Universidad de las Ciencias Informáticas

Facultad 3

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

"Herramienta para la configuración y administración del sistema gestor de base de datos PostgreSQL para el marco de trabajo Sauxe".

Autor: Rafael Ventura Noa Pelier.

Tutor: Ing. Ariel Torres Gálvez.

La Habana, Junio de 2011

"Año 53 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser autor del presente trabajo y reconocer a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

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

Rafael Ventura Noa Pelier

_____

Firma del Autor

Ing. Ariel Torres Gálvez

_____

Firma del Tutor

"La Educación es el pasaporte hacia el futuro, el mañana pertenece a aquellos que se preparan para él en el día de hoy."

Malcolm X.

Agradecimientos

A mis padres por siempre estar a mi lado en todos estos años apoyándome y dándome los consejos necesarios para seguir el largo camino de la vida.

A mi hermana y a mi cuñado Vila que siempre estuvieron presentes dándome aliento para continuar con el cumplimiento de mis sueños.

A mi novia que siempre estuvo conmigo desde el comienzo, brindándome su apoyo y tratando de alegrarme cada día con sus conversaciones.

A mi tutor Ariel por su ayuda y por siempre haberme indicado el camino correcto para dar cumplimiento a los objetivos de mi trabajo de diploma.

El más sincero agradecimiento a todas las personas que de una forma u otra ha colaborado con la elaboración de este trabajo.

Al equipo de compañeros con los cuales he pasado varios años de estudio y a todos mis compañeros de clases.

A mi mamá por siempre haberme brindado su apoyo, comprensión y cariño.

A mi papá que estuvo presente en todo momento dándome el empujón necesario para salir adelante, siendo un guía y transmitiéndome sus experiencias vividas.

A mi hermana y a mí cuñado Vila que siempre confiaron en mí.

A mis familiares que siempre fueron un digno ejemplo para seguir adelante.

A mi novia que estuvo a mi lado todo el tiempo ofreciéndome su incondicional apoyo, siempre ayudándome para cumplir con el sueño de ser un profesional.

Resumen

En la Universidad de las Ciencias Informáticas (UCI) se desarrolla el Sistema de Planificación de Recursos Empresariales (ERP), denominado Cedrux. Para la administración de los servidores de datos con los que se trabaja en este sistema, se desarrolló una herramienta basada en el marco de trabajo del mismo. Dicha herramienta permite realizar tareas de administración y configuración de varios servidores de datos a la vez. Las tareas antes mencionadas pueden realizarse a estos servidores localmente o mediante una conexión remota.

En este documento se presenta el diseño de la herramienta desarrollada así como sus principales funcionalidades. Se exponen las ventajas que ofrece sobre otros sistemas semejantes, pero por razones que se explican más adelante, no son aptos para ser usados en el entorno que se menciona.

Finalmente, se presenta el sistema, como una aplicación web guiada por las políticas de desarrollo del proyecto: de código abierto y multiplataforma. Se explican las métricas aplicadas al diseño, las pruebas de caja blanca y la validación de la herramienta por el departamento de calidad del centro.

Palabras claves: sistema gestor de base de datos, administrar, configurar.

Índice de contenidos

Tabla de contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la UCI se realiza el desarrollo de varios proyectos informáticos, en los cuales la dirección del país presenta cierto interés. En la universidad existen varios centros de desarrollo, en los cuales se encuentran distribuidos los proyectos que se están desarrollando en la institución. El Centro para la Informatización de la Gestión de Entidades (CEIGE), es el encargado del desarrollo del sistema ERP-Cubano y del marco de trabajo Sauxe, el cual contiene un conjunto de componentes reutilizables que provee una estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo.

Para el desarrollo de este sistema ERP, debido a su complejidad y tamaño se tomó la decisión de ubicar el personal vinculado al centro de producción en subdirecciones, para así poder conformar los departamentos productivos.

En el departamento de tecnología es donde se desarrollan las herramientas tecnológicas para facilitar el desarrollo de las soluciones de gestión. En este departamento trabajan los administradores de los servidores de datos, los cuales controlan y definen el ambiente donde funcionan los servidores de las bases de datos que administran.

Entre las principales funciones que debe desarrollar un administrador de Sistemas Gestores de Bases de Datos (SGBD), está la de garantizar confidencialidad, integridad y disponibilidad a la información almacenada. Así como gestionar que los usuarios trabajen de forma cooperativa y complementaria. Los administradores de los SGBD del departamento de tecnología presentan la dificultad de configurar y administrar los servidores de datos que se encuentra remotos. Ya que tienen que realizar las tareas de configuración y administración mediante conexiones remotas a través de la consola (Shell de Unix) en el sistema operativo Linux.

Para cada servidor que se desee configurar o administrar es necesario abrir una nueva conexión lo cual resulta muy tedioso para realizar dichas tareas, además de no poder realizar varias operaciones al mismo tiempo en un único servidor. Cuando es necesario realizar cambios en la configuración o en las conexiones permitidas por el SGBD es muy engorroso localizar los parámetros que se desean modificar debido a que la carga del fichero es en la consola.

En el momento en que se desean aplicar las operaciones de mantenimiento a las bases de datos o de respaldo de la información que se encuentra almacenada en estos servidores remotos, los

administradores pierden tiempo, esperando a que se terminen de aplicar las tareas de administración que se están ejecutando en la consola. Para poder ejecutar otras tareas deben de cambiar a otra herramienta o cambiar de entorno.

De tal modo es necesario que el marco de trabajo Sauxe cuente con herramientas o mecanismos que le permitan realizar tareas administrativas sobre los servidores de bases de datos, mediante la web y usando una estricta seguridad en las conexiones realizadas a los servidores.

De acuerdo a lo antes expuesto se ha definido el siguiente **problema a resolver:**

¿Cómo optimizar el proceso de configuración y administración del gestor de base de datos PostgreSQL en el marco de trabajo Sauxe?

Tomando en este caso como **objeto de estudio:**

Herramientas para la administración de Sistemas Gestores de Base de Datos.

Para dar solución a esta problemática se determinó como **objetivo general:**

Desarrollar una herramienta que permita la configuración y administración del gestor de base de datos PostgreSQL en el marco de trabajo Sauxe.

Del objetivo general se desglosan los siguientes **objetivos específicos:**

1. Fundamentar la investigación mediante la evaluación del marco teórico.
2. Definir una solución de diseño que cumpla con los requerimientos especificados en el análisis.
3. Implementar los componentes de la herramienta.
4. Validar la herramienta.

Centrado en el **campo de acción:**

Administración del Sistema Gestor de Base de Datos PostgreSQL.

En correspondencia con el objetivo general se establecieron las siguientes **tareas investigativas:**

1. Confeccionar el marco teórico conceptual de la investigación a partir de una búsqueda y revisión bibliográfica.
2. Investigar sobre herramientas de administración de base de datos.
3. Investigar arquitectura del gestor PostgreSQL.
4. Asumir los estándares de codificación definidos por el marco de trabajo.
5. Estudiar el marco de trabajo del proyecto.
6. Identificar las herramientas y tecnologías a emplear para el desarrollo de la aplicación.
7. Diseñar las clases por módulos del componente.

8. Analizar los artefactos generados durante la etapa de análisis del componente.
9. Investigar sobre patrones de diseño para la elaboración de los modelos de clases.
10. Implementar los componentes diseñados.

Se define la siguiente **idea a defender:**

Si el marco de trabajo Sauxe contara con una herramienta web que permita realizar tareas de administración y configuración del servidor de datos, se obtendría una mejora en el proceso de configuración y administración del sistema gestor de base de datos.

El presente trabajo se desarrolla en 3 **capítulos**:

Capítulo 1: Fundamentación Teórica. Se realizarán explicaciones de los distintos conceptos utilizados en el documento. Este capítulo incluye un estado del arte del tema tratado a nivel internacional, nacional y de la universidad. Se analizarán también las diferentes tecnologías, lenguajes y metodologías de desarrollo de software empleadas en la realización del sistema.

Capítulo 2: Propuesta de Solución. En este capítulo se hace un diagnóstico del campo de acción y a partir de esto se muestra la propuesta de solución mediante diagramas y documentos de lo que será el futuro sistema.

Capítulo 3: Construcción y validación de la propuesta de solución. Se expondrá el resultado de la implementación del sistema. Se mostrará el resultado final de las pruebas realizadas al sistema para verificar la eficiencia de la aplicación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

En este capítulo se hace referencia a la fundamentación teórica del trabajo, donde se lleva a cabo la realización de un estudio tanto conceptual como del estado del arte. Con el objetivo principal de detectar los diferentes sistemas y aplicaciones que tengan algunas funcionalidades semejantes a las del sistema que se quiere lograr. Se realiza un estudio de las herramientas y tecnologías informáticas que están definidas por el marco de trabajo Sauxe para el desarrollo de aplicaciones web, los lenguajes de programación, los controladores de versiones de proyectos informáticos y navegadores web. También se realiza un estudio de las metodologías empleadas para el desarrollo de la herramienta y definir así los artefactos generados con el objetivo de desarrollar la aplicación.

1.2 Base de datos relacionales.

Las bases de datos relacionales se basan en el modelo relacional, cuya estructura principal es una tabla bidimensional compuesta por filas y columnas. En las bases de datos relacionales toda la información es visible al usuario y está organizada estrictamente como tablas de valores. Las operaciones se realizan sobre estas tablas que se relacionan de forma normalizada y con diversos grados que varían con el tiempo. Entre las ventajas de las bases de datos relacionales se encuentran:

- Garantizar herramientas que eviten la duplicidad de registros, a través de campos clave o llaves.
- Garantizar la integridad referencial. Si se excluye un registro se eliminan todos los registros relacionados dependientes.
- Favorecer la normalización por ser más comprensibles y aplicables.

1.2.1 Sistema Gestor de Base de Datos (SGBD).

Sistema de Gestión de Bases de Datos, es un software muy específico, dedicado a servir de interfaz entre la base de datos y el usuario. Los SGBD proporcionan un interfaz entre aplicaciones y sistema operativo, consiguiendo, entre otras cosas, que el acceso a los datos se realice de una forma más eficiente, fácil de implementar y segura.

Los SGBD cuentan con un lenguaje para que el usuario pueda trabajar con él, el más conocido es el SQL (acrónimo que significa *Structured Query Language*) en español lenguaje estructurado de consultas. (1)

SQL

Es un lenguaje declarativo de base de datos relacionales, que permite especificar diversas operaciones sobre estas. Su manejo del álgebra y el cálculo relacional conforman una de sus principales características, para efectuar consultas con el fin de realizar recuperaciones de la información almacenada en las bases de datos.

SQL está compuesto por dos lenguajes para el trabajo con la información almacenada, como son: (2)

- Lenguajes de definición de datos (DDL): Creación de esquemas, modificación de los mismos.
- Lenguaje de manipulación de datos (DML): Creación, Modificación, Eliminación y Obtención de Datos.

Arquitectura de PostgreSQL

PostgreSQL funciona con una arquitectura Cliente/Servidor, un proceso servidor (*postmaster*) y una serie de aplicaciones cliente que realizan solicitudes de acciones contra la base de datos a su proceso servidor. Por cada una de estas aplicaciones cliente, el proceso *postmaster* crea un proceso postgres.(3) (Ver Ilustración 1)

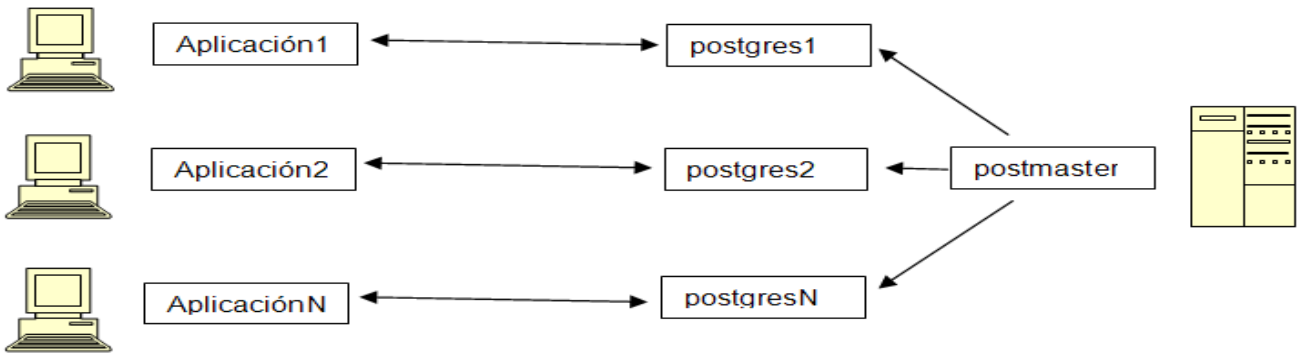

Ilustración 1 Arquitectura de PostgreSQL.

PostgreSQL

PostgreSQL es un potente Sistema de Base de Datos Relacional (RDBMS) libre (*Open Source*, su código fuente está disponible) liberado bajo licencia Berkeley Software Distribución (BSD). Desarrollado en la Universidad de California, en el departamento de ciencias de la computación de Berkeley. Posee más de 15 años de activo desarrollo y arquitectura probada que se ha ganado una muy buena reputación por su confiabilidad e integridad de datos. Funciona en todos los sistemas operativos importantes, incluyendo Linux, UNIX (AIX, HP-UX, SGI IRIX, Mac OS, Solaris, Tru64), y Windows. El nombre como tal de PostgreSQL surge en el año 1996 para establecer una relación entre el nombre original PostgreSQL y las versiones más recientes con capacidades SQL. Entre las principales características del sistema tenemos:

- Soporta casi toda la sintaxis SQL tiene soporte total para llaves foráneas (*foreign keys*), uniones (*joins*), vistas (*views*), disparadores (*triggers*), y procedimientos almacenados (*stored procedures*) en múltiples lenguajes.
- Cliente/Servidor: PostgreSQL usa una arquitectura proceso por usuario cliente/servidor. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.
- Lenguajes Procedurales: PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido además de en C, C++ y Java.
- Interfaces con lenguajes de programación: La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas Interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, Pike, etc.
- Herencia de tablas.
- Incluye la mayoría de los tipos de datos SQL92 y SQL99 (INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, y TIMESTAMP), soporta almacenamiento de objetos grandes binarios, tipos de datos y operaciones geométricas.

- Puntos de recuperación a un momento dado, *tablespaces*, replicación asincrónica, transacciones jerarquizadas (*savepoints*), copia de seguridad en línea.
- Un sofisticado analizador/optimizador de consultas.
- Soporta juegos de caracteres internacionales, codificación de caracteres *multibyte*.
- Alta concurrencia: mediante un sistema denominado MVCC (Acceso concurrente multiversión). PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo envío de transacción. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.
- Amplia variedad de tipos de datos nativos:
  - Números de precisión arbitraria.
  - Texto de largo ilimitado.
  - Figuras geométricas (con una variedad de funciones asociadas).
  - Direcciones IP (IPv4 e IPv6).
  - Bloques de direcciones estilo CIDR.
  - Direcciones MAC.
  - Arreglos.
  - Los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.
- Claves ajenas (Llaves ajenas, Claves Foráneas o *Foreign Keys*).
- Disparadores (*triggers*): un disparador o *trigger* se define en una acción específica basada en algo ocurrente dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica. Ahora todos los disparadores se definen por seis características:
  - El nombre del disparador o *trigger*.
  - El momento en que el disparador debe arrancar.
  - El evento del disparador deberá activarse sobre alguna tabla.
  - La tabla donde el disparador se activará.

- La frecuencia de la ejecución.
- La función que podría ser llamada.

- Vistas (*Views*).
- Integridad transaccional.
- Tipos de datos y operaciones geométricas.
- Soporte para transacciones distribuidas: PostgreSQL permite integrarse en un sistema distribuido formado por varios recursos (por ejemplo, una base de datos PostgreSQL, otra Oracle, una cola de mensajes IBM MQ JMS y un ERP SAP) gestionado por un servidor de aplicaciones donde el éxito ("*commit*") de la transacción global es el resultado del éxito de las transacciones locales.

El sistema cuenta con las siguientes características:

- Máximo tamaño de base de datos ilimitado.
- Máximo tamaño de tabla 32 TB.
- Máximo tamaño de tupla 1.6 TB.
- Máximo tamaño de campo 1 GB.
- Máximo tuplas por tabla ilimitado.
- Máximo columnas por tabla 250 - 1600 dependiendo de los tipos de columnas.
- Máximo de índices por tabla ilimitado. (4)

Estructura de ficheros del sistema gestor PostgreSQL

- **postgresql.conf:** Fichero de configuración principal, contiene la asignación a los parámetros que configuran el funcionamiento del servidor.
- **pg_hba.conf:** Fichero de configuración de la autenticación de los clientes, usuarios y del acceso a las bases de datos del clúster.
- **pg_ident.conf:** Fichero accesorio al anterior, determina como se realiza la autenticación ident que contiene la correspondencia entre usuarios del Sistema Operativo y de PostgreSQL.
- **PG_VERSION:** Fichero de texto con la versión de software Postgres que crea el clúster.

Otros ficheros:

- **postmaster.pid:** Se crea cuando el *postmaster* arranca, contiene el PID del proceso *postmaster*.

- **postmaster.opts:** Contiene las opciones con las que se ha iniciado el *postmaster*.
- **recovery.conf, recovery.done:** Si se quiere o se ha hecho una recuperación.

Estructura de directorios del sistema gestor PostgreSQL

- **base:** En ella se encuentran las plantillas y las bases de datos, contiene un directorio por cada base de datos, dentro hay un fichero por cada tabla o índice de una base de datos.
  - template0 (1): Contiene las definiciones de las tablas del sistema, vistas, funciones y tipos estándar. Nunca se debe modificar ni intentar conectarse a él, existe por si template1 se corrompe.
  - template1 (N): Base de datos plantilla para crear nuevas bases de datos, se puede modificar su estructura, añadiendo tablas, índices, funciones, etc.
- **global:** Posee tablas e índices del catálogo comunes a todas las bases de datos.
  - catálogo compartido: *pg_shadow* (usuarios), *pg_database*, etc.
  - *pgstat.stat:* fichero usado por el monitor de estadísticas.
  - *pg_control:* fichero con parámetros del clúster, algunos inmutables (establecidos en la creación del clúster) y otros variables (establecidos en la puesta en marcha).
- **pg_log:** Ficheros de seguimiento del servidor. Se crea en la versión de Windows, en la de Linux, se debe indicar al arrancar el *postmaster* en qué fichero se hace el seguimiento.
- **pg_xlog:** Ficheros de diario del servidor (WAL).
  - Contiene los diarios de escritura adelantada, para usarlos en las recuperaciones.
  - Implementan un conjunto de segmentos (ficheros) de un tamaño de 16Mb y divididos en páginas de 8Kb.
  - Inicialmente se crea un fichero, y luego el sistema va creando más según las necesidades.
- **pg_clog:** Ficheros de diario para las transacciones (estado de cada transacción).
  - Contiene los ficheros de confirmación.
  - Un diario de confirmación refleja el estado de cada transacción: confirmada, en progreso o abortada.
- **pg_multixact:** Contiene datos sobre el estado multi-transaccional, usado para los bloqueos compartidos de filas.

- **pg_twophase:** Ficheros de estado para las transacciones preparadas.
- **pg_subtrans:** Para realizar los "*savepoints*" en medio de transacciones.
- **pg_tblspc:** Información sobre los *tablespaces*. En Linux/Unix contiene enlaces a los directorios donde se crean los *tablespaces* y hay que controlar, en caso de cambios de ficheros, que estén correctos. (4)

1.2.2 Clientes para administrar PostgreSQL.

PgAdmin3

PgAdmin3 es una herramienta cliente para la administración de PostgreSQL, desarrollado por una comunidad de expertos de todo el mundo, está disponible en más de una docena de idiomas. Es Software Libre publicado bajo la licencia artística.
PgAdmin3 es la más popular de las aplicaciones clientes, usadas para la administración y desarrollo de plataforma de PostgreSQL. PgAdmin3 está diseñado para responder las necesidades de todos los usuarios.
La interfaz gráfica de pgAdmin3 hace fácil la administración. La aplicación también incluye un editor de sintaxis SQL, un servidor, editor de código, un SQL / lote / *shell* de la programación de agente de empleo, el apoyo a la replicación con *Slony-I*. (5)

PhpPgAdmin

Es una herramienta de administración PostgreSQL, basada en una interfaz web, entre sus principales características ofrece la posibilidad de administrar varios servidores, soporte a múltiples versiones de PostgreSQL, administración de usuarios, grupos, bases de datos, esquemas, etc. Manipulación sencilla de datos, exportar los datos a diferentes formatos, importar sentencias SQL y mucho más. (6)
PhpPgAdmin es una Aplicación Web, escrita en PHP, para administrar las bases de datos de PostgreSQL. El proyecto comenzó originalmente como un tenedor de PhpMyAdmin, para proporcionar a los usuarios de PostgreSQL con servicios comparables a lo que los usuarios de MySQL habían estado usando. Sin embargo, en 2002 el código fue reescrito desde cero, y ahora es un código base completamente diferente. (7)
Entre sus principales características se encuentran:

- Administra múltiples servidores.
- Soporte para PostgreSQL 7.4.x, 8.0.x, 8.1.x, 8.2.x, 8.3.x, 8.4.x, 9.0.x.
- Administra todos los aspectos de:
    - Los usuarios y grupos.
    - Bases de datos.
    - Esquemas.
    - Cuadros, índices, restricciones, desencadenadores, reglas y privilegios.
    - Puntos de vista, secuencias y funciones.
    - Avanzada objetos.
    - Informes.
- Fácil manipulación de datos:
    - Ver las tablas de puntos de vista e informes.
    - Ejecución arbitraria de SQL.
    - Seleccionar, insertar, actualizar y eliminar.
- Volcado de datos de la tabla en una variedad de formatos: SQL, COPY, XML, XHTML, CSV, con pestañas, Pgdump.
- Importa secuencias de comandos SQL, copiar datos, XML, CSV y pestañas.
- Apoya el motor Slony amo-esclavo de replicación.
- Excelente compatibilidad con el idioma:
    - Disponible en 27 idiomas.
    - No hay conflictos de codificación.
- Fácil de instalar y configurar. (7)

PgAccess

Para los desarrolladores se desplegó una potente herramienta, llamada *PgAccess*. Está programado utilizando las librerías tcl/tk por lo que es un sistema multiplataforma, tcl/tk interactúa con los sistemas operativos (Windows, Unix, Mac, etc.).

Un detalle importante para utilizar *PgAccess* es que cuando se pone en marcha el servidor Postgres se habilitan conexiones a dicho servidor por TCP/IP, ya que *PgAccess* utiliza este tipo de conexión. El modo de hacer esto es añadir el parámetro -i al iniciar el servidor.

PgAccess permite realizar diversas operaciones en las bases de datos. Desde crear tablas y modificarlas hasta realizar esquemas de la base de datos. Se verá una pequeña muestra de su potencia. Incluso incluye un diseñador visual para realizar consultas, uno de formularios y otro de informes. (8)

Luego de un análisis e investigación realizados acerca de las herramientas que son usadas actualmente como clientes para la configuración y administración del sistema gestor de base de datos PostgreSQL, entre los cuales se puede mencionar Pg_Admin3, PhpPgAdmin, PgAccess. Se llegó a la conclusión de que estos sistemas cuentan con muchas funcionalidades administrativas, las que fueron estudiadas a profundidad y aplicadas en la herramienta que se desea desarrollar, algunas de estas funcionalidades de administración son los mantenimientos de (limpieza, análisis, reindexado), operaciones de respaldo (realización de backup y restauración de backup). Para poder integrar las operaciones de administración al marco de trabajo Sauxe es necesario la implementación de un sistema que permita realizar estas funcionalidades, pero debe ser compatible con las características del marco de trabajo ya que los sistemas mencionados anteriormente no cumplen este requisito de compatibilidad, no pueden ser usados, por tal motivo es necesaria la integración de la herramienta con el mismo.

1.3 Metodología de desarrollo de software.

Conjunto de procedimientos, herramientas y técnicas con una documentación que permiten estructurar, planificar y controlar el proceso de desarrollo de software. Para los desarrolladores de software es una guía en el momento de la implementación de un sistema determinado. Para desarrollar un sistema, se bebe escoger la metodología de desarrollo, pero para realizar esta selección influyen varios factores, que permiten escoger, la más recomendable a aplicar en el sistema que se desea implementar. Entre estos factores se encuentran: el personal que participará en el desarrollo del sistema, los recursos disponibles, las necesidades del cliente. Existen dos tipos fundamentales de metodologías para tratar este tema, las que siguen los métodos tradicionales, que son generalmente para software de gran envergadura, y las que proponen procesos ágiles para el desarrollo, más usadas en sistemas pequeños y con marcadas diferencias entre sí. Para este caso se centra el estudio de las metodologías ágiles

dado el tipo de sistema en cuestión, y poniendo en práctica el modelo de desarrollo utilizado en el proyecto, descrito en el trabajo de diploma de los Ingenieros en Ciencias Informáticas Sergio Hernández Cisneros y Mileidy Magalys Sarduy Pérez. Dicho modelo en resumen, plantea lo siguiente:

"En aras de mejorar la organización, control y rendimiento en el proceso de desarrollo de software tecnológico en la Departamento de Tecnología del Centro de Informatización de la Gestión de Entidades (CEIGE), se ha decidido organizar los procesos que deben desarrollarse en la misma.

Características del modelo de desarrollo:

- Se utilizan solamente los artefactos necesarios para documentar el producto.
- Se basa en la reutilización de componentes.
- Se desarrollan partes pequeñas y se ensambla después el producto.
- Los flujos se integran a través de la arquitectura de software y de negocio. Todos los flujos de desarrollo de cada fase se integran siempre detrás de la arquitectura de software y de negocio.
- Existen áreas dedicadas a tareas específicas y especializadas en temas específicos.
- Se hacen pruebas continuas sobre los compones y/o productos y los cambios se hacen a tiempo. Antes de poner un componente en el repositorio se hacen pruebas unitarias y cuando se va a utilizar como parte de otro producto se hacen pruebas de integración, al igual que antes de liberar el producto también. Todo esto demuestra que se está probando en todo el proceso de desarrollo.
- Las áreas de proceso están especializadas, en temas de apoyo a la producción que es el elemento fundamental de la Subdirección y llevan unido al proceso productivo procesos tales como investigación, formación, gestión del capital humano y calidad.
- Es un método muy estructurado que funciona bien con gente de poca experiencia.
- Reduce los riesgos porque:
  - Provee visibilidad sobre el progreso a través de sus nuevas versiones.
  - Provee retroalimentación a través de la funcionalidad mostrada.
  - Permite atacar los mayores riesgos desde el inicio.
- La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software.
- En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo y según esto ajusta su comportamiento.

- Preocupación por el aprendizaje de los desarrolladores". (24)

En el modelo de desarrollo se generan varios artefactos los cuales son de gran importancia para el desarrollo de un sistema web. Estos artefactos son necesarios para documentar el producto durante el ciclo de vida del desarrllo del software. En cada uno de los flujos de tabajo (Análisis y diseño, Implementación y Prueba) se van elaborando los artefactos correspondientes, para lograr el entendimiento de todas las personas implicadas en la solución que se desea lograr.

Entre los artefactos principales se encuentran:

- Descripción de los procesos del negocio.
- Especificación de los requisitos funcionales del sistema.
- Especificación de los requisitos no funcionales del sistema.
- Diagrama de clases del diseño.
- Diagrama del modelo de datos.
- Diagrama de componentes.
- Diagrama de despliegue.

1.4 Tecnologías y herramientas para el desarrollo.

Las tecnologías son un conjunto de conocimientos técnicos, ordenados científicamente, que permiten diseñar, crear bienes, servicios que facilitan la adaptación al medio ambiente y satisfacer tanto las necesidades esenciales como los deseos de las personas. Con la aplicación de las tecnologías en vista al desarrollo de la sociedad, influye notablemente la disminución del esfuerzo realizado por las personas para lograr algún objetivo, siempre y cuando hagan uso adecuado de las tecnologías.

1.4.1 Herramientas CASE.

CASE es una sigla, que corresponde a las iniciales de: (*Computer Aided Software Engineering*) y en su traducción al español significa Ingeniería de Software Asistida por Computación. Estas herramientas permiten organizar y manejar la información de un proyecto informático. Permitiéndole a los participantes de un proyecto, que los sistemas (especialmente los complejos), se tornen más flexibles, más comprensibles y además mejorar la comunicación entre los participantes.

Utilizándolas se puede mejorar la productividad en el desarrollo, la calidad del software, reducir el tiempo y costo de desarrollo, así como el mantenimiento de los sistemas informáticos, mejorar la planificación de un proyecto, aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos, automatizar el desarrollo del producto, la documentación, la generación de código, las pruebas de errores y la gestión del proyecto, ayuda a la reutilización del software, portabilidad y estandarización de la documentación, gestión global en todas las fases de desarrollo de software con una misma herramienta y facilitar el uso de las distintas metodologías propias de la ingeniería del software. (9)

Entre sus características fundamentales tenemos:

- Permitir la aplicación práctica de metodologías estructuradas, las cuales al ser realizadas con una herramienta se consigue agilizar el trabajo.
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
- Simplificar el mantenimiento de los programas.
- Mejorar y estandarizar la documentación.
- Aumentar la portabilidad de las aplicaciones.
- Facilitar la reutilización de componentes software.
- Permitir un desarrollo y un refinamiento visual de las aplicaciones, mediante la utilización de gráficos.

Visual Paradigm

Visual Paradigm es una herramienta profesional, que utiliza “UML”: como lenguaje de modelaje. Esta herramienta está diseñada con el objetivo de la creación de diseños basados en el paradigma de la programación orientada a objetos. Incluye una herramienta llamada *“Visual Architect”* que permite la generación de código para el manejo de la base de datos y puede generar códigos para lenguajes como PHP, JAVA, C# y para los gestores de base de datos PostgreSql, SQL, MySQL, entre otros. Esta herramienta es muy utilizada por analistas, arquitectos de software, diseñadores y desarrolladores con el fin de realizar sus diseños y modelos durante el desarrollo de software. Realizando una buena explotación de la herramienta se puede ganar mucho tiempo y agilizar la realización de los diseños y diagramas necesarios en el desarrollo de un software de buena calidad. (10)

1.4.2 Herramienta para el desarrollo colaborativo.

RapidSVN

RapidSVN es un cliente gráfico que permite manipular los repositorios de Subversión. Es una de las alternativas más conocidas para el sistema GNU/Linux para poder interactuar con el repositorio, es muy intuitivo y fácil de utilizar. Una de las características más importantes de este sistema de control de versiones es que puede acceder al repositorio a través de diferentes redes lo que permite que varias personas puedan hacer uso del mismo conjuntamente, posibilitando que el trabajo se realice con mayor rapidez. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer porque la calidad del mismo vaya a verse afectada, si se ha hecho un cambio incorrecto a los datos, simplemente basta con deshacer ese cambio. Entre las ventajas de RapidSVN se encuentran:

- Sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- La creación de ramas y etiquetas es una operación más eficiente.
- Puede ser servido mediante Apache.
- Maneja eficientemente archivos binarios.
- Permite selectivamente el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez.

Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor web provee en el momento de autentificar archivos. (11)

1.4.3 Herramientas para el desarrollo.

Apache

Apache es uno de los mayores triunfos del software libre. Es un servidor web HTTP de código abierto para plataformas como GNU/Linux, Microsoft Windows, Macintosh y otras, dentro de sus principales características están las siguientes:

- Corre en disímiles sistemas operativos, lo que lo hace prácticamente universal.
- Apache es una tecnología gratuita de código fuente, abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si queremos ver que es lo que estamos instalando como servidor, lo podemos saber, sin ningún secreto.
- Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este y están ahí para que se instalen cuando se necesiten.
- Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.

Apache y GNU/Linux es una combinación que se está utilizando en el mundo empresarial, ha ayudado a que el campo de GNU/Linux se amplíe de forma muy sólida en el mundo Internet. Este servidor web acapara casi el 60 por ciento de cuota de mercado y desde su nacimiento en 1995 ha sido el servidor web preferido por millones de *webmaster* y es una de las aplicaciones de código abierto más importantes del mercado del software. (12)

PostgreSQL 8.3

Es un sistema de gestión de bases de datos libre basado en el proyecto Postgres, perteneciente a la Universidad de Berkeley. Es un sistema objeto-relacional, que incluye características como la herencia, valores no atómicos (atributos basados en vectores y conjuntos), funciones, disparadores, entre otras. Es altamente extensible, permitiendo el uso de operadores, funciones y tipos de datos definidos por el usuario. Soporta la integridad referencial garantizando la integridad de los datos en la base de datos. PostgreSQL permite realizar múltiples conexiones desde procesos clientes, existiendo un proceso maestro en el servidor que siempre se ejecuta y que está a la espera de nuevas conexiones clientes, de forma tal que cuando alguien se conecta, se inicia un nuevo proceso, asegurando que el cliente y la nueva conexión no necesitan del proceso postgres original. Una de sus principales características es la alta concurrencia. Esto permite que mientras se realizan cambios en una tabla, otros procesos accedan a la misma sin la necesidad de bloqueos, además cada usuario tiene visión de la última modificación.

Presenta el inconveniente que para bases de datos pequeñas su velocidad de respuesta no es muy eficiente en comparación con otras relativamente grandes. (13)

NetBeans

IDE (acrónimo que significa *Integrated Development Environment)* en español entorno de desarrollo integrado. Es un conjunto de herramientas de programación que entre sí conforman un programa informático y de esta forma dan soluciones a problemas informáticos. Como el ambiente de trabajo de los IDE se puede utilizar uno o varios lenguajes de programación, estos se empaquetan conformando una única aplicación. Un IDE consiste en un editor de código, un compilador, un depurador, un intérprete, un sistema de control de versiones y un constructor de interfaz gráfica GUI (del inglés *Graphical User Interface*). Los IDE pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes. (14)

Presenta una fuerte integración con los Framework de PHP como son *Symfony* y *ZenFramework*, de hecho, realizar aplicaciones con estos Framework es muy ágil.

Entre las ventajas de NetBeans se encuentran:

- Soporte a Java Script.
- Interprete de Fondo (*Background Parser*) capaz de identificar errores sintácticos en tiempo de edición.
- Completamiento de código.
- Marcado sintáctico que presenta en diferentes estilos de letras palabras claves, identificadores estándares, y literales en general facilitando la claridad del código.
- Integración con Subversión.
- Soporte a documentación tanto para Java Script como para PHP.

1.4.4 Librerías y marco de trabajo.

"*Framework* se define en términos generales como un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que puede ser utilizada como referencia para enfrentar y resolver nuevos problemas de índole similar.

Más asociada a la informática en el desarrollo de software es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual

otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otras aplicaciones para ayudar a desarrollar y unir los diferentes componentes de un proyecto, es decir, son soluciones completas que llevan incorporado herramientas de apoyo a la construcción (ambiente de trabajo o desarrollo) y motores de ejecución (ambiente de ejecución).

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio". (15)

Los *Frameworks* son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

ExtJS

Es una librería de JavaScript de un alto rendimiento, es compatible con una gran cantidad de los navegadores web existentes en la actualidad. Permite realizar páginas e interfaces web dinámicas mediante el uso de AJAX, DHTML y DOM. La librería incluye componentes de Interfaz de Usuario (UI) que pueden ser modificados al gusto del desarrollador, modelo de componentes extensibles, una Interfaz de Programación de Aplicaciones (API) fácil de usar y licencia de *open source* (Código Abierto) y comerciales.

Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de *layouts (capas)* 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 uno (Firefox, IE, Safari, etc.).

Usar un motor de *render* como ExtJS permite tener estos beneficios:

- Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.

- 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, dándole la libertad de cargar información sin que el cliente se dé cuenta.
- 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. (16)

Zend Framework

"Zend Framework es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP5. Zend Framework es una implementación que usa un código totalmente orientado a objetos. La estructura de los componentes de Zend Framework es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. A menudo se refiere a este tipo de diseño como "use-at-will" (uso a voluntad).

Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de Zend Framework conforman un potente y extensible framework de aplicaciones web al combinarse. Zend Framework ofrece un gran rendimiento y una robusta implementación del patrón Modelo Vista-Controlador (MVC), una abstracción de base de datos fácil de usar y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos.

El principal patrocinador del proyecto Zend Framework es Zend Technologies, pero muchas empresas han contribuido con componentes o características importantes para el marco de trabajo. Empresas como Google, Microsoft y StrikeIron se han asociado con Zend para proporcionar interfaces de servicios web y otras tecnologías que desean poner a disposición de los desarrolladores de Zend Framework". (17)

Doctrine

Doctrine es un Mapeo Objeto-Relacional o ORM del inglés (*Object-Relational Mapping*) que posee una poderosa capa de abstracción de BD. Una de sus características es la opción de escribir consultas de

BD en un objeto apropiado orientado al dialecto SQL (*Structured Query Language*) y que se le denomina Lenguaje de Consulta de Doctrine o DQL del inglés (*Doctrine Query Language)*, inspirado por el Lenguaje de Consulta de Hibernate o HQL del inglés (*Hibernate Query Language*). Este proporciona a los desarrolladores una poderosa alternativa al SQL que mantiene la flexibilidad sin requerir duplicación de código innecesario. (18)

Principales características de Doctrine

Doctrine es un framework para el mapeo objeto – relacional para PHP que está dividido en dos capas principales, la DBAL del inglés (*Database Abstraction Layer)* y el ORM. (18)
Doctrine es un framework que está principalmente construido alrededor de los patrones de *ActiveRecord*, *Data Mapping* y *Meta Data Mapping*. (18)
A través de una clase base especifica llamada **Doctrine_Record**, todas las clases hijas obtienen la interfaz típica *ActiveRecord* (salvar, eliminar, etc.) y esto permite a Doctrine fácilmente participar y monitorear el ciclo de vida de sus registros. El trabajo real, sin embargo, es mayormente reenviado a otros componentes como la clase **Doctrine_Table**. Esta clase tiene la típica interfaz de mapeo de datos **createQuery()**, **find(id)**, **findAll()**, **findBy*()**, **findOneBy*()**. (18)

1.4.5 Lenguajes de modelado y desarrollo.

1.4.5.1 Lenguajes de Modelado.

UML

El lenguaje unificado de modelado UML por sus siglas en inglés (*Unifid Modeling Lenguage*) prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos. Es capaz de describir la semántica general de los diagramas y los significados de los símbolos utilizados. Actualmente es el más conocido y utilizado.
En la actualidad UML está consolidado como un lenguaje estándar en el análisis y el diseño de sistemas de cómputo. Mientras más grande sea el sistema a desarrollar, más beneficios presenta el uso de UML. Entre los cuales se pueden mencionar diseño y documentación, código reutilizable, descubrimiento de

falla, ahorro en el tiempo de tiempo de desarrollo de software, modificaciones más simples, mejor comunicación entre los programadores. (19)

BPMN

La Notación para el Modelado de Procesos de Negocio o más conocida como BPMN por sus siglas en inglés (*Business Process Modeling Notation*) es una notación gráfica estandarizada que permite el modelado de procesos de negocio, en un formato de flujo de trabajo. BPMN está planeada para dar soporte únicamente a aquellos procesos que sean aplicables a procesos de negocios. Esto significa que cualquier otro tipo de modelado realizado por una organización con fines distintos a los del negocio no estará en el ámbito de BPMN.

La propuesta de Gestión de Procesos de Negocios (BPM) ha adquirido una atención considerable recientemente tanto por las comunidades de administración de negocios como por la de ciencia de la computación. BPM provee un conjunto de metodologías para el análisis, comprensión y documentación de los procesos de negocios. En este dominio, el lenguaje de modelado estándar del OMG (*Object Management Group*), se ha tornado popular debido a su gran importancia. (20)

BPMN está dirigido a usuarios, proveedores y prestadores de servicios que necesitan una manera estándar para comunicar los procesos de negocio. El modelado en BPMN se realiza mediante diagramas muy simples con un conjunto muy pequeño de elementos gráficos. Con esto se busca que para los usuarios del negocio y los desarrolladores técnicos sea fácil entender el flujo y el proceso. Las cuatro categorías básicas de elementos son:

- Objetos de flujo: Eventos, Actividades, Rombos de control de flujo (*Gateways*).
- Objetos de conexión: Flujo de Secuencia, Flujo de Mensaje, Asociación.
- *Swimlanes* (Carriles de piscina): *Pool*, *Lane*.
- Artefactos: Objetos de Datos, Grupo, Anotación.

Estas cuatro categorías de elementos dan la oportunidad de realizar un diagrama simple de procesos de negocio (en inglés *Business Process Diagram* o BPD). En un BPD se permite definir un tipo personalizado de objeto de flujo o un artefacto, así se hace el diagrama más comprensible.

El principal objetivo de BPMN es: Resolver las dificultades de comunicación que tiene el lenguaje común:

- Proporciona un método normalizado para representar procesos de negocio
- Facilita su entendimiento debido a la poca complejidad de su notación
- Proporciona un lenguaje común entre los usuarios de negocio y los técnicos
- Facilita la diagramación de los procesos de negocio

1.4.5.2 Lenguajes de desarrollo.

Programación Orientada a Objetos (POO)

La programación orientada a objetos (POO), intenta simular el mundo real a través del significado de objetos que contiene características y funciones. Los lenguajes orientados a objetos se clasifican como lenguajes de quinta generación.
En la POO se define la herencia como una jerarquía de extracciones, y la relación entre clases, donde se comparte la estructura y el comportamiento de una o más clase considerada como clases superiores o una superclase, con lo cual se resume que la herencia es una unidad independiente por si misma heredada de una abstracción o superclase. (21)

Lenguaje de programación

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente. Aunque muchas veces se usa lenguaje de programación y lenguaje informático como si fuesen sinónimos, no tiene por qué ser así, ya que los lenguajes informáticos engloban a los lenguajes de programación y a otros más, como, por ejemplo, el HTML.

PHP

PHP (acrónimo que significa *Hipertext Preprocessor*). Es un lenguaje de programación del lado del servidor que es independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Al ser un lenguaje del lado del servidor, puede utilizarse para realizar conexiones en la red o acceder a base de datos, así los datos procesados y obtenidos pueden ser enviados al navegador

y ser mostrados a los usuarios. Una de sus principales ventajas, es ser un lenguaje libre, por tanto es una opción de fácil acceso.
Es muy utilizado para la realización de páginas web dinámicas, por su gran dinamismo y adaptabilidad. Los navegadores web no tienen necesariamente que soportar a PHP, pero para que se puedan ejecutar las páginas realizadas en PHP si es necesario que el servidor web soporte este lenguaje. Es un lenguaje interpretado de alto nivel embebido en páginas HTML. (22)

JavaScript

JavaScript es un lenguaje de scripting basado en objetos, que se utiliza principalmente para crear páginas web dinámicas y permite el desarrollo de interfaces de usuario mejoradas. Una página web dinámica es aquella que permite la interacción entre el contenido de la misma y el usuario. JavaScript permite incorporar a dichas páginas efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. A pesar de su nombre, este no guarda relación directa con el lenguaje Java, sino que simplemente la compañía dueña del mismo lo adoptó por una cuestión de mercado. (16)

1.4.6 Navegador Mozilla Firefox.

Mozilla Firefox es un navegador de código abierto, multiplataforma. Es uno de los navegadores más usados a nivel mundial para la navegación por internet y el desarrollo de aplicaciones web, debido a que proporciona una gran rapidez al ejecutar el código JavaScript, presenta una gran seguridad al realizar transferencias de información al establecer conexiones con algún servidor web. Entre sus principales características encontramos las siguientes:

- Velocidad: las páginas se abren en menor tiempo y se navega con comodidad en ellas.
- Seguridad: es un software de código abierto, esto permite que las fallas de seguridad se corrijan al instante, tienen un grupo de colaboradores encargados de esta cuestión. Al mismo tiempo, Mozilla Firefox usa su propio motor de dibujado de páginas, *Gecko*, lo que lo hace

inmune a las fallas de seguridad del Internet Explorer, que también tienen todos los navegadores basados en sí mismo.

- Desde sus inicios de modo sencillo y eficaz Mozilla Firefox procuró evitar las molestas ventanas emergentes de publicidad.
- La navegación por pestañas permite tener varias páginas abiertas en una misma ventana, lo que garantiza una navegación mucho más cómoda.
- Se pueden usar varios perfiles de usuario con configuraciones diferentes para cada uno.

Este navegador presenta un innumerable número de extensiones, temas, *plugins* y complementos que conforman el paquete de herramientas del navegador, proporcionándole así a sus usuarios una gran comodidad, ya que todos las herramientas pueden ser configuradas y modificadas a las necesidades y gustos de los usuarios. Durante el desarrollo de aplicaciones web es muy utilizado por las comodidades que brinda el complemento *Firebug* a los desarrolladores, ya que estos pueden llevar un control del tráfico de información desde el cliente hasta el servidor y desde el servidor al cliente, brindando muchas facilidades en el momento en que es necesario realizar la conexión de la capa de presentación a la capa de negocio. (23)

1.5 Resultados esperados.

Se propone para la solución del problema existente, la implementación de una aplicación web que cumpla los estándares y normativas del marco de trabajo Sauxe. Usando tecnologías libres y de código abierto, que sea capaz de solucionar las necesidades de los administradores de los sistemas gestores de bases de datos PostgreSQL. Brindando la posibilidad de poder acceder a la aplicación desde cualquier lugar del dominio donde se utilice. Permite la configuración de los ficheros de configuración de los sistemas gestores de base de datos mediante la web. Dado el caso de ser una aplicación web debe permitir a los usuarios acceder de alguna manera a los ficheros ya configurados, descargándolos en la PC, para si el usuario desee replicar la configuración a varios sistemas gestores de base de datos PostgreSQL. Así también se aplicara esta característica de usabilidad a los ficheros (*.backup) generados, como a los ficheros (*.backup) que se deseen restaurar, también a los reportes realizados por el sistema de las operaciones realizadas por el mismo. Se contará con las tecnologías y herramientas seleccionadas y con el modelo de desarrollo a seguir para obtener los mejores resultados.

1.6 Conclusiones parciales.

Durante este capítulo se realiza el esclarecimiento de algunos términos y conceptos. Se efectúa un estudio de las principales herramientas y tecnologías que propone el marco de trabajo Sauxe para la realización de aplicaciones web. Se realiza un estudio de aplicaciones clientes que permiten la configuración y administración del sistema gestor de base de datos PostgreSQL.

Con el resultado obtenido en el análisis realizado a las herramientas semejantes se llegó a la conclusión que ninguno de los clientes de administración del SGBD PostgreSQL, cumplen con los estándares que define el marco de trabajo Sauxe para la integración de estos al marco de trabajo. Por tal motivo es necesario el desarrollo de una herramienta que permita la administración y configuración del SGBD.

La administración debe ser realizada de forma remota, tiene que efectuar cambios a los parámetros de los ficheros de configuración en los servidores remotos mediante el uso de una aplicación web. La herramienta que se desea obtener mejorará en gran medida el proceso de realizar las tareas administrativas a los SGBD que se encuentran instalados en servidores remotos.

CAPÍTLUO 2: PROPUESTA DE SOLUCIÓN

2.1 Introducción.

En este capítulo se recogen los resultados obtenidos durante el proceso de desarrollo de la solución, como algunos de los artefactos generados. Se describen los procesos del negocio relativos al campo de acción y los requisitos funcionales de la solución para lograr un mejor entendimiento del sistema que se quiere lograr. Se realiza la descripción del diseño elaborado para así lograr los objetivos propuestos.

2.2 Descripción de los procesos del negocio.

Un proceso de negocio no es más que un conjunto de tareas relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio definido. La descripción de los procesos de negocio hace más viable el paso a las actividades del análisis ya que posibilita una compresión más clara de los procesos en cuestión y contribuye a que los requisitos que se definan satisfagan las necesidades del usuario. Estos procesos son descritos a continuación.

2.2.1 Descripción del proceso: Configuración de archivos.

La herramienta permitirá a sus usuarios realizar la modificación de los parámetros de los archivos de configuración del gestor de bases de datos, entre los cuales se encuentran (postgresql.conf, pg_hba.conf, pg_ident.conf), estas modificaciones serán ejecutadas a través de una conexión segura mediante SSH (*Secure Shell*) la cual permite realizar varias operaciones con ficheros haciendo uso de SFTP (en inglés *Secure File Transfer Protocol*), en español significa protocolo para la transferencia segura de ficheros. En caso que algún cambio requiera el reinicio del servidor para que el mismo adopte las nuevas configuraciones, la herramienta permite realizar la operación de reiniciar el servidor.

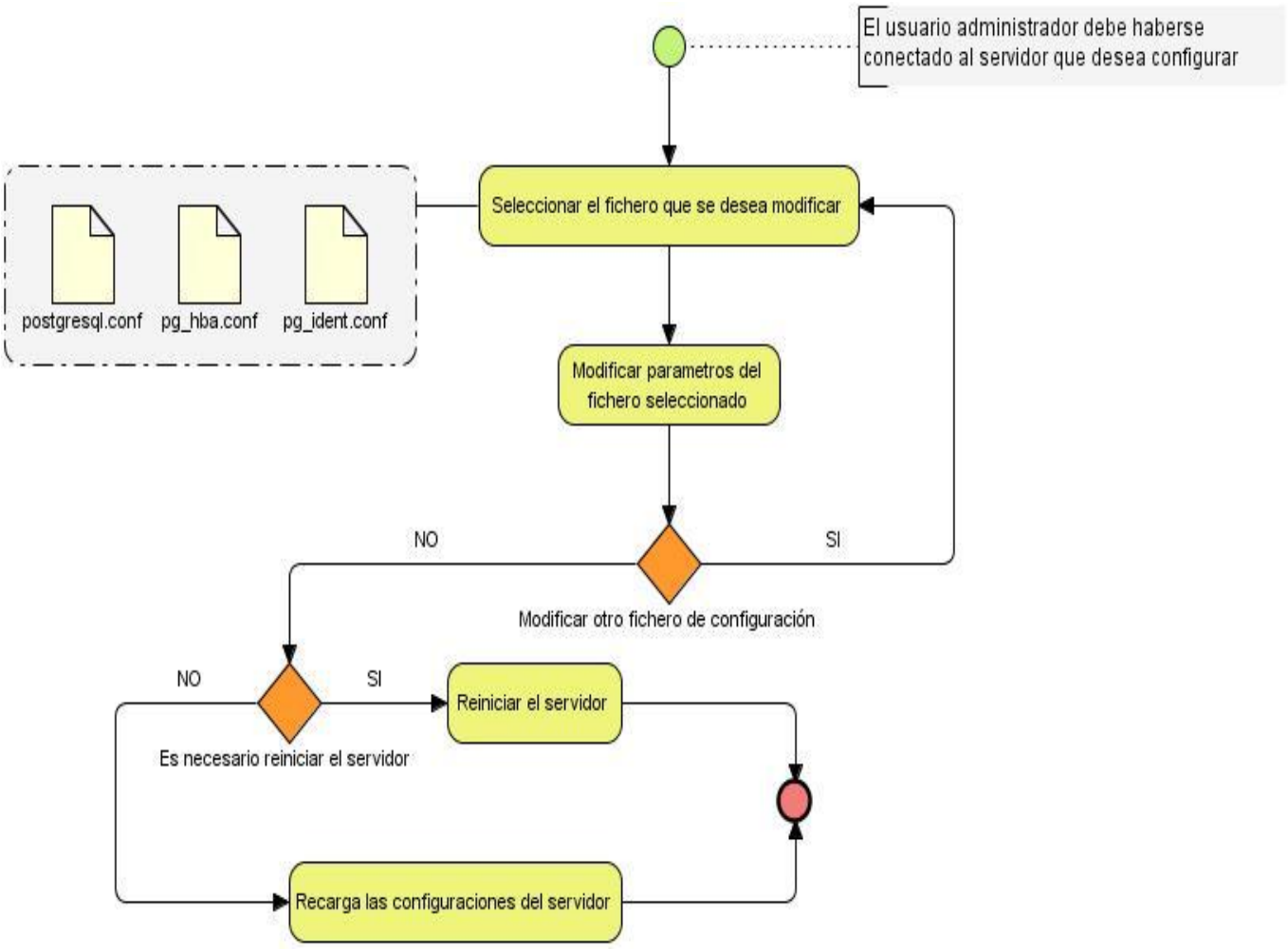

Ilustración 2 Proceso configuración de archivos del gestor de base de datos.

2.2.2 Descripción del proceso: Realización de mantenimientos.

El usuario cuenta con la posibilidad de realizar mantenimientos ya sean instantáneos o programados (El usuario realiza la programación, en cuanto a tiempo de ejecución), estos mantenimientos pueden ser de varios tipos (limpieza, análisis, reindexado). Los mantenimientos programados, son similares al de los mantenimientos instantáneos, la diferencia radica en que luego de la selección de los objetos a mantener el usuario debe de realizar la configuración temporal del mantenimiento, esta configuración

temporal se basa en la definición del tiempo que puede ser: Diario, Semanal, Mensual, Solo una vez, seleccionando también la hora y los minutos en la que será ejecutado el mantenimiento.

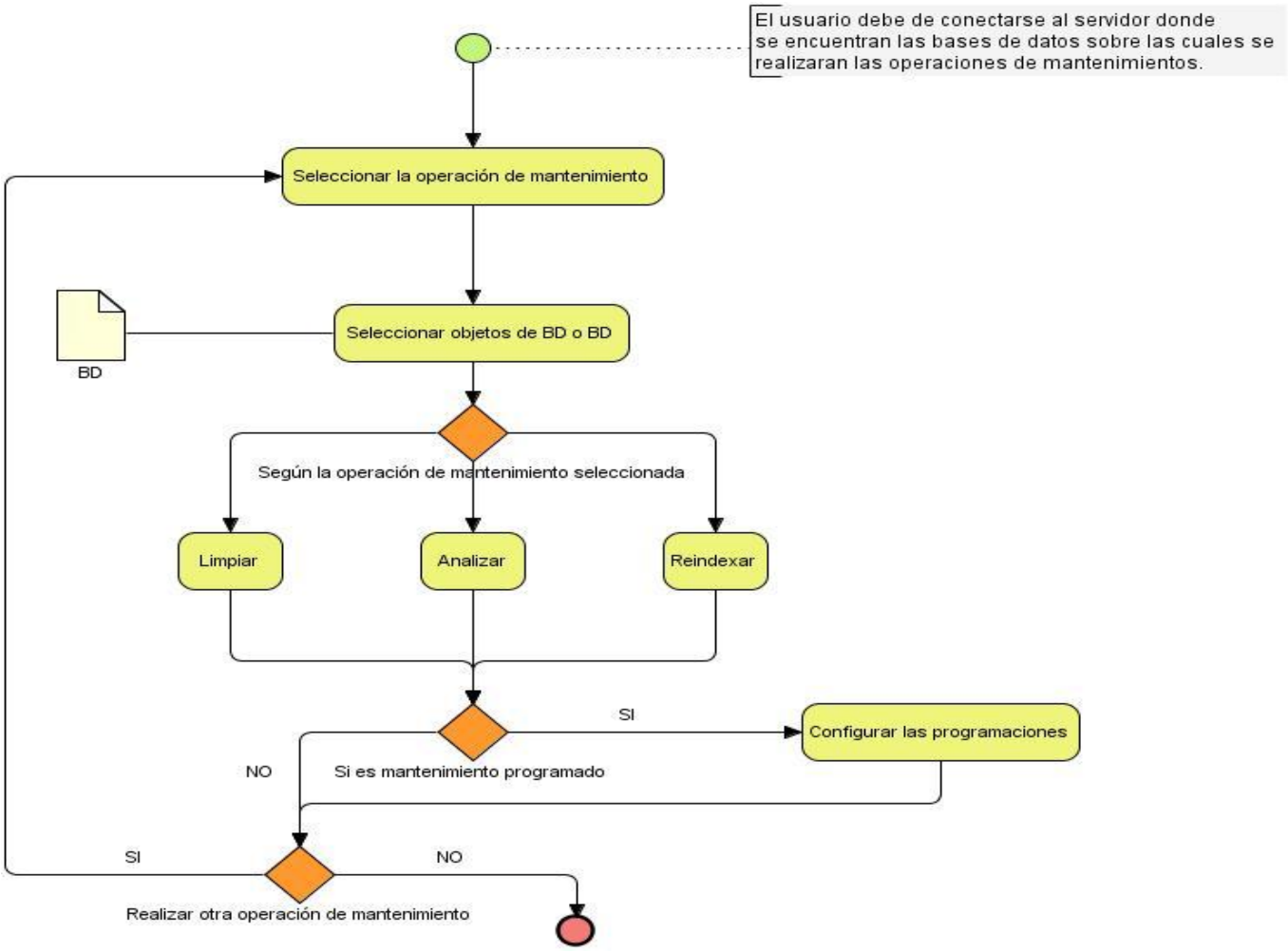

Ilustración 3 Proceso mantenimiento al gestor de base de datos.

2.2.3 Descripción del proceso: Realización de salvas de respaldo.

La solución facilita al usuario la gestión de las salvas de respaldo, las cuales pueden realizarse de dos formas, una es instantánea donde el usuario puede realizar su salva (*.backup) y descargarla en la PC cliente donde se encuentre. La otra forma es configurando una programación para realizar la salva, especificando el objeto de la base de datos a salvar y el tiempo en que se realizara la salva, ídem

(mantenimiento programado). Para dar solución a este proceso el sistema realiza la carga de las bases de datos del servidor, para así seleccionar a la que se le va a realizar la salva y ejecutar el respaldo, en caso que sea programado, entonces realizar la configuración de la programación temporal.

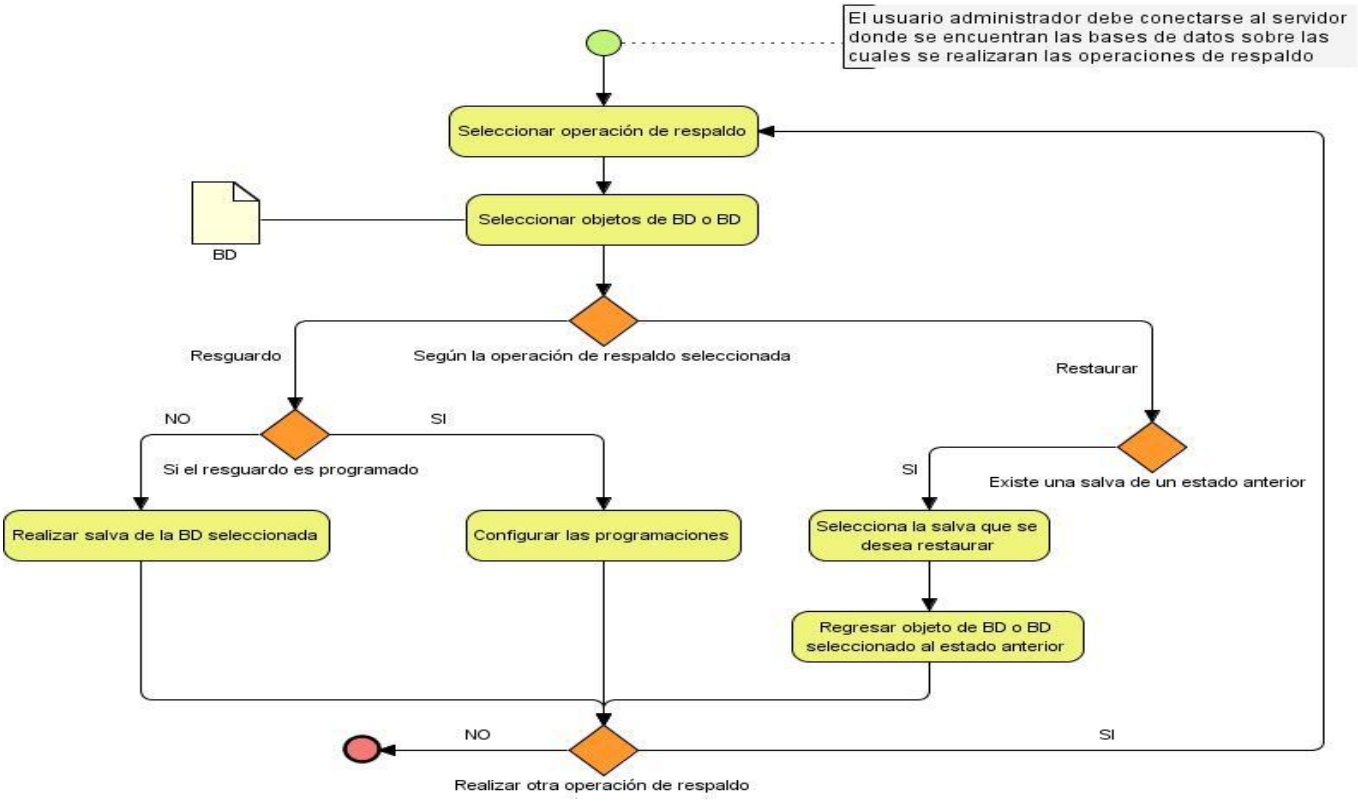

Ilustración 4 Proceso respaldo de objetos del gestor de base de datos.

2.3 Requisitos de software.

Los requisitos de software son unos de los principales elementos en el proceso de desarrollo de sistemas, ya que estos permiten la comunicación entre los usuarios y el equipo de desarrollo para llegar a un entendimiento de lo que se desea realizar. Por tal motivo son punto clave en la producción de algún sistema informático.

Existen distintas definiciones de requisito de software dadas por diversos autores entre las que podemos citar:

- Los requisitos son expresiones de las necesidades de *stakeholders* para alcanzar una meta particular. (25)
- Un requisito es una condición o capacidad necesaria dada por un usuario con el objetivo de resolver un problema o alcanzar un objetivo. (26)
- Los requisitos expresan las necesidades y restricciones atribuibles a un producto de software que contribuye a la solución de algún problema del mundo real. (27)

Los requisitos de software se clasifican en funcionales y no funcionales. Los requisitos funcionales describen qué es lo que el sistema debe hacer para dar soporte a las funciones y objetivos del usuario. Los requisitos no funcionales imponen restricciones de cómo los requisitos funcionales deben ser implementados. (28)

2.3.1 Requisitos funcionales.

Los requisitos funcionales definen las acciones y funciones que debe de ser capaz de realizar el sistema. Los mismos se centran en *qué* y *cómo* se deben de hacer las funciones.

Descritos los procesos de negocio, se identificaron los requisitos funcionales a cumplir por el sistema, los cuales son listados a continuación:

1. -Gestionar operaciones con el clúster de BD y el SGBD.
   1.1. -Autenticar los usuarios del clúster de BD y el del SGBD.
   1.2. -Desconectar el gestor de bases de datos.
   1.3. -Modificar los datos de la conexión al gestor.
   1.4. -Eliminar el gestor de bases de datos.
2. -Modificar parámetro en el fichero postgresql.conf.
3. -Gestionar el fichero hba.conf.
   3.1. -Adicionar una nueva conexión en el fichero hba.conf.
   3.2. -Modificar alguna conexión existente en el hba.conf.
   3.3. -Eliminar alguna conexión existente en el hba.conf.
4. -Gestionar el fichero pg_ident.conf.
   4.1. -Adicionar un nuevo usuario en el fichero pg_ident.conf.
   4.2. -Modificar algún usuario existente en el pg_ident.conf.

4.3. -Eliminar algún usuario existente en el pg_ident.conf.

5. -Realizar operaciones de mantenimientos a objetos de BD.
   5.1. -Realizar limpieza a objetos de BD.
   5.2. -Realizar análisis a objetos de BD.
   5.3. -Realizar reindexado a objetos de BD.
6. -Realizar mantenimientos programados.
   6.1. -Adicionar nuevo mantenimiento programado.
   6.2. -Modificar algún mantenimiento programado existente.
   6.3. -Eliminar algún mantenimiento programado existente.
7. -Realizar operaciones de respaldos a objetos de BD.
   7.1. -Realizar resguardo a objetos de BD.
      7.1.1. -Realizar descarga de los respaldos realizados.
   7.2. -Realizar restauración a objetos de BD.
8. -Realizar respaldos programados.
   8.1. -Adicionar nuevo respaldo programado.
   8.2. -Modificar algún respaldo programado existente.
   8.3. -Eliminar algún respaldo programado existente.
9. Operaciones con el servidor de bases de datos.
   9.1. -Iniciar el servidor en caso de ser necesario.
   9.2. -Detener el servidor en caso de ser necesario.
   9.3. -Reiniciar el servidor en caso de ser necesario.
   9.4. -Recargar el servidor en caso de ser necesario.

Tabla 1 Especificación de requisito Autenticar los usuarios del clúster de BD y el del SGBD.

Precondiciones	N/A
Flujo de eventos	
Flujo básico Autenticar los usuarios del clúster de BD y el del SGBD	
1	Seleccionar la opción conectar servidor.
2	Se introducen los datos de la autenticación al clúster de datos: Descripción Dirección de IP Usuario Contraseña

	Puerto SSH
3	Se introducen los datos de la autenticación a la base de datos: Usuario Contraseña Puerto Base de Datos Comentario
4	Se selecciona la opción de almacenar contraseñas: Almacenar No Almacenar
5	El sistema valida los datos introducidos (ver validaciones 1 y 2)
6	Si la información es correcta, el sistema confirma que se conectó al servidor.
7	Concluye el requisito.
Pos-condiciones	
1	Se ha adicionado un nuevo usuario.
Flujos alternativos	
Flujo alternativo 6.a Información errónea	
1	El sistema señala los datos erróneos y permite corregirlos.
2	El usuario corrige los datos.
3	Volver al paso 5 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo 6.b Información incompleta	
1	El sistema señala los datos vacíos y permite corregirlos.
2	El usuario corrige los datos.
3	Volver al paso 5 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo *.a El usuario cancela la acción	
1	Concluye el requisito.
Pos-condiciones	
1	No se autentica el servidor.
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual.
2	El campo comentario es opcional, puede quedar vacío.
Conceptos	**Autenticación** — **Visibles en la interfaz:** Descripción al clúster Dirección de IP Usuario del clúster Contraseña del clúster Puerto SSH Usuario de la Base de Datos

	Contraseña de la Base de Datos Puerto Base de Datos Comentario **Utilizados internamente:** N/A
Requisitos especiales	N/A.
Asuntos pendientes	N/A.

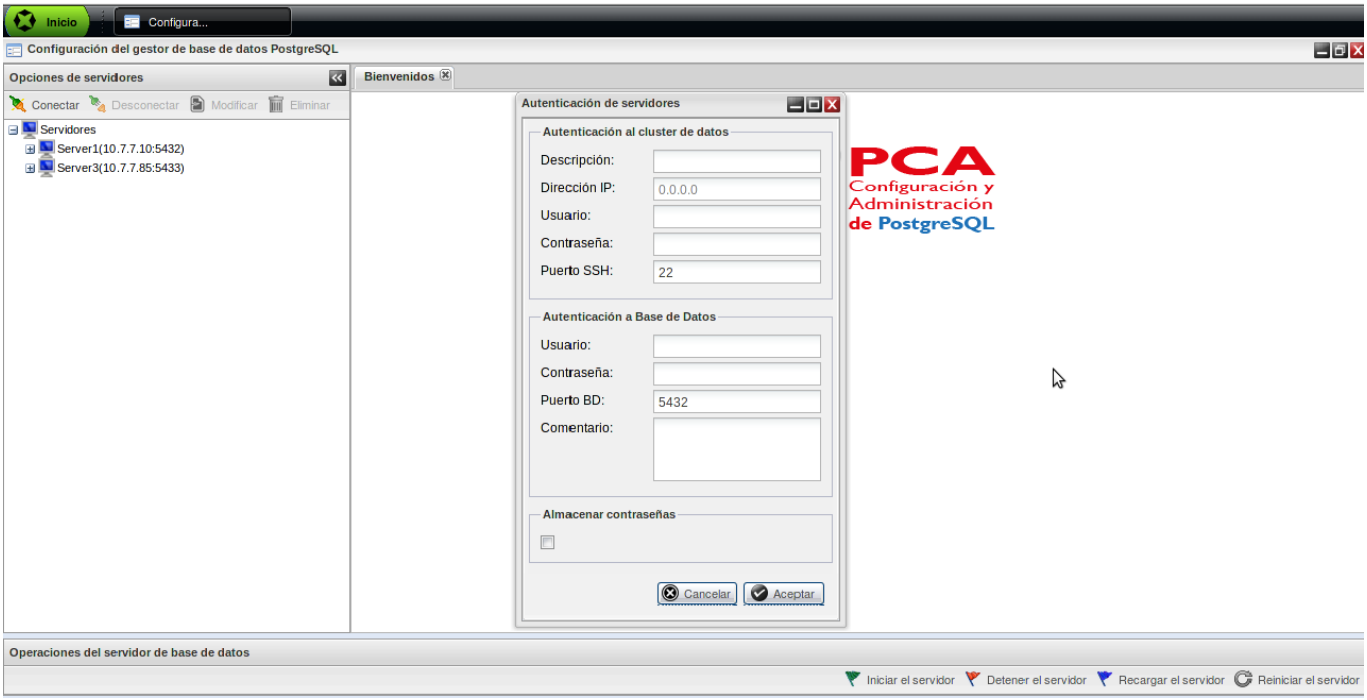

Ilustración 5 Prototipo de interfaz Autenticar los usuarios del clúster de BD y el del SGBD.

Las demás especificaciones elaboradas de los requisitos se encuentran en el Anexo 2 del documento digital.

2.3.2 Requisitos no funcionales.

Como se ha mencionado con anterioridad el presente trabajo forma parte de un proceso productivo, el cual es iniciado y llevado a cabo por el CEIGE, y los resultados formaran parte del marco de trabajo desarrollado por el mismo. Por tal motivo los requisitos no funcionales a los que se debe de acoger la

aplicación a desarrollar son los establecidos por el centro al inicio del proceso de desarrollo, a continuación de describen algunos de mayor importancia.

2.3.2.1 Autenticación

Autenticación y Autorización (Contraseña de acceso). Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos. El sistema realizara una petición y chequeo de los usuarios y contraseñas para así poder definir si el usuario tienes permisos para realizar algunas acciones que puedan causar daños severos a la información almacenada.

2.3.2.2 Software

Para el cliente:

- Navegador Mozilla Firefox 3.0 o superior.
- Sistema operativo Windows 98 o superior, Linux.

Para el servidor:

- Sistema operativo Linux en cualquiera de sus distribuciones.
- Un servidor Apache 2.0 o superior con módulo PHP 5.0 disponible. Este debe estar configurado con la extensión “pgsql” y “libssh2-php” incluida.
- Un servidor de base de datos PostgreSQL 8.3 o superior.
- Debe de estar configurado con SSH, para la transferencia de ficheros de forma segura a través del SFTP.
- Debe tener instalado la librería “*tar”* para la compresión de los ficheros (*.backup).

2.3.2.3 Hardware

Para el servidor:

- Requerimientos mínimos: Procesador Pentium IV a 2GHz de velocidad de procesamiento y 1Gb de memoria RAM.
- Al menos 40Gb de espacio libre en disco duro.
- Tarjeta de red.

Para el cliente:

- Requerimientos mínimos: Procesador Pentium III a 1GHz con 256Mb de memoria RAM.
- Tarjeta de red.

2.4 Modelo de diseño.

El Modelo de diseño es una abstracción del Modelo de implementación, y el código fuente se emplea fundamentalmente para representar y documentar el diseño. Es utilizado como entrada de gran importancia en las actividades relacionadas con la implementación. El modelo de diseño puede contener: diagramas, clases, paquetes, subsistemas, cápsulas, protocolos e interfaces.
En esta sección se verán los elementos que se tuvieron en cuenta durante el diseño de la solución, dando paso a la exposición de los resultados, lo cual refleja cómo será implementado el sistema en términos de clases del diseño.

2.4.1 Diagrama 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, contiene información como clases, asociaciones, atributos, métodos y dependencias. A continuación se muestra el diagrama de clases del diseño basado en estereotipos web que se realizó durante el proceso de desarrollo.
En el diagrama de clases del diseño que hace referencia al módulo de Configuración (Ver Ilustración 6), se utilizan varias clases entidades para persistir la información generada y así tener una constancia de los cambios que pueden sufrir los parámetros de los ficheros de configuración en determinados SGBD, los cuales se encuentran registrados previamente en la herramienta. El diagrama diseñado presenta en la capa de presentación la clase llamada (ConfiguracionBD.js), la cual se encarga de realizar peticiones al servidor a través de AJAX o mediante los envíos de datos realizados por los formularios (*submit*).
Para dar solución al problema se cuenta con tres clases entidades en la capa de datos (FicheroConfiguracion, FicheroHBA, FicheroIdent), las mismas escriben en las tablas que persistirán la información de los cambios realizados sobre estos ficheros. Así como las clases (PostgresqlModel, HBAModel, IdentModel) en la capa del negocio, las mismas contienen los métodos que gestionan la información que es persistida. Las clases entidades (ServidorSSH, ServidorBD), son las encargadas de persistir la información relacionada con los servidores remotos y los SGBD que se encuentren instalados en los respectivos servidores, que son registrados mediante el uso de la herramienta. Estas

clases entidades contribuyen a la solución obtenida ya que son las clases principales dentro de los diagramas de clases del diseño que se explican a continuación.

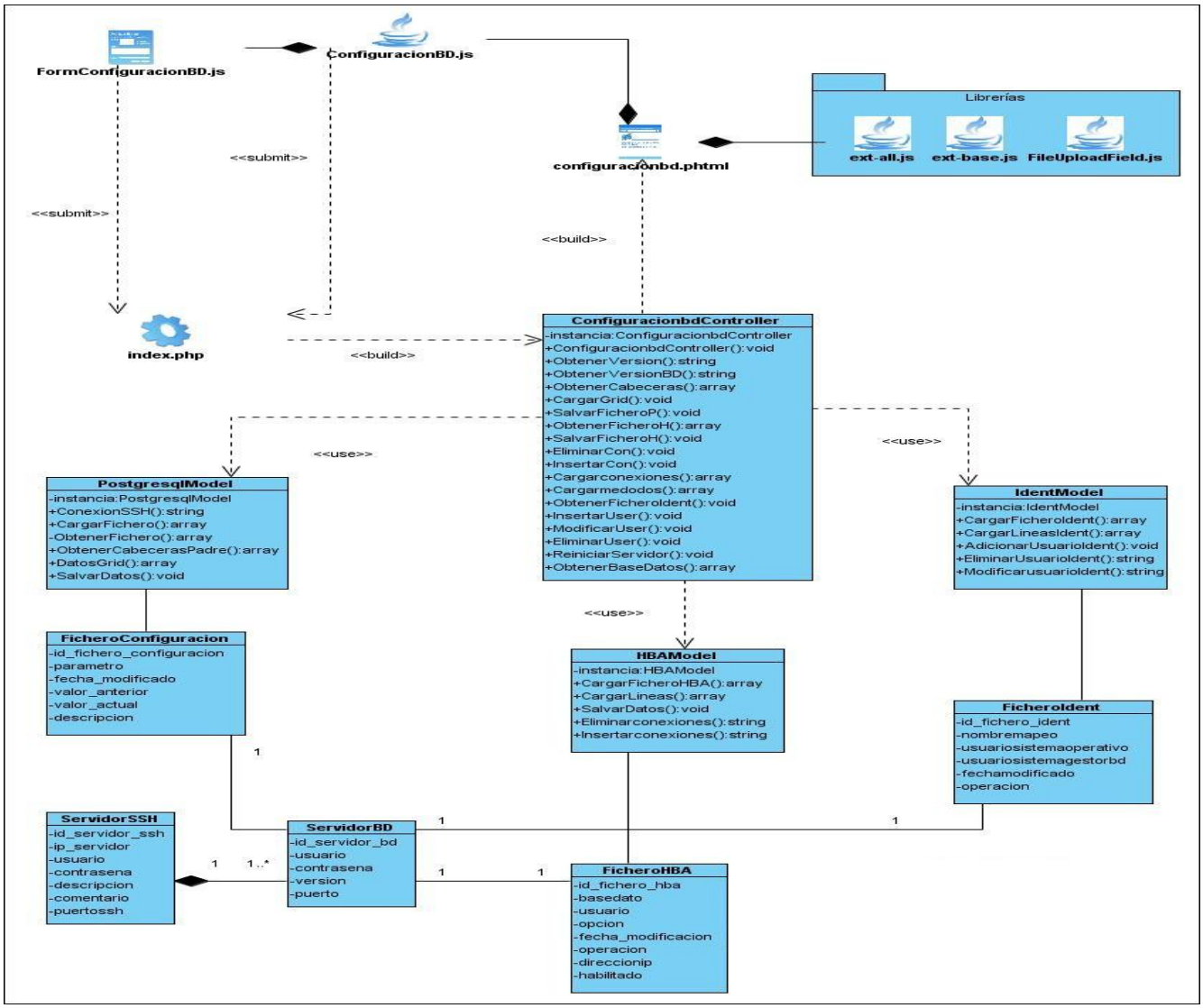

Ilustración 6 Diagrama de clases del diseño del módulo de Configuración.

El diagrama de clases del diseño que referencia al módulo de Mantenimiento (Ver Ilustración 7), presenta en la capa de presentación la clase llamada (Mantenimiento.js), la cual se encarga de realizar peticiones al servidor a través de AJAX o mediante los envíos de datos realizados por los formularios (*submit*).

Para dar solución al problema se tomó como propuesta contar con dos clases entidades en la capa de datos (Mantenimiento, MantenimientoProgramado), las cuales escriben en las tablas que persistirán la información de los mantenimientos que se realicen instantáneamente o automáticamente usando las tareas programadas del sistema operativo Linux. La clase (MantenimientoModel) en la capa de negocio, la misma contiene los métodos que gestionan la información que será persistida, correspondientemente a cada uno de los mantenimientos que se han ejecutado sobre el SGBD que se esté administrando.

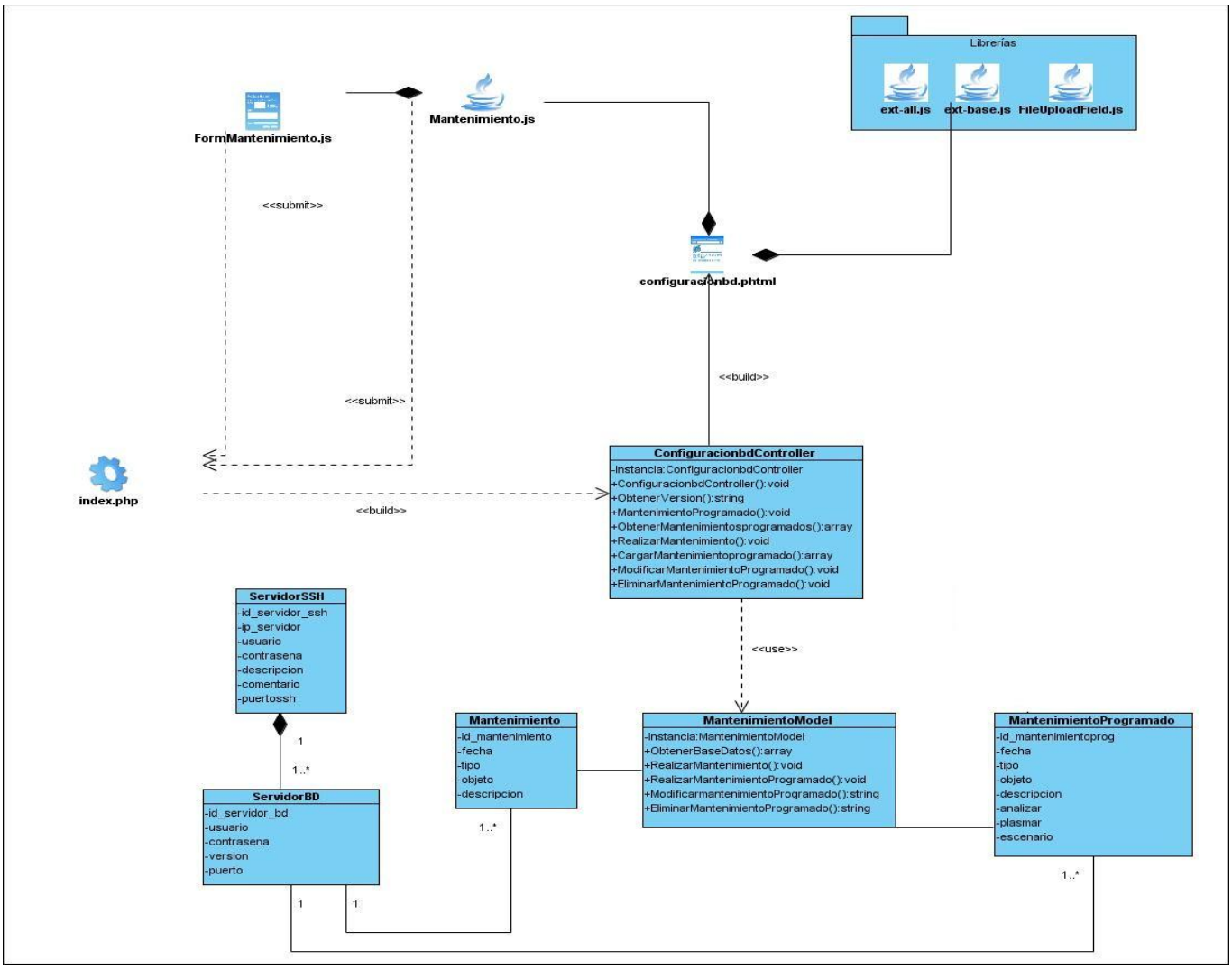

Ilustración 7 Diagrama de clases del diseño del módulo de Mantenimiento.

El diagrama de clases del diseño que referencia al módulo de Respaldo (Ver Ilustración 8), presenta en la capa de presentación la clase llamada (Respaldo.js), la cual se encarga de realizar peticiones al servidor a través de AJAX o mediante los envíos de datos realizados por los formularios (*submit*).

También es la encargada de gestionar las descargas de los respaldos realizados instantáneamente en la PC cliente. Además de permitir la subida de ficheros al servidor web de hasta 2 MB de tamaño, en el fichero se debe encontrar el código de los objetos de bases de datos que se desean restaurar en algún SGBD que se encuentre remoto.

Para dar solución al problema se tomó como propuesta contar con tres clases entidades en la capa de datos (Respaldo, Restauro, RespaldoProgramado), las cuales escribirán en las tablas que persistirán la información de las operaciones de respaldo que se realicen instantáneamente o automáticamente usando las tareas programadas del sistema operativo Linux y de los restauros realizados mediante la aplicación. Así como la clase (RespaldoModel) en la capa del negocio, la misma contiene los métodos que gestionan la información que será persistida, correspondientemente a cada uno de los respaldos que se han ejecutado sobre el SGBD que se esté administrando.

En la clase entidad RespaldoProgramado es la que permite almacenar toda la información generada por las programaciones de los respaldos que se realizan en los distintos servidores remotos registrados. Así como las opciones de las configuraciones de los respaldos, que pueden ser modificadas o eliminadas en algunos casos, en dependencia con las preferencias de los usuarios.

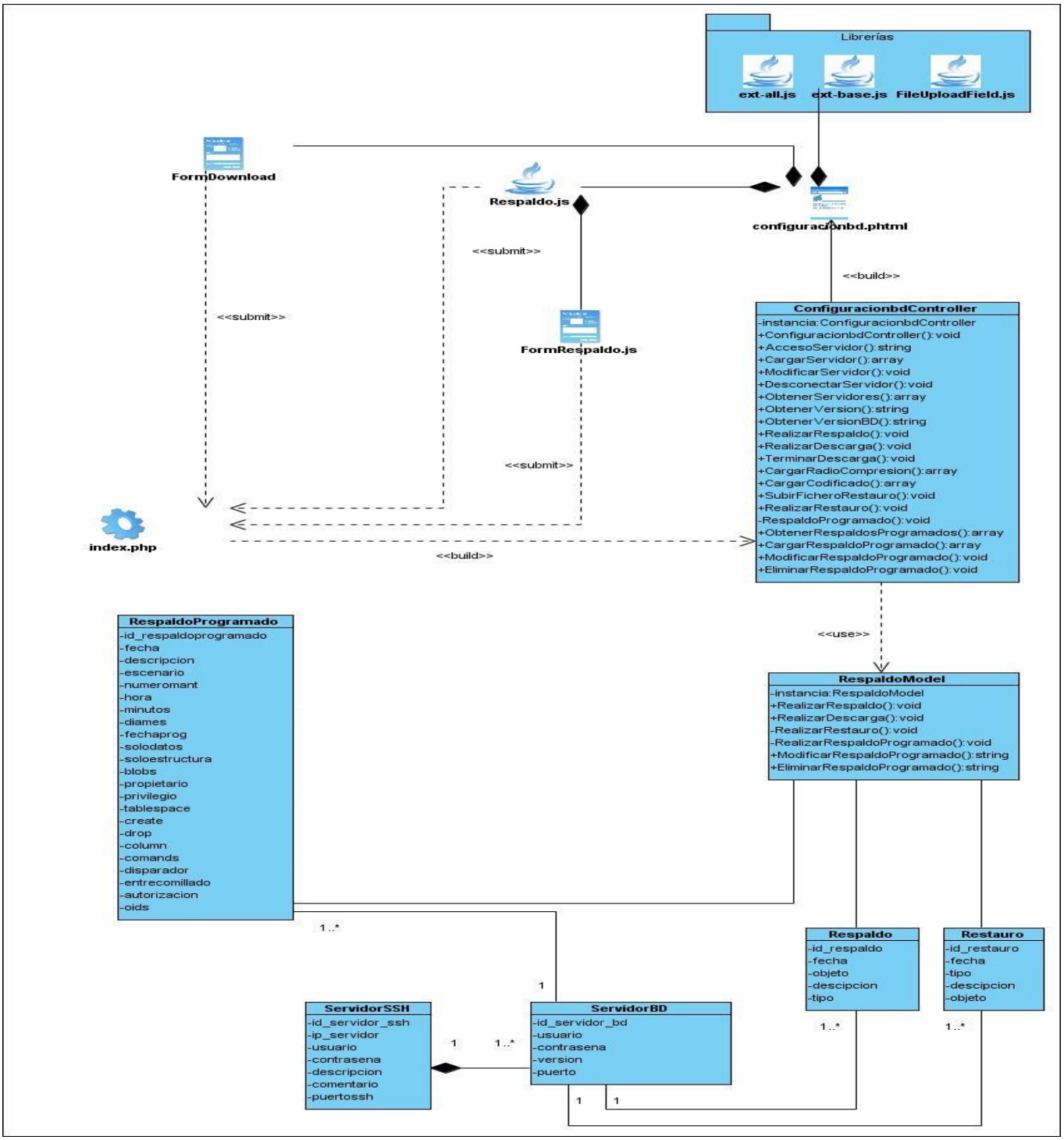

Ilustración 8 Diagrama de clases del diseño del módulo de Respaldo.

2.4.2 Patrones utilizados.

2.4.2.1 Patrón arquitectónico Modelo-Vista-Controlador (MVC)

El patrón Modelo-Vista Controlador MVC (del inglés *Model-View Controller*) es un patrón de arquitectura utilizado en sistemas web para separar los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos, permitiendo flexibilidad y facilidad a la hora de hacer futuros cambios.

La **Vista** es la información presentada al usuario. Una vista puede ser una página Web o una parte de una página.

El **Controlador** actúa como intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para generar una página, es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo.

El **Modelo** representa las estructuras de datos. Típicamente el modelo de clases contendrá funciones para consultar, insertar y actualizar información de la base de datos.

Este patrón es empleado en la capa de control del marco de trabajo Sauxe y determina la estructura de los paquetes internos de los componentes a desarrollar.

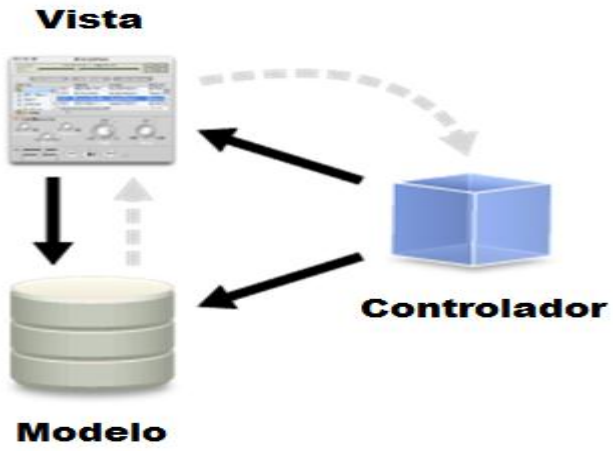

Ilustración 9 Patrón Modelo Vista-Controlador.

2.4.2.2 Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores y debe ser aplicable a diferentes problemas de diseño en distintas circunstancias. (29)
Para hacer un diseño eficiente se tomaron en cuenta un conjunto de patrones, que al ser experiencias de diseñadores expertos en orientación a objetos permiten dar soluciones eficientes a problemas existentes, facilitando notablemente el trabajo posterior. Algunos de estos patrones son:

- Fachada

Es un patrón estructural y consiste en crear una única clase de manejo más fácil, que permita acceder a un conjunto numeroso y complicado de clases. La fachada satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas. (29) En la solución se pone de manifiesto este patrón en las clases pertenecientes al paquete "*Zend::RSA_Facade*".

Los patrones GRASP acrónimo de (*General Responsability Assignment Software Patterns)* en español Patrones de Software para la Asignación General de Responsabilidad, describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades.

Para el diseño del módulo se tuvieron en cuenta los 5 patrones GRASP: Experto, Creador, Bajo acoplamiento, Alta cohesión y Controlador. El marco de trabajo utilizado en el desarrollo del módulo busca un máximo rendimiento y flexibilidad en sus soluciones y pone en práctica estos patrones para lograr un sistema reusable y flexible.

Patrón experto:

Problema*:* ¿De qué forma podemos saber qué responsabilidad delegar a cada objeto? (30)

Solución*:* Asignar una responsabilidad al experto en información; la clase que tiene la información necesaria para llevar a cabo la responsabilidad. (30)

El patrón se utilizó específicamente: en la clase PostgresqlModel, HBAModel, IdentModel en la configuración de los ficheros, las cuales serán responsables de efectuar las operaciones que conciernen a las funciones: listar, adicionar, eliminar y actualizar los parámetros, asumiendo toda la lógica para cada una de ellas.

Patrón creador:

Este patrón es adaptable a las clases del paquete Domain, quienes son las encargadas de crear los objetos de tipo Doctrine_Query, para permitir el acceso a la información almacenada a nivel de datos. (30)

Patrón controlador:

Problema: ¿Quién gestiona un evento del sistema? (30)

Solución: Asignar la responsabilidad de gestionar un mensaje de un evento del sistema a una clase que represente una de estas dos opciones:

La clase controladora definida: ConfiguracionBDController, es un ejemplo de la aplicación de este patrón la misma tendrá a cargo la responsabilidad de manejar los eventos dentro del componente.

Patrón alta cohesión:

Problema: ¿Cómo mantener manejable la complejidad? (30)

Solución: Asignar responsabilidades de manera que la información que almacena una clase sea coherente y esté relacionada con la clase. (30)

Este patrón fue utilizado en el diseño del componente de manera general; donde se agruparon las clases en dependencia de los requerimientos, según la premisa de que cada clase debe implementar las operaciones que estén sobre la misma área funcional.

Patrón bajo acoplamiento:

Problema: ¿Cómo dar soporte a las bajas dependencias y al incremento de la reutilización? (30)

Solución: Diseñar con el objetivo de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. (30)

En el modelo de datos se definieron un conjunto de clases persistentes, entre las cuales se establecieron las relaciones necesarias de manera que fueran más independientes y reutilizables para reducir el impacto de los cambios y acrecentar la oportunidad de una mayor productividad.

2.5 Modelo de datos.

Un modelo de datos es una colección de conceptos bien definidos matemáticamente que ayudan a expresar las propiedades estáticas y dinámicas de una aplicación con un uso de datos intensivo. (31)
En el diseño del modelo de datos realizado (Ver Ilustración 10), fue necesaria la creación de varias tablas con el objetivo de persistir la información generada por las operaciones que se realizan a los servidores de datos donde se encuentran los SGBD que seran configurados y administrados.
Entre las principales tablas que se utilizan en la propuesta de solución se encuentan (dat_servidor_ssh, dat_servidor_bd), en las cuales se almacenan los servidores registrados asi como el SGBD al cual se le realizó la conexión a través del puerto expecificado en la autenticación.
Las tablas (dat_fichero_configuracion, dat_fichero_hba, dat_fichero_ident), usadas para persistir la información referente a los cambios que sufran estos ficheros de configuración en su respectivo SGBD.

Así como las tablas (dat_mantenimientoprog, dat_respaldoprog), las cuales se utilizan para darle solución al problema de las configuraciones de las tareas de mantenimientos y respaldos programados, en las mismas se almacena información acerca de las configuraciones establecidas por los administradores de los SGBD, estas configuraciones se almacenan con el objetivo, de si en algún momento el administrador desea realizar cambios sobre las mismas o desea eliminar algunas de las configuraciones realizadas, estas se encuentren disponibles para ser reconfiguradas y almacenadas nuevamente, para su posterior ejecución.

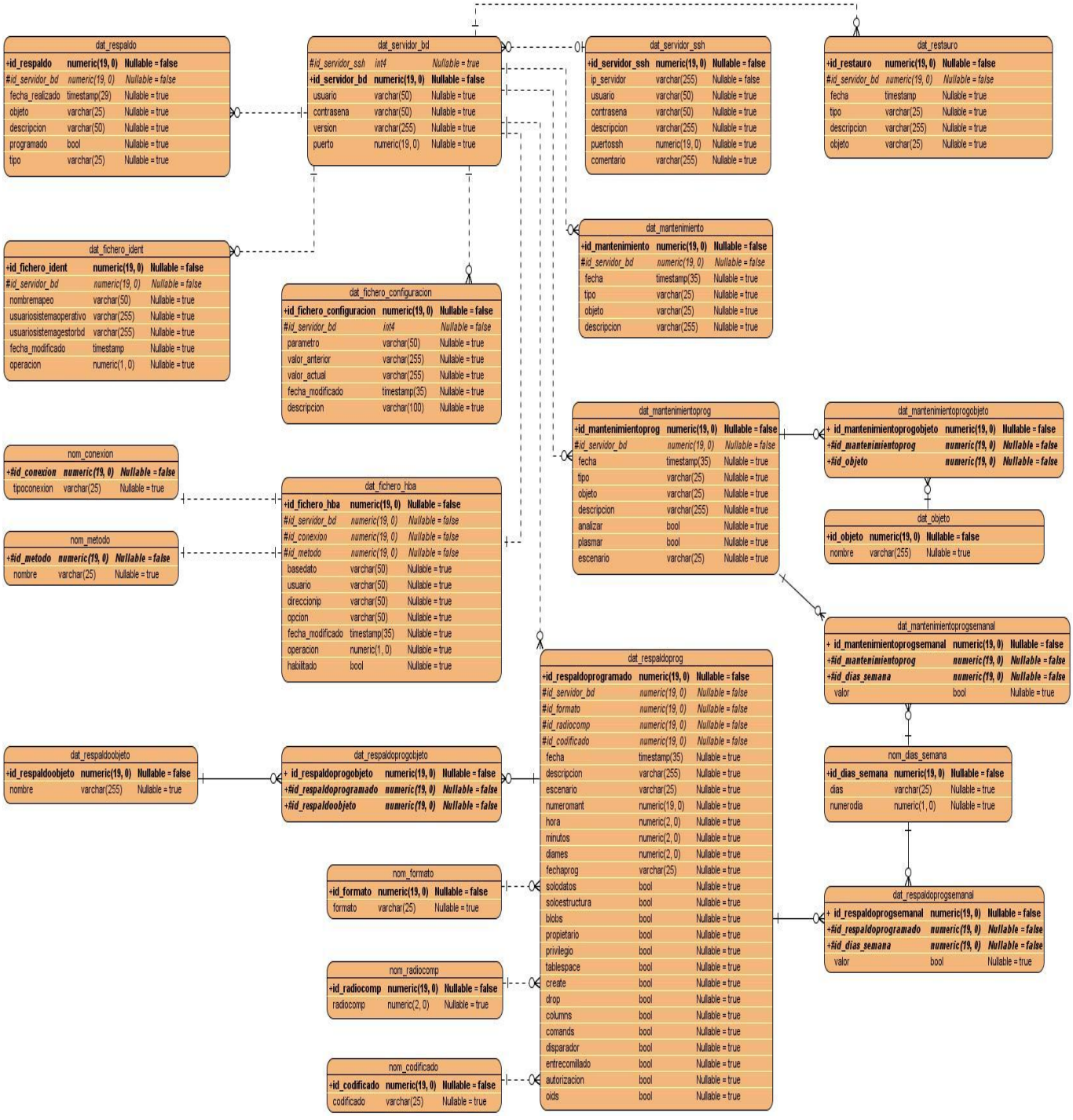

Ilustración 10 Diagrama del Modelo de Datos.

2.6 Conclusiones parciales.

Durante la captura de los requisitos a través de las entrevistas realizadas al cliente, se logró entender la necesidad del mismo y definir los requisitos que presentaban mayor importancia para el cliente, para centrar la solución en estos principalmente.

Se obtuvo como resultado el diagrama del diseño de clases para una mejor comprensión de cómo están estructuradas las clases que conforman el componente. Esto posibilita conocer las clases principales del negocio para luego diseñar el modelo de datos correspondiente a la solución propuesta.

Teniendo los resultados mencionados anteriormente se realiza la entrada a la fase de implementación del sistema ya que se tienen los principales artefactos para el desarrollo.

La aplicación del patrón Modelo-Vista Controlador permite un mejor flujo de información entre las capas de presentación, negocio y datos, lo que permitió separar cada una de estas capas para que cumplan con sus funcionalidades independientemente.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

En este capítulo se muestra el modelo de implementación que se aplica en el diseño de la solución que fue descrito en el capítulo anterior y se especifica un conjunto de validaciones y pruebas que evalúan la calidad del sistema.

3.1 Modelo de implementación.

El modelo de implementación describe cómo los elementos del modelo de diseño, son implementados en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos estructurales y modulares disponibles en el entorno de implementación y en los lenguajes de programación utilizados, cómo dependen algunos componentes de otros y los recursos necesarios para poder ejecutar el sistema desarrollado.

3.1.1 Diagrama de componentes.

La solución propuesta cuenta con un solo componente llamado Configuración de Sistemas Gestores de Base de Datos el cual se encuentra dentro del paquete de Herramientas e interactúa con los demás componentes que se encuentran dentro del paquete de Sauxe.

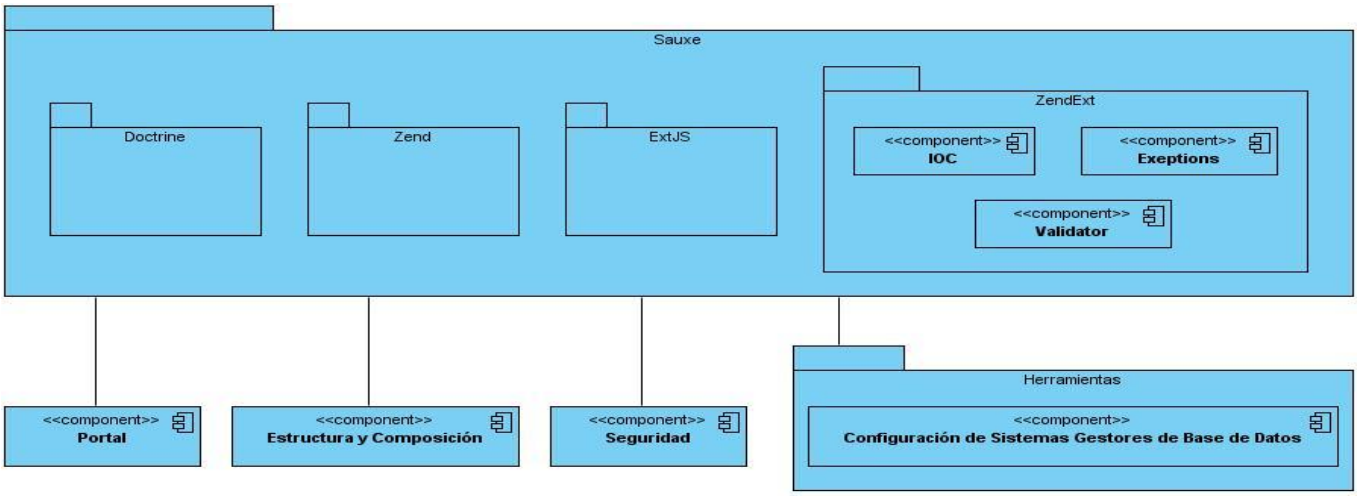

Ilustración 11 Diagrama de Componentes.

3.1.2 Diagrama de despliegue.

El usuario, desde una estación de trabajo, podrá acceder a la aplicación, la cual estará desplegada en el mismo servidor web. Dicho servidor estará conectado a un servidor de bases de datos en el cual se almacenará la información de interés para la solución. A continuación se muestra el diagrama de despliegue elaborado.

Ilustración 12 Diagrama de Despliegue.

3.2 Métricas de software.

Las métricas de software son cuantitativamente una medida, la cual permite a los desarrolladores tener una visión de la eficacia en los procesos de software desarrollados. Es donde se reúnen los datos básicos de productividad y calidad, los cuales son analizados y comparados con resultados obtenidos anteriormente y evaluados para determinar las mejoras en la calidad y la productividad.
Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:

- ***Responsabilidad.*** Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- ***Complejidad de implementación.*** Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- ***Reutilización.*** Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- ***Acoplamiento.*** Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.

- ***Complejidad del mantenimiento.*** Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- ***Cantidad de pruebas.*** Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, modulo, clase, conjunto de clases, etc.) diseñado.

Las métricas escogidas como instrumento para evaluar la calidad del diseño descrito en el capítulo anterior y su relación con los atributos de calidad son las siguientes:

Tamaño operacional de clase (TOC): Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

Tabla 2 Atributos de calidad evaluados por la métrica TOC.

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Tabla 3 Criterios de evaluación para la métrica TOC.

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
	Baja	>2*Promedio

Reutilización	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Relaciones entre clases (RC): Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Tabla 4 Atributos de calidad evaluados por la métrica RC.

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Tabla 5 Criterios de evaluación para la métrica RC.

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

3.2.1 Resultados obtenidos de la aplicación de las métricas TOC.

Obtenidos los resultados de la aplicación de las métricas TOC, se llega a la conclusión que el diseño propuesto tiene una calidad aceptable, teniendo en cuenta que el 88% de las clases empleadas cuenta con 3 operaciones o menos, lo cual ofrece evaluaciones positivas de los atributos de calidad involucrados (responsabilidad, complejidad de implementación y reutilización). A continuación se muestran los resultados obtenidos.

Tabla 6 Instrumento de evaluación de la métrica TOC.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
ConfiguracionBDController	43	Alta	Alta	Baja
PostgresqlModel	5	Alta	Alta	Baja
HBAModel	5	Alta	Alta	Baja
IdentModel	5	Alta	Alta	Baja
MantenimientoModel	6	Alta	Alta	Baja
RespaldoModel	6	Alta	Alta	Baja
DatFicheroConfiguracionModel	0	Baja	Baja	Alta
DatFicheroHbaModel	0	Baja	Baja	Alta
DatFicheroIdentModel	0	Baja	Baja	Alta
DatMantenimientoModel	0	Baja	Baja	Alta
DatMantenimientoprogModel	0	Baja	Baja	Alta
DatServidorBdModel	0	Baja	Baja	Alta

DatServidorSshModel	0	Baja	Baja	Alta
NomConexionModel	0	Baja	Baja	Alta
NomMetodoModel	0	Baja	Baja	Alta
DatMantenimientoprogobjetoModel	0	Baja	Baja	Alta
DatMantenimientoprogsemanalModel	0	Baja	Baja	Alta
DatObjetoModel	0	Baja	Baja	Alta
DatRespaldoModel	0	Baja	Baja	Alta
DatRespaldoobjetoModel	0	Baja	Baja	Alta
DatRespaldoprogModel	0	Baja	Baja	Alta
DatRespaldoprogobjetoModel	0	Baja	Baja	Alta
DatRespaldoprogsemanalModel	0	Baja	Baja	Alta
DatRestauroModel	0	Baja	Baja	Alta
NomCodificadoModel	0	Baja	Baja	Alta
NomDiasSemanaModel	0	Baja	Baja	Alta
NomFormatoModel	0	Baja	Baja	Alta
NomRadiocompModel	0	Baja	Baja	Alta
DatFicheroConfiguracion	0	Baja	Baja	Alta
DatFicheroHba	0	Baja	Baja	Alta
DatFicheroIdent	0	Baja	Baja	Alta
DatMantenimiento	0	Baja	Baja	Alta
DatMantenimientoprog	0	Baja	Baja	Alta
DatServidorBd	0	Baja	Baja	Alta
DatServidorSsh	0	Baja	Baja	Alta
NomConexion	0	Baja	Baja	Alta

NomMetodo	0	Baja	Baja	Alta
DatMantenimientoprogobjeto	0	Baja	Baja	Alta
DatMantenimientoprogsemanal	0	Baja	Baja	Alta
DatObjeto	0	Baja	Baja	Alta
DatRespaldo	0	Baja	Baja	Alta
DatRespaldoobjeto	0	Baja	Baja	Alta
DatRespaldoprog	0	Baja	Baja	Alta
DatRespaldoprogobjeto	0	Baja	Baja	Alta
DatRespaldoprogsemanal	0	Baja	Baja	Alta
DatRestauro	0	Baja	Baja	Alta
NomCodificado	0	Baja	Baja	Alta
NomDiasSemana	0	Baja	Baja	Alta
NomFormato	0	Baja	Baja	Alta
NomRadiocomp	0	Baja	Baja	Alta

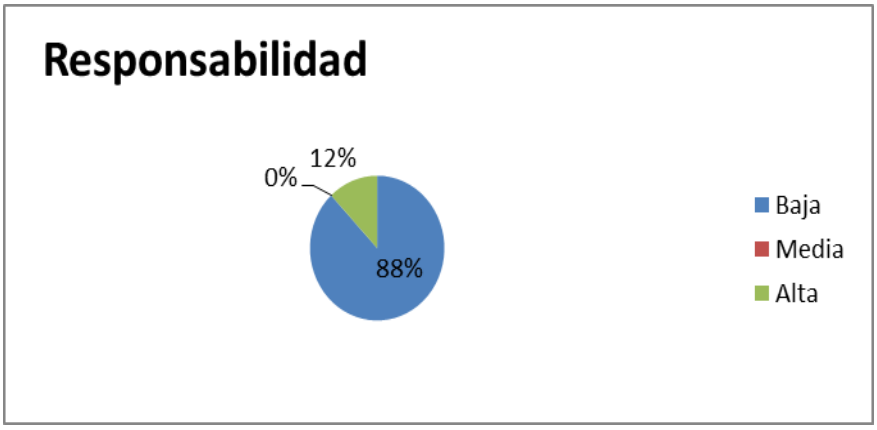

Ilustración 13 Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad.

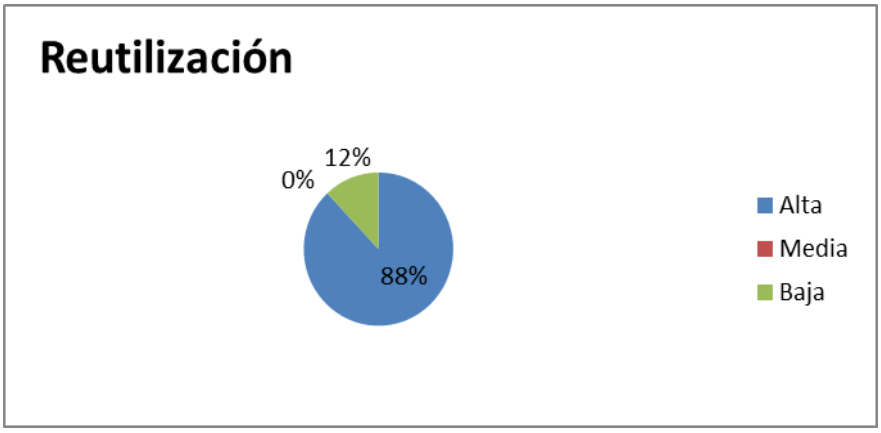

Ilustración 14 Resultados de la evaluación de la métrica TOC para el atributo Reutilización.

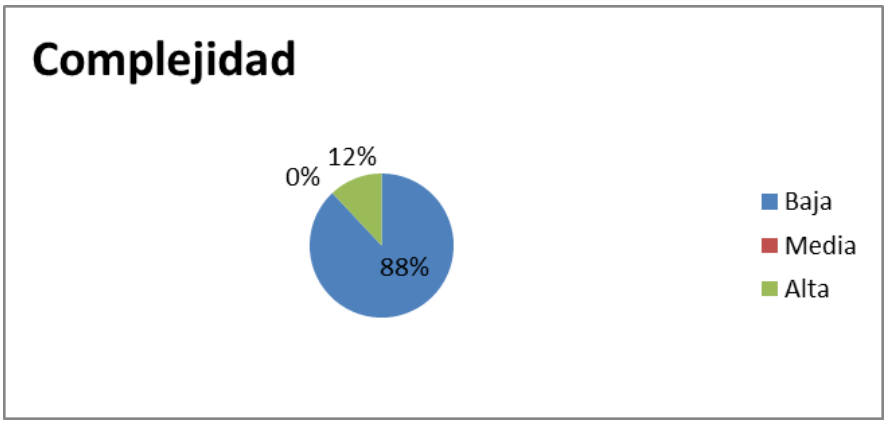

Ilustración 15 Resultados de la evaluación de la métrica TOC para el atributo Complejidad.

3.2.2 Resultados obtenidos de la aplicación de la métrica RC

Obtenidos los resultados de la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 50% de las clases empleadas posee menos de 2 dependencias de otras clases lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización). A continuación se muestran los resultados obtenidos.

Tabla 7 Instrumento de evaluación de la métrica RC.

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
ConfiguracionBDController	5	Alto	Alta	Baja	Alta
PostgresqlModel	2	Medio	Media	Media	Media
HBAModel	2	Medio	Media	Media	Media
IdentModel	2	Medio	Media	Media	Media
MantenimientoModel	2	Medio	Media	Media	Media
RespaldoModel	3	Alto	Media	Media	Media
DatFicheroConfiguracionModel	1	Bajo	Baja	Alta	Baja
DatFicheroHbaModel	3	Alto	Media	Media	Media
DatFicheroIdentModel	1	Bajo	Baja	Alta	Baja
DatMantenimientoModel	1	Bajo	Baja	Alta	Baja
DatMantenimientoprogModel	3	Alto	Media	Media	Media
DatServidorBdModel	7	Alto	Alta	Baja	Alta
DatServidorSshModel	1	Bajo	Baja	Alta	Baja
NomConexionModel	0	Ninguno	Baja	Alta	Baja
NomMetodoModel	0	Ninguno	Baja	Alta	Baja
DatMantenimientoprogobjetoModel	2	Medio	Media	Media	Media
DatMantenimientoprogsemanalModel	2	Medio	Media	Media	Media
DatObjetoModel	1	Bajo	Baja	Alta	Baja
DatRespaldoModel	1	Bajo	Baja	Alta	Baja
DatRespaldoobjetoModel	1	Bajo	Baja	Alta	Baja
DatRespaldoprogModel	6	Alto	Alta	Baja	Alta
DatRespaldoprogobjetoModel	2	Medio	Media	Media	Media
DatRespaldoprogsemanalModel	2	Medio	Alta	Media	Media
DatRestauroModel	1	Bajo	Baja	Alta	Baja
NomCodificadoModel	0	Ninguno	Baja	Alta	Baja
NomDiasSemanaModel	0	Ninguno	Baja	Alta	Baja

NomFormatoModel	0	Ninguno	Baja	Alta	Baja
NomRadiocompModel	0	Ninguno	Baja	Alta	Baja

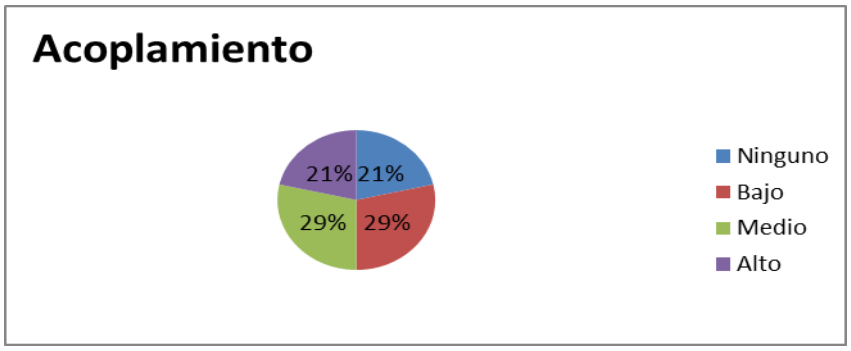

Ilustración 16 Resultados de la evaluación de la métrica RC para el atributo Acoplamiento.

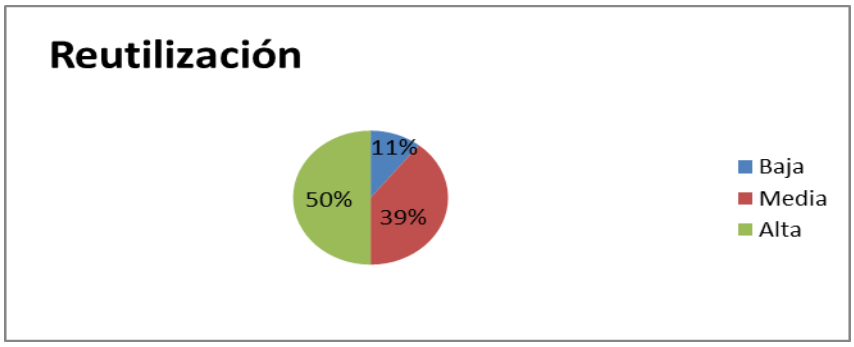

Ilustración 17 Resultados de la evaluación de la métrica RC para el atributo Reutilización.

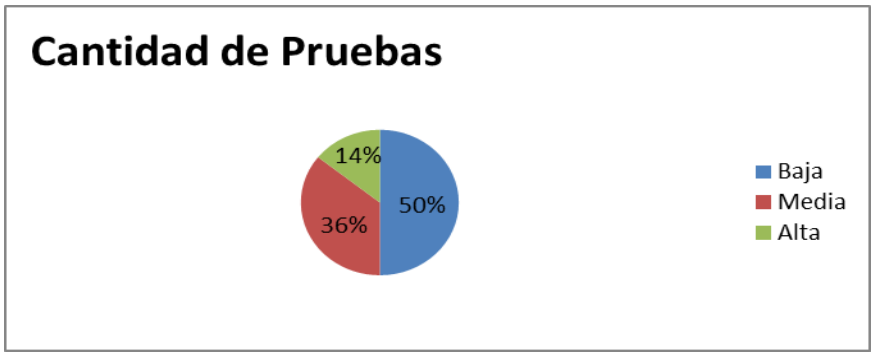

Ilustración 18 Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas.

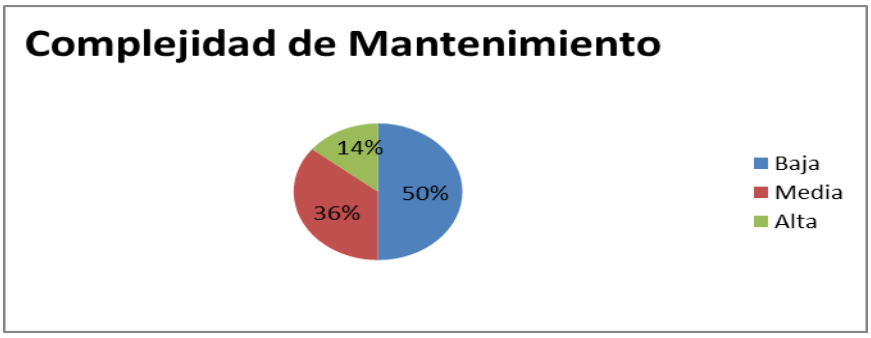

Ilustración 19 Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento.

3.2.3 Matriz de inferencia de indicadores de calidad.

La matriz de inferencia de indicadores de calidad, también conocida como matriz de cubrimiento, es una representación estructurada de los atributos de calidad y métricas utilizadas para evaluar la calidad del diseño propuesto. La misma permite conocer si los resultados obtenidos de las relaciones atributo/métrica son positivos o no, llevando estos resultados a una escalabilidad numérica donde, si los resultados son positivos se le asigna el valor de 1, si son negativos toma valor 0 y si no existe relación es considerada como nula y se representa con un guión simple (-). Luego se puede obtener un resultado general para cada atributo promediando todas sus relaciones no nulas.

A continuación se muestran los resultados obtenidos.

Tabla 8 Resultados de la evaluación de la relación atributo/métrica.

Atributos\Métricas	TOC	RC	Promedio
Responsabilidad	1	(-)	1
Complejidad de Implementación	1	(-)	1
Reutilización	1	1	1
Acoplamiento	(-)	1	1
Complejidad de Mantenimiento	(-)	1	1
Cantidad de pruebas	(-)	1	1

Tabla 9 Rango de valores para la evaluación de la relación atributo/métrica

Categoría	Rango de valores
Malo	<=0.4
Regular	>0.4 y <0.7
Bueno	>=0.7

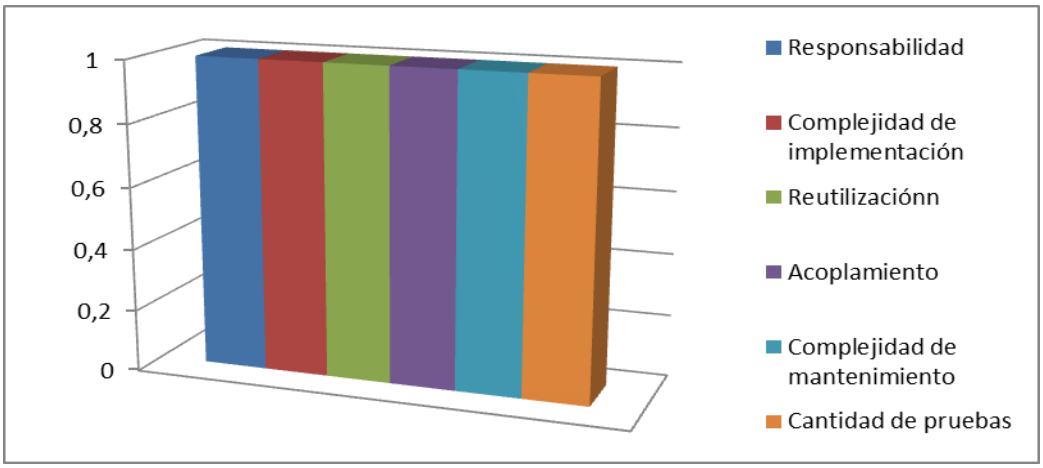

Ilustración 20 Resultados obtenidos de la evaluación de los atributos de calidad.

3.3 Pruebas de software.

Las pruebas del software son un elemento crítico para la garantía de la calidad del sistema informático que se desea realizar, permiten verificar y revelar la calidad de un producto software y representa una revisión final de las especificaciones, del diseño y de la codificación. Dichas pruebas son realizadas con el objetivo de detectar errores en el sistema, por lo que se llevan a cabo durante todo el ciclo de vida del producto.

Los casos de prueba especifican una forma de probar el sistema, incluyendo las entradas con las que se ha de probar, las condiciones bajo las que ha de probarse, así como los resultados esperados.

3.3.1 Pruebas estructurales o de caja blanca.

Las pruebas de caja blanca o técnicas de caja transparente o de cristal, proponen una manera de diseñar los casos de prueba centrándose en el comportamiento interno y la estructura del programa. De esta manera se examina solo la lógica interna del programa, dejando fuera los aspectos de rendimiento del mismo. En los sistemas orientados a objetos, las pruebas de caja blanca pueden aplicarse a los métodos de la clase.

El empleo de este tipo de pruebas permitirá diseñar casos de prueba que comprueben que todas las sentencias del sistema y las condiciones, tanto verdaderas como falsas se ejecuten al menos una vez. Para esto primero se procede a enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo asociado.

```
public function DatosGrid($cabecera, $ip, $userserver, $contraserver, $puertossh, $version) {

    $resul = array();   (1)
    $fichero = $this->ObtenerFichero($ip, $userserver, $contraserver, $puertossh, $version); (1)

    for ($f = 0; $f < count($fichero); $f++) {    (2)
        if (stristr($cabecera, $fichero[$f]['cabecerapadre'])) {   (3)
            $resul[] = $fichero[$f];    (4)
        }
    }    (5)
    return $resul;    (6)
}
```

Ilustración 21 Código fuente de la funcionalidad Cargar Grid.

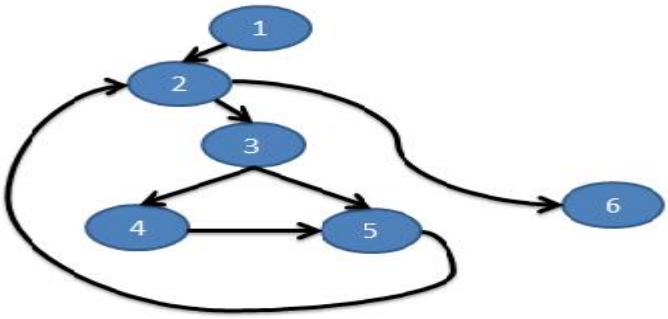

Ilustración 22 Grafo de flujo asociado a la funcionalidad Cargar Grid.

Luego de haber realizado la construcción del grafo se realiza el cálculo de la complejidad ciclomática mediante las tres fórmulas descritas a continuación, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

1. V (G) = (A – N) + 2

Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.

V (G) = (7 - 6) + 2

V (G) = 3

2. V (G) = P + 1

Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = 2 + 1

V (G) = 3

3. V (G) = R

Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del cálculo.

V (G) = 3

El cálculo efectuado utilizando las tres fórmulas antes descritas ha dado como resultado el mismo valor en todos los casos, el resultado fue de 3, lo que indica que existen 3 posibles caminos por donde el flujo puede circular, y determina el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Seguidamente se representan los caminos básicos por los que puede recorrer el flujo:

- Camino básico #1: 1-2-3-4-5-2-6
- Camino básico #2: 1-2-3-5-2-6
- Camino básico #3: 1-2-6

Luego de haber determinado los caminos, se le realiza un caso de prueba a cada uno.

- **Caso de prueba para el Camino básico #1:**

  Descripción: Los datos de entrada cumplirán con los siguientes requisitos: La cabecera, el usuario y la contraseña del servidor serán cadenas. La dirección ip del servidor será un número ej: 10.*.*.*, el puerto será el definido por el que está publicado el protocolo SSH ej: 22, la versión del sistema gestor de base de datos que se encuentra instalada en el servidor con el IP anteriormente definido. Los datos no deben de ser nulos.

  Condición de ejecución: La cabecera será “CONECTION AND AUTHENTICATION”, el ip del servidor 10.7.7.10, el puerto 22, el servidor 8.3, 8.4 o 9.0.

  Entrada: $cabecera = “CONECTION AND AUTHENTICATION”, $ip = 10.7.7.10, $userserver = “postgres”, $contraserver = “postgres”, $puertossh = 22, $version = 8.4.

  Resultados esperados: Se realiza la carga de un arreglo con todos los parámetros pertenecientes a la sección CONECTION AND AUTHENTICATION del fichero postgresql.conf en el servidor 10.7.7.10.

- **Caso de prueba para el Camino básico #2:**

  Descripción: Los datos de entrada cumplirán con los siguientes requisitos: La cabecera, el usuario y la contraseña del servidor serán cadenas. La dirección ip del servidor será un número ej: 10.*.*.*, el puerto será el definido por el que está publicado el protocolo SSH ej: 22, la versión del sistema gestor de base de datos que se encuentra instalada en el servidor con el IP anteriormente definido. Los datos no deben de ser nulos.

  Condición de ejecución: La cabecera será “CONECTION AND AUTHENTICATION”, el ip del servidor 10.7.7.10, el puerto 22, el servidor 8.3, 8.4 o 9.0.

  Entrada: $cabecera = “CONECTION AND AUTHENTICATION”, $ip = 10.7.7.10, $userserver = “postgres”, $contraserver = “postgres”, $puertossh = 22, $version = 8.4.

  Resultados esperados: No se realiza la carga de un arreglo con todos los parámetros pertenecientes a la sección CONECTION AND AUTHENTICATION del fichero postgresql.conf en el servidor 10.7.7.10.

- **Caso de prueba para el Camino básico #3:**

  Descripción: Los datos de entrada cumplirán con los siguientes requisitos: La cabecera, el usuario y la contraseña del servidor serán cadenas. La dirección ip del servidor será un número ej: 10.*.*.*, el puerto será el definido por el que está publicado el protocolo SSH ej: 22, la versión del sistema gestor de base de datos que se encuentra instalada en el servidor con el IP anteriormente definido. Los datos no deben de ser nulos.

  Condición de ejecución: La cabecera será “CONECTION AND AUTHENTICATION”, el ip del servidor 10.7.7.10, el puerto 22, el servidor 8.3, 8.4 o 9.0.

  Entrada: $cabecera = “CONECTION AND AUTHENTICATION”, $ip = 10.7.7.10, $userserver = “postgres”, $contraserver = “postgres”, $puertossh = 22, $version = 8.4.

  Resultados esperados: No se realiza la carga de un arreglo con todos los parámetros pertenecientes a la sección CONECTION AND AUTHENTICATION del fichero postgresql.conf en el servidor 10.7.7.10. El fichero no fue cargado, no hay datos de donde leer.

Luego de realizar las comprobaciones de los casos de pruebas básicos, se llega a la conclusión que el flujo de ejecución de la función es correcto y que cumple con las condiciones que se habían planteado para su buen funcionamiento.

3.4 Conclusiones parciales.

En este capítulo fueron generados los artefactos del modelo de implementación de la solución propuesta, tales como el diagrama de despliegue el cual permite conocer cómo debe ser la infraestructura necesaria para poder implantar el sistema. El diagrama de componentes desarrollado posibilita conocer como son las interacciones entre los distintos componentes del sistema en conjunto con los del marco de trabajo.

Se realizó una validación del diseño expuesto en el capítulo anterior mediante la aplicación de métricas para la evaluación de atributos de calidad, lo cual permitió valorar el diseño propuesto de muy bueno, dado los excelentes resultados obtenidos. Se describieron las pruebas estructurales realizadas que permitieron comprobar el correcto funcionamiento del sistema y garantizan la eficiencia, la solidez y el buen funcionamiento del código implementado.

Las pruebas exploratorias realizadas al sistema, fueron desarrolladas en varias iteraciones con dos probadores y un especialista de calidad, para así validar los requisitos del sistema. Todo esto permitió verificar que las funcionalidades implementadas responden a las necesidades y propósitos de los clientes.

CONCLUSIONES

Al desarrollar la versión 1.0 de la Herramienta para la configuración y administración del gestor de base de datos PostgreSQL, haciendo uso de las tecnologías propuestas, se logró que formara parte del paquete de Herramientas que componen el marco de trabajo Sauxe.

Con la realización del estudio del arte desarrollado sobre las herramientas, tecnologías y técnicas aplicadas como parte de la fundamentación teórica del sistema a desarrollar, se llega a obtener un sistema informático el cual presenta funcionalidades de gran utilidad para sus usuarios.

Los resultados obtenidos fueron analizados mediante la aplicación de las métricas para la validación del diseño y una etapa de pruebas que aseveró el cumplimiento de los requerimientos propuestos.

La herramienta obtenida permitirá a los usuarios administradores de servidores en los cuales se encuentre instalado el gestor de base de datos PostgreSQL, realizar tareas administrativas y de configuración del sistema gestor, de manera fácil y segura.

RECOMENDACIONES

Una vez concluido el proceso de desarrollo se recomienda llevar a cabo la construcción de una segunda versión donde se incorporen las siguientes funcionalidades.

- Realizar un módulo de Reportes el cual se encargará de llevar a cabo la realización de informes en formato XML o PDF de todas las operaciones realizadas por los administradores mediante el uso de la herramienta.
- Realizar un módulo de monitoreo que informará el uso del SGBD de los recursos del servidor donde se encuentra instalado (Memoria RAM, disco duro, etc.).
- Realizar un módulo de Administración de roles, para que así los administradores puedan asignar o denegar permisos a los roles sobre los distintos objetos de las bases de datos.

REFERENCIAS

1. **Clemente, Maikel Yulier Sañudo.** *Desarrollo de la versión 2.0 de la herramienta.* La Habana : s.n., 2010.

2. **Erich Mario Gómez Pérez, Ariel Torres Gálvez.** *Administración y optimización de un Sistema de Base de Datos Descentralizado, en PostgreSQL.* La Habana : s.n., 2008.

3. **PostgreSQL, Equipo de Desarrollo.** *Tutorial de PostgreSQL.* 1999.

4. **Alarcón, Jose Manuel.** Administración de SGBD PostgreSQL. [En línea] 10-11 de 2006. [Citado el: 4 de 3 de 2011.] http://es.scribd.com/doc/32185176/Manual-de-PostgreSQL-Server.

5. **PgAdmin.** PgAdmin. [En línea] PgAdmin. [Citado el: 5 de 3 de 2011.] www.pgadmin.org/.

6. **Caliz Ramos, Doris Cruz.** Generador de reporte para postgres. [En línea] Escuela Politécnica Nacional españa, 2004. [Citado el: 8 de 3 de 2011.] http://bibdigital.epn.edu.ec/handle/15000/805.

7. **Tobias Ratschiller, Rob Casson,Marcellus Barrus,Brian Budnick,Dan Wilson.** PhpPgAdmin. [En línea] PhpPgAdmin, 28 de 2 de 2011. [Citado el: 20 de 3 de 2011.] http://phppgadmin.sourceforge.net/doku.php?id=start.

8. **TLDP-ES/LuCAS, Forma parte de.** PgAccess - a Tcl/Tk interface for PostgreSQL. [En línea] LWP, 2011. [Citado el: 23 de 3 de 2011.] http://www.lawebdelprogramador.com/temas/PostgreSQL/1697-PgAccess_-_a_Tcl_Tk_interface_for_PostgreSQL.html.

9. **Prada Nicot, Héctor y Sánchez González, Kenner.** *Desarrollo de los componentes Puesto de Trabajo y Pagos Adicionales del subsistema Capital Humano integrado al sistema integral de gestión CEDRUX.* La Habana : s.n., 2009.

10. **Asensio, Rafael Menéndez-Barzanallana.** Ingenieria de Sofware. [En línea] Universidad de Murcia, 2011. [Citado el: 5 de 12 de 2010.] http://www.um.es/docencia/barzana/IAGP/Enlaces/CASE_principales.html.

11. **RapidSVN .** RapidSVN . [En línea] RapidSVN , 11 de 2 de 2011. [Citado el: 11 de 12 de 2010.] http://rapidsvn.com/index.php/Documentatio.

12. **Apache.** Apache. [En línea] The Apache Software Foundation, 2011. [Citado el: 15 de 1 de 2011.] http://httpd.apache.org/docs/2.2/.

13. **Aquino, Aylienn y Linares, Yasser.** *Implementación del módulo de Contabilidad General del Sistema Integral de Gestión Cedrux. Universidad de las Ciencias Informáticas.* La Habana : s.n., 2009.

14. NetBeans Docs & Support. [En línea] [Citado el: 23 de 1 de 2010.] http://netbeans.org/kb/index.html.

15. **Commons, Creative.** SOA Agenda. [En línea] [Citado el: 10 de 1 de 2011.] http://soaagenda.com/journal/articulos/que-son-los-frameworks/.

16. **Frederick, Shea, Ramsay, Colin y Blades, Steve 'Cutter'.** *Learning Ext JS.* Birmingham - Mumbai : s.n., 2008. 978-1-847195-14-2.

17. **Ltd, Zend Technologies.** Sitio Zend. [En línea] 2009. [Citado el: 14 de 1 de 2011.] http://www.zend.com/en/products/studio/.

18. **Camejo, Rene Bauta.** *Desarrollo de una herramienta generadora de ficheros de mapeo para la persistencia de objetos de esquemas relacionales basada en doctrine. Universidad de las Ciencias informáticas.* La Habana : s.n., 2009.

19. **Leisis bacallao masid, Laneska Suarez Lamos.** *Análisis y diseño del subsistema cuentas de clientes del proyecto Modernización del sistema bancario cubano. Universidad de las Ciencias informáticas.* La Habana : s.n., 2009.

20. **Roxana Giandini, Gabriela Pérez, Claudia Pons.** *Un lenguaje de Transformación específico para Modelos de Proceso del Negocio.* La Habana : s.n., 2009.

21. **Marley, Jimi.** Programación Orientada a Objetos. [En línea] Programación en Castellano, 2011. [Citado el: 5 de 2 de 2011.] http://www.programacion.com/articulo/programacion_orientada_a_objetos_279.

22. **PHP, Grupo de documentación de.** *Manual de PHP.* 2001.

23. **Foundation, Mozilla Europe y Mozilla.** Características de Firefox. [En línea] 2011. [Citado el: 14 de 2 de 2011.] http://www.mozilla-europe.org/es/firefox/features/.

24. **Hernández Cisneros, Sergio y Sarduy Pérez, Mileidy Magalys.** *Propuesta de modelo de desarrollo de software tecnológico del Centro de Soluciones de Gestión.* 2009.

25. **Nuseibeh, Bashar y Easterbrook, Steve.** *Proceedings of the Conference on The Future of Software Engineering.* 2000.

26. **IEEE.** *Compilation of IEEE Standard Computer Glossaries.* 1991. 1559370793.

27. **Kotonya, Gerald y Sommerville, Ian.** *Requirements Engineering: Processes and Techniques.* s.l. : John Wiley & Sons, 1998. 978-0-471-97208-2.

28. **Sommerville, Ian y Sawyer, Pete.** *Requirements Engineering: A good practice guide.* s.l. : Lancaster University, 1997.

39. **Gamma, Erich, y otros.** *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 1994. 0-201-63361-2.

30. **Grosso, Andrés.** Prácticas de Software. [En línea] 21 de 3 de 2011. [Citado el: 12 de 4 de 2011.] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

31. *Un método para el diseño de la base de datos a partir del modelo orientado a objetos.* **Hernández González, Dra. Anaisa.** México DF : Computación y Sistemas, 2004, Vol. 7. 1405-5546.