Universidad de las Ciencias Informáticas
Facultad 15

Título: "Análisis y Diseño de la Herramienta de Administración y Configuración de Chronos."

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores:

Lázaro Lemes Rosales.

Carlos José Batista Villar

Tutores:

Ing. Sergio Hernández Cisneros.

Ing. Mario M. Concepción Milanés.

CIUDAD DE LA HABANA, 2010.

"La sencillez es el mejor ropaje de los pensamientos grandes."

DECLARACIÓN DE AUTORÍA

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

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

Lázaro Lemes Rosales.

_____

Firma del Autor

Carlos José Batista Villar

_____

Firma del Autor

Ing. Sergio Hernández Cisneros

_____

Firma del Tutor

Ing. Mario Michel Concepción Milanés.

_____

Firma del Tutor

DATOS DE CONTACTO

Síntesis de los tutores:

Ing. Sergio Hernández Cisneros (scisneros@uci.cu): Graduado de Ingeniería en Ciencias Informáticas en el año 2009 en la Universidad de las Ciencias Informáticas. Profesor del Departamento de Ciencias Básicas de la Facultad 15.

Ing. Mario Michel Concepción Milanés (**mmconcepcion@uci.cu**)**:** Graduado de Ingeniería en Ciencias Informáticas en el año 2009 en la Universidad de las Ciencias Informáticas. Profesor del Departamento de Ciencias Básicas de la Facultad 15.

AGRADECIMIENTOS

Agradecemos de manera especial a nuestro tutor porque durante el desarrollo de la tesis siempre nos apoyó y creyó en nosotros, incluso cuando nosotros no.

Los autores.

A mis padres, Fidel y Lolita por haber tenido una enorme paciencia.

A todos mis abuelos, Alicia, Dulce y José Luis por su gran cariño.

A mi tío José Luis.

A mis hermanos Chuko y Gabriel por ser ejemplos para mí.

A los ojos que me iluminan, Dayana.

A mis hermanos el Vla, Julito, el Mede y Patato.

A Diolexis, Daniel y a todos los que de una forma u otra aportaron su granito de arena para hacer posible este sueño de Fidel.

Carlos José Batista Villar.

Quiero agradecer primeramente a mi dios, porque siempre escucha mis oraciones.

A mis padres porque siempre han estado cuando los he necesitado dándome aliento y esperanzas para seguir adelante.

A mi hermana por ser una excelente hermana y por tener tanta paciencia.

A mis abuelos que de una forma u otra han contribuido en mi formación.

A mis tíos que también han aportado su granito de arena a través de toda mi vida.

A los compañeros que he tenido a lo largo de estos cinco años de carrera y que me ayudaron y que compartimos momentos buenos y malos dentro de la universidad.

Lázaro Lemes Rosales.

DEDICATORIA

A la memoria de mi abuelo Carlos Villar y a la veterana más linda del mundo Alicia Cruz.
Carlos

A mis padres y a mi hermana porque son los que siempre han estado conmigo.
A la memoria de Manina y Paino porque influyeron en mi como persona y aunque ya no se encuentren físicamente siempre estarán en mi memoria.
Lázaro Lemes.

RESUMEN

La réplica de bases de datos es una de las soluciones más generalizadas para alcanzar los retos que imponen la alta disponibilidad, tolerancia a fallos y escalabilidad en los sistemas de software actuales. La amplia gama de soluciones existentes incluyen sistemas que involucran replicación síncrona y asíncrona, desarrolladas para entornos de réplica multi-maestro y maestro-esclavo. Chronos es una herramienta de réplica asíncrona para entornos multi-maestro que pretende satisfacer la mayor parte de los requerimientos de este tipo de soluciones. Este, como cualquier Sistema de Réplica, necesita una Herramienta de Administración y Configuración. Esta herramienta de administración y configuración pretende tener una interfaz de usuario Web, amigable y fácil de usar, donde la configuración de los distintos sistemas no deba realizarse a través de la consola (Shell del sistema operativo) y que cualquier usuario que no tenga grandes conocimientos de informática pueda interactuar con la misma. Como parte del diseño, la herramienta debe permitir la configuración remota del sistema, así como las actividades de monitorización y administración. Además permitirá una serie de opciones que marcará la diferencia de la Herramienta de Administración de Chronos con otras similares que se utilizan en la actualidad en el mundo.

Palabras Claves: Réplica asíncrona, Bases de Datos, PostgreSQL

TABLA DE CONTENIDO

INTRODUCCIÓN

Hoy en día, en un mundo globalizado, de alta incertidumbre y competitivo, la gestión de la información se convierte en una forma de marcar la diferencia y obtener mayores ventajas respecto al resto de las empresas. En este sentido, simples formatos y registros son calificados como herramientas básicas de recopilación de información, en especial de necesidades de clientes, de quejas, reclamos e incluso de nuevos servicios solicitados. Esto ayuda también a la incorporación de factores de innovación en las entidades corporativas. La explosión de nuevas tecnologías que empezó con la introducción de los ordenadores y la llegada de Internet en los noventa, le ha brindado a las empresas nuevas opciones y herramientas que son explotadas con gran intensidad. Una de ellas es la utilización de instrumentos de información en la generación de bases de datos y estas en la actualidad, ocupan un lugar determinante en cualquier área del quehacer humano, comercial y tecnológico. Es importante tener un dominio sobre las diferentes técnicas, herramientas y metodologías por parte de todo el personal administrativo, ya que esto posibilitaría realizar una acertada gestión de los datos y conducir de esta manera a la estabilidad de cualquier sistema.

Cada día crece vertiginosamente la necesidad de información actualizada, confiable y segura para cualquier empresa o entidad, surgiendo el concepto y utilización de clúster para garantizar estas premisas.

Un cluster de base de datos ofrece un rendimiento mayor para cualquier aplicación que requiera la manipulación de una gran cantidad de datos ó transacciones en breves períodos de tiempo. La función principal de este tipo de dispositivo es la replicación de datos, la cual puede ser asíncrona o síncrona y puede desarrollarse en entornos multi-maestro o maestro-esclavo.

En Cuba a partir de la desintegración del campo socialista y el arreciamiento del bloqueo por parte del gobierno estadounidense, la economía se vio imposibilitada de mantenerse al tanto de los adelantos tecnológicos que en el mundo iban surgiendo. El país en aras de revertir esta situación ha destinado grandes cantidades de recursos de acuerdo a las posibilidades económicas para el desarrollo informático. De este modo, durante los primeros años de este siglo, se ha propuesto informatizar la sociedad, lo que condujo a la creación de nuevos sistemas a la altura de las nuevas tecnologías emergentes.

Al calor de la batalla de ideas, como una idea del Comandante en Jefe Fidel Castro, surgió la Universidad de las Ciencias Informáticas (UCI), para poner en marcha la total explotación de las tecnologías con el objetivo de formar profesionales altamente calificados y contribuir a la informatización de la sociedad.

En la facultad 15, perteneciente a la mencionada universidad, se desarrolla el proyecto Aduana, el cual tiene la misión de informatizar gran cantidad de los procesos que se llevan a cabo en las aduanas de Cuba. Aduana utiliza Oracle como sistema gestor de Bases de Datos ya que permite el almacenamiento y la gestión de grandes volúmenes de datos de manera eficiente, sin embargo, el mismo es software propietario, requiriendo el pago de patentes y licencias para su utilización. Como Cuba es un país bloqueado y del tercer mundo se le imposibilita el pago por este tipo de herramienta, por lo que es de vital importancia la migración hacia un sistema gestor de bases de datos bajo licencias públicas (GPL) como PostgreSQL, debido a las ventajas que ofrece este tipo de licencias:

- Ahorros multimillonarios en la adquisición de licencias de software propietarios como Oracle.
- Eliminación de barreras presupuestales.
- Combate efectivo a la copia ilícita de software.
- Gran cantidad de colaboradores de primera línea dispuestos a ayudar.
- Los tiempos de desarrollo de aplicaciones son menores por la amplia disponibilidad de herramientas y librerías.
- Las necesidades diferentes de los contribuyentes hacen que el software esté adaptado a una cantidad más grande de problemas.
- Contribuyen al beneficio social y la soberanía tecnológica del país.

El proyecto Aduana comprende la replicación de los datos entre nodos y la sincronización de los mismos, además de otros requisitos que hacen que esto sea altamente complicado, pues no existe una herramienta para PostgreSQL que resuelva el problema por si sola. Teniendo en cuenta que la adaptación de éstas a este escenario descrito provocaría mayor utilización de recursos, escasa personalización de la solución y partes incompletas que no podrían resolverse, se quiere desarrollar una herramienta de réplica de bases de datos cuyo nombre es Chronos.

Chronos es una herramienta de réplica asíncrona para entornos multi-maestro que pretende satisfacer la mayor parte de los requerimientos de este tipo de soluciones. Se desea diseñar sus componentes a partir de necesidades particulares de un escenario de

réplica, asumiendo las principales potencialidades de las soluciones similares en la actualidad para PostgreSQL e implementando nuevas mejoras que la distinguen del resto.

Chronos como cualquier Sistema de Réplica necesita una Herramienta de Administración y Configuración, requisito fundamental para la optimización del funcionamiento de dicho software. En la actualidad no existe una herramienta que posea una interfaz amigable y de fácil entendimiento por el usuario; que la configuración de los distintos sistemas no requiera amplios conocimientos de bases de datos y comandos complejos de la consola (Shell del Sistema Operativo). Además de las limitaciones mencionadas también podemos significar la configuración no centralizada, ya que actualmente esta debe realizarse nodo por nodo. Además, no existe una herramienta que realice el tratamiento de errores, ya que al realizarse la configuración a través de comandos en la consola o mediante scripts aumenta la ocurrencia de los mismos.

Ante la situación descrita anteriormente se plantea el siguiente **problema a resolver**: ¿Cómo modelar los procesos de administración y configuración de Chronos en un lenguaje entendible por los desarrolladores, que permita su implementación?

Después de identificar el problema a resolver se plantea como **objeto de estudio**: El Sistema de Replicación de Bases de Datos Chronos, enmarcado en el siguiente **campo de acción**: Las herramientas de configuración y administración de los sistemas de replicación de bases de datos.

Como **objetivo general** se tiene: Modelar el análisis y diseño de la Herramienta de Administración y Configuración de Chronos, derivado en los siguientes **objetivos específicos**:

- Desarrollar el marco teórico de la investigación.
- Identificar los procesos relacionados con la Herramienta de Administración y Configuración de Chronos.
- Definir el Análisis y Diseño de la Herramienta de Administración y Configuración de Chronos.

Tareas a cumplir:

- Realizar un estudio del estado del arte de los procesos de administración y configuración de Sistemas de Réplica existentes.
- Fundamentar el uso de metodologías, tecnologías y herramientas a utilizar.
- Caracterizar la arquitectura propuesta para el sistema.
- Realizar el Análisis y Diseño de la Herramienta de Administración y Configuración de Chronos.

Posibles resultados:

- Apropiarse de información acerca del método utilizado por los distintos Sistemas de Réplica para PostgreSQL sobre Sistema Operativo Linux.
- Obtener una propuesta de solución para la Herramienta de Administración y Configuración de Chronos.

El documento está estructurado en 3 capítulos:

Capítulo 1: Fundamentación Teórica: Se describen algunos conceptos importantes como la réplica de datos, se realiza un estudio de los principales sistemas de réplica existentes para PostgreSQL en el mundo. Se lleva a cabo un estudio de los principales lenguajes, metodologías, herramientas y Frameworks, haciendo al final una fundamentación de los mismos que formarán parte de la propuesta que se hace para llevar a cabo la implementación.

Capítulo 2: Características del Sistema: En dicho capítulo se explican una serie de técnicas que se llevaron a cabo para la captura y validación de los requisitos, además del patrón utilizado para englobar los mismos, se describe el modelo de dominio, se determina el actor encargado de inicializar cada uno de los Casos de Usos del Sistema (CUS), se describen los requisitos funcionales y no funcionales que deberá cumplir el sistema, se realiza el diagrama de casos de uso del sistema, así como una pequeña descripción de los CUS y una descripción ampliada que incluye cada uno de los flujos normales y alternos y por ultimo se hace un pequeño sumario de los aportes y beneficios que la implementación de esta herramienta puede traer consigo.

Capítulo 3: Análisis y Diseño del Sistema: En este capítulo se modelan los diagramas de clases del análisis y del diseño, así como los diagramas de interacción correspondientes. Se hace referencia a la arquitectura y patrones utilizados para la construcción del diseño propuesto. Se muestra el modelo lógico y físico de la base de datos, así como el diagrama de despliegue y por último se aplican algunas métricas para llevar a cabo la validación del diseño.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 INTRODUCCIÓN

En el presente capítulo se abordan los principales conceptos a tratar relacionados con la réplica de datos y se estudia la existencia de sistemas o herramientas empleadas en los procesos relacionados con la misma. Se estudian las tecnologías, los lenguajes de programación, el sistema gestor de base de datos, las herramientas, metodologías y lenguajes con los cuales se realizará la propuesta del sistema.

1.2 PUNTOS IMPORTANTES SOBRE LA RÉPLICA DE DATOS

Cada día crece la importancia de mantener un control y acceso a la información. En su mayoría la misma se encuentra almacenada en bases de datos. Muchos de los sitios Web conocidos mundialmente poseen gran cantidad de información y usuarios accediendo a ésta y modificándola, por lo que teniendo el control de estos datos en un solo nodo de base de datos (BD) se hace casi imposible proveer la información disponible a todas las personas que acceden simultáneamente. Muchas de las compañías no resguardan la información en una sola BD, sino que utilizan la técnica de replicación de datos, teniendo varios nodos que dividen la carga de operaciones, comparten y actualizan la información entre ellos posibilitando que los datos no se pierdan ante cualquier fallo.

La replicación de datos es el envío de información entre dos o más nodos, permitiendo que ciertos datos se almacenen en más de un sitio, aumentando la disponibilidad de los mismos y mejorando la eficiencia del acceso a dicha información.

La réplica es una de las soluciones más generalizadas para alcanzar los retos que imponen la alta disponibilidad, tolerancia a fallos y escalabilidad en los sistemas de software actuales. La amplia gama de soluciones existentes incluye sistemas que involucran replicación síncrona y asíncrona desarrolladas para entornos de réplica multi-maestro y maestro-esclavo.

La replicación síncrona permite la réplica instantánea de los datos introducidos en un nodo del sistema. La misma se confirma una vez estén estos datos actualizados en los demás nodos. Cada transacción solamente es aceptada si todos los sistemas implicados en la réplica están conectados y listos para

recibirla, si al menos uno falla, todo el proceso es anulado. La réplica síncrona asegura la consistencia de datos en todos los sitios en tiempo real.

La replicación asíncrona, permite la réplica de los datos introducidos en un nodo del sistema con un desfasaje de tiempo (hay un período de tiempo antes de que todos los sitios alcancen la convergencia de datos) sujeto a las reglas existentes. La misma se confirma una vez introducidos los datos en uno de los nodos y luego se replica en el resto.

En un entorno maestro-esclavo los datos se ingresan y modifican en un nodo (maestro o "master") y se replican al resto de los nodos (esclavos o "slaves") donde solo pueden ser leídos. En un entorno multi-maestro los datos se ingresan, leen y modifican en cualquier nodo del sistema.

Ventajas:

La réplica es una técnica que permite copiar y distribuir idénticamente las tablas de una base de datos en múltiples bases de datos ubicadas en diferentes nodos de la red. La replicación asegura que los datos correctos estén siempre disponibles en el momento y en el lugar necesario. Posee las siguientes características:

Alta Disponibilidad: La información se encuentra en varios servidores, lo que implica que si por algún fallo uno de estos servidores no está activo la información continua estando disponible.

Tolerancia a fallos: Garantiza un comportamiento correcto, donde efectivamente pueden existir un número finito de fallos y tipos de fallos, sin afectar el estado y la disponibilidad de la información.

Rendimiento: Provee un rápido acceso local a datos compartidos ya que balancea la actividad sobre múltiples sitios. Algunos usuarios pueden acceder a un servidor mientras que otros acceden a diferentes servidores, reduciendo la carga de todos los servidores. También, los usuarios pueden acceder a datos de un sitio replicado que tiene menos costo de acceso, típicamente un sitio que se encuentra geográficamente más cerca de ellos.

Reducción de la carga: Puede ser utilizada para distribuir datos sobre lugares en múltiples regiones. Así, las aplicaciones pueden acceder a varios servidores regionales en vez de acceder a un servidor central. Esta configuración puede reducir la carga de la red.

Reducción de costo: Es menos costoso acceder a un servidor local o geográficamente más cercano (o varios en algunos casos), que a uno que se encuentre demasiado lejos.

1.3 ESTUDIO DE SISTEMAS DE RÉPLICA

Existen en el mundo múltiples soluciones que permiten llevar a cabo la réplica de datos enfocadas al gestor de base de datos PostgreSQL. Estas aplicaciones se dividen de acuerdo a los entornos de réplica donde pueden ser aplicadas, para el entorno “maestro-esclavo” la más popular es:

1.3.1 SLONY I

Slony es el plural de elefante en el idioma ruso. La mascota para Slony, es una muy buena variación del usual elefante mascota de PostgreSQL. Es un sistema de replicación “maestro a múltiples esclavos”, soporta la réplica en cascada y permite a un esclavo ser a su vez maestro para otro servidor; que a pesar de ser maestro también, es de solo lectura para los usuarios finales, o sea que solo el sistema de réplica puede escribir en él. Incluye los tipos de funcionalidades necesarias para replicar bases de datos grandes a un número razonablemente limitado de sistemas esclavos. “Razonable,” en este contexto, un orden de una docena de servidores. Si el número de servidores crece más allá de ese, el coste de comunicaciones aumentará prohibitivamente y las ventajas incrementales de tener servidores múltiples fallarán en ese punto. Además permite indicar qué cambios replicar de un servidor a otro. Slony-I implementa la réplica asíncrona, usando disparadores para determinar las actualizaciones de las tablas, donde un solo “origen” (maestro) se puede replegar a los suscriptores múltiples” (esclavos) incluyendo suscriptores conectados en cascada. Slony I realiza una réplica de espejos, exactamente igual al origen de datos, no es posible actualizar los datos a medida que se produce algún cambio en ellos.

La instalación, administración y configuración es muy compleja pues se debe realizar nodo por nodo mediante scripts y comandos, lo que implica que estas operaciones solo puedan ser realizadas por usuarios avanzados. (1)

1.3.2 PgCLUSTER

Es un sistema de replicación síncrona multi-maestro para PostgreSQL. PgCLUSTER consiste en tres tipos de servidores, un servidor para balance de carga, un cluster BD y un servidor de réplica. Tiene dos funciones principales:

Compartir carga: La carga de la sesión de las demandas es distribuida. Es efectivo en aplicaciones Web donde existe gran demanda por el número de peticiones.

Alta disponibilidad: Cuando ocurre un fallo en el cluster BD, el servidor de balance de carga y el de replicación separan el modo que falló del sistema y continúa el servicio que usa el cluster restante. El cluster BD cuando es reparado puede restaurarse dinámicamente a un sistema, sin detener el servicio. Los datos son copiados automáticamente a la BD restaurada o añadidos desde otra BD. Este último con objetivos definidos para soportar una alta disponibilidad, para controlar los fallos de los servidores, es un sistema propietario además. (2)

1.3.3 PgPOOL II

PgPOOL-II es un programa de balance de carga y réplica síncrona que trabaja entre los servidores de datos y el cliente transparentemente (una aplicación de base de datos cree que PgPOOL-II es el verdadero servidor de PostgreSQL y el servidor ve a PgPOOL-II como uno de sus clientes). Debido a esto una aplicación de base de datos existente puede empezar a usarse con PgPOOL-II casi sin ningún cambio en su código fuente.

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

Proporciona las siguientes características:

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

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

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

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

Paralelización de consultas: Al usar la función de paralelización de consultas, los datos pueden dividirse entre varios servidores, de modo que la consulta puede ejecutarse en todos los servidores de manera concurrente para reducir el tiempo total de ejecución. La paralelización de consultas es una solución adecuada para búsquedas de datos a gran escala.

La configuración de este software es compleja, pues es necesario utilizar scripts y comandos específicos del sistema lo que resulta tedioso. Al fallar un nodo del sistema, este se separa continuando el servicio sin interrupción, luego de restablecerse la conexión la BD se sincroniza con el resto de los nodos. PgPOOL-II brinda la opción de ejecutar un comando determinado por el usuario en el nodo restablecido.

1.4 TENDENCIAS Y DESARROLLO DEL SOFTWARE LIBRE

El software libre es un movimiento tecnológico que ha revolucionado la sociedad. Presenta características especiales que han permitido la experimentación de nuevas formas de desarrollo y mantenimiento de programas, nuevos modelos económicos y nuevas normas legales. Es un asunto de libertad no de precio. Para entender el concepto se debe pensar en libre como en libertad de expresión.

Software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De modo más preciso se refiere a cuatro libertades de los usuarios del software: La libertad de usar el programa con cualquier propósito. La libertad de estudiar cómo funciona el programa y adaptarlo a las necesidades específicas. La libertad de distribuir copias. La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie (3).

Esto da la medida de la viabilidad económica de este sistema libre, que por ser libre no es necesariamente gratuito, sino que da la posibilidad de comercializarlo, distribuirlo, prestarlo con total libertad y protegerlo legalmente. Evidentemente es la alternativa para los países subdesarrollados.

Ventajas:

- **Costo:** el costo total de propiedad del sistema operativo libre Linux es menos de la mitad que el de Windows. Gran parte del ahorro proviene de no tener que pagar licencia y de sus menores costos de administración.
- **Innovación tecnológica:** el desarrollo en comunidad de este sistema y el conocimiento del código fuente, propician que a cada instante, un desarrollador necesite nuevas actualizaciones y las realice él mismo, proponiendo nueva funcionalidad al programa.
- **Escrutinio público:** el proceso de revisión pública al que está sometido el desarrollo del software libre imprime un gran dinamismo al proceso de corrección de errores. Cada mejora es socializada libremente, la comunidad puede cambiar la realidad de las innovaciones.
- **Independencia del proveedor:** el Software Libre garantiza una independencia con respecto al proveedor gracias a la disponibilidad del código fuente.
- **Desarrollo de la industria local:** en el Software Libre al disponer del código fuente de la aplicación es posible desarrollar internamente las mejoras o las modificaciones necesarias. De este modo se contribuye a la formación de profesionales en nuevas tecnologías, al desarrollo local y de la industria nacional de software.
- **Privacidad y seguridad:** el Software Libre por su carácter abierto dificulta la introducción de código malicioso, espía o de control remoto, debido a que el código lo revisan muchos usuarios y desarrolladores que pueden detectar posibles puertas traseras. En el mundo del software libre cualquier programador puede realizar una auditoría para comprobar que no se ha introducido ningún código malicioso y a su vez, cualquier entidad puede añadir libremente encriptación adicional a la aplicación que utilice para proteger sus datos.

Desventajas:

- **La curva de aprendizaje es mayor:** No es fácil aprender a trabajar con el software libre si ya se ha usado un software propietario. Dos personas que nunca han tocado una computadora posiblemente tarden lo mismo en aprender a utilizar el software libre, que el software propietario, pero si ya ha usado un software propietario, generalmente se tarda más en aprender a usar el software libre.

Se necesita dedicar recursos a la reparación de errores: El software libre se adquiere sin garantías explícitas, aunque pueden existir garantías específicas para determinadas situaciones, por lo que hay que

dedicar recursos para reparar cualquier daño del software. Por otra parte, en el software propietario es imposible reparar errores, el usuario debe obtener una nueva versión.

En Cuba se ha desencadenado un auge nacional del software libre, principalmente en los órganos legislativos del estado y gobierno para garantizar la seguridad y la integridad de la información confidencial. El país se encuentra enfrascado en realizar un crecimiento en la industria del software que da sus primeros pasos y cuya única alternativa sostenible para su avance es la migración al software libre, dándole un impulso gigantesco a esta corriente de desarrollo.

Existen ejemplos como el de la Universidad de las Ciencias Informáticas (UCI), que acertadamente ha dedicado esfuerzos al desarrollo de las tecnologías libres y hoy es pilar en este campo con un trabajo ascendente al contar con una distribución de GNU/Linux orientada a Escritorio llamada Nova que responde a las necesidades de las instituciones cubanas.

Hoy el desarrollo de software con la utilización de las tecnologías libres hará que Cuba sea respetada y obtenga como valores fundamentales el logro de un estado de soberanía e independencia tecnológica. (4)

1.5 LENGUAJE DE PROGRAMACIÓN Y TECNOLOGÍAS DEL LADO DEL CLIENTE

1.5.1 HTML

HTML, siglas de HyperText Markup Language (Lenguaje de Marcas de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas Web (5). Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de etiquetas, rodeadas por corchetes angulares. HTML también puede describir, hasta un cierto punto, la apariencia de un documento y puede incluir un script (por ejemplo Java Script), el cual puede afectar el comportamiento de navegadores Web y otros procesadores de HTML.

Ventajas:

- **Información por hipertexto:** Diversos elementos (texto o imágenes) de la información que se muestra en la pantalla están vinculados con otras informaciones que pueden ser de otras fuentes. Para mostrar en pantalla esta otra información bastará con hacer clic sobre ellos.
- **Gráfico:** En la pantalla aparece simultáneamente texto, imágenes e incluso sonidos.

- **Global:** Se puede acceder a él desde cualquier tipo de plataforma, usando cualquier navegador y desde cualquier parte del mundo.
- **Pública:** Toda su información está distribuida en miles de ordenadores que ofrecen su espacio para almacenarla. Toda esta información es pública y toda puede ser obtenida por el usuario.
- **Dinámica:** La información, aunque está almacenada, puede ser actualizada por el que la publicó sin que el usuario deba actualizar su soporte técnico.
- **Independiente:** Dada la inmensa cantidad de fuentes, es independiente y libre.

Desventajas:

- No tiene semántica. Uso de etiquetas con nombres diferentes.
- El contenido no puede ser reconocido ni procesado por programas
- Tiene un costoso mantenimiento de las páginas
- No tiene estándares comunes.
- Solo tiene hiperenlaces simples
- Los programas de texto son poco amigables y tienen una interfaz restringida.

1.5.2 JAVASCRIPT

Fue creado en la empresa Netscape Communications, específicamente para su uso en el desarrollo de sitios Web. Es un lenguaje interpretado ya que no necesita de ningún compilador, solamente de un navegador, esta orientado a objetos de forma limitada. Es un lenguaje de script multiplataforma [cross-platform]. Es un lenguaje pequeño y ligero; no es útil como un lenguaje independiente, más bien está diseñado para una fácil incrustación en otros productos y aplicaciones, tales como los navegadores Web. Dentro de un entorno anfitrión, Javascript puede ser conectado a los objetos de su entorno para proveer un control programable sobre éstos (6).

Ventajas:

- El lenguaje de scripting es seguro y fiable.
- El código es visible y puede ser leído por cualquiera, incluso si está protegido con las leyes del copyright.
- El código Java Script se ejecuta en el cliente por lo que el servidor no es solicitado más de lo debido.
- Es soportado por la mayoría de los navegadores Web.

- Es multi-plataforma.

Desventajas:

- Los script tienen capacidades limitadas, por razones de seguridad.
- El código del script debe descargarse completamente antes de poderse ejecutar.

1.6 LENGUAJE DE PROGRAMACIÓN DEL LADO DEL SERVIDOR

Existe una multitud de lenguajes concebidos o no para Internet. Cada uno de ellos explota más a fondo ciertas características que lo hacen más o menos útiles para los desarrolladores de software en la actualidad y para todas aquellas personas que de una forma u otra se benefician o utilizan los mismos. En el dominio de la red, algunos de los lenguajes del lado del servidor más ampliamente utilizados para el desarrollo de páginas dinámicas son el PHP y JSP (7).

1.6.1 JSP

"*JSP (Java Server Pages) es una tecnología Web, del lado del servidor, que se usa generalmente para generar documentos XHTML y XML dinámicos. JSP es un producto de la compañía Sun Microsystems, su funcionamiento se basa en scripts, con una sintaxis similar a la de Java.*

JSP es una tecnología similar a PHP, ASP y demás. Y permite incrustar código JSP dentro del HTML, para crear información dinámicamente (basándose en operaciones o acceso a bases de datos, por ejemplo). El código JSP se incrusta en el HTML dentro de las marcas, a esto se le llama scriptled" (8).

Para correr un programa en JSP es necesario tener instalado un servidor que soporte dicha tecnología.

Ventajas:

- Te permite programar siguiendo un patrón de diseño y unas técnicas que forman parte del legado cultural de los programadores y analistas.
- El punto anterior hace que las aplicaciones sean más mantenibles, ya que no tienes por qué aprender cómo lo hizo el programador, simplemente debes conocer las técnicas y los patrones de diseño.
- Las aplicaciones hechas en un Framework en Java son por lo general más robustas.

Desventajas:

- El hosting en Java es más caro.

- Se necesita tiempo de estudio, porque no es sencillo, ni tan fácil de aprender como otros lenguajes.

1.6.2 PHP

El PHP (acrónimo de PHP: Hipertext Preprocessor), es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor."*Es un lenguaje de script incrustado dentro del HTML. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características especificas de si mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas". "Con PHP se puede hacer cualquier cosa que podamos realizar con un script CGI, como el procesamiento de información en formularios, foros de discusión, manipulación de cookies y páginas dinámicas*" (9).

Ventajas:

- Posee una comunidad muy grande de desarrolladores.
- PHP es fácil de aprender comparado con otros lenguajes de programación.
- Posee un rendimiento muy eficiente.
- Es gratuito y se puede descargar desde www.php.net .
- Es Open Source.
- Este lenguaje fue diseñado para trabajar sobre la Web por ello trae un conjunto muy amplio de funciones para ser utilizadas en diferentes tareas.
- Está disponible para la mayoría de sistemas operativos existentes. Desde Unix, Linux, Microsoft Windows, MAC, entre otros.
- La versión 5 de PHP esta diseñada para que soporte características de programación orientada a objetos.
- Tiene soporte para conectarse a una gran variedad de base de datos como: MySQL, PostgreSQL, Oracle, entre otras.

Desventajas:

- El manejo de errores no es tan sofisticado como en otros lenguajes.
- PHP consume mas recursos (y por tanto es ligeramente mas lento) llamando y ejecutando una función que ejecutando código que encuentra en línea, embebido en el script.

1.7 GESTOR DE BASES DE DATOS

"Un *sistema gestor de base de datos se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos. Ayuda a realizar las siguientes acciones:*

- *Definición de la información.*
- *Mantenimiento de la integridad de los datos dentro de la base de datos.*
- *Control de la seguridad y privacidad de los datos.*
- *Manipulación de los datos*"(10).

1.7.1 POSTGRESQL

Es un sistema de gestión de bases de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD. El desarrollo de PostgreSQL no es manejado por una sola empresa sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

Ventajas:

- Implementación del estándar SQL92/SQL99.
- Soporta una gran variedad de tipos de datos.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas, por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- Posee una gran escalabilidad. Es capaz de ajustarse al número de CPU y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta.

- Implementa el uso de rollback's, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz.
- Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel.
- Es un producto Open Source, sin costos de licencia.

1.8 FRAMEWORKS

"*El concepto de Framework se emplea en muchos ámbitos del desarrollo de sistemas de software, no solo en el ámbito de aplicaciones Web. Podemos encontrar Frameworks para el desarrollo de aplicaciones médicas, de visión por computador, para el desarrollo de juegos y para cualquier ámbito que pueda ocurrírsenos. En general, con el término Framework, nos estamos refiriendo a una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un Framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta. Los objetivos principales que persigue un Framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones" (11).*

Algunos de los Frameworks más usados en la actualidad y de los fundamentales considerados para realizar la Herramienta de Administración de Chronos son:

Para Javascript ExtJS, para PHP Zend y para acceso a datos Doctrine.

1.8.1 DOCTRINE

"*Doctrine es un potente y completo sistema ORM (object relational mapper) para PHP 5.2+ con un DBAL (database abstraction layer) incorporado*" (12).

Ventajas:

- Entre muchas otras cosas tienes la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos.
- Desarrollado completamente en PHP 5.3

- Uso de nombres de espacios.
- Incluye lo relevante de los ORM de Java, basado en Hibernate, EntityManager y JPA (Java Persistence API)
- Posee estándares y buenas prácticas POO e implementación de patrones de diseños.

1.8.2 ZEND

Zend Framework se trata de un marco de trabajo para desarrollo de aplicaciones y servicios Web con PHP, brindando soluciones para construirlos robustos y seguros.

Ventajas:

- Es de código abierto.
- Trabaja con PHP en su versión 5.0.
- Cuenta con módulos para manejar archivos PDF, canales RSS, Servicios Web (Amazon, Flickr, Yahoo), entre otros.
- Incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple consultar sus datos sin tener que escribir ninguna consulta SQL.
- Posee amplia documentación.
- Posee robustas clases para autenticación y filtrado de entrada.

Desventajas:

- Es más difícil de aprender que otras alternativas.
- Existen reportes que indican que también tiene un rendimiento menor que otros Frameworks.
- Es difícil y pesado de instalar, ya que hay que apuntar las librerías desde php y para una persona que empieza a programar no es tarea sencilla.
- No usa el Modelo Vista Controlador (MVC) aunque tenga componentes para ello.

1.8.3 EXTJS

ExtJS es una biblioteca de Javascript para el desarrollo de aplicaciones Web interactivas usando tecnologías como AJAX, DHTML y DOM ,que permite construir aplicaciones enriquecidas de Internet(RIA). Esta librería incluye:

- Componentes GUI (Graphical User Interface) de alto rendimiento y personalizables.

- Modelo de componentes extensibles.
- Un API fácil de usar.
- Licencias Open Source y comercial.

"*Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layout similar al que provee Java Swing, gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada navegador*" (13).

Además la ventana flotante que provee ExtJS es excelente por la forma en la que funciona. Al moverla o redimensionarla solo se dibujan los bordes haciendo que el movimiento sea fluido lo cual proporciona facilidades a los desarrolladores (13).

Ventajas:

- **Balance entre Cliente – Servidor**: La carga de procesamiento se distribuye, permitiendo que el servidor al tener menor carga pueda manejar peticiones de varios clientes simultáneamente.
- **Comunicación asíncrona**: En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, brindando la libertad de cargar información sin conocimiento del cliente.
- **Eficiencia de la red**: El tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

Desventajas:

- No existe una forma fácil de realizar binding entre los componentes visuales con el respectivo modelo, lo cual genera que el programador tenga que escribir más código para validar y enlazar los formularios.
- Es difícil hacer que el servidor haga push de información desde el servidor Web hacia el navegador, haciendo que el cliente tenga que constantemente hacer pooling para obtener datos actualizados del servidor.
- La falta de un diseñador gráfico limita la difusión de la aplicación al no proveer una forma fácil y rápida de desarrollar en él.

- Un problema grave del layout manager es que si se usa código personalizado dentro de un componente, como por ejemplo un iframe, entonces se hace complicado lograr que se redimensione junto con el layout pues cae fuera de su control.
- Su sistema de licenciamiento no contempla la licencia LGPL, o el código es 100% GPL o se debe pagar por la licencia de desarrollo.

1.9 METODOLOGÍA, HERRAMIENTA PARA LA MODELACIÓN DEL SOFTWARE

En todo proceso de desarrollo de software se hace necesario el uso de una guía que posibilite el progreso eficiente en la construcción de aplicaciones. Como procedimientos y técnicas para toda la documentación y generación de artefactos del ciclo de vida del desarrollo del software, se requiere el uso de una metodología. Una decisión muy importante a tomar sería entonces cuál de todas las metodologías puede ser usada. El desarrollo de software es riesgoso y difícil de controlar en todos lo casos, sea un proyecto a corto plazo o un proyecto a largo plazo. Sin embargo muchas veces no se toma en cuenta el utilizar una metodología adecuada, sobre todo en los casos en que el proyecto es de poca duración.

Una metodología para el desarrollo de un proceso de software no es más que un conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de software.

Actualmente existen diversas metodologías de desarrollo de software, algunas de las más importantes y utilizadas son: Extreme Programing (XP) y Rational Unified Process (RUP).

1.9.1 RATIONAL UNIFIED PROCESS (RUP)

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

***Inicio**: El Objetivo en esta etapa es determinar la visión del proyecto.*

***Elaboración**: En esta etapa el objetivo es determinar la arquitectura óptima.*

***Construcción**: En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.*

***Transmisión**: El objetivo es llegar a obtener el release del proyecto.*

"Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los Objetivos de una iteración se establecen en función de la evaluación de todas las iteraciones precedentes".

Disciplina de Desarrollo:

Ingeniería de Negocios: *Entendiendo las necesidades del negocio.*

Requerimientos: *Trasladando las necesidades del negocio a un sistema automatizado.*

Análisis y Diseño: *Trasladando los requerimientos dentro de la arquitectura de software.*

Implementación: *Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.*

Pruebas: *Asegurándose que el comportamiento requerido es el correcto y que todo los solicitado esta presente.*

Disciplina de Soporte:

Configuración y administración del cambio: *Guardando todas las versiones del proyecto.*

Administrando el proyecto: *Administrando horarios y recursos.*

Ambiente: *Administrando el ambiente de desarrollo.*

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

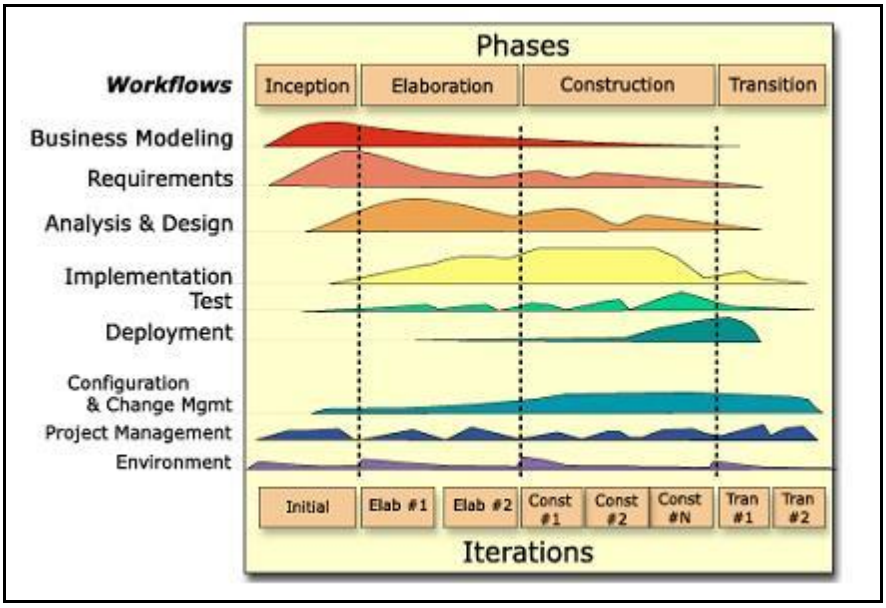

Figura 1.1 Fases e Iteraciones de la Metodología RUP (14)

Los elementos del RUP son:

Actividades*: Son los procesos que se llegan a determinar en cada iteración.*

Trabajadores*: Vienen hacer las personas o entes involucrados en cada proceso.*

Artefactos*: Un artefacto puede ser un documento, un modelo, o un elemento de modelo.*

"Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software". (14)

1.9.2 EXTREME PROGRAMING (XP)

"*Es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizadas para proyectos de corto plazo, corto equipo y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, que tiene como particularidades tener como parte del equipo, al usuario final y a expertos como parte del equipo, pues son algunos de los requisitos para llegar al éxito del proyecto*".

Características de XP

- ***Pruebas Unitarias:*** *se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se puedan hacer pruebas de las fallas que pudieran ocurrir.*
- ***Refabricación:*** *se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.*
- ***Programación en pares:*** *una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce el otro consulta el mapa.*

¿Qué es lo que propone XP?

- *Empieza en pequeño y añade funcionalidad con retroalimentación continua.*
- *El manejo del cambio se convierte en parte sustantiva del proceso.*
- *El costo del cambio no depende de la fase o etapa.*
- *No introduce funcionalidades antes que sean necesarias.*
- *El cliente o el usuario se convierte en miembro del equipo.*

Derechos del Cliente

- *Decidir que se implementa.*
- *Saber el estado real y el progreso del proyecto.*
- *Añadir, cambiar o quitar requerimientos en cualquier momento.*
- *Obtener lo máximo de cada semana de trabajo.*
- *Obtener un sistema funcionando cada 3 o 4 meses.*

Derechos del Desarrollador

- *Decidir como se implementan los procesos.*
- *Crear el sistema con la mejor calidad posible.*
- *Pedir al cliente en cualquier momento aclaraciones de los requerimientos.*
- *Estimar el esfuerzo para implementar el sistema.*
- *Cambiar los requerimientos en base a nuevos descubrimientos.*

Lo fundamental en este tipo de metodología es:

- *La comunicación, entre los usuarios y los desarrolladores.*
- *La simplicidad, al desarrollar y codificar los módulos del sistema.*

La retroalimentación concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales. (14)

1.10 LENGUAJE DE MODELADO

El lenguaje de modelado es una serie de símbolos y de modos de utilizarlos para modelar un diseño de software orientado a objetos. Algunas entidades los usan combinándolos con una metodología de desarrollo de software para lograr de una especificación inicial un plan de implementación y que el mismo sirva de puente de comunicación con un equipo de desarrolladores.

1.10.1 UML

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Este ofrece un estándar para describir un “plano” del sistema incluyendo aspectos conceptuales tales como procesos de negocio y funcionalidades del sistema, esquemas de Base de Datos y componentes de software reutilizables. Pueden ser artefactos un modelo, una descripción que comprende el desarrollo de software. El lenguaje UML tiene una notación gráfica muy

expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, etc., hasta la implementación y configuración con los diagramas de despliegue.

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.

Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.

Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.

Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Un modelo UML esta compuesto por tres clases de bloques de construcción:

Elementos: Los elementos son abstracciones de cosas reales o ficticias (objetos, acciones, etc.)

Relaciones: Relacionan los elementos entre sí.

Diagramas: Son colecciones de elementos con sus relaciones.

Es importante destacar que UML es un lenguaje para especificar y no para describir métodos o procesos. Además UML es desde finales de 1997, orientado a objetos de acuerdo con el Object Management Group, siendo utilizado por grandes empresas de desarrollo de software tales como: Microsoft, Oracle, entre otras. (15)

1.11 HERRAMIENTA CASE

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Como es sabido, los estados en el Ciclo de Vida de desarrollo de un Software son: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación.

CASE se define también como:

- Conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases.
- La sigla genérica para una serie de programas y una filosofía de desarrollo de software que ayuda a automatizar el ciclo de vida de desarrollo de los sistemas.

- Una innovación en la organización, un concepto avanzado en la evolución de tecnología con un potencial efecto profundo en la organización. Se puede ver al CASE como la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales.(16)

1.11.1 VISUAL PARADIGM

Visual Paradigm es una herramienta CASE (Ingeniería de Software Asistida por Ordenador, Computer Aided Software Engineering,) que utiliza "UML": como lenguaje de modelado (17). Desarrollada para diseñar software con Programación Orientada a Objetos, busca reducir la duración del ciclo de desarrollo brin-dando ayuda tanto a arquitectos, analistas, diseñadores y desarrolladores. Busca también automatizar tareas tediosas que pueden distraer a los desarrolladores. Es una herramienta colaborativa porque soporta a varios usuarios trabajando en un mismo proyecto, genera la documentación del proyecto automática-mente en varios formatos como son Web o .pdf y permite control de versiones. Brinda la posibilidad de generar código a partir de los diagramas, para plataformas como .Net, Java y PHP, así como obtener diagramas a partir de código. Esta es precisamente una gran ventaja puesto que el sistema será desarrollado en PHP. Visual Paradigm para UML es multiplataforma, lo cual le permite al usuario utilizar esta herramienta en varios sistemas operativos como Windows, Linux, Unix y otros; además se encuentra disponible en distintas versiones: Enterprise, Professional, Standard, Modeler, Personal y Community. Facilitando también las licencias especiales para fines académicos (18). Entre sus características se encuentra que es muy fácil de instalar y actualizar. Posee compatibilidad entre sus ediciones y soporte de UML versión 2.0.

1.12 ENTORNO DE DESARROLLO INTEGRADO DE PROGRAMACIÓN

"*Un entorno de desarrollo integrado (Integrated Development Environment o IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos.*

Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un

sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones" (19).

1.12.1 ZEND STUDIO

Sirve como editor de texto para páginas PHP permite la creación, gestión de proyectos y la depuración de código. Es un programa hecho por la Zend, que son los impulsores del lenguaje PHP. Todo el programa está hecho en Java. Es multi-plataforma incluyendo versiones para Windows, Linux etc. Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene la interfaz de edición y la ayuda. La parte del servidor, que instala Apache y el módulo PHP o en caso de que estén instalados, se pueden configurar juntos para trabajar en depuración.

1.12.2 NETBEANS

NetBeans IDE es un entorno de desarrollo visual para aplicaciones programadas mediante Java, de modo que puede ejecutarse en cualquier ambiente que ejecute Java. Es un producto de código abierto, con todos los beneficios del software disponible en forma gratuita, el cual ha sido examinado por una comunidad de desarrolladores. Es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas.

Está escrito completamente en Java usando la plataforma NetBeans pero soporta el desarrollo de todos los tipos de aplicación Java (J2SE, Web, EJB y aplicaciones móviles), puede servir además para cualquier otro lenguaje de programación.

Este IDE ofrece a los desarrolladores numerosas ventajas en la creación de nuevas aplicaciones multi-plataforma.

1.13 FUNDAMENTACIÓN DE LAS HERRAMIENTAS, LENGUAJES Y TECNOLOGÍAS

Después de haber hecho un estudio del arte de las diferentes tecnologías, herramientas y lenguajes además de tener en cuenta las características de la herramienta que se quiere desarrollar, así como las características de la UCI que es donde se va a llevar a cabo dicha herramienta, se propone utilizar:

Como lenguaje del lado del servidor: PHP porque:

- Es gratis.
- Es libre.
- Dispone de una gran cantidad de bibliografía.
- Tiene una comunidad que cada día crece más.

Como Framework: El Marco de Trabajo Tipo del Centro de Soluciones de Gestión (CESGE), utilizado en el proyecto ERP-Cuba que utiliza a su vez los Frameworks siguientes:

- Doctrine
- Zend
- ExtJS
- Zend_Ext, desarrollado por el Equipo de la Dirección Técnica del CESGE (la línea de arquitectura).
  - Porque es desarrollado en la UCI.
  - Es libre hasta el momento.
  - Se pone en práctica la reutilización de código dentro de la Universidad.
  - Está certificado por la dirección de calidad de la IP.

Como gestor de Bases de Datos: PostgreSQL porque:

- La herramienta que se quiere desarrollar es para PostgreSQL
- Es libre.
- Es multi-plataforma.
- La UCI forma parte de una de las comunidades de desarrollo del mismo.

Como metodología de desarrollo: RUP por ser:

- Centrado en la arquitectura.
- Guiado por casos de uso.
- Iterativo e incremental

- Permite ingeniería inversa.
- En cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

Como lenguaje de modelado: UML porque:

- Es orientado a objetos.
- Tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático.

Como IDE de desarrollo: NetBeans porque:

- Posee una gran versatilidad.
- Facilidades para programar cualquier lenguaje a través de la instalación de plugins.
- Es libre.
- Es multi-plataforma

Como Herramienta Case: Visual Paradigm porque:

- Es muy completa.
- Fácil de usar.
- Multi-plataforma.
- Proporciona excelentes facilidades de interoperabilidad con otras aplicaciones.
- Suministra características tales como: generación del código, ingeniería reversa y generación de informes.
- La UCI posee la licencia de este software y está abogando porque los proyectos productivos la utilicen.

1.14 CONCLUSIONES

En este capítulo se hizo un estudio y valoración de las principales herramientas, metodologías y lenguajes que se utilizan para el desarrollo de proyectos informáticos en la actualidad, así como de las principales herramientas de réplica existentes para PostgreSQL en el mundo, centrándose en la administración y configuración de las mismas. Después de realizado todo el estudio del arte pormenorizado de las distintas herramientas, se hizo una propuesta para llevar a cabo la implementación de la Herramienta de Administración y Configuración de Chronos.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 INTRODUCCIÓN

Antes de comenzar a desarrollar un sistema es necesario comprender el entorno de trabajo realizando un estudio de los procesos que en él se desarrollan. En este capítulo se hace un análisis de las distintas técnicas utilizadas a la hora de llevar a cabo la ingeniería de requisitos (IR), se analiza de forma objetiva el entorno del sistema específicamente la administración y configuración de la herramienta. Se realiza el modelo de dominio, se analizan los requisitos funcionales y no funcionales para dar paso a la solución del problema. Se elabora una pequeña descripción de los patrones utilizados para los casos de uso, se describen detalladamente todos los Casos de Usos del Sistema analizando punto por punto cada escenario posible, se exponen los aportes y beneficios que traerá el Análisis y Diseño de la Herramienta de Administración y Configuración de Chronos y su posterior implementación.

2.2 TÉCNICAS UTILIZADAS EN LAS ACTIVIDADES DE LA INGENIERÍA DE REQUISITOS

Existen varias técnicas propuestas para la ingeniería de requerimientos que son aplicables para la captura y validación de los mismos. Es importante resaltar que estas técnicas pueden ser aplicables a las distintas fases del proceso de la IR y es necesario tener en cuenta las características propias del proyecto en desarrollo para aprovechar al máximo la utilidad de las mismas.

2.2.1 ENTREVISTAS

Las entrevistas se emplean para reunir información proveniente de personas o de grupos. Durante la entrevista el analista conversa con el encuestado para obtener la información deseada. Las entrevistas se emplean para recolectar información en forma verbal a través de preguntas elaboradas por el analista, estas preguntas se les realiza a quienes se encuentren afectados por la aplicación, a usuarios con gran nivel de conocimiento del sistema o a personas que pueden proporcionar datos. El éxito de esta técnica depende de la habilidad del entrevistador y de su preparación para la misma (20).

2.2.2 SISTEMAS EXISTENTES

La técnica de Sistemas existentes consiste en analizar distintos sistemas ya desarrollados que estén relacionados con el sistema a ser construido. Por un lado, se pueden analizar las interfaces de usuario, observando el tipo de información que se maneja y cómo es manejada. Esto puede ser útil para descubrir información importante a tener en cuenta, información que tal vez el cliente/usuario haya fallado en comunicar. Es recomendable que luego de haber analizado el sistema, este sea mostrado al cliente/usuario, ya que por su experiencia puede sugerir importantes ideas nuevas (21).

2.2.3 PROTOTIPOS

Durante la actividad de extracción de requerimientos, puede ocurrir que algunos requerimientos no estén demasiado claros o que no se esté muy seguro de haber entendido correctamente los requerimientos obtenidos hasta el momento, todo lo cual puede llevar a un desarrollo no eficaz del sistema final. Entonces, para validar los requerimientos hallados, se construyen prototipos. Los prototipos son simulaciones del posible producto, que luego son utilizados por el usuario final, permitiendo conseguir una importante retroalimentación en cuanto a si el sistema diseñado con base a los requerimientos recolectados le permite al usuario realizar su trabajo de manera eficiente y efectiva (22).

2.3 MODELO DE DOMINIO

A partir de una evaluación del estado del negocio, cuyo objetivo para la administración informática es la gestión de información y configuración del sistema, se determinó realizar lo que se conoce como modelo del dominio. En este se representan los principales conceptos y las relaciones entre ellos, permitiendo emplear un vocabulario común que posibilita tanto a los desarrolladores como a los clientes entender el contexto y lograr una solución que cumpla con las expectativas de estos últimos.

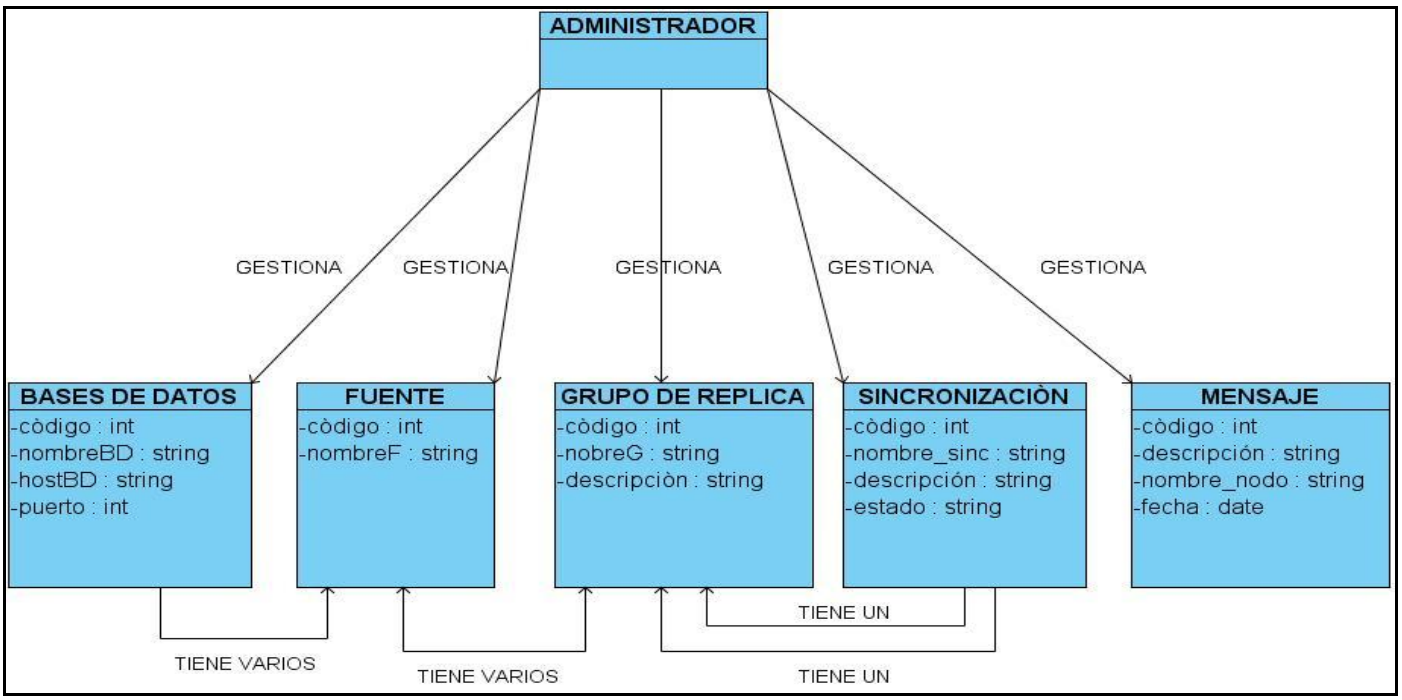

Figura 2.1 Modelo de Dominio

Administrador: Es el encargado de administrar y configurar todo lo referente a la réplica de los datos entre los nodos (Bases de datos) implicados.

Bases de datos: Tiene la información de los nodos que están implicados en la réplica.

Grupos de réplica: Forma de agrupar las fuentes (tablas de las bases de datos) para hacer la configuración más sencilla.

Fuentes por Grupos de réplica: Conjunto de fuentes que formarán parte de un grupo de réplica.

Sincronizaciones: Tiene la información relacionada con el evento de réplica en cuestión, almacenando: grupo de réplica origen y destino, tipo y método de sincronización y otros parámetros necesarios.

Mensajes: Guarda información referente al comportamiento de todo el clúster de réplica, así como de la propia herramienta de administración y configuración.

2.4 PATRÓN UTILIZADO PARA LOS REQUISITOS.

¿Qué es un patrón?

El patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas (23).

Patrón CRUD (Creating, Reading, Updating, Deleting)

Este patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual.

Completo

Este patrón consta de un caso de uso, llamado Información CRUD o gestionar información, que modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio y estos a su vez son cortos y simples.

Parcial

Este patrón alternativo modela una de las vías de los casos de uso como un caso de uso separado. Es preferiblemente utilizado cuando una de las alternativas de los casos de uso es mas significativa, larga o más compleja que las otras.

En los requerimientos identificados se pone de manifiesto el patrón CRUD, que aunque está dirigido a casos de uso están presentes tanto el completo como el parcial en los requisitos de software definidos.

2.5 ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE

Un requerimiento es una representación documentada de una condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo, que puede ser alcanzada por un sistema o componente de un sistema para satisfacer un contrato, u otro documento impuesto formalmente. Permite definir el ámbito del sistema, define una interfaz de usuario para el sistema enfocada a las necesidades y metas de este. Establece y mantiene un acuerdo entre clientes y otros involucrados sobre lo que el sistema debería hacer.

2.5.1 REQUERIMIENTOS FUNCIONALES

Los requerimientos funcionales no son más que las condiciones o capacidades que el sistema debe cumplir. Los requerimientos funcionales que debe cumplir la aplicación son los siguientes:

RF1 Adicionar BD

1.1. Se registran los siguientes datos:

1.1.1. Dirección IP del nodo.

1.1.1.1. La dirección tiene que ser válida.

1.1.2. Puerto del servidor PostgreSQL.

1.1.3. Nombre de la Base de Datos.

1.2. Si no hay conexión con la base de datos especificada no será adicionada esta.

RF2 Mostrar BD

2.1 Se muestran los siguientes datos:

2.1.1 Dirección IP del nodo.

2.1.2 Puerto del servidor PostgreSQL

2.1.3 Nombre de la base de datos

2.1.4 Si existe conexión con la base de datos.

RF3 Modificar BD

3.1 Solo se podrá modificar

3.1.1 Dirección IP

3.1.2 Puerto del servidor PostgreSQL

RF4 Eliminar BD

4.1 Si la base de datos a eliminar esta implicada en una o más sincronizaciones no podrá ser eliminada hasta tanto no se eliminen las sincronizaciones

RF5 Adicionar Grupos

5.1. Se registran los siguientes datos:

5.1.1. Nombre del grupo de réplica.

5.1.2. Descripción del grupo de réplica.

5.2. No pueden existir dos grupos de réplica con el mismo nombre.

RF6 Mostrar Grupos

6.1. Se muestran los siguientes datos:

6.1.1. Nombre del grupo de réplica.

6.1.2. Descripción del grupo de réplica.

RF7 Modificar Grupo

7.1. Se podrán modificar los siguientes datos:

7.1.1. Nombre del grupo de réplica.

7.1.2. Descripción del grupo de réplica.

RF8 Eliminar Grupo

8.1. Si el grupo de réplica a eliminar está implicado en una o más sincronizaciones no podrá ser eliminado hasta tanto no se eliminen las sincronizaciones.

RF9 Adicionar Fuentes

9.1. El grupo de réplica al que se le adiciona la fuente no debe estar implicado en ninguna sincronización.

9.2. Las fuentes con igual nombre deben tener igual composición.

9.3. Los grupos de réplica solo pueden estar formados:

9.3.1. Una fuente.

9.3.2. Varias fuentes todas iguales.

9.3.3. Varias fuentes todas distintas

RF10 Mostrar Fuentes

10.1. Se muestran los siguientes datos:

10.1.1. Dirección IP del nodo a la que pertenece.

10.1.2. Nombre de la Base de Datos.

10.1.3. Nombre de la fuente.

RF11 Eliminar Fuentes

11.1. El grupo de réplica al que se le elimina la fuente no debe estar implicado en ninguna sincronización.

RF12 Adicionar Sincronización

12.1. Se registran los siguientes datos:

12.1.1. Nombre de la sincronización.

12.1.2. Descripción.

12.1.3. Grupo de réplica origen.

12.1.4. Grupo de réplica destino.

12.1.5. Método de sincronización (Adicionar diferencia, Copia Total o Intercambio).

12.1.6. Tipo de sincronización (Instantánea o Programada).

12.1.6.1. Si el tipo de sincronización es programada se registra también un parámetro que representa la fecha y hora a ejecutar dicha sincronización.

RF13 Mostrar Sincronizaciones

13.1.Se muestran los siguientes datos:

13.1.1. Nombre de la sincronización.

13.1.2. Descripción.

13.1.3. Grupo de réplica origen.

13.1.4. Grupo de réplica destino.

13.1.5. Método de sincronización.

13.1.6. Tipo de sincronización.

13.1.7. Estado de la sincronización.

RF14 Reconfigurar Sincronización

14.1.Si existe algún fallo con la configuración de una sincronización se detecta y debe ser arreglado.

RF15 Eliminar Sincronización

RF16 Eliminar Mensajes.

RF17 Mostrar Mensajes

17.1 Se muestran los siguientes datos:

17.1.1 Dirección IP del nodo responsable.

17.1.2 Tipo de mensaje (Error, Error Grave o Advertencia).

17.1.3 Descripción del mensaje.

RF18 Buscar Mensajes.

18.1 Se realiza un filtrado teniendo en cuenta los siguientes criterios:

18.1.1 Tipo de mensaje.

18.1.2 Dirección IP del nodo responsable.

2.5.2 REQUISITOS NO FUNCIONALES

Los requerimientos no funcionales especifican propiedades del sistema, como restricciones del entorno o la implementación, rendimiento o facilidad de mantenimiento (24).

Usabilidad

- El sistema debe tener una interfaz que le sea familiar al usuario para aprovechar sus conocimientos en el manejo de herramientas de software.
- Debe ser de fácil aprendizaje para que usuarios que no sean expertos en sistema operativo Linux puedan familiarizarse rápidamente.

Apariencia o interfaz externa

- Aplicación Web de administración y configuración en línea del sistema.
- Debe poseer una Interfaz Gráfica de Usuario amigable y enfocada al usuario final.
- A través de esta debe ser posible realizar todas las configuraciones de sincronización para todos los nodos del cluster de manera sencilla y rápida.
- Deberá permitir monitorizar el estado de la réplica haciendo interactuar al usuario con este como parte del sistema.

Rendimiento

- El sistema debe ser lo más eficiente posible para poder lograr un tiempo de respuesta adecuado.
- La velocidad de procesamiento de la información debe ser rápida, acorde con las necesidades del usuario.
- Aplicación de las diferentes técnicas de elaboración en el sistema para facilitar el rápido acceso a los datos.

Soporte

- El sistema deberá permitir posteriores modificaciones y actualizaciones a fin de alcanzar mayor funcionalidad.

Portabilidad

- El sistema operará sobre las plataformas basadas en Linux.

Seguridad

- La información manejada por el sistema debe estar protegida de acceso no autorizado y divulgación.

- La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes por las personas autorizadas.
- A los usuarios autorizados se les garantizará el acceso al sistema y que los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los mismos para realizar las transferencias necesarias en un momento dado.
- Se deben crear usuarios que tendrán asignados permisos de acción sobre cada información manejada por el sistema para lo cual se requiere la autenticación del usuario.

Requerimientos de Hardware

Para el cliente:

- Se requiere que las computadoras estén conectadas a la red.

Para el servidor:

Las condiciones mínimas de hardware son:

- CPU similar al Intel Celeron de 2.80 GHz o superior.
- HDD de 5 GB y 128 MB de RAM.

Requerimientos de Software

Los requerimientos mínimos de software necesarios son:

- Como fuente de almacenamiento de la información se utilizará el sistema gestor de base de datos PostgreSQL.
- El lenguaje de programación deberá ser PHP
- Como Framework de Javascript ExtJS.
- Como servidor de aplicaciones Web se utilizará el Apache 2.0 o superior.
- La aplicación funcionará en Sistema Operativo Linux.

Requerimientos en el diseño y la implementación

Para organizar el análisis y el diseño del sistema se utilizará:

- La metodología RUP.
- UML como lenguaje de modelado.
- Visual Paradigm como herramienta CASE.
- PHP será el lenguaje de programación a ser usado para la implementación.
- Los nombres de los métodos y de las clases deben ser lo más sencillo posible para un mejor entendimiento entre el equipo de desarrolladores.

2.6 MODELAMIENTO DEL SISTEMA

2.6.1 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

El sistema de réplica de base de datos que se pretende implantar en las aduanas de Cuba para facilitar gran parte de las funciones que en estas se realiza, es una herramienta de réplica asíncrona para entornos multi-maestro llamada Chronos. Chronos ha sido diseñado a partir de necesidades particulares de un escenario de réplica, asumiendo las principales potencialidades de las soluciones similares en la actualidad para PostgreSQL e implementando nuevas mejoras que la distinguen del resto. Constituido por diversos componentes como:

LibTransaccionesSQL.so: Librería de enlace dinámico (share object) que extiende la funcionalidad del gestor capturando los cambios que se producen en cualquiera de las tablas contenidas.

LibComunicacion.so: Librería de enlace dinámico (share object) que contiene una capa de servicios para la comunicación con el Motor de Procesamiento de Sentencias (MPS).

Motor de Procesamiento de Sentencias (MPS): Constituye el núcleo de la aplicación.

AdminChronos: Aplicación de administración, configuración y monitorización en línea del sistema.

AdminChronos es una aplicación Web al que solo pueden acceder y tienen permiso de ejecución los usuarios previamente autenticados que tengan el rol de administradores de la misma. Tiene como propósito garantizar la réplica de la información, para lo cual gestiona fuentes y sincronizaciones entre otras funcionalidades. Se divide en varios módulos: administración de fuentes, centro de sincronización y buzón de mensajes.

En el módulo de administración de fuentes se agregan las Bases de Datos (BD) y tablas de las mismas con las que se trabajará. En el centro de sincronización se determina el método (adicionar diferencia, copia total o intercambio) y tipo (instantánea o programada) de esta además deben crearse los grupos de origen y destino de la réplica. En el buzón de mensajes se registran los errores ocurridos de los cuales se almacenan: dirección IP del nodo responsable, tipo de error y una breve descripción de este. El administrador tiene la tarea de realizar estas operaciones creando de esta manera una configuración de réplica que será establecida en cada nodo involucrado en el proceso.

2.6.2 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. A continuación se muestra el actor que interviene en el sistema.

Actor	Descripción
Administrador.	Persona que interactúa con el componente AdminChronos. Es el único responsable de gestionar toda la información relacionada con las réplicas de BD, sincronizaciones y mensajes de error. Puede además, buscar cualquier información sobre estas entidades por diferentes criterios de búsqueda y ver los datos de las mismas.

2.6.3 DIAGRAMA DE CASOS DE USO DEL SISTEMA.

Es un modelo de las funciones deseadas para el sistema y su entorno y sirve como contrato entre el cliente y los desarrolladores. Es utilizado como entrada esencial para las actividades de análisis, diseño y prueba. Se diferencia de los casos de uso a implementar en la primera versión del sistema.

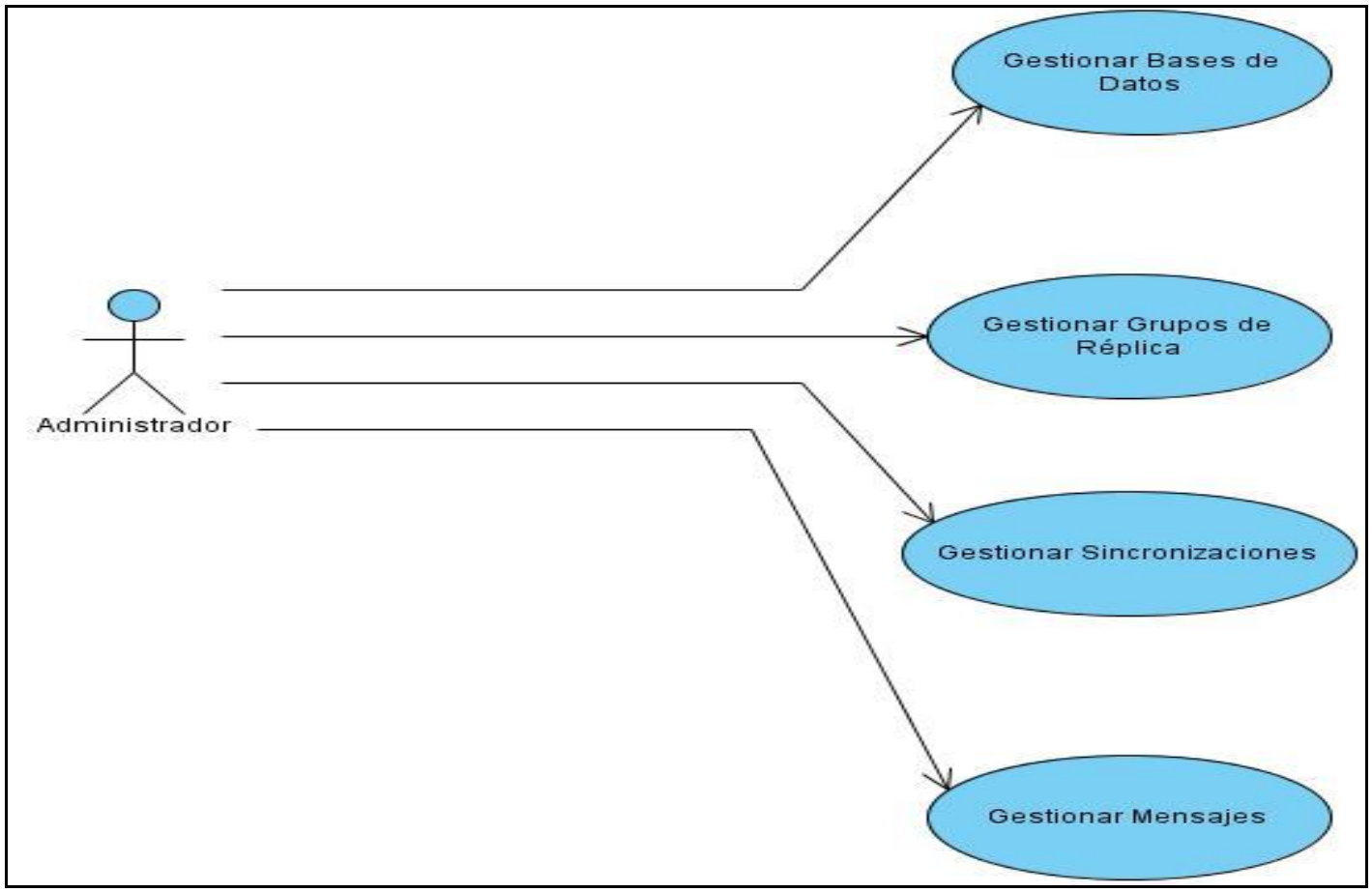

Figura 2.2 Diagrama de Casos de Uso del Sistema

El diagrama de CUS engloba una serie de Casos de Uso para su mayor entendimiento y claridad. A continuación se enumeran los Casos de Uso por los que está compuesto cada gestionar.

Gestionar BD:

AdicionarBD.

MostrarBD.

ModificarBD.

EliminarBD.

Gestionar Grupos:

Adicionar Grupo.

Mostrar Grupo.

Modificar Grupo.
Eliminar Grupo.
Adicionar Fuentes.
Mostrar Fuentes.
Eliminar Fuentes.
Gestionar Sincronizaciones:
Adicionar Sincronización.
Mostrar Sincronizaciones.
Reconfigurar Sincronización.
Eliminar Sincronización.
Gestionar Mensajes:
Mostrar Mensajes.
Eliminar Mensajes.
Buscar Mensajes.

2.6.4 EXPANSIÓN DE LOS CASOS USO DEL SISTEMA

Los casos de uso expandidos describen detalladamente la secuencia de pasos que realizan los actores en su interacción con el sistema para completar cada proceso.
(Ver anexo #1)

2.7 APORTES Y BENEFICIOS

El sistema propuesto permitirá gestionar la replicación de bases de datos con PostgreSQL de una manera muy flexible, amigable y fácil de usar, ya que brindará la posibilidad de configurar de manera remota y centralizada los distintos nodos que estén conectados y disponibles para replicar. Además brindará una serie de opciones desde su interfaz visual que la convertirán en única de su tipo. La misma inicialmente se utilizará en las aduanas de Cuba. Por su repercusión y utilidad pudiera expandirse a otras áreas y empresas que deseen mantener su información segura mediante la replicación utilizando un gestor de BD tan robusto como PostgreSQL. Algunas de las opciones que tendrá la herramienta son:

- **Configuración en línea del sistema:** AdminChronos permitirá la configuración remota de todo el sistema, así como las actividades de administración. Todo esto sin intervenir en el proceso de

réplica, solo conectándose a la base de datos del nodo en cuestión.

- **Configuración de fuentes de datos *ad-hoc*:** Para Chronos, la unidad más pequeña de la réplica de datos es una fuente, equivalente a una tabla de la base de datos. AdminChronos permitirá la adición de fuentes de datos *ad-hoc* a una sincronización ya configurada, lo que indica que esta se restablecerá desde cero.
- **Agrupamiento de fuentes de datos:** Para facilitar el proceso, Chronos hará posible la unión de muchas fuentes de datos en grupos, estos se corresponden como orígenes y destinos de una sincronización, lo que ahorraría el tedioso proceso de crear sincronizaciones para cada par de fuentes de datos involucradas. Para establecer sincronizaciones entre grupos, los criterios de agrupación son:
    - **De un origen a uno o múltiples destinos:** El grupo origen contiene una tabla de una base de datos y el grupo destino contiene todas las tablas que se sincronizarán con esta en la réplica. Por supuesto las tablas agrupadas deben cumplir con el criterio de igualdad entre fuentes de datos y corresponden a bases de datos diferentes.
    - **De múltiples orígenes a múltiples destinos:** El grupo origen contiene un conjunto de fuentes de datos relacionadas o no entre sí y el grupo destino contiene otro conjunto de fuentes de datos de igual forma. Existe una función entre ambos grupos, a cada fuente origen le corresponde una fuente destino, de manera que puedan sincronizarse entre sí homólogamente. Este criterio es utilizado para configurar réplicas de bases de datos completas, en ese caso cada base de datos representaría un grupo de fuentes y se configuraría la réplica entre ellas.
- **Configuración básica del sistema.** Incluirá la configuración básica de sincronizaciones de fuentes de datos.
- **Validación de la configuración:** AdminChronos validará la configuración establecida, lo que reduce un gran volumen de errores que se pueden cometer en el diseño de la réplica de datos para un escenario determinado y que no necesariamente estén asociados al intelecto del administrador, sino a lo tedioso que puede resultar este trabajo.

2.8 CONCLUSIONES

En este capitulo se hizo un profundo estudio de las características que tendrá el sistema a implementar, nos apropiamos de elementos concisos al analizar el entorno del problema desde el punto de vista de administración y configuración de la futura herramienta de réplica, lo que nos acerca más a la propuesta de implementación y crea los cimientos para el análisis y diseño. Se valoraron los requisitos funcionales y no funcionales para de esta forma ir definiendo y pormenorizando los detalles que debe tener el mismo. Se realizó además una propuesta de interfaz de usuario no funcional y se llegó a la conclusión de que la misma cumplía los requisitos para ser la interfaz funcional de la aplicación con algunas pequeñas modificaciones. Se llegó a conclusiones importantes sobre los aportes y beneficios que este trabajo y la consiguiente implementación de la aplicación traerán.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 INTRODUCCIÓN

En este capítulo se procederá a realizar los restantes pasos para que todo quede listo para la implementación del sistema. Primeramente se realizará el modelo de análisis, así como las clases que el incluye y los diagramas de colaboración, lo que servirá de entrada para el diseño. También se describen los patrones de diseño usados, ya que los mismos permiten solucionar problemas de reutilización de código, organización y mayor claridad para los programadores cuando se dispongan a implementar. Luego se pasará a realizar las clases del diseño utilizando estereotipos Web, así como la descripción de las clases más significativas y por último se aplicarán métricas para la evaluación del diseño, lo cual garantizará una correcta implementación.

3.2 MODELO DE ANÁLISIS

El Análisis es el flujo de trabajo donde se refinan y estructuran los requisitos obtenidos con anterioridad, con el objetivo de facilitar la compresión, preparación y modificación de los mismos. Un modelo de análisis es aquel que estructura los requisitos de un modo que facilita su comprensión y puede considerarse además como una primera aproximación al Modelo del Diseño.

Una realización de casos de uso proporciona por tanto, una traza directa hacia un caso de uso concreto del modelo de casos de uso, además de ayudar a comprender los requisitos del software y no cómo se implementará la solución; logrando una profundización más precisa de los requerimientos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero, incluyendo su arquitectura. (25)

3.2.1 DIAGRAMAS DE CLASES DEL ANÁLISIS

Un diagrama de clases del análisis es un artefacto en el que se representan los conceptos del dominio del problema. Representan las cosas del mundo real, no de la implementación automatizada. En general se siguen directrices muy parecidas a las que se usan en la construcción del modelo conceptual. Este se

realiza para cada uno de los CUS y muestra las clases participantes que se clasifican en tres tipos: interfaces, controladoras y entidades.

Clase Interfaz: Se utilizan para modelar la interacción entre el sistema y sus actores, cada interfaz debe asociarse con al menos un actor y viceversa.

Clase Controladora: Representan coordinación, secuencia, transacciones y control de otros objetos.

Clase Entidad: Se utilizan para modelar la información que posee una larga vida y que es a menudo persistente. (26)

(Ver anexo #2)

3.2.2 DIAGRAMAS DE COLABORACIÓN

(Ver anexo #3)

3.3 DISEÑO

En el diseño se determina la arquitectura general del sistema y su comportamiento dinámico, adaptando la especificación realizada en la etapa anterior. En esta fase se establece el comportamiento dinámico del sistema, es decir, como debe reaccionar ante los acontecimientos (27).

El resultado obtenido de la etapa de diseño facilita enormemente la implementación posterior del sistema, pues proporciona la estructura básica del sistema y como los diferentes componentes actúan y se relacionan entre ellos.

El diseño del sistema se realizará teniendo en cuenta el patrón de arquitectura Modelo Vista Controlador (MVC), el Marco de Trabajo Tipo del Centro de Soluciones de Gestión (CESGE), utilizado en el proyecto ERP-Cuba que utiliza a su vez los Frameworks siguientes:

- Doctrine
- Zend
- ExtJS
- Zend_Ext, desarrollado por el Equipo de la Dirección Técnica del CESGE (la línea de arquitectura).

3.3.1 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.

3.3.1.1 PATRONES GOF

Decorator: este patrón permite añadir funcionalidad a una clase dinámicamente. Zend Framework implementa dicho patrón en la clase Zend_View, encargada de asignarle responsabilidades a objetos de manera dinámica y configurarlos con nuevos atributos.
Singleton: este patrón garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Zend Framework tiene una instancia única del controlador frontal disponible mediante este patrón para lograr una vía de entrada única a las solicitudes.
Facade: este patrón simplifica los accesos a las clases de la capa de acceso a datos proporcionando un objeto que todas las clases de capas superiores utilizarán para acceder a las clases contenidas en la capa del modelo. Define una interface de más alto nivel que permite usar el sistema más fácil. El objetivo de la aplicación de este patrón es reducir la dependencia entre clases. Se utilizará una clase intermediaria entre las clases controladoras de Zend Framework y las de acceso a datos de Doctrine, la que brindará, de las operaciones de acceso a datos, sólo las que necesiten los controladores para su funcionamiento, lo que reduce la dependencia de estos entre las múltiples clases de acceso a datos existentes en el sistema.
Factory: Este patrón proporciona una interfaz para crear familias de objetos relacionados o dependientes sin especificar su clase concreta. Permite configurar en tiempo de ejecución un sistema con una familia u otra de objetos. Además garantiza que un conjunto de clases se usen a la vez. Zend Framework provee una API para el acceso a datos conformada por un conjunto de clases que implementa dicho patrón. (28)

3.3.1.2 PATRONES DE ASIGNACIÓN DE RESPONSABILIDADES (GRASP)

(Ver anexo #4)
Controlador: Este patrón se tiene en cuenta para realizar las asignaciones en cuanto al manejo de los eventos del sistema y definir sus operaciones. Zend Framework contribuye a la utilización de este patrón

ya que define un Controlador Frontal (Front Controller) que implica que todas las solicitudes son dirigidas a un único script PHP que se encarga de instanciar al controlador frontal y redirigir las llamadas.

Experto: Patrón que propone como solución asignar la responsabilidad a la clase que cuenta con la información necesaria para cumplir la responsabilidad. Las clases que brinda el Framework Ext_JS se encargarán de visualizar las interfaces ya que cuentan con la información para crear los diferentes componentes visuales, las clases controladoras del Zend Framework manejarán las peticiones del cliente y las clases que genera y utiliza el Doctrine serán las encargadas del acceso a datos pues contienen y representan los datos que manejará el sistema. Permitiendo que se conserve el encapsulamiento, soportando un bajo acoplamiento y una alta cohesión.

Creador: Tiene en cuenta para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto solo pueda ser creada por el objeto que contiene la información necesaria para ello. Existe un único script PHP que se encarga de instanciar al controlador frontal, este último es el encargado de instanciar las clases controladoras y estas, a su vez, instancian objetos de la clase Zend_View. El uso de este patrón permite crear las dependencias mínimas necesarias entre las clases, favoreciendo al mantenimiento del sistema.

Bajo Acoplamiento: Este patrón brinda como solución asignar responsabilidades de manera que las clases no dependan fuertemente de otras. Ofreciendo como beneficio que son fáciles de entender por separadas, fáciles de reutilizar y no se afectan por cambios de otros componentes. Dicho patrón se tiene en cuenta debido a la importancia de realizar un diseño de clases independientes que soporten los cambios.

Alta cohesión: El siguiente patrón propone asignar la responsabilidad de manera que la complejidad se mantenga dentro de límites manejables asumiendo solamente las responsabilidades que deben manejar, evadiendo un trabajo excesivo. Su utilización mejora la claridad y facilidad con que se entiende el diseño, simplifica el mantenimiento y las mejoras de funcionalidad, generan un bajo acoplamiento, soporta mayor capacidad de reutilización. (29)

3.3.1.3 PATRONES DE ACCESO A DATOS

Active Record: Este patrón representa de forma Orientada a Objetos los datos de una Base de Datos Relacional, definiendo interfaces sencillas para acceder y manipular esos datos. Es un enfoque al problema de acceder a los datos de una base de datos. Una fila en la tabla de la base de datos (o vista)

se envuelve en una clase, de manera que se asocian filas únicas de la base de datos con objetos del lenguaje de programación usado. Cuando se crea uno de estos objetos, se añade una fila a la tabla de la base de datos. Cuando se modifican los atributos del objeto, se actualiza la fila de la base de datos. Doctrine usa este patrón para manejar la base de datos.

Row Data Gateway: Patrón que asume el comportamiento de un objeto que actúa como puerta de enlace a una fila de una tabla de la base de datos. Tiene propiedades que reflejan las columnas de la tabla y métodos de actualización en la base. Zend Framework provee una API para el acceso a datos conformada por un conjunto de clases que implementa dicho patrón.

Table Data Gateway: Parecido a Row Data Gateway, pero a diferencia de este define la estructura de acceso por registros en las entidades de la base de datos, específica su acceso a nivel de tabla, proponiendo un objeto que se comporte como puerta de enlace a cada tabla de la base de datos. Contiene una interface que permite actualizar, buscar, borrar e insertar en la tabla y puede retornar un registro, un grupo de registro y hasta un objeto del dominio. Zend Framework provee una API para el acceso a datos conformada por un conjunto de clases que implementa dicho patrón. (30)

3.3.1.4 PATRÓN DE ARQUITECTURA MODELO VISTA CONTROLADOR

El patrón de arquitectura conocido como Modelo-Vista-Controlador (MVC), separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario; es decir separa en tres capas diferentes los datos de una aplicación, la interfaz de usuario y la lógica de control: (Ver Anexo #4)

Modelo: Esta capa administra el comportamiento y los datos del dominio de la aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: Esta capa maneja la visualización de la información, es decir que presenta el modelo en un formato adecuado para interactuar, que usualmente es la interfaz de usuario.

Controlador: Esta capa controla el flujo de datos entre la vista y el modelo; es decir que responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases.

Esta separación permite construir y probar el modelo, independientemente de la representación visual. En el caso de los diagramas de clases realizados para el diseño la página phtml representa la vista, las

clases controller el controlador y en el modelo se representan las clases de la lógica del negocio y las de dominio que son las de acceso a dato, quedando así hecha la representación de este patrón y logrando que cualquier cambio que se realice en la vista no afecte ni la lógica del negocio, ni el dominio. (30)

3.4 DIAGRAMAS DE CLASES DEL DISEÑO

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Normalmente contienen la siguiente información:

- Clases, asociaciones y atributos
- Interfaces, con sus operaciones y constantes
- Métodos
- Información sobre los tipos de atributos
- Navegabilidad
- Dependencias

A diferencia del modelo conceptual, un diagrama de este tipo contiene las definiciones de las entidades del software en vez de conceptos del mundo real. (31)

3.4.1 DIAGRAMA DE CLASES CON ESTEREOTIPOS WEB

(Ver anexo #5)

3.4.2 DESCRIPCIÓN DE LAS CLASES MÁS SIGNIFICATIVAS

Con este epígrafe se ofrece la información detallada de la estructura y funcionalidades que se muestran en las clases identificadas durante el diseño. Se toman como referencia las clases que controlan las operaciones y gestionan el intercambio de información entre el usuario y la base de datos del sistema. De las mismas se describen las operaciones más significativas.
(Ver anexo # 6)

3.5 DISEÑO DE LA BD

En el desarrollo de una aplicación informática, el diseño de la BD es de gran importancia, ya que en ella se almacenan todos los datos que son necesarios en la modelación del problema que se desea resolver, además ésta es la fuente de obtención de toda la información que se quiera recuperar del sistema. Las bases de datos necesitan de una definición de su estructura que le permitan almacenar datos, reconocer el contenido y recuperar la información.

3.5.1 MODELO LÓGICO

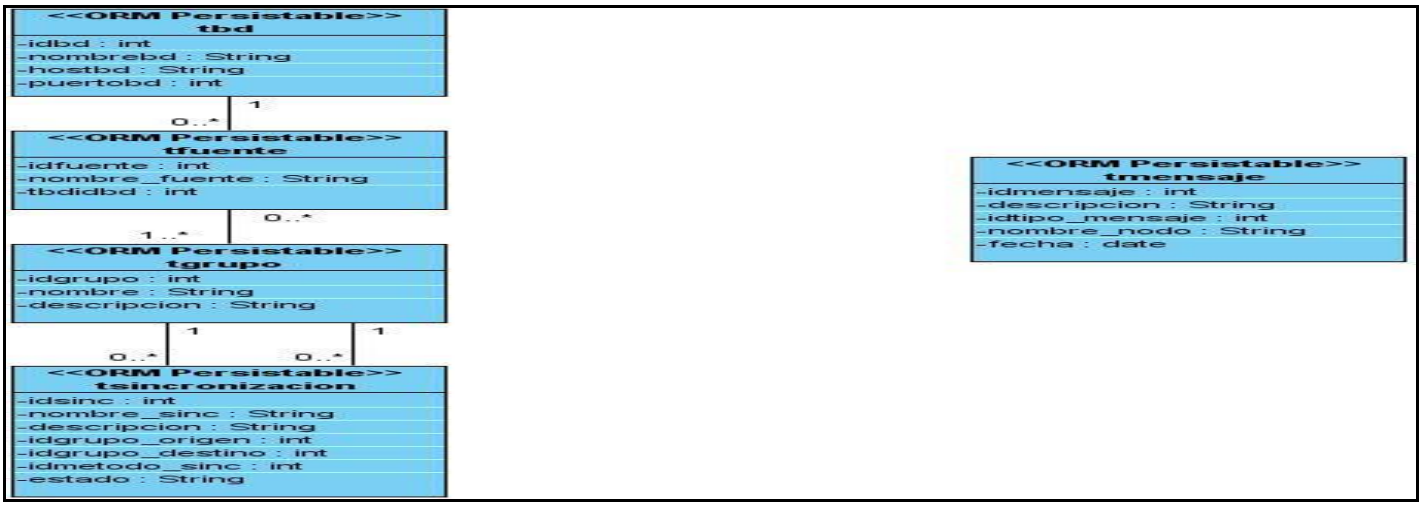

Figura 3.9 Modelo Lógico de Datos

3.5.2 MODELO FÍSICO

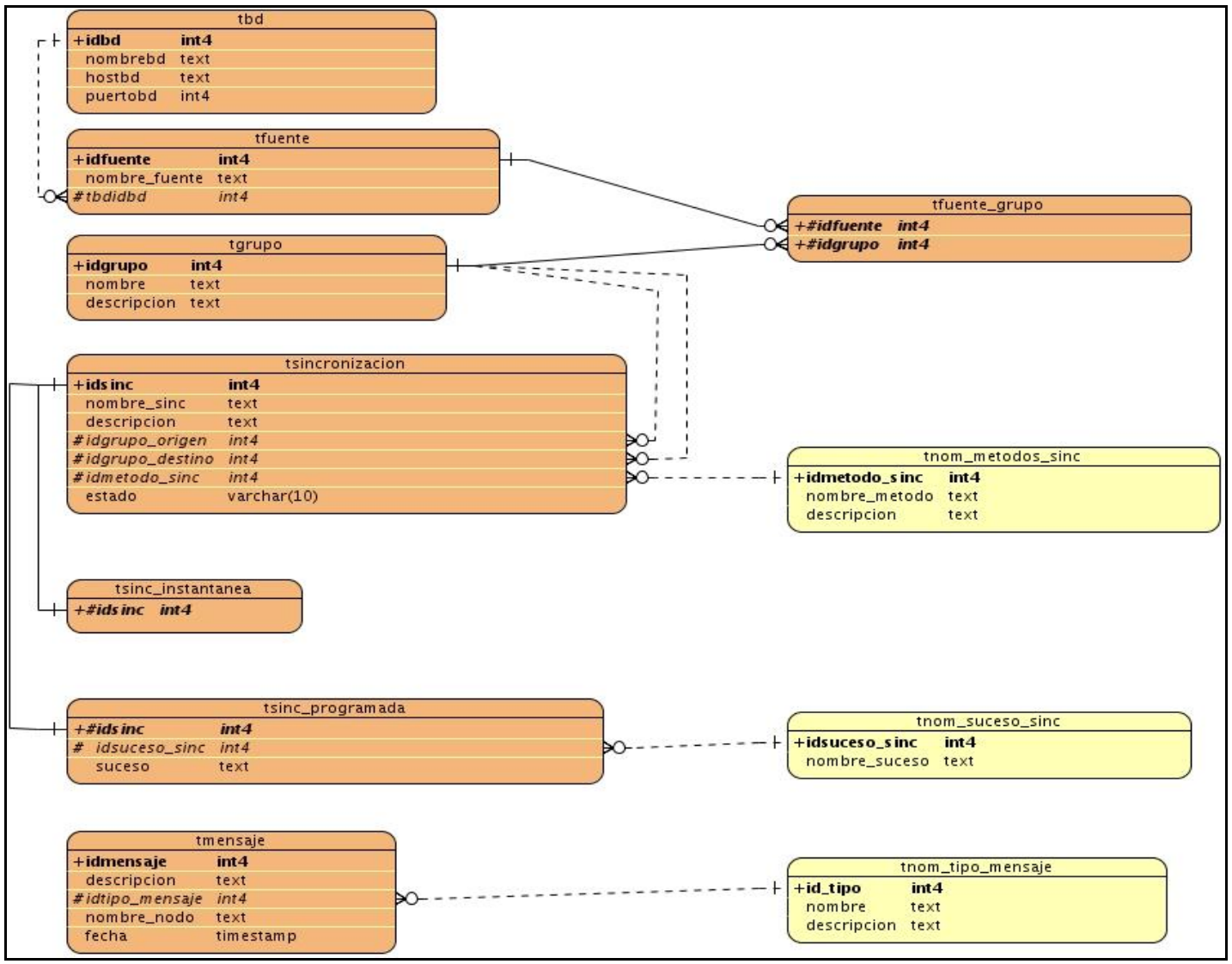

Figura 3.10 Modelo Físico de Datos

3.6 DIAGRAMA DE DESPLIEGUE

El modelo de despliegue describe cómo una aplicación se despliega a través de una infraestructura. La intención del modelo de despliegue no es para describir la infraestructura, es el camino en el cual los componentes específicos deben corresponder a una aplicación que despliega a través de él. El modelo de

despliegue muestra la configuración (relaciones físicas) de los nodos que participaron en la ejecución y de los componentes hardware y software que residen en ellos (32).

En el siguiente epígrafe se muestra la distribución que deberá tener el sistema para su correcto funcionamiento, tanto en la arquitectura de software como de hardware, a través de un diagrama de despliegue.

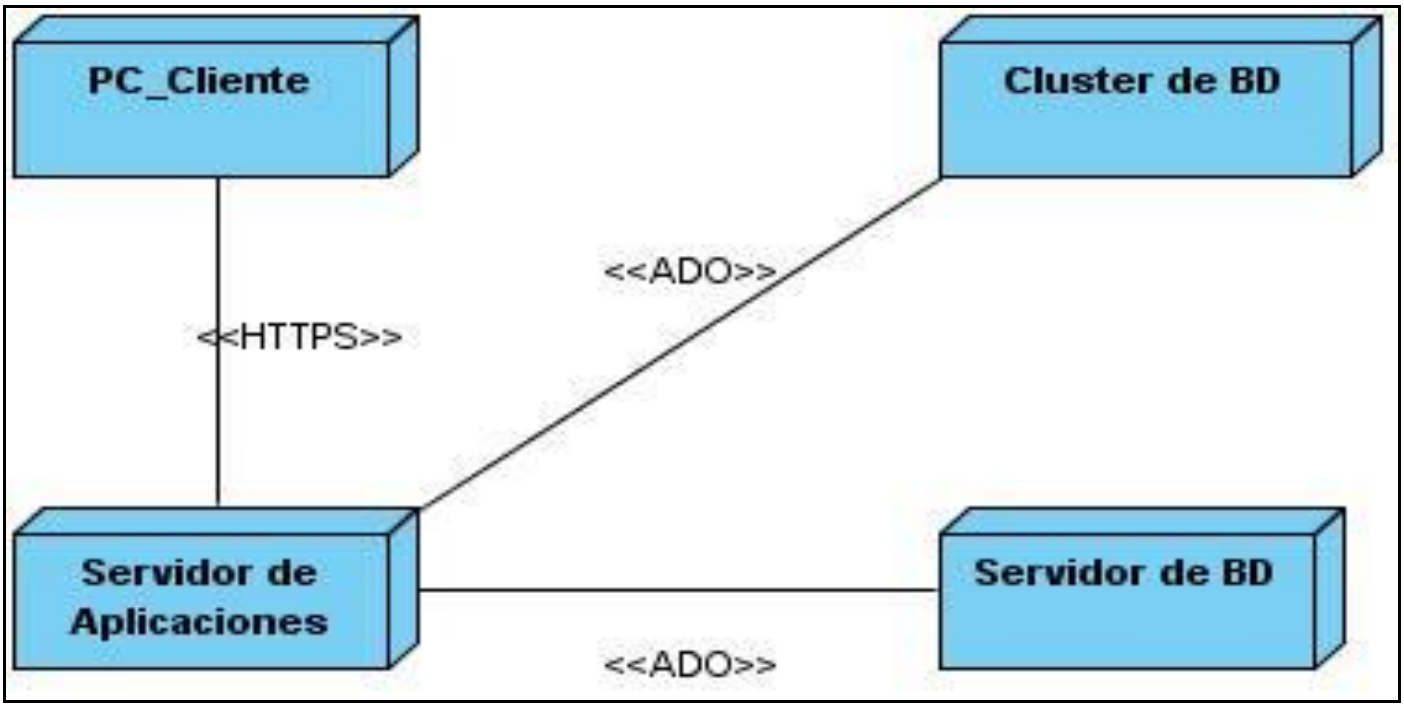

Figura 3.11 Diagrama de Despliegue

3.7 MÉTRICAS PARA LA EVALUACIÓN DEL DISEÑO

Cuando los modelos de diseño aumentan en tamaño y complejidad resulta beneficioso aplicar métricas para verificar la calidad y objetividad del mismo. La visión objetiva de un modelo de diseño Orientado a Objetos (OO) se obtiene a través de la adecuada aplicación de métricas, cuyo resultado aporta el componente cualitativo en la evaluación que se hace. Dentro del diseño OO, se definen nueve características medibles: tamaño, complejidad, acoplamiento, suficiencia, integridad, cohesión, originalidad, similitud y volatilidad (33).

Las métricas buscan comprobar, de manera cuantitativa, en qué grado el software posee las distintas características que definen la calidad de un producto software, de ahí que su utilización sea una fase

importante dentro de la evaluación del diseño. Entre las referenciadas por Pressman (34), se han seleccionado las que se aplican con mayor facilidad dentro del sistema en cuestión, considerando que este estudio brinda un modelo sencillo de implementar y que a su vez cubre los principales atributos de calidad de software. Lo referido al aseguramiento de la calidad es la principal razón por la que se conciben estas métricas.

La valoración de las actividades realizadas dentro del modelo de diseño se ha realizado con la aplicación de las métricas siguientes:

- **Tamaño operacional de clase (TOC).**

Esta métrica expresa el número de métodos asignados a una clase. El impacto que tiene su aplicación en los atributos de calidad mencionados se evidencia mayormente en la responsabilidad, complejidad y reutilización. La relación está expresada porque el aumento de la cantidad de métodos de la clase es directamente proporcional a la responsabilidad y complejidad de la clase, lo que supone un aumento de los mencionados indicadores. De igual manera, se traduce en una reducción de la reutilización que pueda tener la misma.

- **Relaciones entre clases (RC)**

Métrica que está dada por el número de relaciones de uso de una clase con otras. El aumento de las RC se traduce en un aumento del acoplamiento, complejidad de mantenimiento, cantidad de pruebas que requiere la clase; al tiempo que, de la misma forma que la métrica anterior, implica una reducción en la reutilización de la clase.

Luego de efectuar los cálculos para un:

Total de clases	**22**
Promedio de procedimientos	**4.27272727272727**

Tabla 1 Cantidad de procedimientos por clases.

No	Nombre	Procedimientos
1	AdminController	14
2	CentrosincController	29
3	BuzonmsgController	5
4	TbdModel	6
5	TfuenteModel	7

6	TgrupoModel	5
7	Tfuente-grupoModel	6
8	TsincronizacionModel	7
9	TmensajeModel	4
10	TsincProgramadaModel	4
11	TsincInstantaneaModel	4
12	Tnom_metodo_sinc	1
13	Tnom_suceso_sinc	1
14	Tnom_tipo_mensaje	1
15	Tbd	0
16	TfuenteModel	0
17	TgrupoModel	0
18	Tfuente-grupo	0
19	Tsincronizacion	0
20	Tmensaje	0
21	TsincProgramada	0
22	TsincInstantanea	0

Se obtuvieron las siguientes gráficas para la Responsabilidad, Complejidad y Reutilización, utilizando las siguientes tablas para obtener las categorías:

Tabla 2 Criterio para calcular la responsabilidad

Responsabilidad	Categoría	Criterio
	Baja	< =Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	> 2* Prom.

Tabla 2 Criterio para calcular la Complejidad de implementación

Complejidad implementación	Categoría	Criterio
	Baja	< =Prom.
	Media	Entre Prom. y 2* Prom.

	Alta	> 2* Prom.

Tabla 4 Criterio para calcular la reutilización

Reutilización	Categoría	Criterio
	Baja	> 2* Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	<= Prom.

Tabla 5 Responsabilidad y Complejidad de las clases

Responsabilidad y Complejidad	Cantidad de clases
Baja	14
Media	6
Alta	2

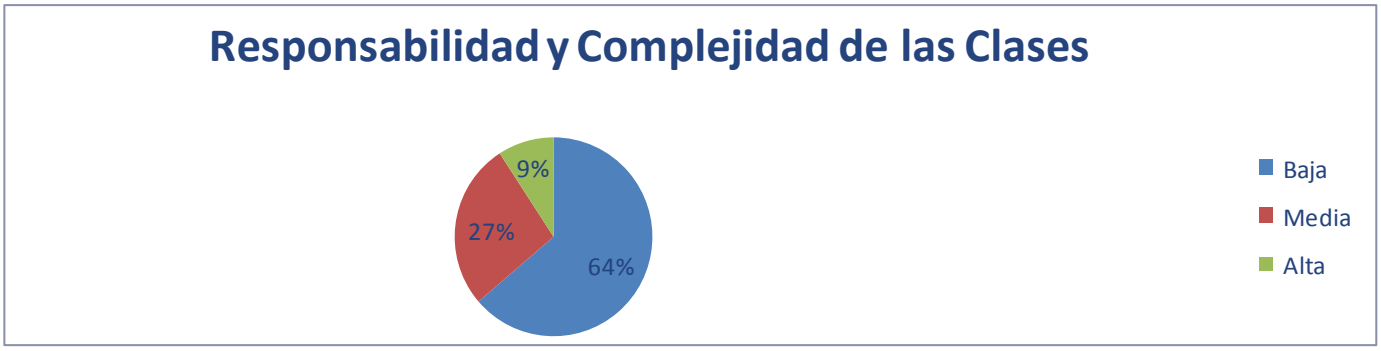

Figura 3.12 Representación de la responsabilidad y complejidad.

Tabla 6 Reutilización de las clases

Reutilización	Cantidad de clases
Baja	2
Media	6
Alta	14

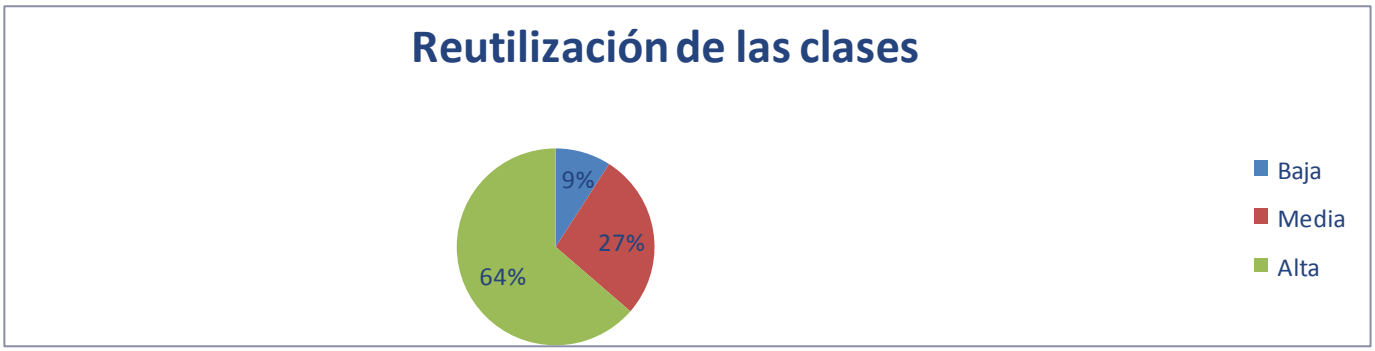

Figura 3.13 Representación de la reutilización

Cuando la responsabilidad (Figura 3.12) de las clases entre media y baja es mayor que el 80% se puede decir que cumple con el patrón de alta cohesión, además de que presentan una reutilización alta.

Para aplicar la métrica RC se tuvo en cuenta la cantidad de relaciones que tenían las clases, a partir del promedio de las relaciones y mediante un criterio se obtuvo la categoría (baja, media y alta en el caso del Acoplamiento) y (baja, media, alta) para la Reutilización.

	Categoria	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2

	Categoria	Criterio
Reutilización	Bajo	>2* Prom.
	Medio	Entre Prom. y 2*Prom
	Alto	<= Prom.

Luego de efectuar los cálculos para un:

Total de clases	**22**
Promedio de asociaciones de uso	**1.09090909090909**

Tabla 7 Relaciones de uso por clases

No	Nombre	Relaciones
1	AdminController	0
2	CentroSincController	0
3	BuzonmsgController	0
4	TbdModel	1
5	TfuenteModel	1
6	TgrupoModel	1
7	Tfuente-grupoModel	2
8	TsincronizacionModel	1
9	TmensajeModel	1
10	TsincProgramadaModel	1
11	TsincInstantaneaModel	1
12	Tnom_metodo_sinc	0
13	Tnom_suceso_sinc	0
14	Tnom_tipo_mensaje	0
15	Tbd	1
16	Tfuente	2
17	Tgrupo	2
18	Tfuente-grupo	2
19	Tsincronizacion	5
20	Tmensaje	1
21	TsincProgramada	1
22	TsincInstantanea	1

Tabla 8 Acoplamiento de las clases

Acoplamiento	Cantidad de clases
Ninguno	6
Bajo	11
Medio	4
Alto	1

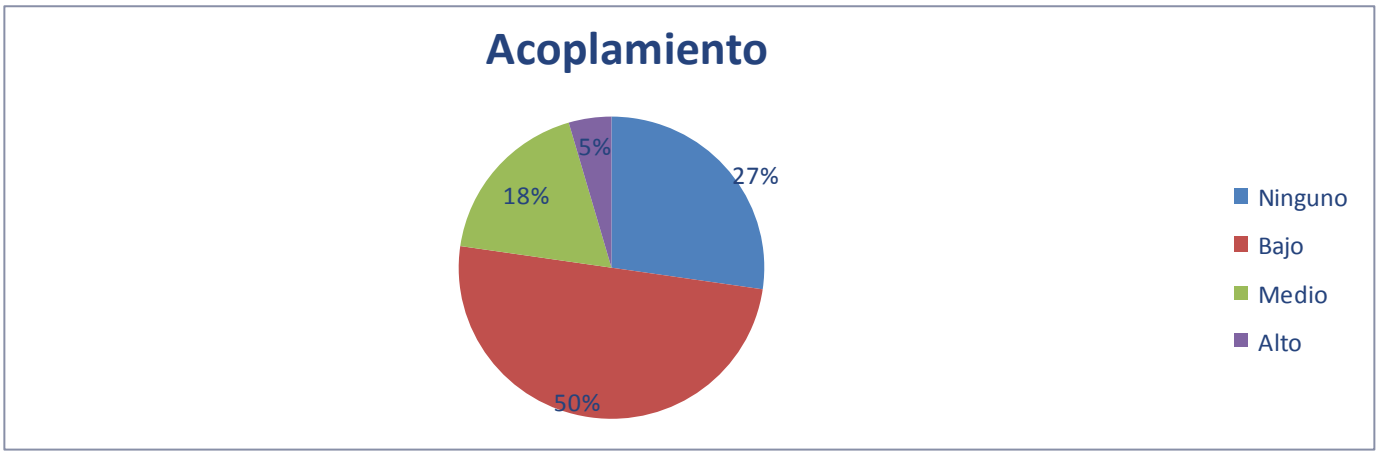

Figura 3.14 Representación del acoplamiento entre las clases

Tabla 9 Reutilización de las clases

Reutilización	Cantidad de clases
Bajo	1
Medio	4
Alto	17

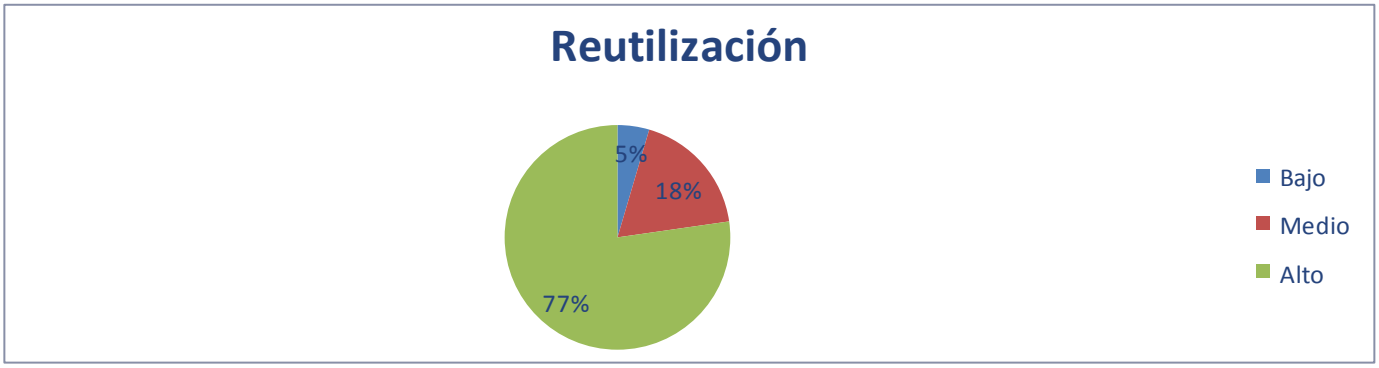

Figura 3.15 Representación de la reutilización

De manera general los resultados de la aplicación de esta métrica son positivos. El acoplamiento existente entre bajo y ninguno de las clases es de 77%. El nivel de reutilización de las clases es alto, ya que el 95% de las clases pueden ser reutilizadas.

3.8 CONCLUSIONES DEL CAPÍTULO 3

En el presente capítulo se abordaron los aspectos básicos relacionados con el análisis y diseño del sistema, además se mostraron los diagramas correspondientes: los de clases del análisis y de clases del diseño así como los diagramas de colaboración para cada una de las funcionalidades descritas. Se describe la arquitectura y los patrones que se tuvieron en cuenta para el diseño de los casos de uso. Además, se muestran los modelos físico y lógico de la base de datos los cuales son de gran importancia para la implementación del sistema, así como el diagrama de despliegue. También y no por último menos importante se hace una validación del diseño a través de una serie de métricas las cuales demuestran que lo que se hizo en este capitulo fue correcto y garantiza una buena implementación.

CONCLUSIONES:

Con el desarrollo del siguiente trabajo de diploma se arriba a las siguientes conclusiones:

- Se realizó un minucioso estudio de los sistemas de réplica existentes y se concluyó que no existe una herramienta de administración y configuración que posea las características especificas para este entorno de réplica.
- Se aplicaron una serie de técnicas para la captura de requisitos y se realizó la descripción de un sistema de administración y configuración para Chronos.
- Se realizó el diseño del sistema utilizando patrones, logrando una estructura general robusta y de fácil mantenimiento.
- Se validó el diseño del sistema aplicando métricas que demostraron que el diseño de clases no presenta dificultades.

RECOMENDACIONES:

A partir de todo el estudio y la investigación realizada para llevar a buen término el presente trabajo recomendamos:

- Realizar la implementación de la herramienta teniendo en cuenta la propuesta de diseño.
- Presentar el trabajo en eventos y publicarlo.
- Incluir en futuras versiones del sistema un modulo que gestione las configuraciones para la resolución de conflictos.

REFERENCIAS BIBLIOGRÁFICAS:

1. **Mustain, A. Elein.** [Online] Noviembre 18, 2004. [Citado el: Enero 20, 2010.] www.onlamp.com/pub/a/onlamp/2004/11/18/slony.html.

2. [Online] Noviembre 3, 2005. [Citado el: Enero 20, 2010.] http://pgcluster.projects.postgresql.org/feature.html.

3. **Rodriguez, Carlos Moreno**. Las esferas socioculturales del Software Libre. [Online] 3 2005. [Citado el: Enero 20, 2010.] http://www.derecho-internet.org/node/293..

4. **Valle, Amaury** E. del. juventudrebelde. [Online] Febrero 14, 2008. [Citado el: Enero 25, 2010.] http://www.juventudrebelde.cu/cuba/2008-02-14/software-libre-ii-una-estrategia-decisiva-de-desarrollo/.

5. [Online] 2010. [Citado el: Febrero 3, 2010.] http://www.masadelante.com/faqs/html.

6.[Online], Febrero 5, 2008. [Citado el: Febrero 3, 2010.] https://developer.mozilla.org/index.php?title=Es/Gu%C3%ADa_JavaScript_1.5/Concepto_de_JavaScript.

7. **Alvarez, Rubén**. [Online] Julio 14, 2009. [Citado el: Febrero 3, 2010.] http://www.desarrolloweb.com/articulos/243.php.

8. [Online] Enero 14, 2007. [Citado el: Febrero 3, 2010.] http://casidiablo.net/%C2%BFque-es-jsp/.

9. **S, Christian Van Der Henst**. [Online] Mayo 23, 2001. [Citado el: Febrero 10, 2010.] http://www.maestrosdelweb.com/editorial/phpintro/.

10. **Alvarez, Sara**. [Online] Julio 31, 2007. [Citado el: Febrero 13, 2010.] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

11. **Gutiérrez., Javier J**. ¿Qué es un framework web? 2008.

12.**Garnet,Joan.** [Online] Octubre 25, 2007. [Citado el: Febrero 13, 2010.] http://www.joangarnet.com/blog/?p=415.

13. **Corzo, Giancarlo**. [Online] Octubre 22, 2008. [Citado el: Febrero 13, 2010.] http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

14. **Sanchez, María A. Mendoza**. [Online] Junio 7, 2004. [Citado el: Febrero 15, 2010.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

15. **Orallo, Enrique Hernández**. El Lenguaje Unificado de Modelado(UML). Valencia,España : s.n., 2007.

16. **INFORMATICA, INSTITUTO NACIONAL DE ESTADISTICA E.** *Herramientas Case.* 1999.

17. Build Quality Applications Faster, Better and Cheaper . [En línea] 2009. [Citado el: 18 de enero de 2010.] http://www.visual-paradigm.com/product/vpuml/modeleredition.jsp.

18. Paradigm Visual. visual-paradigm.com/product/vpuml/. [En línea] Visual Paradigm. , 2008. [Citado el: 18 de Febrero de 2010.] http://www.visual-paradigm.com/product/vpuml/.

19. **Martínez, Jaime Solís.** [En línea] 2008. [Citado el: 5 de febrero, 2010.] http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.

20. **Kontoya, G y Sommerville, I.** Requirements Engineering: Processes and Techniques. 2000.

21. **Young, Ralph R.** The Requirements Engineering Handbook. Londres: s.n., 2004. s.n.

22. **Chaves, Michael Arias.** La ingenieria de requerimientos y su importancia en el desarrollo de proyectos de software. San José : s.n., 2005.

23. **Larman, Craig**. UML y Patrones. 1999.

24. **Jacobson, I. and Booch, G. y Rumbaugh, J**. El Proceso Unificado de Desarrollo de Software. p. 110.

25. **Jacobson, I. and Booch, G. y Rumbaugh, J.** El Proceso Unificado de Desarrollo de Software. p. 168.

26. **Jacobson, I. and Booch, G. y Rumbaugh, J.** El Proceso Unificado de Desarrollo de Software. p. 174-176.

27. **Jacobson, I. and Booch, G. y Rumbaugh, J**. El Proceso Unificado de Desarrollo de Software. p. 208.

28. Vistos desde otros horizontes. Revista Atix. 8, 2009. http://www.softwarelibre.org.bo/esteban/files/86/226/atix08.pdf

29. **Marcello Visconti, Hernán Astudillo**. Fundamentos de Ingeniería de Software. [En línea] [Citado el: 05 de abril de 2009.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

30. **Colectivo de autores**. Arquitectura y Patrones de diseño.2008-2009.

31**. Instituto de Ingeniería Eléctrica (IIE).** Instituto de Ingeniería Eléctrica. sitio Web Instituto de Ingeniería Eléctrica de la Universidad de la República. [En línea] 2007. [Citado el: 20 de Febrero de 2010.] http://iie.fing.edu.uy.

32. **Jacobson, I. and Booch, G. y Rumbaugh, J.** El Proceso Unificado de Desarrollo de Software. p. 217.

33. **Torres, Manuel Lagos.** El Rincón del Programador. sitio Web El Rincón del Programador. [En línea] 28 de Diciembre de 2002. [Citado el: 10 de Marzo de 2010.] http://www.elrincondelprogramador.com/default.asp?pag=infoAutor.asp.

34. **Pressman, Roger S.** Ingeniería de Software, un enfoque práctico. Quinta edición. s.l. : McGraw-Hill Companies, 2002. pág. 640. ISBN: 8448132149.

BIBLIOGRAFÍA:

BRUEGGE, B. D., A. *Ingeniería de Software Orientado a Objetos*. Prentice Hall Pearson Educación. 2002.

Desarrollo de aplicaciones multiplataforma con NetBeans IDE. (n.d.). Obtenido de http://www.sun.com/emrkt/innercircle/newsletter/latam/0207latam_feature.html

El Lenguaje Unificado de Modelado (UML). (n.d.). Obtenido de http://www.disca.upv.es/enheror/pdf/ActaUML.PDF

Fundamentos de Ingeniería de Software. (n.d.). Obtenido de http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf

HANSEN, G. W. H., JAMES V. . . Diseño y Administración de Bases de Datos HALL., P.

Jacobson, I. and Booch, G. y Rumbaugh, J. "El Proceso Unificado de Desarrollo de software". 2000. 2000.

Joshua Drake, J.W., Practical PostgreSQL. 530.

Lenguajes de Programación. Programación Java. (n.d.). Obtenido de http://www.lenguajes-de-programacion.com/programacion-java.shtml

Pressman, R.S. Ingeniería de Software. Un enfoque práctico (reproducción). 2005, La Habana: Félix Varela.

Pressman, Roger S. Ingeniería de Software, un enfoque práctico. Quinta edición. s.l. : McGraw-Hill Companies, 2002.

¿Qué significa y qué ventajas aporta la arquitectura en tres capas? . (n.d.). Obtenido de http://www.solmicro.es/inicio.php?ID_CATEGORIA=25

REYNOSO, C. B. Introducción a la Arquitectura de Software, 2004.

RUMBAUGH, J. J., IVAR; BOOCH, GRADY El proceso unificado de desarrollo. Addison Wesley. 2000a. p.

RUMBAUGH, J. J., IVAR; BOOCH, GRADY. El lenguaje unificado de modelado. Addison Wesley. 2000b. p.

Visual Paradigm for UML. (n.d.). Obtenido de http://www.versioncero.com/?pg=34

ANEXOS:

Anexo #1: Artefacto CU Expandidos.

Anexo #2: Artefacto Diagramas de Clases de Análisis.

Anexo #3: Artefacto Diagramas de Colaboración.

Anexo #4: Artefacto Patrones de Diseño.

Anexo #5: Artefacto Diagramas de Clases del Diseño.

Anexo #6: Artefacto Clases más Significativas.

GLOSARIO DE TÉRMINOS:

A

AD-HOC: Es posible realizar cambios sin tener que detener el sistema (en caliente).

APACHE: Servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1.

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

Array: Es un conjunto o agrupación de variables del mismo tipo cuyo acceso se realiza por índices. Existen dos tipos de array: array asociativos y array Índice numéricos.

B

Binding: En informática, un binding es una "ligadura" o referencia a otro símbolo más largo y complicado y que se usa frecuentemente. Este otro símbolo puede ser un valor de cualquier tipo, numérico, de cadena, etc. o el nombre de una variable que contiene un valor o un conjunto de valores. En el campo de la programación, un binding es una adaptación de una biblioteca para ser usada en un lenguaje de programación distinto de aquél en el que ha sido escrita.

Bugs: Es un error o un defecto en el software o hardware que hace que un programa funcione incorrectamente.

C

C: Es un lenguaje de programación creado en 1972 por Kenneth L. Thompson y Dennis M. Ritchie en los Laboratorios Bell.

CGI: Common Gateway Interface / Interface de Acceso Común o Interfaz Común de Puerta de Enlace. Es una importante tecnología de la World Wide Web que permite a un cliente (explorador web) solicitar datos de un programa ejecutado en un servidor web.

Cluster: Es un grupo de múltiples ordenadores unidos mediante una red de alta velocidad, de tal forma que el conjunto es visto como un único ordenador, más potente que los comunes de escritorio. De un sistema de este tipo se espera que presente combinaciones de los siguientes servicios:

1. Alto rendimiento
2. Alta disponibilidad

3. Equilibrio de carga

4. Escalabilidad

Copyleft: Comprende a un grupo de derechos de autor caracterizados por eliminar las restricciones de distribución o modificación impuestas por el copyright, con la condición de que el trabajo derivado se mantenga con el mismo régimen de derechos de autor que el original.

E

EntityManager: Con la llegada de EJB 3 y JPA nace la figura del EntityManager para simplificar la persistencia de objetos.

Esclavo: En la replicación de bases de datos los servidores Slave (esclavo) leen las consultas almacenadas en el log binario del servidor Master y las ejecutan localmente, de esta manera mantienen una copia exacta de los datos almacenados en el Master.

G

GPL: Es una licencia creada por la Free Software Foundation y orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software Libre (General Public License).

H

Hibernate: Es una herramienta de mapeo objeto-relacional 38 para la plataforma Java. Facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos XML. Hibernate busca solucionar el problema de la diferencia entre estos dos modelos: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional).

Hosting: El alojamiento web (en inglés web hosting) es el servicio que provee a los usuarios de Internet un sistema para poder almacenar información, imágenes, vídeo, o cualquier contenido accesible vía Web. Es una analogía de hospedaje o alojamiento en hoteles o habitaciones donde uno ocupa un lugar específico, en este caso la analogía alojamiento web o alojamiento de páginas web, se refiere al lugar que ocupa una página web, sitio web, sistema, correo electrónico, archivos etc. en Internet o más específicamente en un servidor que por lo general hospeda varias aplicaciones o páginas web.

I

Interfaz: En informática, una interfaz es la parte del programa informático que permite el flujo de información entre varias aplicaciones o entre el propio programa y el usuario. En software también se

habla de interfaz gráfica de usuario, que es un método para facilitar la interacción del usuario con el ordenador o la computadora a través de la utilización de un conjunto de imágenes y objetos pictóricos (iconos, ventanas, formularios, páginas web...).

J

Java: Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los 90.

JPA: Java Persistence API, más conocida por su sigla JPA, es la API de persistencia desarrollada para la plataforma Java EE e incluida en el estándar EJB3. Esta API busca unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional.

L

Layout: El layout es el esquema de distribución, lógico y ordenado de un sistema y es usado como herramienta para optimizar procesos o sistemas.

M

Maestro: En la replicación de bases de datos el servidor Master (maestro o amo) es donde se realizan todas las consultas de modificación de datos las cuales se almacenan en el log binario del servidor.

Maestro-Esclavo: Un Esclavo puede tener un único maestro. Un maestro puede tener múltiples esclavos. Los maestros envían las modificaciones realizadas en las BD a los esclavos. Los esclavos no pueden modificar la BD a su antojo, solamente recibir las modificaciones del maestro y aplicarlas. En resumidas cuentas, como máximo un maestro (escritura) y múltiples esclavos (lectura).

Multi-Maestro: La replicación multi-maestro también conocida peer to peer o replicación de n visas está compuesta de múltiples sitios maestros que participan igualmente en un modelo que soporta actualizaciones desde cualquier sitio. Las actualizaciones a un sitio individual son propagadas a todos los demás sitios maestros que participan.

N

Nodo: En términos de la propuesta para replicar base de datos fragmentados, son dos bases de datos PostgreSQL que intervienen en la réplica.

O

Open Source: Es el término con el que se conoce al software distribuido y desarrollado libremente.

ORM: El mapeo objeto-relacional (más conocido por su nombre en inglés, Object-Relational mapping) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.

P

Perl: Practical Extraction and Report Language. Es un lenguaje de programación desarrollado por Larry Wall inspirado en otras herramientas de UNIX como son: sed, grep, awk, c-shell.

Plugins: Un plugins o componente enchufable es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad muy específica.

Pooling: Memoria caché de conexiones de base de datos gestionada por la base de datos.

Push: Los servicios Push están basados a menudo en preferencias de información a medida. Es decir, un modelo publicador/suscriptor. Un cliente deberá suscribirse a varios canales de información. Cuando el nuevo contenido está disponible en uno de estos canales, el servidor deberá enviar la información al usuario. Las conferencias sincronizadas y la mensajería instantánea son ejemplos típicos de los servicios tipo 'push'. Los mensajes de chat y, en ocasiones archivos, son enviados al usuario tan pronto estos son recibidos por el sistema de mensajería. Los programas descentralizados P2P (como WASTE) y los centralizados (como IRC o Jabber) permiten hacer 'push' de archivos. Es decir, el remitente inicia la transferencia de datos, en vez del destinatario.

R

Refactorización: El término se usa a menudo para describir la modificación del código fuente sin cambiar su comportamiento. Su objetivo es el mantenimiento del código que no arregla errores ni añade funcionalidad, sólo el mejorar la facilidad de comprensión del código o cambiar su estructura y diseño y eliminar código muerto.

Release: Nueva versión de una aplicación informática.

Render: Renderizado (render en inglés) es un término usado en jerga informática para referirse al proceso de generar una imagen desde un modelo.

Replicación en cascada: En una configuración de replicación en cascada, un servidor que actúa como un concentrador recibe actualizaciones desde un servidor que actúa como proveedor. El centro de repeticiones actualiza los cambios a los consumidores. Replicación en cascada es útil en las situaciones siguientes:

- Cuando hay una gran cantidad de consumidores.

- Debido a que los maestros en una topología de replicación de manejar todo el tráfico de actualización, que podría ponerlos bajo una carga pesada para soportar el tráfico de replicación a los consumidores.
- Puede cargar el tráfico de replicación a varios centros en cada replicación del servicio de actualizaciones a un subconjunto de los consumidores.
- Para reducir los costes de conexión mediante el uso de un centro local en entornos distribuidos geográficamente.

Rollback's: En tecnologías de base de datos, un rollback es una operación que devuelve a la base de datos a algún estado previo. Los Rollback's son importantes para la integridad de la base de datos, a causa de que significan que la base de datos puede ser restaurada a una copia limpia incluso después de que se han realizado operaciones erróneas. Son cruciales para la recuperación de crashes de un servidor de base de datos; realizando rollback (devuelto) cualquier transacción que estuviera activa en el tiempo del crash, la base de datos es restaurada a un estado consistente.

S

Scripts: Un conjunto de comandos escritos en un lenguaje interpretado.

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

Subconsultas: Una subconsulta es una instrucción SELECT anidada dentro de una instrucción SELECT, SELECT…INTO, INSERT…INTO, DELETE, o UPDATE o dentro de otra subconsulta.

T

Tipeados: Teclear, escribir a máquina.

Trigger(o disparador): En una Base de datos, es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).

U

UNIX: Sistema operativo portable, flexible, potente, con entorno programable, multiusuario y multitarea, muy difundido.