 0
```

Para poder ver cómo se balancean las consultas, teniendo activada la directiva *log_statement = 'all'* en */etc/postgresql/8.3/main/postgresql.conf* de ambos PostgreSQL, se puede utilizar el siguiente script para ver qué consultas aparecen en los registros de cada nodo:

```
#!/bin/sh

PGSQL=/usr/bin/psql
HEAD=/usr/bin/head
TAIL=/usr/bin/tail
CUT=/usr/bin/cut
IP_LIST="192.168.0.3"
PORT=9999

for ip in $IP_LIST
do
    echo "ip address: $ip"
    for t in branches tellers accounts history
    do
        echo -n "table $t: "
        COUNT=`$PGSQL -h $ip -p $PORT -U pgpool2 -d bench_replication -c "SELECT count(*) FROM $t" | $HEAD -n 3 | $TAIL -n 1`
        echo $COUNT
    done
done

exit 0
```

En media, se debería haber ejecutado dos (de las cuatro) consultas *SELECT* en cada una de las bases de datos, si bien esto no tiene porqué ser siempre así.

A continuación se procede a ejecutar el benchmark básico de *pgbench*, de modo que se podrá apreciar el comportamiento del clúster bajo continuas inserciones, actualizaciones y consultas. Desde la consola se ejecutará:

```
/usr/lib/postgresql/8.3/bin/pgbench -h pgsql1 -p 9999 -U pgpool2 -d bench_replication -c 10 -t 1000
```

El resultado que se obtuvo fue el siguiente:

Número Clientes.	10
Transacciones por cliente.	1000
Total de transacciones.	10 000
Procesadas.	10 000 / 10 000
Transacciones ps – 75.094259	Incluyendo establecimiento de la conexión.
Transacciones ps – 75.188971	Excluyendo establecimiento de la conexión.

Si se monitoriza el registro de PgPool-II en */var/log/syslog* y los registros de ambas instancias de PostgreSQL, se notará como en el primer nodo se ejecutan todas las consultas (actualización, selección, modificación, inserción) mientras que en el segundo sólo las inserción y las modificación. Esto se debe a que cada transacción está explícitamente declarada (BEGIN...END) y, en ese caso, PgPool-II no hace uso más que del nodo principal.

4.5 Conclusiones

En este capítulo queda especificada la estructura física del sistema, así como la forma en que se intercambia información con los usuarios, mediante los diagramas de componentes y despliegue. Se identificaron los casos y métodos de prueba con los cuales se midió la calidad del sistema desarrollado, lo cual demuestra que la aplicación cuenta con las características y funcionalidades por la que fue concebida.

Conclusiones

Al término de la investigación se pueden arribar a las siguientes conclusiones:

- Se realizó un extenso estudio sobre la réplica de datos a nivel internacional y en Cuba, lo que posibilitó obtener el conocimiento para la implementación de la solución propuesta. Se fundamentaron las tecnologías usadas para el desarrollo de la aplicación sentando las bases en las ventajas que ofrecían al desarrollo.
- Se realizó el modelo de dominio así como la identificación de los requerimientos del módulo a desarrollar, los casos de uso y los actores que interactuaban con el sistema, quedando plasmados en las descripciones de los casos de uso, permitiendo así el análisis profundo de las funcionalidades que los requisitos deben cumplimentar y que fueron definidas con el cliente así como también se definieron los requisitos no funcionales que definen comportamientos específicos del sistema.
- Se logró realizar el análisis y el diseño del módulo de réplica de datos para “GeForza” sobre tecnologías libres, definiéndose la arquitectura.
- Se describieron los diagramas de componentes y de despliegue a partir de la arquitectura, se logró la implementación de las funcionalidades así como se probaron las mismas a través del método de caja negra aplicando la técnica de particiones equivalentes, se le realizó además una prueba de rendimiento obteniendo resultados satisfactorios.
- Se obtuvo el módulo de réplica de datos para GeForza desarrollado con herramientas libres además de los artefactos generados por la metodología usada que sirven como documentación de la solución, facilitando la realización de posteriores cambios.

Recomendaciones

- Realizar la réplica a nivel de esquemas de BD disminuyendo el tráfico de información por la red.

- Posibilitar la réplica de datos sin importar el sistema gestor de base de datos al cual se le enviará la información.

Referencias bibliográficas

1. *Informatización de la sociedad cubana*. Disponible en: http://www.mic.gov.cu/hinfosoc.aspx.

2. MARTÍN GARCÍA, A. *Propuesta de un modelo para la gestión del alcance en el proyecto "Fuerza de Trabajo Calificada"* [Consultado el: 10 febrero de 2010]. Disponible en: http://www.buenastareas.com/ensayos/Propuesta-De-Un-Modelo-Para-La/81272.html.

3. *Sistemas de gestión de bases de datos*. Disponible en: http://www.um.es/geograf/sigmur/sigpdf/temario_9.pdf

4. ALVAREZ, S. *Sistemas gestores de bases de datos* [Consultado el: enero 15 de 2010]. Disponible en: http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html

5. PÉREZ VALDÉS, D. *¿Qué son las bases de datos?* Disponible en: http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/

6. VALLE PÉREZ, P. *Bases de datos. Concepto, características, funcionalidades* Disponible en: http://www.mailxmail.com/curso-informatica-administracion-publica-3/bases-datos-concepto-caracteristicas-funcionalidades

7. *Diseño y Optimización de Bases de Datos: Bases de Datos Distribuidas* Disponible en: http://cmapspublic.ihmc.us/rid=1161027378031_1484918508_472/12.pdf

8. *Bases de datos distribuidas*. Departamento de computacion CINVESTAV, Disponible en: http://www.cs.cinvestav.mx/SC/prof_personal/adiaz/Disdb/Cap_3.html

9. OLARTE, C. A. *Bases de Datos Distribuidas* Disponible en: http://atlas.puj.edu.co/~caolarte/puj/cursos/cc100/files/clases/BDDistribuidas.pdf

10. *Réplica de datos en sistemas distribuidos*. [Consultado el: 15-01 de 2010]. Disponible en: http://www.worldlingo.com/ma/enwiki/es/Replication_%28computer_science%29

11. REINGART, M. y FRANCO, E. C. *Simple Python-Based PostgreSQL replication system.* [Consultado el: 20 enero de 2010]. Disponible en: http://www.fl.unc.edu.ar/dptoi.blog/wp-content/uploads/2009/11/PyReplica.pdf.

12. *The multimaster and synchronous replication system for PostgreSQL* [Consultado el: 15 enero de 2010]. Disponible en: http://pgcluster.projects.postgresql.org/feature.html

13. PEÑA MONTERO, A. M. *Cluster de base de datos sobre el SGBD PostgreSQL para Almacenes de Datos* [Consultado el: 20 febrero de 2010]. Disponible en: http://postgresql.uci.cu/attachments/101/GESEMP124.pdf.

14. *PgPool-II*. [Consultado el: 18 febrero de 2010]. Disponible en: http://pgpool.projects.postgresql.org/.

15. SABATER, J. *Replicación y alta disponibilidad de PostgreSQL con pgpool-II* [Consultado el: 18 febrero de 2010]. Disponible en: http://linuxsilo.net/articles/postgresql-pgpool.html.

16. *Slony*. [Consultado el: 20 enero de 2010]. Disponible en: http://www.slony.info/.

17. *Introducción a la Ingeniería de Software*. Entorno Virtual de Aprendizaje, Disponible en: http://eva.uci.cu/mod/resource/view.php?id=11402.

18. *RUP*. [Consultado el: 19 febrero de 2010]. Disponible en: http://comunidades.uci.cu/viewtopic.php?f=193&t=1079&highlight=rup.

19. *PHP*. [Consultado el: 20 febrero de 2010]. Disponible en: http://php.uci.cu/?q=node/9.

20. EGUÍLUZ PÉREZ, J. *Introducción a JavaScript.*

21. *Filosofía del software libre*. [Consultado el: 5 febrero de 2010]. Disponible en: http://www.gnu.org/philosophy/free-sw.es.html.

22. *Software libre*. [Consultado el: 15 febrero de 2010]. Disponible en: http://www.softwarelibre.org/.

23. *Visual Paradigm*. [Consultado el: 22 febrero de 2010]. Disponible en: http://www.visual-paradigm.com.

.

24. *Sobre PostgreSQL*. [Consultado el: 19 febrero de 2010]. Disponible en: http://www.postgresql-es.org/sobre_postgresql.

25. *PostgreSQL Documentation*. [Consultado el: 19 febrero de 2010]. Disponible en: http://www.postgresql.org/docs/.

26. *Introducción, características e historia.* Portal en español sobre PostgreSQL, de 2010]. Disponible en: http://www.postgresql-es.org/sobre_postgresql.

27. *ExtJs*. [Consultado el: 20 febrero de 2010]. Disponible en: http://www.extjses.com/.

28. *Manual del Núcleo de Ext*. [Consultado el: 20 enero de 2010]. Disponible en: http://extjsmexico.com/extcore/manual/.

29. *¿Qué es un servidor web?* de 2010]. Disponible en: http://www.misrespuestas.com/que-es-un-servidor-web.html.

30. *Introducción a APACHE*. de 2010]. Disponible en: http://linux.ciberaula.com/articulo/linux_apache_intro/.

31. *Servidor Apache HTTP*. de 2010]. Disponible en: http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-httpd.html.

32. *Fase de Inicio. Disciplina de Requisitos.* . de 2010]. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=22095.

33. *Fase de Elaboración. Flujo de trabajo de Análisis y Diseño.* de 2010]. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=22668.

34. *Disciplina de Análisis y Diseño.* . de 2010]. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=14069.

35. *Fase de Elaboración. Flujo de trabajo de Análisis y Diseño.*
. de 2009]. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=22668.

36. *Arquitectura y patrones de diseño.* de 2010]. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=14075.

37. *Arquitectura y Patrones de diseño.*
. Disponible en: http://eva.uci.cu/mod/resource/view.php?id=14075.

Glosario de términos

API: Una interfaz de programación de aplicaciones o API (del inglés application programming interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos)., 27

Atómicas: Las transacciones son atómicas cuando al ejecutarse una operación esta se ejecuta de comienzo a fin, de llegarse a presentar algún problema deshace toda la operación (todo o nada)., 10

Backends: Es la parte del software que procesa la entrada desde el front-end. La separación del sistema en "front ends" y "back ends" es un tipo de abstracción que ayuda a mantener las diferentes partes del sistema separadas., 15

Backups: Una copia de seguridad o backup en informática es un archivo digital, un conjunto de archivos o la totalidad de los datos considerados lo suficientemente importantes para ser conservados., 30

Caché: Es un conjunto de datos duplicados de otros originales, con la propiedad de que los datos originales son costosos de acceder en tiempo, respecto a la copia en la caché., 24

Clúster: En términos de réplica de datos, un clúster es uno de los servidores de base de datos que interviene en la réplica., 13

Concurrente: Juntarse o coincidir en un mismo lugar o tiempo, diferentes personas, sucesos o cosas, 7

FTP: File Transfer Protocol - Protocolo de Transferencia de Archivos, en informática, es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (Transmission Control Protocol), basado en la arquitectura cliente-servidor., 16

HTML: (Lenguaje de Marcas de Hipertexto, HyperText Markup Language en Inglés). Es el lenguaje de marcado predominante para la construcción de páginas web., 16, 18, 19

Multihilos: Característica que permite a una aplicación realizar varias tareas a la vez concurrentemente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente., 23

Paralelización: En términos de réplica de datos ocurre cuando llegan consultas al mismo tiempo al servidor de BD., 14

Particionar: Dividir o repartir en términos de réplica de datos se le llama particionar a dividir los datos entre los servidores., 11

Postmaster: En servidores web, postmaster es un término utilizado para identificar al administrador de un servidor de correo electrónico., 24

RAM: Es la memoria de acceso aleatorio (en inglés random-access memory cuyo acrónimo es RAM) es la memoria desde donde el procesador recibe las instrucciones y guarda los resultados., 26

Replicación: Es el proceso de duplicar o mas los datos existentes en una base de datos a otra(s). En muchos entornos se usan la réplica por espejo y la réplica por fragmentos de datos., 2, 5, 9, 10, 11, 13, 14, 15, 16, 17, 28, 30, 33

Sincrónico: Se refiere al proceso de propagación de los cambios en los datos y el esquema entre la fuente de datos y los destinos después de haber aplicado la instantánea inicial en el destino., 13

TCP/IP: Es un conjunto de protocolos de red en la que se basa Internet y que permiten la transmisión de datos entre redes de computadoras. En ocasiones se le denomina “conjunto de protocolos TCP/IP”, en referencia a los dos protocolos más importantes que la componen, 16, 24