Facultad 4

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

Desarrollo del Módulo Réplica de la Plataform a para la gestión del aprendizaje ZERA del proyecto Alfaomega

Autor: Mario Martínez Torne.

Tutor: Ing. Enrique José Altuna Castillo.

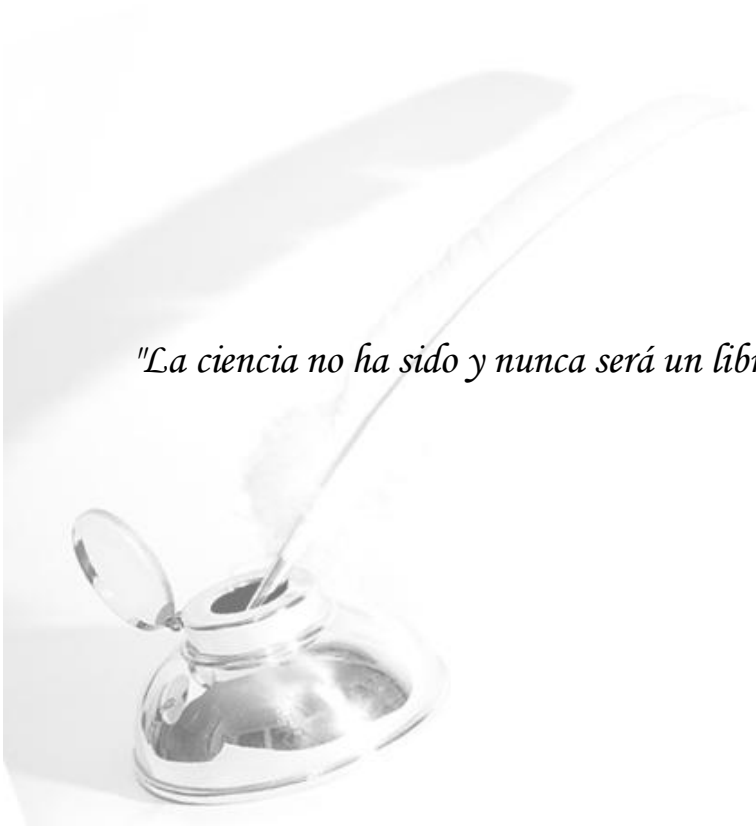
Cotutor: Ing. Yuneikys Recio Miranda.

MSc. Roberto López Dosagües.

Ing. Tamara Rodríguez Sánchez.

2010-2011

Pensamiento



"La ciencia no ha sido y nunca será un libro terminado. Cada éxito importante trae nuevos interrogantes."

Albert Einstein

Declaración de autoría:

Declaramos que somos los únicos autores del trabajo “Desarrollo del Módulo Réplica de la plataforma para la gestión del aprendizaje ZERA del proyecto Alfaomega” y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____	_____
Mario Martínez Torne	Ing. Enrique J. Altuna Castillo
Autor	Tutor

Agradecimientos:

A mis profesores que durante estos años me han brindado todos sus conocimientos.

A mis tutores que han sido maravillosos y nos guiaron durante todo este tiempo.

A mis compañeros de estudio.

A la Revolución por darme esta oportunidad.

A todos aquellos que han aportado el más mínimo granito para alcanzar la mayor de mis metas.

A mis padres Ángela Idalmis y Mario que es lo más grande que tengo en la vida, que me han dado todo de sí, que han sabido brindarme su amor y guiarme por el camino del estudio y que se han sacrificado por ver este sueño hecho realidad.

A mi hermanita por existir.

A Susan que siempre ha sido incondicional conmigo.

A mis abuelos por su cariño.

A Ary por su amor, paciencia y comprensión.

A mis primos Dorge, Nelsi, Yanet, Maye y Mayelin por ser más que eso, unos hermanos.

A mis tíos y tías: Elsa, Mevis, Dalgis, Suset, Gusta, Guille, Ernesto, Nelson.

A mi antigua compañera de tesis Miro por su contribución.

Al profesor Arcel Labrada por sus aportes.

A mis amigos todos, esos amigos que forman mi nueva familia, que han sido maravillosos y a los cuales les debo mucho: Carlos, Ale, Guille, Joel, Ancel, Yunier, May, a los alphas, a mi incansable combo de los party, al piquete del apto 93104 por soportar mis pesadeces, a los de la vieja escuela... a todos los que me han brindando ese bello sentimiento que es la amistad...

Dedicatoria:

Mario:

A mis padres les dedico este trabajo y todos mis logros en la vida…

RESUMEN

Las tecnologías de la información y las comunicaciones han permitido avances sustanciales en todos los procesos de la vida cotidiana. Los procesos de enseñanza y aprendizaje no han estado exentos de las potencialidades y los beneficios de las mismas, permitiendo mejoras en la obtención de conocimiento. Las plataformas educativas, son resultado de la aplicación de tecnologías en los procesos de enseñanza. Su perfeccionamiento constituye objeto de investigación constante en los centros de enseñanza universitaria, con el objetivo de mejorar sus funcionalidades y una mayor interacción del triángulo interactivo profesor-estudiante-contenidos. El objetivo de la presente investigación, está encaminada a perfeccionar el cumplimiento de los objetivos mencionados, en el que se desarrolla un módulo Réplica que forma parte de la plataforma educativa nombrada ZERA del proyecto Alfaomega desarrollada por profesores y estudiantes de la Universidad de las Ciencias Informáticas. Este módulo permite efectuar la réplica y sincronización de los datos entre los servidores de base datos de la plataforma. En la investigación se describe un estudio sobre los sistemas de réplica y los principales ambientes en la sincronización de los datos. Se puntualiza un estudio sobre las metodologías de desarrollo de software y se decide la selección de la metodología idónea para su desarrollo. La investigación describe además las funcionalidades del módulo Réplica. Forma parte de la investigación la validación del módulo Réplica lo que justifica la validez del cumplimiento de los objetivos propuestos.

Palabras clave: réplica, sincronización, módulo.

Índice de contenido

Índice de Figuras

Índice de Figuras

Índice de figuras

Índice de tablas

Introducción

La llamada tercera Revolución Científico-Tecnológica, ha dado un salto sustancial en todas las ramas y esferas de la sociedad. En particular, en la esfera de la educación, el desarrollo de tecnologías de la información, constituye una herramienta que facilita y permite la realización de los procesos de enseñanza y aprendizaje, en el que se abren espacios para que estudiantes y profesores administren y tengan acceso a la información.

El apoyo brindado a los procesos de enseñanza y aprendizaje por la tecnología informática a través del Internet, ha proporcionado métodos para reforzar la educación a distancia y el apoyo a las clases presenciales. El empleo de las tecnologías de la información y las comunicaciones (TIC), ha favorecido el desarrollo de los Sistemas para la Gestión del Aprendizaje en inglés *Learning Management System* (LMS) en línea. Los LMS son plataformas a través de las cuales se administran y gestionan cursos en línea, que brinda los componentes que integran los procesos de formación, con la capacidad de integrar a personas, procesos y equipos en aulas virtuales para el aprendizaje. Se basan en los principios del aprendizaje colaborativo y garantiza a los sujetos actuantes del proceso, el acceso a materiales de trabajo, comunicación, recursos virtuales educativos, seguimiento y acceso del proceso evaluativo. En la actualidad son utilizados para crear grupos, espacios y comunidades virtuales para la discusión, investigación y obtención de conocimientos sobre temas en específicos.

Cuba como potencia educacional prolifera en sus centros educacionales el uso de las LMS y se encuentra inmersa en su desarrollo. La Universidad de las Ciencias Informáticas (UCI), es un centro de educación superior de nuevo tipo, que tiene entre de sus objetivos, desarrollar proyectos de carácter informático para Cuba y el resto del mundo. Entre estos se distingue, por su impacto en el ámbito social y educacional, el proyecto Alfaomega, que consiste en el desarrollo de una plataforma para la gestión del aprendizaje, nombrada ZERA, de tipo e-learning. Integra los principales conceptos de los hiperentornos, las mejores prácticas y elementos arquitectónicos de soluciones similares y las principales especificaciones y estándares educativos desarrollados y utilizados a nivel mundial en plataformas de aprendizaje colaborativo.

La plataforma ZERA gestiona todo lo referente a los estudiantes, profesores y asignaturas de varias instituciones educacionales. Esta plataforma cuenta con una base de datos local en cada centro afiliado, que tiene solo la información referente al mismo, y una central, con la totalidad de los datos, para un

control de cada uno de los centros de estudios asociados a ella. La plataforma debe mantener una estrecha relación entre la base de datos central e instituciones asociadas de manera que la información se actualice en ambos sentidos desde varias localizaciones. Actualmente la plataforma no cuenta con un mecanismo que garantice que se actualicen los datos y que exista coherencia entre la información del servidor central y la información contemplada en el servidor local.

A partir del análisis de lo antes expuesto, se define como **problema de investigación**:

¿Cómo garantizar la réplica y sincronización de los datos entre los servidores de base de datos de la plataforma para la gestión del aprendizaje ZERA?

Objetivo general:

Desarrollar el análisis, diseño, implementación y prueba del módulo Réplica de la plataforma para la gestión del aprendizaje ZERA del proyecto Alfaomega.

Objeto de Estudio:

Los procesos de Réplica y sincronización de base de datos.

Campo de Acción:

Base de datos de la plataforma para la gestión del aprendizaje ZERA.

Idea a defender:

Si se desarrolla un módulo para realizar los procesos de Réplica en la plataforma ZERA, se garantiza la sincronización de los datos entre los servidores centrales y los servidores locales asociados.

Objetivos específicos:

1. Realizar el estudio del marco teórico conceptual que permitirá el posterior desarrollo de la investigación.
2. Realizar el análisis del módulo Réplica de la plataforma para la gestión del aprendizaje ZERA del proyecto Alfaomega.
3. Diseñar el módulo Réplica de la plataforma para la gestión del aprendizaje ZERA del proyecto Alfaomega.
4. Implementar el módulo Réplica de la plataforma para la gestión del aprendizaje ZERA del proyecto Alfaomega, sin dejar constancia en el documento la escritura integral del desarrollo módulo.

5. Probar el módulo Réplica de la plataforma para la gestión del aprendizaje ZERA del proyecto Alfaomega.

Para dar cumplimiento a los objetivos específicos antes señalados se cumplirán las siguientes **tareas de investigación:**

1. Revisión bibliográfica para conformar los fundamentos teóricos de la investigación.
2. Investigación del estado del arte de herramientas para ejecutar los procesos de réplica.
3. Identificación de requerimientos funcionales y no funcionales.
4. Revisión de la propuesta de arquitectura para realizar la réplica.
5. Descripción del Módulo Réplica de la plataforma ZERA.
6. Descripción de las funcionalidades del sistema.
7. Modelación de los requerimientos.
8. Realización del diseño de los prototipos de interfaz de usuario.
9. Realización del diseño del módulo.
10. Implementación del módulo Réplica teniendo en cuenta el alcance de la investigación.
11. Realización de pruebas al mecanismo de réplica.

Los **métodos de investigación** que soportan el desarrollo del presente trabajo son la combinación dialéctica de los métodos Teóricos y Empíricos. Entre los Teóricos se emplearán:

- ***Analítico – Sintético:*** la investigación se encaminará a partir del análisis de los conceptos y métodos que permitirán la realización de sincronización de datos a través de la réplica de base de datos y métodos para realizar réplicas de datos y la correspondiente selección y/o síntesis de las posibles herramientas a utilizar para el desarrollo del producto informático.
- ***Histórico – Lógico:*** la investigación se realizará a partir del estudio del estado del arte de la problemática analizada, que permitirá conocer la trayectoria y desarrollo de aplicaciones para garantizar la sincronización de datos, teniendo en cuenta sus antecedentes históricos, las investigaciones realizadas y los resultados obtenidos por otros autores al efectuar dichas investigaciones, lo que dará la posibilidad de seleccionar aquellas herramientas, lenguajes y metodologías lógicas y adecuadas para el desarrollo de la aplicación.

Métodos Empíricos:

- **Observación**: permitirá la ejecución correcta del proyecto, alertando sobre posibles atrasos y fallas en el cumplimiento de las tareas propuestas.
- **Revisión de documentos**: permitirá la elaboración del marco teórico conceptual y el desarrollo del diseño de la propuesta de solución del módulo.

Estructura capitular:

Capítulo I: Fundamentación teórica

*S*e abordan los conceptos y características fundamentales de las bases de datos, los procesos de réplica y sincronización, el estudio del estado del arte de soluciones de réplica existentes. Se realiza un estudio y selección, de las metodologías y software usado actualmente, para dar solución a los problemas de sincronización de datos.

Capítulo II: Características del sistema

Describe las características del módulo propuesto para darle solución al problema, se describen los procesos que generan la situación problemática y los procesos que se automatizarán. Se especifica el modelo de negocio, las características del sistema planteadas en requisitos funcionales, no funcionales y descripción de los casos de uso.

Capítulo III: **Análisis y diseño del módulo Réplica**

Agrupa la propuesta de análisis y el diseño del módulo Réplica. Incluye la definición del Modelo del Análisis, los Diagramas de interacción que representan el flujo principal y alterno de los eventos de los casos de uso del sistema y se describen las clases del diseño.

Capítulo IV: **Validación de la solución propuesta.**

Concentra los aspectos de las pruebas que se le realizan al módulo Réplica. Validando en cierta medida la calidad del módulo a través de los métodos de prueba aplicados, y la descripción de los casos de prueba de integración.

Capítulo I: Fundamentación teórica

Introducción

El objetivo que sigue este capítulo es la selección de la metodología a seguir y herramientas a utilizar en el desarrollo de la propuesta del módulo Réplica para la plataforma ZERA del proyecto Alfaomega. Se resumen los principales conceptos y características de las bases de datos, los procesos de réplica y sincronización, así como los diferentes métodos existentes para implementarlos. Se realiza un estudio del estado del arte de soluciones de réplica existentes y un análisis de las tendencias, tecnologías y metodologías usadas actualmente para el desarrollo de sistemas, a través de un análisis crítico y valorativo de las posibles herramientas.

1.1 Base de datos

Con el creciente flujo de datos existentes en la actualidad, la acumulación y organización de grandes cantidades de información, puede convertirse en un inconveniente para las empresas y organizaciones, lo que provoca en muchos casos fallos en la búsqueda y pérdida de datos de importancia. Para solventar esta situación es muy común el empleo de las Base de datos.

Se han dado distintas definiciones de este concepto, tales como:

El término de Bases de Datos fue escuchado por primera vez en 1963, en un simposio celebrado en California, USA. “Una base de datos se puede definir como un conjunto de información relacionada que se encuentra agrupada o estructurada.” (1)

“Grupo de archivos relacionados. Las bases de datos informatizadas facilitan un rápido acceso a la información necesaria para la toma de decisiones.” (2)

A partir de los conceptos anteriores, para esta investigación se asume como Base de Datos (BD), a una colección de datos o información almacenados de forma organizada y relacionados entre sí. Estas permiten almacenar grandes cantidades de información con el objetivo de facilitar la búsqueda y actualización de la misma, permitiendo el acceso directo a los datos y la administración de estos por una serie de aplicaciones.

Las BD se pueden clasificar en estáticas, que son aquellas que sus datos son de sólo lectura; y las dinámicas en las que sus datos pueden variar a través el tiempo. Además, existen varios modelos de base

de datos, entre los que se encuentra, la base de datos distribuidos (BDD), siendo estas una colección de múltiples instancias de BD, distribuidas a través de un variado número de sitios en una red.

1.1.1 Sistema de Bases de datos distribuidos (SBDD)

Un sistema de bases de datos distribuidas, es donde los datos se encuentran ubicados en distintos ordenadores, que pueden estar o no en la misma localización geográfica. Existe comunicación entre las máquinas que pertenecen al SBDD y comparten el mismo esquema global de la base de datos. El control de los datos en los distintos nodos del sistema, es estructurado por un sistema central, donde un administrador central es responsable de los datos a nivel global, y en un administrador local por cada nodo con un nivel de autonomía local diferente. Los usuarios de un nodo pueden acceder a sus datos desde otro nodo. Cuando en un SBDD surge alguna falla en un nodo, esto no afecta a los nodos restantes y cuando son duplicados los datos entre los nodos, estos se pueden encontrar en cualquiera de los nodos que pertenecen al sistema. (3)

De las ventajas de uso de SBDD, existen varios escritos en el que las relacionan. Para esta investigación se asume las definidas en un artículo emitido por la Universidad de Colima: (4)

- ✓ Los datos pueden colocarse cerca del punto de su utilización, de forma que el tiempo de comunicación sea más corto. Varias computadoras operando en forma simultánea pueden entregar más volumen de procesamiento que una sola computadora.
- ✓ Los datos duplicados aumentan su confiabilidad. Cuando falla una computadora, se pueden obtener los datos extraídos de otras computadoras. Los usuarios no dependen de la disponibilidad de una sola fuente para sus datos.
- ✓ Pueden variar su tamaño de modo sencillo. Se pueden agregar computadoras adicionales a la red conforme aumentan el número de usuarios y su carga de procesamiento. Frecuentemente, es más fácil y más barato agregar una nueva computadora más pequeña que actualizar una computadora única y centralizada.
- ✓ Se pueden adecuar de una manera más sencilla a las estructuras de la organización de los usuarios.

En resumen, un SBDD es un conjunto de BD distribuidas en la red, que comparten un mismo esquema global y se comunican entre ellas. En estas se pueden realizar operaciones tanto locales como distribuidas, permitiendo que los usuarios accedan a sus datos desde cualquier sitio del SBDD.

1.2 Réplica de base de datos

Al definir este concepto se encuentra una variada literatura que hace referencia, la que se comenta en lo adelante.

“La replicación es el proceso de copia de datos de un alojador a otro por bloques y de forma diferencial. Por lo tanto, como la replicación se suele llevar a cabo a nivel de archivo o de libro, a medida que cambian cosas en el libro en cuestión, los bloques que han cambiado en la fuente se replican inmediatamente en el destino.” (5)

La Réplica de datos, es el proceso de copiar y de mantener los objetos de la base de datos en las múltiples bases de datos que incorporan un sistema de base de datos distribuida. (6)

El término replicación se refiere, al conjunto de tecnologías destinadas a la copia y distribución de datos y objetos de base de datos desde una base de datos a otra, para luego sincronizar ambas bases de datos y mantener su coherencia. (7)

La réplica de BD es la capacidad que tienen varias aplicaciones, para duplicar y distribuir datos y objetos entre las bases de datos en un SBDD. Tiene como objetivo brindar copias para la seguridad y es una forma de prevenir la inactividad de la base datos. Mejora el funcionamiento y garantiza la disponibilidad e integridad de los datos de las aplicaciones. La replicación de datos permite eliminar el impacto de los cortes en la red, restringir el acceso a los datos, reduce y balancea la carga en los servidores centralizados. (8)

Con la réplica de datos, una aplicación puede tener acceso a una base de datos local más que al servidor central, reduciendo así el tráfico en la red y optimizar su funcionamiento. También garantiza que si ocurre una falla en el servidor local, la aplicación pueda seguir funcionando ya que otros servidores replicados permanecen disponibles. (6)

Existen distintos entornos o ambientes de replicación como son:

- ✓ Maestro-Esclavo (master-slave): o de solo lectura, permite a un solo maestro (servidores que pueden modificar los datos) recibir consultas de lectura/escritura, mientras los esclavos solo pueden aceptar consultas de lectura. Si se necesita hacer algún cambio en la base de datos este debe ser realizado en el esquema del maestro, evitando así problemas en el momento de la sincronización. Cuando se desea emplear una arquitectura Maestro – esclavo debe ser creada la conexión de forma abierta en el maestro para el esclavo o los esclavos. A través de esta los esclavos leen las modificaciones realizadas en el maestro y sincronizan sus datos con los del maestro. Soporta replicación en un sentido exclusivamente. (9)
- ✓ Maestro-Maestro (multi-master): permite que varios sitios actúen como pares iguales, para manejar los grupos de objetos replicados en la base de datos. Cada servidor en los distintos nodos de la red es maestro y por tanto la conexión entre los servidores de la red es de maestro a maestro. A diferencia de maestro-esclavo, en este tipo de réplica se puede modificar el esquema de la base de datos de cualquier réplica en cualquier sitio. Permite leer o escribir consultas que serán enviadas a varias computadoras replicadas, aumentando el rendimiento mediante la sincronización de los cambios entre servidores. Puede ser usado con el objetivo de tener una forma de recuperar los sitios ante posibles desastres o incidentes y ofrecer sitios con sistemas de alta disponibilidad y para balancear la carga de peticiones desde cualquier sitio. (9)

El ambiente de replicación efectivo para darle respuesta a la problemática planteada, es el entorno maestro-maestro, por su capacidad de replicación bidireccional. Permite que las transferencias de datos se efectúen en ambos sentidos, pudiéndose replicar la información desde el servidor central al local o viceversa. Con esto se reduce el tráfico en la red y el acceso a la aplicación central, optimizando su rendimiento.

1.2.1 Modelos de distribución de datos

La distribución de datos consta con dos modelos de distribución básicamente aplicados a cada uno de los ambientes de replicación antes vistos, basados en la sincronización de base de datos. La sincronización, es la coherencia entre los datos de las tablas de las bases de datos, que intervienen en el ambiente de replicación en un sistema distribuido. Es la actualización de las distintas bases de datos de un SBDD en un ambiente de replicación, mediante el intercambio de los registros actualizados en cada nodo. La

sincronización no es completa hasta que los cambios realizados en cada nodo han sido actualizados en el otro.

Estos modelos de distribución son:

- ✓ Asincrónica: conocida como almacena-y-reenvía, captura cualquier cambio local, los almacena en una cola, y, a intervalos regulares, replica estos cambios en los servidores remotos. Con esta forma de réplica, hay un período de tiempo antes de que todos los sitios alcancen la convergencia de datos. (10)
- ✓ Sincrónica: llamada réplica en tiempo real, aplica cualquier cambio o ejecuta cualquier procedimiento reproducido en todos los sitios que participan en el ambiente de réplica como parte de una sola transacción. Si el procedimiento falla en cualquier sitio, entonces la transacción entera se anula. La réplica sincrónica asegura la consistencia de datos en todos los sitios en tiempo real. (10)

La réplica sincrónica, es más eficiente en entornos distribuidos donde los nodos se encuentren a corta distancia unos de otros, pues esta incide en el rendimiento de la aplicación y aumenta el tráfico en la red. Por otra parte, la réplica asincrónica, es más óptima para ambientes distribuidos donde los nodos estén a larga distancia unos de otros, mejorando el rendimiento de la aplicación, y reduce el tráfico en la red.

Para la replicación de las bases de datos de la plataforma ZERA, se empleará el modelo de distribución asíncrono, con el fin de mantener coherentes y actualizados los datos entre los servidores de la plataforma. Con la aplicación de la réplica maestro-maestro asíncrona, los cambios podrán actualizarse en ambos sentidos, la aplicación será tolerante a fallos en la red, los usuarios podrán tener acceso a su información desde cualquier sitio. El acceso a la aplicación se realizará a nivel local más que a nivel central, aumentando el funcionamiento del sistema. En el caso de que el servidor local este averiado, no existe interferencia en la disponibilidad de los datos de los usuarios de la plataforma ZERA, y viceversa.

1.3 Análisis de otras soluciones existentes

Del estudio realizado para la conformación del estado del arte, se determinó, que en el mundo existen varias soluciones para resolver problemas de sincronización de datos entre servidores.

A continuación se exponen algunos ejemplos de las existentes a nivel internacional, las cuales son privativas:

1. **EMC MirrorView:**
    - Replicación remota basada en arreglos.
    - Espejeado de datos independiente del host.
    - Síncrono y Asíncrono.
    - Administración centralizada. (11)
2. **RecoverPoint:**
    - Protección continua de datos.
    - Replicación remota continua.
    - Protección de datos local y remota simultánea.
    - Reducción del ancho de banda.
    - Registro diario de los cambios en los datos a nivel de bloques. (12)

En Cuba, en la UCI es donde existe la mayor experiencia de las propuestas de soluciones para la réplica de base de datos a nivel nacional, destacándose:

1. **Réplica bidireccional basada en control de cambios:** fue propuesta por el Proyecto Identidad de la Facultad 1, se caracteriza por ser bidireccional y basada en control de cambios no en colas, lo que evita las transmisiones redundantes. Soporta ambientes heterogéneos y no tiene en cuenta el gestor de base de datos. Soporta particionamiento horizontal de los datos y garantiza la integridad de los datos, ya que provee mecanismos de detección y resolución de conflictos, está implementado sobre la plataforma .NET y posee una topología flexible y escalable. No soporta el entorno de replicación maestro-esclavo. (13)
2. **Solución para la réplica de datos del Proyecto Prisiones:** soporta la replicación de transacciones a través de *Hibernate*, JDBC y la replicación de acciones a través de disparadores, se integra con el sistema gestor de base de datos Oracle en estos momentos, aunque se encuentran en fase de desarrollo el soporte para PostgreSQL, MySQL y Microsoft SQL Server. Utiliza los protocolos TCP/IP, FTP y HTTP para la transferencia de datos de replicación, los archivos de gran tamaño son enviados por FTP permitiendo resumir la transmisión en caso de

interrupciones en la red. Presenta interfaz visual web, por lo que se puede administrar y configurar remotamente solo con emplear un navegador web. (13)

3. **Chronos: sistema de replicación asíncrona multi-maestro para postgresql:** herramienta de réplica multi-maestro asíncrona que pretende satisfacer la mayor parte de los requerimientos de este tipo de soluciones. Ha sido diseñada a partir de necesidades particulares de un escenario de réplica asumiendo las principales potencialidades de las soluciones similares en la actualidad para Postgresql. (14)

- ✓ Las soluciones antes mencionadas, no satisfacen todas las necesidades de la plataforma para la gestión del aprendizaje ZERA, pues:
- ✓ EMC MirrorView esta implementada sobre software propietario.
- ✓ Solución para la réplica de datos del Proyecto Prisiones presentan incompatibilidad con el sistema gestor de base de datos Postgres.
- ✓ Chronos no se puede embeber en la plataforma para la gestión del aprendizaje ZERA.

Por todo lo antes expuesto se arriba a la conclusión de que, es necesaria la investigación, para desarrollar una nueva propuesta de solución a los problemas de réplica y sincronización entre servidores de base de datos.

1.4 Herramientas de replicación

1.4.1 Pyreplica

Es una herramienta simple que permite la replicación asincrónica de datos para PostgreSQL, utilizando disparadores implementados en python, señales, secuencias y un script cliente influenciado por Slony pero mucho más simple y fácil. Soporta además la replicación en un entorno maestro-maestro, donde se replican datos en ambas direcciones. Está programado en Python por lo que es simple y flexible, permitiendo una fácil instalación, no se necesita aprender nuevos comandos para su administración, es muy fácil de adaptar manualmente, es eficiente pues tiene un bajo impacto en el uso de memoria y la red al no utilizar transmisión secuencial (*polling*) y es multiplataforma. (15)

1.4.2 PgCluster

Es un sistema de replicación sincrónico multi-maestro para PostgreSQL. Consiste en tres tipos de servidores, un servidor para balance de carga, un cluster de base de datos y un servidor de réplica. Tiene dos funciones principales:

- ✓ ***Compartir carga:*** **l**a carga de la sesión de las demandas es distribuida. Es efectivo en aplicaciones Web donde existe gran demanda por el número de peticiones.
- ✓ ***Alta disponibilidad:*** **c**uando ocurre un fallo en el Cluster DB, el servidor de balance de carga y el de replicación separan el fallo del sistema, y continúa el servicio que usa el DB restante.

PgCluster realiza la replicación a nivel de sentencias, por lo que, en algunos casos no se replican correctamente. El Cluster DB cuando es reparado puede restaurarse dinámicamente a un sistema, sin detener el servicio. Los datos son copiados automáticamente a la DB restaurada o añadidos desde otra DB. (16)

1.4.3 SymmetricDS

Es un producto desarrollado en Java funciona con el *driver Java Database Connectivity* (JDBC), está habilitado para la web permitiendo replicar tablas entre base de datos relacionales en tiempo real, con base de datos independientes. El *software* está diseñado a escala de un gran número de bases de datos (1000 nodos), trabaja a través de conexiones de baja velocidad y puede soportar los períodos de interrupción de la red. Puede funcionar con los siguientes gestores: MySQL, Oracle, SQL Server, PostgreSQL, DB2, Firebird, HSQLDB, H2, y Apache Derby. La sincronización puede ser configurada para enviar o extraer datos en un intervalo de tiempo. Además, se comporta como multi-maestro. Es multiplataforma, posibilita adicionar nodos al sistema con solo agregar algunas tuplas en las tablas de configuración; permite definir el intervalo de tiempo con que se desea replicar, requiere de una máquina virtual de Java para su ejecución, además de que soporta cortes en la red. Está licenciado como software libre bajo licencia LGPL. (17)

1.5 Metodologías

1.5.1 Proceso Unificado de Rational (RUP)

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unido técnicas de desarrollo, a través de UML, y trabajo de distintas metodologías utilizadas por los clientes. Es una

metodología de desarrollo de software robusta debido a la alta cantidad de procesos y documentos que requiere. Describe cómo aplicar efectivamente las mejores prácticas comprobadas comercialmente para el desarrollo de software. Proporciona una guía para ordenar las actividades de un equipo, dirige las tareas de cada desarrollador por separado y del equipo como un todo, especifica los artefactos que deben desarrollarse y ofrece criterios para el control y la medición de los productos y actividades de proyectos. (18)

La existencia de un sistema acontece a través de ciclos de desarrollo, en cada uno se repite el proceso unificado de desarrollo. Cada ciclo consta de 4 fases, Inicio, Elaboración, Construcción y Transición. El ciclo de vida de RUP se caracteriza por: dirigido por casos de uso que orientan el proyecto a la importancia para el usuario y lo que este quiere; centrado en la arquitectura relacionando la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden; iterativo e incremental pues divide el proyecto en pequeños proyectos donde los casos de uso y la arquitectura cumplen sus objetivos de manera más depurada. (19)

En su modelación define como sus principales elementos:

- ✓ Trabajadores: define el rol de los integrantes del equipo de trabajo, estos realizan las actividades y son propietarios de elementos.
- ✓ Actividades: tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- ✓ Artefactos: productos tangibles del proyecto que son producidos, modificados y usados por las actividades.
- ✓ Flujo de actividades: secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

En RUP las actividades se agrupan en grupos lógicos, quedando estructurado en 9 flujos de trabajo, los 6 primeros se conocen como flujos de ingeniería y los 3 últimos como de apoyo.

Flujos de trabajo de ingeniería:

- 🗁 **Modelamiento del negocio:** describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.

- **Requerimientos:** define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- **Análisis y diseño:** describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- **Implementación:** define cómo se organizan las clases y objetos en componentes, cuáles nodos que se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- **Prueba (Testeo):** busca los defectos a lo largo del ciclo de vida.

1.5.2 Extreme Programming (XP)

XP es una metodología ágil actualmente muy exitosa, usada para proyectos a corto plazo y equipo pequeño, con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico. Está integrada por cuatro partes elementales: historia de usuarios, roles, procesos y prácticas. Consiste en la programación rápida, donde se debe tener al usuario final como parte del equipo de desarrollo, pues se basa en la retroalimentación continua entre el cliente y el equipo, además de la sencillez en las soluciones implementadas. Se centra en fomentar las relaciones interpersonales, promoviendo el trabajo en equipo. (18)

Se caracteriza por: (20)

- ✓ Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- ✓ Frecuente interacción del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- ✓ Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- ✓ Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- ✓ Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que en más sencillo

hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

Se basa fundamentalmente en: (18)

- ✓ **Pruebas Unitarias:** realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se pueda hacer pruebas de las fallas que pudieran ocurrir.
- ✓ **Re-fabricación:** reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- ✓ **Programación en pares:** consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

1.5.3 SCRUM

SCRUM es una metodología ágil desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Es iterativa e incremental donde cada ciclo o iteración termina con una pieza de software ejecutable que incorpora nueva funcionalidad. Las iteraciones en general tienen una duración entre dos y cuatro semanas. Se focaliza en priorizar el trabajo en función del valor que tenga para el negocio, maximizando la utilidad de lo que se construye y el retorno de inversión. Está diseñado especialmente para adaptarse a los cambios en los requerimientos, por ejemplo en un mercado de alta competitividad. Los requerimientos y las prioridades se revisan y ajustan durante el proyecto en intervalos muy cortos y regulares. De esta forma, se puede adaptar en tiempo real el producto que se está construyendo a las necesidades del cliente. Se busca entregar software que realmente resuelva las necesidades, aumentando la satisfacción del cliente. (20)

Tiene un conjunto de reglas muy pequeño y muy simple y está basado en los principios de inspección continua, adaptación, auto-gestión e innovación. El cliente se entusiasma y se compromete con el proyecto dado que ve crecer el producto iteración a iteración y encuentra las herramientas para alinear el desarrollo con los objetivos de negocio de su empresa. Por otro lado, los desarrolladores encuentran un ámbito propicio para desarrollar sus capacidades profesionales y esto resulta en un incremento en la motivación de los integrantes del equipo. (20)

Esta metodología es aconsejable utilizarla en proyectos donde los requerimientos cambien constantemente, es necesaria la reunión diaria entre los integrantes del equipo de desarrollo, para así poder lograr la integración de todos.

1.5.4 Lenguaje Unificado de Modelado (UML)

UML (Unified Modeling Language) permite la modelación de sistemas con tecnología orientado a objetos, es un lenguaje para visualizar un sistema a través de gráficos, especificar las características que debe tener el sistema, que permite a partir de los modelos diseñados construir el sistema y documentar a través de los elementos gráficos el sistema desarrollado. Es una forma de modelar los procesos de negocio, funciones del sistema, escribir clases, esquemas de base de datos y componentes de software reusables.

Su uso es independiente del lenguaje de programación y de las características del proyecto, pues este fue diseñado para modelar cualquier tipo de proyectos. Está formado por varios diagramas que son la representación gráfica de elementos y sus relaciones que visualizan el sistema desde distintas perspectivas. (21)

Un modelo UML describe lo que debería hacer un sistema, pero no dice cómo implementarlo.

1.6 Herramientas para la modelación del software

1.6.1 Visual Paradigm para UML

Es una herramienta profesional para el modelado visual de un software, es propiedad de la compañía Visual Paradigm con licencia gratuita y comercial. Resume un conjunto de funcionalidades para el desarrollo de software. Es una aplicación multiplataforma que soporta el ciclo de vida del desarrollo de software. Visual Paradigm admite el modelado de los requerimientos, procesos de negocio y modelado de base de datos. Proporciona la generación automática de documentos y código a partir de los diagramas en diferentes formatos.

Es fácil de usar, soporta la última notación UML 2.1, ingeniería inversa, generación de código, exportación/importación XML, generador de informes, editor de figuras, integración con MS Visio, plug-in, integración IDE con Visual Studio, IntelliJ IDEA, Eclipse, NetBeans y otros. Entre sus nuevas características se incluyen el modelado colaborativo con CVS y Subversion. (22)

Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. (23)

Visual Paradigm se encuentra actualmente es muy utilizado, está entre las aplicaciones más potente para el modelado de aplicaciones. Es sencilla de usar y presenta la gran ventaja de ser multiplaforma.

1.6.2 ArgoUML

ArgoUML es una herramienta para modelar sistemas, mediante el cual son realizados diseños en UML, que puede crear algunos de los diagramas estándares de este lenguaje. Sus principales características son que tiene muy buenas ayudas, soporta todas las especificaciones UML y está integrado con la WEB.

Características: (24)

- ✓ Las ayudas de ArgoUML abren extensiones UML, XML, SVG, OCL y otros de los estándares.
- ✓ ArgoUML es 100% puro de Java. Esto permite que ArgoUML funcione en todas las plataformas para las cuales un puerto confiable de la plataforma Java2 esté disponible.
- ✓ ArgoUML es un proyecto abierto. La disponibilidad del código fuente asegura de que una nueva generación de diseñadores y de investigadores de software tenga un marco probado del cual puedan conducir el desarrollo y la evolución de las tecnologías de la herramienta CASE.

Esta aplicación es multiplataforma y genera código en distintos lenguajes. Los diseños son exportables a XML y pueden ser importados por algunos Frameworks. Por otra parte su instalación es costosa, es poco amigable y difícil de aprender.

1.6.3 Balsamiq Mockups

El Balsamiq es una herramienta para crear prototipos de interfaz gráfica de usuario, está creado en Adobe AIR pudiéndose utilizar tanto en Windows como en Linux, es fácil y amigable para utilizar. Los elementos que brinda para el prototipado se asemejan a dibujos creados a mano alzada. Permite exportar los diseños a PNG, PDF y al portapapeles, así como compartirlos con distintos usuarios que lo utilicen.

1.6.4 Evolus Pencil

Pencil es una aplicación gratuita y de código abierto, para crear diagramas y prototipos de interfaz gráfica de usuario. Es fácil de usar e instalar, consume pocos recursos del sistema y ocupa poco espacio en el

disco duro. Su interfaz es simple y amigable para el usuario, posibilitando el trabajo con este a usuarios inexpertos. Puede ser instalada tanto en LINUX como en Windows.

Entre sus principales funciones se encuentran: construcción de diagramas y prototipos, vinculación entre páginas, edición de texto en pantalla, exportación a PNG, HTML, a documentos Openoffice.org, Word y PDF, soporta deshacer y rehacer, instalación de plantillas y estilos definidos por el usuario, operaciones de dibujo estándar, importar objetos externos. (25)

1.7 Lenguajes de programación del lado del cliente

1.7.1 Lenguaje Extensible de Marcado de Hipertexto (XHTML)[1]

Es una versión avanzada de Lenguaje de Marcado de Hipertexto (HTML)[2] y basada en Lenguaje eXtensible de Marcado (XML)[3]. La primera versión de XHTML se denomina XHTML 1.0 y se publicó el 26 de Enero de 2000.XHTML 1.0 es una adaptación de HTML 4.01 al lenguaje XML, por lo que mantiene casi todas sus etiquetas y características, pero añade algunas restricciones y elementos propios de XML. Técnicamente, HTML es descendiente directo del Estándar de Lenguaje de Marcado Generalizado (SGML)[4], mientras que XHTML lo es del XML, que a su vez, también es descendiente de SGML. Actualmente, entre HTML 4.01 y XHTML 1.0, la mayoría de los diseñadores escogen XHTML. Definiéndolo de forma sencilla, “HTML es lo que se utiliza para crear todas las páginas web de Internet”. Más concretamente, HTML es el lenguaje con el que se “escriben” la mayoría de las páginas web. El propio W3C define el Lenguaje de Marcado de Hipertexto como “un lenguaje reconocido universalmente y que permite publicar información de forma global”. Desde su creación, el lenguaje HTML ha pasado a ser un lenguaje utilizado exclusivo para crear documentos electrónicos a ser un lenguaje que se utiliza en muchas aplicaciones electrónicas como buscadores, tiendas en línea y banca electrónica. (26)

1.7.2 Hojas de Estilos en Cascadas (CSS)[5]

Es un lenguaje de hojas de estilo creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. Separar la definición de los

1 En inglés: eXtensible *Hypertext Markup Language.*
2 En inglés: *Hypertext Markup Language.*
3 En inglés: *eXtensible Markup Language.*
4 En inglés: ***S****tandard* ***G****eneralized* ***M****arkup* ***L****anguage.*
5 En inglés: Cascading Style Sheets

contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados “documentos semánticos”). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. (27)

1.7.3 JavaScript

Lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java. Legalmente, JavaScript es una marca registrada de la empresa Sun Microsystems. (28)

1.7.4 Ajax

El término es un acrónimo de *Asynchronous* JavaScript + XML, que se puede traducir como “JavaScript asíncrono + XML”. En el artículo publicado por Jesse James Garrett en el 2005 define Ajax como: “Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes”. Las tecnologías que forman Ajax son XHTML y CSS para crear una presentación basada en estándares, *Document Object Model* (DOM) para la interacción y manipulación dinámica de la presentación. XML, y *JavaScript Object Notation* (JSON) para el intercambio y la manipulación de información, *XHR o Extensible Markup Language / Hypertext Transfer Protocol* **(**XMLHttpRequest) para el intercambio asíncrono de la información y JavaScript para unir todas las demás tecnologías. (29)

1.8 Lenguajes de programación del lado del servidor

1.8.1 Java

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

manipulación directa de punteros o memoria. Las aplicaciones Java están típicamente compiladas en un código representado por la unidad básica de almacenamiento de información (bytecode), aunque la compilación en código máquina nativo también es posible. En tiempo de ejecución, este código es normalmente interpretado o compilado a código nativo, aunque también es posible la ejecución directa por hardware por un procesador Java. La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través de la Comunidad de Java (*Java Community Process*), otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre. (13)

1.8.2 Pre-procesador de hipertexto (PHP)

Es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo web y puede ser embebido dentro de código HTML. Es usado principalmente en interpretación del lado del servidor (*server-side scripting*), pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas QT o GTK+. Fue creado originalmente por Rasmus Lerdorf en 1994; sin embargo la implementación principal de PHP es producida ahora por El Grupo PHP (*The PHP Group*). Publicado bajo la licencia PHP, la Fundación de Software Libre considera esta licencia como software libre. Entre las ventajas de este lenguaje se encuentran que es un lenguaje multiplataforma, la capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destacándose su conectividad con MySQL, su capacidad de expandir su potencial utilizando la enorme cantidad de módulos (extensiones), posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda, es libre, por lo que se presenta como una alternativa de fácil acceso para todos. Además, permite las técnicas de programación orientada a objetos, no requiere definición de tipos de variables y tiene manejo de excepciones a partir PHP5. (30)

1.9 Servidores web

1.9.1 Lighttpd

Es un servidor web diseñado para ser rápido, seguro, flexible, y fiel a los estándares. Está optimizado para entornos donde la velocidad es muy importante, y por eso consume menos CPU y memoria RAM que

otros servidores. Por todo lo que ofrece, lighttpd es apropiado para cualquier servidor que tenga problemas de carga. Lighttpd es software libre y se distribuye bajo la Licencia BSD. Funciona en GNU/LINUX y UNIX de forma oficial. Para *Microsoft Windows* actualmente hay una distribución conocida como *Lighttpd for Windows* mantenida por Kevin Worthington. (31)

1.9.2 Apache

Es el servidor más utilizado. Parte de su éxito se debe a que es multiplataforma y a su estructura modular, que permite emplear diversos lenguajes en el lado del servidor (PHP, Python y Perl principalmente), así como incorporar características como la compresión de datos, las conexiones seguras y la utilización de URLs amigables. (31)

1.10 Framework para el desarrollo

Symfony: es un framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. (32)

Symfony se diseñó para que se ajustara a los siguientes requisitos: (32)

- ✓ Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- ✓ Independiente del sistema gestor de bases de datos.
- ✓ Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- ✓ Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- ✓ Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- ✓ Preparado para aplicaciones empresariales. Adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.

1.11 Sistema Gestor de Base de Datos

1.11.1 Firebird

Es un SGBD relacional de código abierto, basado en la versión 6 de Interbase, cuyo código fue liberado por *Borland* en el 2000. Su código fue reescrito de C a C++. Sus principales características son: Es multiplataforma y actualmente puede ejecutarse en los sistemas operativos: Linux, HP-UX, FreeBSD, Mac OS, Solaris y Microsoft Windows. Ejecutable pequeño, con requerimientos de hardware bajos. Arquitectura Cliente/Servidor sobre protocolo TCP/IP. Soporte de transacciones ACID y claves foráneas. Es medianamente escalable. Buena seguridad basada en usuarios (*roles*). Bases de datos de sólo lectura, para aplicaciones que corran desde dispositivos sin capacidad de escritura, como CD-ROM. Existencia de controladores ODBC, OLEDB, JDBC, PHP, Perl, .net, etc. Pleno soporte del estándar SQL-92, tanto de sintaxis como de tipos de datos. Completo lenguaje para la escritura de disparadores y procedimientos almacenados denominado PSQL. (33)

1.11.2 PostgreSQL

Es un SGBD relacional orientado a objetos de software libre, liberado bajo la licencia BSD, consiste básicamente en ver el código, poder redistribuirlo y además modificarlo. Su desarrollo no es manejado por una sola compañía, como otros proyectos Open Source, es dirigido y desarrollado por la comunidad PGDG (PostgreSQL Global Development Group). Está caracterizado por su alta concurrencia, pues mediante su sistema denominado Control de la Concurrencia Multi-Versión (MVCC) permite el acceso a las tablas que están bajo un proceso que se está ejecutando en ellas sin necesidad de bloqueos. Es multiplataforma, está disponible para 34 plataformas en su última versión estable, incluyendo Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), y Windows. También es compatible con el almacenamiento de objetos binarios, incluyendo imágenes, sonidos o vídeos. Soporta gran parte del SQL estándar y muchas modernas funcionalidades como: adaptable a las necesidades de los clientes (ver código, redistribuirlo y modificarlo), soporta distintos tipos de datos (el uso de índice, reglas y vistas). Permite la gestión de varios usuarios, declaración de funciones propias, definición de disparadores. Soporta alta concurrencia que facilita y permite, que mientras un proceso escribe en una tabla otros accedan a la misma tabla sin necesidad de bloqueos. (34)

1.12 Entorno de desarrollo integrado seleccionado

El entorno de desarrollo integrado es una aplicación compuesta por varias herramientas de programación. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación generalmente integran un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

NetBeans IDE es un entorno de desarrollo, una herramienta para programadores con el objetivo de escribir, compilar y depurar programas. El IDE NetBeans es de código abierto y gratuito para uso tanto comercial como no comercial. El código fuente está disponible para su reutilización. NetBeans IDE brinda soporte para PHP 5.3 así como también para Symfony. Algunas de las posibilidades que hasta ahora brinda el IDE son: (35)

PHP 5.3

- ✓ Soporte para las palabras claves (keywords).
- ✓ Soporte para espacios.
- ✓ Soporte para cierres.

Symfony

- ✓ Ejecución de comando Symfony.
- ✓ Limpiar la caché.
- ✓ Ayuda para encontrar comandos y para conocer los parámetros.
- ✓ Muestra toda la ejecución de cada comando (creación de ficheros, directorios, etc.)
- ✓ Configuración de atajos del teclado.

1.13 Fundamentación de las tecnologías y herramientas a utilizar

Luego de un crítico y valorativo análisis de algunas de las tendencias, tecnologías y herramientas existentes, para el desarrollo de la réplica se ha tenido en cuenta las ventajas y desventajas de su aplicación, para darle solución al problema de investigación, y teniendo en cuenta las licencias de las herramientas, se han determinado las tecnologías y herramientas a utilizar.

El ambiente de replicación a aplicar en el SBDD es el multi-maestro, pues este permite la réplica bidireccional, que garantiza que las transacciones de los datos entre los servidores centrales y locales se puedan efectuar en ambos sentidos. Además, con el objetivo de aumentar la coherencia entre los datos de los servidores y el rendimiento de la aplicación a través de la sincronización, el modelo que se utilizará es el asíncrono.

Como metodología a seguir para el desarrollo es seleccionada RUP, pues es la más completa, flexible, robusta y aplicable a cualquier tipo de proyecto. Es muy utilizada actualmente, existe una vasta documentación y conocimientos sobre su modo de aplicación. En ella se explica de forma clara y organizada, lo que debe hacer cada integrante del proyecto y cuando hacerlo. Por otra parte, las metodologías XP, SCRUM y DSDM, necesitan de la participación activa del cliente durante el proceso de desarrollo.

Para la modelación del módulo se utilizará el programa Visual Paradigm para UML, pues es multiplataforma, soporta el ciclo de vida del desarrollo de software, y la universidad tiene licencia para su uso. Se puntualiza además, que el empleo de la herramienta ArgoUML, es difícil su aprendizaje e instalación, y no soporta todos los diagramas de UML, y el Rational Rose, aunque soporta la metodología RUP completamente, no es multiplataforma, por tanto, no puede ser ejecutado en el sistema operativo LINUX. Para el diseño de los prototipos de interfaz de usuario se utilizará el Evolus Pencil pues, aparte de ser multiplaforma, es libre y fácil de usar.

Para el desarrollo del módulo de Réplica se utilizará el lenguaje PHP, ya que su eje central está enfocado a la web. El tiempo de desarrollo en PHP es bajo, y aún más con el apoyo de frameworks como Symfony, ya que la mayoría de las funcionalidades comunes ya están implementadas (conectividad con BD, comercio electrónico, redes sociales, correo), solo hay que adaptarlas al proyecto y como la mayoría son software libre están mejoradas y optimizadas. Para el desarrollo de la réplica de la plataforma se utilizará SymmetricDS pues soporta el ambiente de replicación maestro-maestro asíncrono, presenta gran robustez en cuanto al número de nodos que soporta, es una herramienta libre, multiplataforma y funciona con un gran número de gestores de BD incluyendo PostgreSQL. Para el desarrollo de la solución se utilizará PostgreSQL por ser un SGBD rápido, ocupa poco espacio, su licencia BSD lo certifica libre con posibilidad de modificar y distribuir, además de poseer mayor facilidad en el proceso de réplica de datos.

Conclusiones Parciales

En este capítulo:

- ✓ Se investigó sobre las distintas tendencias para los Procesos de Réplica y Sincronización de datos entre servidores.
- ✓ Se seleccionó el entorno de replicación multi-maestro, con el modelo asíncrono, para garantizar la correcta sincronización entre los servidores de base de datos de la plataforma para la gestión del aprendizaje ZERA, del proyecto Alfaomega.
- ✓ Se realizó un análisis y selección de las distintas herramientas y metodologías actuales a emplear, en el desarrollo de la propuesta de solución del problema a resolver.
- ✓ Se seleccionó RUP como metodología, Visual Paradigm para el modelado y el Evolus Pencil para el diseño de los prototipos de interfaz de usuario.
- ✓ Se utilizará para el desarrollo, el lenguaje PHP, el framework Symfony, como herramienta de réplica SymmetricDS, como sistema gestor de base de dato PostgreSQL.
- ✓ Se propone desarrollar un módulo para la gestión de réplicas, en la plataforma para la gestión del aprendizaje ZERA.

Capítulo II: Características del Sistema

Introducción

En el presente capítulo se describen las características del módulo propuesto para dar solución al problema de investigación. Se describen los procesos que generan la situación problemática y los procesos que se automatizarán. Se especifica el modelo de dominio, las características del sistema planteadas en requisitos funcionales, no funcionales y descripción de los casos de uso.

2.1 Descripción de la propuesta de solución

En el Capítulo I, luego del estudio y análisis realizado se concluyó que como herramienta de replicación se utilizará el SymmetricDS, la cual se configura a través de archivos de forma manual, por lo que se propone desarrollar un módulo para la gestión de Réplica en la plataforma ZERA, que permita la configuración de la herramienta de réplica, por usuarios conocedores del negocio de réplicas, y no necesariamente tengan que tener conocimientos avanzados sobre la aplicación e informática.

2.1.1 Funcionamiento del SymmetricDS

Primeramente se debe identificar los nodos implicados, que no son más que los servidores de BD de la plataforma, donde se ejecutarán instancias del SymmetricDS. En el negocio de réplica, se cuenta con varios nodos, de ellos, uno central, en el cual se almacenarán los datos de cada una de las escuelas. Es importante organizar los nodos pues constituyen un factor primordial para la sincronización de los datos, clasificándolos según la información que manejarán y de esta manera definir los grupos de nodos, los cuales son los encargados de encaminar los datos.

Aunque los datos que se deben sincronizar son prácticamente los mismos, cada institución maneja particularidades diferentes, la ruta para garantizar la sincronización de los datos es configurada mediante el grupo de nodo.

Los enlaces entre nodos no son más que la vinculación del nodo origen y destino, especificando las acciones a realizar. Las mismas pueden ser de entrada de datos o salidas.

Los canales constituyen el medio por el cual viaja la información. En este se definen las prioridades de sincronización de los datos y pueden ser habilitados o deshabilitados.

Los disparadores se utilizan para registrar y capturar los cambios. Cada disparador se define teniendo en cuenta una tabla, el canal de transmisión, además puede especificar cambios definidos como: actualizar, eliminar o simples inserciones. Se define tantos disparadores como tablas a replicar.

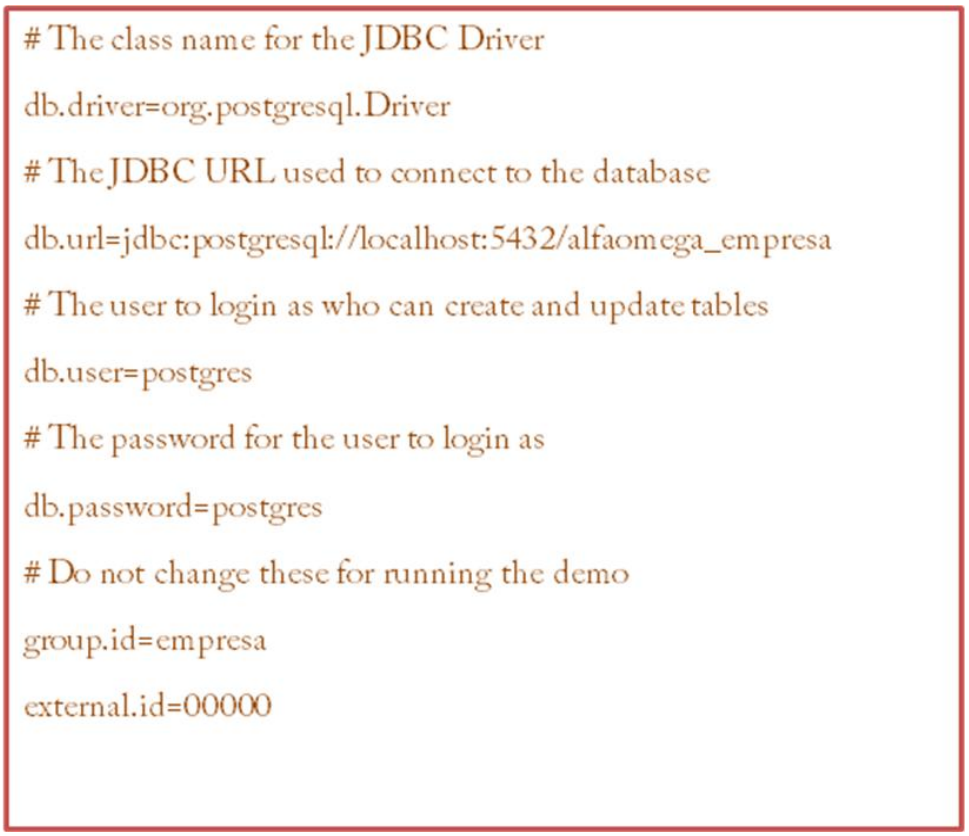
El funcionamiento elemental del SymetricDS se explica a continuación:

En primer lugar, se deben modificar los archivos:

- samples/***root.propeties*** que no es más que su nodo raíz. Solo se configura en el servidor central. Ver fig.1
- samples/***client.propeties*** identifica a cada uno de los nodos asociados. Este archivo se modifica en cada una de las instituciones asociadas. Ver fig.2

En cada uno de ellos se modifican estos parámetros, suelen ser los más importantes para la configuración de un nodo.

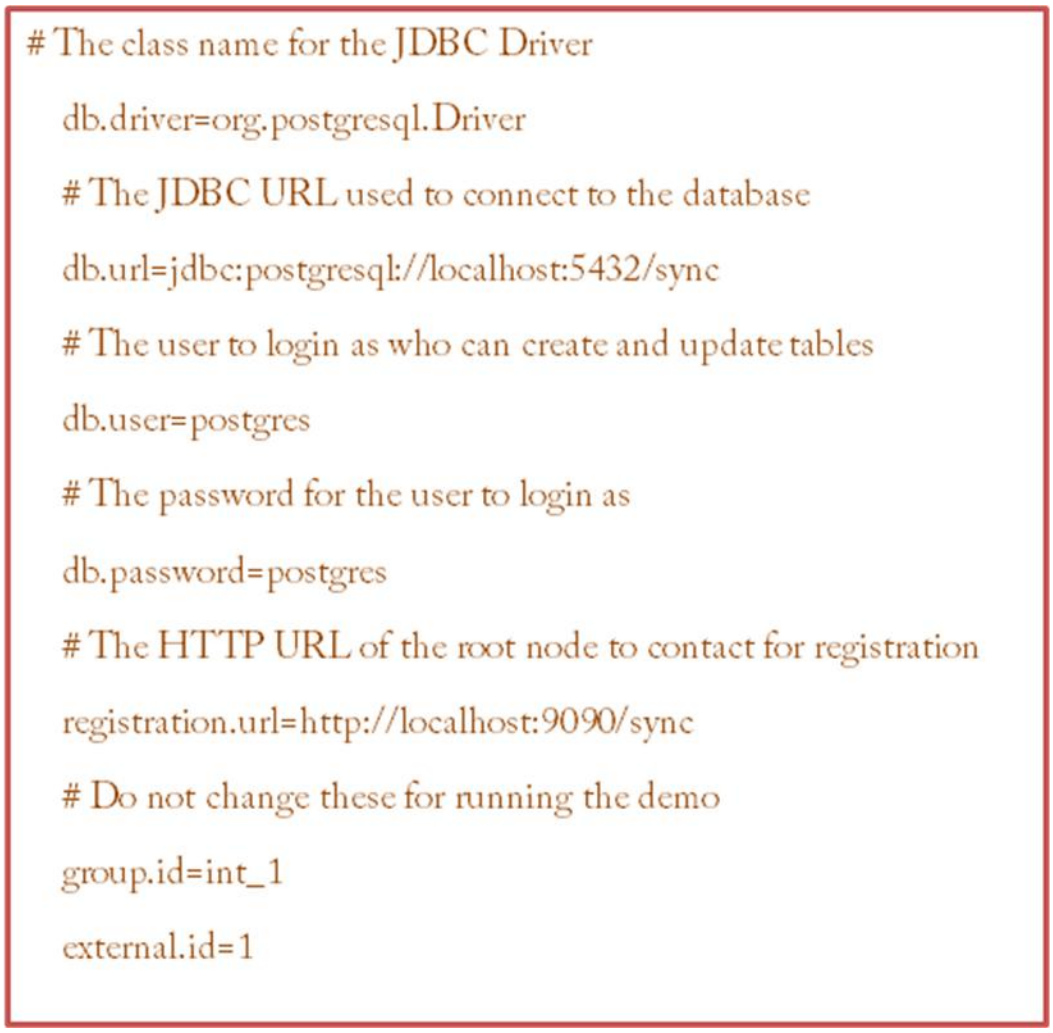
- **db.driver:** representa el nombre de la clase del controlador JDBC en este caso para PostgreSQL.
- **db.url:** sería la dirección URL JDBC que usa para conectarse a la base de datos. **db.user** usuario de la base de datos con privilegios para acceder, crear y actualizar las tablas de SymmetricDS.
- **db.paswword:** contraseña para el usuario de la base de datos.
- **group.id:** representa identificador del grupo al que pertenece el nodo. Para lograr la sincronización se especifica entre grupos de nodos.
- **external.id:** no es más que el identificador externo de un nodo, es muy utilizado en las expresiones condicionales y un subconjunto de datos para la sincronización. Puede coincidir en varios nodos siempre y cuando compartan los mismos intereses.



```
# The class name for the JDBC Driver
db.driver=org.postgresql.Driver
# The JDBC URL used to connect to the database
db.url=jdbc:postgresql://localhost:5432/alfaomega_empresa
# The user to login as who can create and update tables
db.user=postgres
# The password for the user to login as
db.password=postgres
# Do not change these for running the demo
group.id=empresa
external.id=00000
```

Figura 1 Ejemplo del archivo *.properties* del nodo.

En el caso de los nodos asociados, debe especificar en el parámetro **registration.url** la dirección del nodo central al que se puede contactar.



```
# The class name for the JDBC Driver
db.driver=org.postgresql.Driver
# The JDBC URL used to connect to the database
db.url=jdbc:postgresql://localhost:5432/sync
# The user to login as who can create and update tables
db.user=postgres
# The password for the user to login as
db.password=postgres
# The HTTP URL of the root node to contact for registration
registration.url=http://localhost:9090/sync
# Do not change these for running the demo
group.id=int_1
external.id=1
```

Figura 2 Ejemplo del archivo client.properties.

El SymetricDS tiene varias tablas de configuración, en las que son definidos los pormenores. Inicialmente están vacías y la configuración se puede agregar en cada una de las tablas correspondientes para esta operación, también se configura a través del archivo insert_sample.sql.

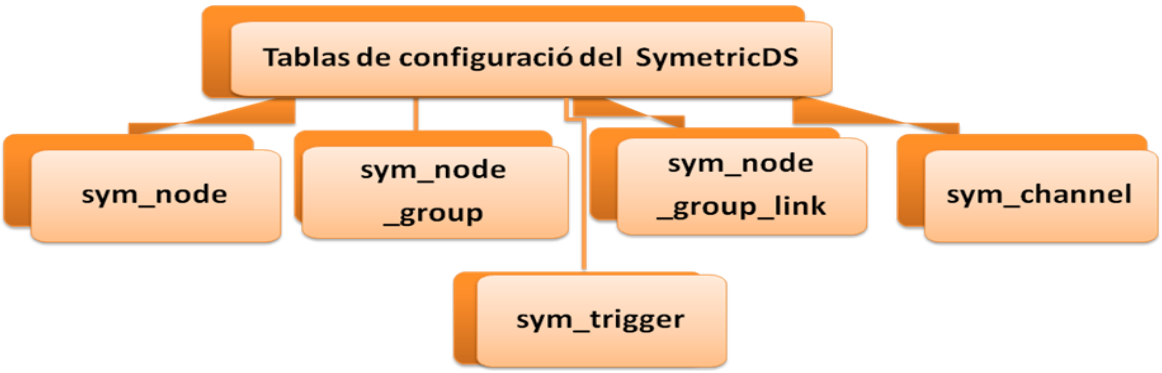


Figura 3 Principales tablas de configuración del SymmetricDS.

En la tabla ***sym_node*** se registran uno a uno los nodos, especificando los siguientes campos de la tabla:

- **node_id:** identificador único del nodo, es de tipo varchar.
- **node_group_id:** identificador del grupo al que pertenece, es de tipo varchar. **external_id:** un identificador de dominio específico para el contexto en el sistema local, es de tipo varchar.
- **sync_enabled:** es el parámetro que precisa si el nodo estará habilitado (0) o deshabilitado (1) para la sincronización, es de tipo entero.
- El ***external_id*** y ***node_group_id*** se definen en los archivos **.properties**.

Los grupos de nodos son sencillos de configurar y se definen en la tabla **sym_node_group**.

- **node_group_id:** identificador único para un grupo de nodos.
- **description:** descripción de este grupo de nodos. Ambos parámetros son de tipo *varchar*.

Mediante el enlace, un grupo de nodos origen envían sus actualizaciones de datos a un grupo de nodos destino. Se configuran en la tabla ***sym_node_group_link,*** agregando los siguientes campos:

- **source_node_group_id:** grupo de nodos en los cuales los cambios serán capturados.

- **target_node_group_id**: grupo de nodos en cual los cambios de los datos serán enviados.
- **data_event_action:** esquema de notificación usado para enviar los cambios en los datos al grupo de nodos destinos.

Los canales de transmisión se definen en la tabla ***sym_channel***, la misma se puede controlar, el máximo de información que será procesada. Estos permiten además, la sincronización de SymmetricDS, ya que pueden encontrarse habilitados o deshabilitados.

- **channel_id:** identificador único del canal.
- **processing_order:** prioridad para procesar los datos del canal.
- **max_batch_size:** número máximo de datos a procesar por este canal.
- **enabled:** indica si el canal está habilitado o no para la transmisión.
- **description:** descripción del canal.

Los disparadores de SymmetricDS se definen en la tabla ***sym_trigger,*** llenando los siguientes campos:

- **source_table_name:** nombre de la tabla de origen que tendrá un disparador para observar los cambios de los datos.
- **source_node_group_id:** identificador del grupo de nodos, donde se instalarán los disparadores para registrar los cambios y luego replicarlos.
- **target_node_group_id:** identificador del grupo de nodos, a los que serán enviados los datos.
- **sync_on_insert:** se dispara al insertarse algún dato de la tabla, es de tipo entero con valores de 0 ó 1.
- **sync_on_update:** se dispara al actualizarse algún dato de la tabla, es de tipo entero con valores de 0 ó 1.
- **sync_on_delete:** se dispara al eliminarse algún dato de la tabla, es de tipo entero con valores de 0 ó 1.
- **initial_load_order:** orden de secuencia de esta tabla, cuando un inicial la carga

se envía a un nodo.

- **last_updated_by:** especifica el usuario que actualizó la última vez esta entrada.
- **channel_id:** identificador del canal por el cual fluyen los cambios de los datos.
- **last_update_time:** establece la hora en la que un usuario actualizó la última entrada.
- **create_time:** establece la hora cuando fue creada la entrada.
- **sync_on_insert_condition:** se especifica la condición, bajo la cual se activara el disparador al realizar alguna inserción.
- **sync_on_update_condition:** se especifica la condición, bajo la cual se activara el disparador al realizar alguna actualización.
- **sync_on_delete_condition:** se especifica la condición, bajo la cual se activara el disparador al eliminar. Estos tres últimos parámetros, se utilizan para definir disparadores que se activaran al coincidir ciertas condiciones.

Para que la herramienta de réplica funcione correctamente en PostgreSQL, debe permitir la conexión entre los servidores mediante la configuración de los archivos:

pg_hba.conf

```
#TYPE DATABASE USER CIDR-ADDRESS
METHOD
# IPv4 local connections:
host all all 127.0.0.1/32 md5
host all all 10.34.0.0/16 md5
host all all 10.0.0.0/8 md5
# IPv6 local connections:
#host all all ::1/128 md5
```

Figura 4 Ejemplo de configuración del archivo pg_hba.conf.

postgresql.conf

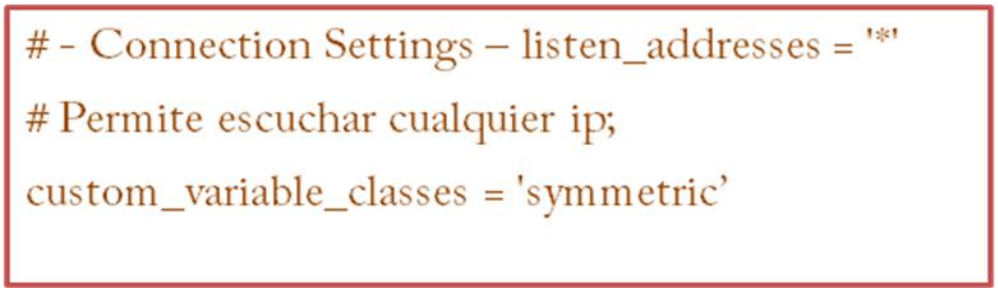


Figura 5 Ejemplo de configuración del archivo *postgresql.conf.*

Luego de realizado estos cambios debe reiniciar el postgres, con la siguiente instrucción.

/etc/init.d/postgresql-8.4 restart

Se debe crear el lenguaje plpgsql sobre la base de datos, se puede realizar corriendo un script en la base de datos con las siguientes instrucciones.

CREATE FUNCTION plpgsql_call_handler() RETURNS language_handler AS '$libdir/plpgsql' LANGUAGE C; CREATE FUNCTION

En el proceso de ejecución del comando open-registration el cual se ejecuta desde el servidor central en el momento de activar la réplica para un servidor local se inicializa una cadena de inserción de datos en las tablas del SymmetricDS *sym_node* y *sym_security*, registrando así el identificador del nodo activado y datos característicos del mismo. Este proceso garantiza que de no estar autorizado un nodo cliente en el servidor central se verá imposibilitado de establecer conexión.

Ejemplo:

../bin/sym -p root.properties --open-registration "store,1"

2.2 Modelo de Dominio

A partir de la anterior descripción de la propuesta de solución al problema, se comprueba que el negocio estudiado posee un bajo nivel de estructuración, por tanto, se concibe enfocar los procesos de réplica y sincronización de datos a través de un modelo de dominio. La realización de este modelo posibilita la comprensión de los conceptos utilizados por los usuarios, con los que trabajan y los que deberá trabajar la aplicación. Permite separar el código tanto como sea posible del resto del desarrollo de la plataforma, reduciendo las probabilidades de cambios en otros módulos y subsistemas que

provoquen que este se interrumpa, reduciendo el tiempo de tardanza en la estabilización del sistema. El modelo puede ser tomado como punto de partida para el diseño del sistema, pues el mapa de conceptos, constituye una versión inicial de la aplicación, que tiene como función fundamental: ayudar a comprender el negocio.

2.2.1 Análisis de las clases conceptuales del Dominio

Tabla 1 Descripción de los conceptos del dominio.

Clases Conceptuales	Descripción
nodo_Central	Representa al servidor central de la plataforma ZERA o la institución central.
nodo_Escuela	Representa a un servidor local de la plataforma ZERA o una escuela asociada a la institución central.
Sistema	Es la asociación del gestor de base de datos PostgreSQL con la información gestionada en la plataforma ZERA.
Administrador	Representa a las personas con las facultades para realizar operaciones de administración de la plataforma en la plataforma ZERA. Este puede ser local o central.

Estas clases conceptuales se relacionan entre ellas y dan paso al siguiente dominio:

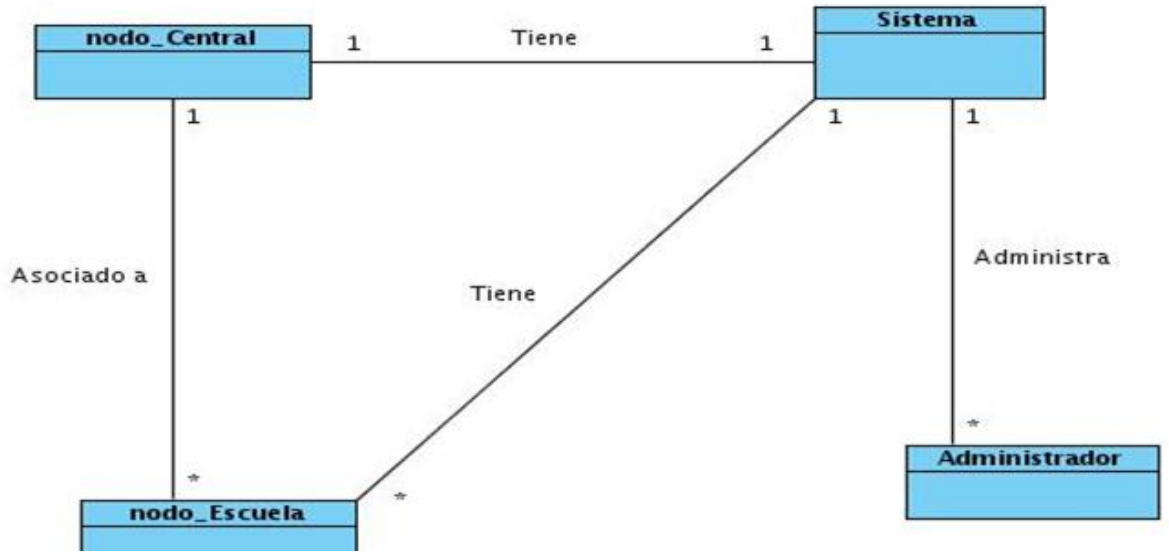


Figura 6 Diagrama del Modelo de Dominio

2.3 Especificación de los requisitos de software

La especificación de requisitos del software es la descripción de cómo debe comportarse el sistema a desarrollar. Según Roger S. Pressman para que un esfuerzo de desarrollo de software tenga éxito, es esencial comprender perfectamente los requisitos del software.

El profesor Ian Sommerville define que: "*Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema.*"

El módulo de Réplica de la plataforma ZERA funcionalmente debe cumplir con una serie de características, que tienen como objetivo darle solución al problema. Así mismo se tendrá en cuenta cualidades no funcionales que debe tener el sistema.

2.3.1 Requisitos funcionales

Jacobson escribe que los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Son el punto de partida para identificar qué debe hacer el sistema. Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. (36)

El módulo Réplica de la plataforma ZERA debe permitir:

RF-1 Configurar el ambiente inicial para la replicación.

1.1 Configurar el ambiente inicial para el servidor central.

1.2 Configurar el ambiente inicial para las escuelas.

1.3 Generar archivos de configuración.

RF-2 Activar la réplica: inicializar los procesos de replicación.

2.1 Activar la replicación en el servidor central.

2.2 Activar la replicación en el servidor local.

RF-3 Revisar el estado de la réplica.

3.1 Obtener datos del estado de replicación, a través de filtrado por las fechas de inicio y fin de las sincronizaciones, así como por escuelas.

3.2 Mostrar el estado de replicación de los datos entre las fechas seleccionadas y para la escuela seleccionada.

3.3 Generar un reporte sobre el estado.

RF-4 Mostrar estado de configuración de las escuelas: listar de las escuelas creadas cuáles fueron configuradas para la replicación y cuáles no.

4.1 Mostrar escuelas configuradas.

4.2 Mostrar escuelas sin configurar.

RF-5 Mostrar últimas escuelas sincronizadas: listar los identificadores de los últimos nodos que han sincronizado los datos entre las bases de datos locales y la central.

RF-6 Mostrar información general: listar la cantidad de escuelas configuradas y sin configurar, además de la cantidad de sincronizaciones realizadas.

2.3.2 Requisitos no funcionales

Son varios los autores que abordan el tema. De estos se coincide, para esta investigación con Jacobson en el que plantea que los requisitos no funcionales especifican propiedades del sistema, como restricciones del entorno o de la implementación, rendimiento, dependencias de la plataforma, facilidad de mantenimiento, extensibilidad y fiabilidad. (36)

El módulo Réplica de la plataforma ZERA debe cumplir con las siguientes especificaciones:

- ✓ **Usabilidad:** el módulo Réplica debe ser usado solamente por los usuarios que tengan permisos para configurar la réplica y verificar el estado en que se encuentra la misma entre los servidores de la plataforma ZERA.
- ✓ **Interfaz externa:** debe ser sencilla y amigable permitiéndole a los usuarios utilizar fácilmente el sistema. La interfaz principal debe dar una pequeña introducción a la configuración de Réplicas en general, y cada una de las adyacentes debe ir orientando al usuario de que trata específicamente.
- ✓ **Portabilidad:** la aplicación debe ser soportada en Windows y en Linux.
- ✓ **Seguridad:** el módulo podrá ser utilizado por administradores o personas autorizadas.
- ✓ **Restricciones en el Diseño y la Implementación:**
  - Las herramientas para el modelado deben ser Visual Paradigm, el Evolus Pencil para el diseño de los prototipos de interfaz de usuario y UML como lenguaje de modelado.
  - Se debe usar para el desarrollo, el lenguaje PHP, el framework Symfony, como herramienta de réplica SymmetricDS y SGBD PostgreSQL.
- ✓ **Confiabilidad:** debe tener un alto grado de confiabilidad, dada la importancia de la sincronización de la información.

2.4 Definición y descripción de los casos de uso del sistema

2.4.1 Definición de los actores

Tabla 2 Descripción de los actores

Actor	Descripción

Administrador Local	Realiza las tareas de administración en la plataforma instalada en su escuela y puede ver el espacio de la escuela en la plataforma instalada en el servidor central. Es quien en la escuela va a revisar el estado en que está la réplica de sus datos.
Administrador Central	Se encarga de la administración global del sitio y tiene permiso sobre todos los componentes de la plataforma. Es el que va a configurar la réplica y todas las otras funciones referentes a la misma.
Director	Mediante el rol Usuario, consulta los reportes de estado de réplica y registro de últimas sincronizaciones.
Profesor	Mediante el rol Usuario, consulta los reportes de estado de réplica y registro de últimas sincronizaciones.

2.4.2 Definición de los casos de uso

Tabla 3 Definición de los casos de uso.

Caso de Uso	Nombre
CU-1	Configurar el ambiente inicial de réplica.
CU-2	Mostrar estado de réplica.
CU-3	Mostrar últimas sincronizaciones.
CU-4	Mostrar escuelas sin configurar.
CU-5	Mostrar escuelas configuradas.

CU-6	Mostrar Información General.
CU-7	Configurar Réplica Local.

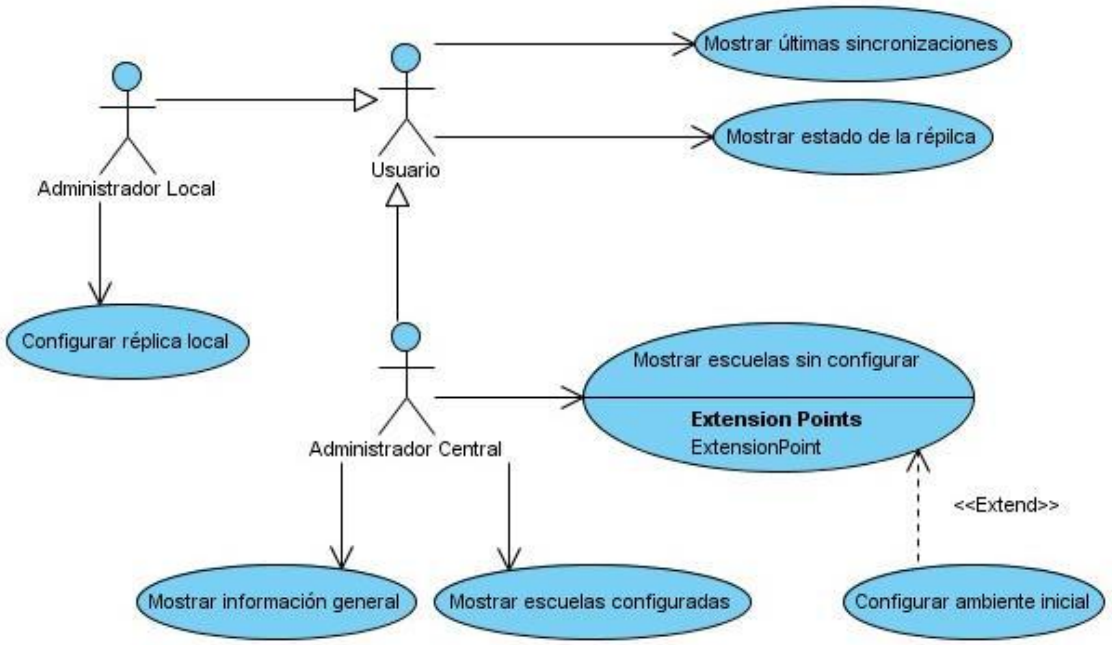


Figura 7 Descripción de Casos de Uso del Sistema.

Tabla 4 CU Configurar el Ambiente Inicial de Réplica.

Objetivo	Configurar el ambiente inicial para la realización de la réplica.
Actores	Administrador central.
Resumen	El caso de uso se inicia cuando el Administrador central selecciona la opción *Configure* que le permite configurar el ambiente inicial para la réplica de la base de datos. El actor del sistema selecciona la opción *Save* y el sistema genera el archivo de configuración inicial y el caso de uso termina.

Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. El servidor central debe estar configurado. Para configurar la réplica de la base de datos, el actor debe ser el responsable temporal de la acción.
Poscondiciones	Se generó un fichero con la configuración del ambiente inicial.

Flujo de eventos.	
Flujo básico	
Acciones del actor	**Respuesta del sistema**
1. El caso de uso se inicia cuando el actor selecciona la opción “Configure”.	2. El sistema muestra los campos necesarios a llenar para la correcta configuración inicial. • User. • Password. • Database name. • Port.
3. El actor introduce los datos. • User. • Password. • Database name. • Port.	4. El sistema permite: • Save. • Cancel.
5. El actor selecciona la opción “Save”.	6. El sistema verifica que los campos no estén vacíos, que el puerto introducido sean números enteros entre 0/65535 y

	que el formato del Ip del servidor sea correcto. A partir de estos datos genera un fichero de configuración inicial del servidor.
	7. El sistema registra la escuela cliente en el servidor central y le asigna a este un sistema de autenticación para el acceso desde este al servidor central.
	8. Se registra la escuela como configurada y se elimina del listado escuelas sin configurar.
	9. El caso de uso termina
Flujo alterno	
5.a El actor selecciona la opción “Cancel”	5.a.1 El sistema elimina los datos entrados.
	5.a.2 Regresa a la vista anterior.
	5.a.3 El caso de uso termina.
Flujo alterno	
6. a Existen datos incorrectos.	6.a.1 Muestra un mensaje de color rojo indicando los campos vacíos o con datos incorrectamente escritos.
	6.a.2 Regresa al paso 2.
Referencia	RF-1, 1.1, 1.2, 1.3, RF-2, 2.1

Tabla 5 CU Mostrar Escuelas sin Configurar.

Objetivo	Listar los nombres de las escuelas que no están configuradas para replicar.
Actores	Administrador central
Resumen	El caso de uso inicia cuando el actor entra al módulo Réplica y en el *Dashboard* en la página principal de este, selecciona la opción *Unconfigured Schools*. El sistema muestra un listado con los nombres de las escuelas que no han sido configuradas para la replicación.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. El servidor central debe estar configurado.
Poscondiciones	Lista los nombres de las escuelas que no están configuradas para replicar.
Flujo de eventos.	
Flujo básico	
Acciones del actor	**Respuesta del sistema**
1. El actor selecciona la opción para entrar al módulo Réplica.	2. El sistema muestra en el *Dashboard* debajo de la opción *Unconfigured Schools* una lista con los nombres de las 7 primeras escuelas sin configurar y permite: ✓ *Show all.*
3. El actor selecciona la opción "*Unconfigured Schools*" o	4. El sistema muestra una lista con los nombres de las escuelas que no están

"*Show all*" del menú en el *Dashboard* del módulo Réplica.	configuradas para la réplica. Y permite: ✓ *Configure.*
5. El actor puede o no seleccionar la opción Configure. En caso de que la seleccione: Ver caso de uso "Configurar ambiente Inicial".	6. El caso de uso termina.
Referencia	RF-4, 4.2

Tabla 6 CU Mostrar Escuelas Configuradas.

Objetivo	Listar las escuelas que están configuradas para la replicación.
Actores	Administrador central.
Resumen	El caso de uso inicia cuando el actor entra al módulo Réplica y en el *Dashboard* en la página principal de este, selecciona la opción *Configured Schools for Replication*. El sistema muestra un listado con los nombres de las escuelas configuradas. El caso de uso termina.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. El servidor central debe estar configurado.
Poscondiciones	Lista los nombres de las escuelas que están configuradas para replicar.
Flujo de eventos.	

Objetivo	Listar las escuelas que están configuradas para la replicación.
Flujo básico	
Acciones del actor	**Respuesta del sistema**
1. El actor selecciona la opción para entrar al módulo Réplica.	2. El sistema muestra en el *Dashboard* debajo de la opción *Configured Schools for Replica* una lista con los nombres de las 7 últimas escuelas configuradas para replicar y permite: ✓ *Show all*.
3. El actor selecciona la opción "*Configured Schools for Replica*" o "Show all" del menú en el *Dashboard* del módulo Réplica.	4. El sistema muestra una lista con los nombres de las escuelas configuradas para la réplica. Y permite: ✓ *Stop*.
5. El actor puede o no seleccionar la opción Stop.	6. En caso que la seleccione el sistema detiene la réplica para esa escuela. Y permite: ✓ *Activate*.
	7. El caso de uso termina.
Referencia	RF-4, 4.1

Tabla 7 CU Configurar Réplica Local.

Objetivo	Configurar la réplica para la escuela local.
Actores	Administrador Local
Resumen	El actor selecciona la opción *Configure Replica* en el *Dashboard* del módulo Réplica, el sistema le permite activar la replicación y el caso de uso termina.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. Para configurar la réplica de la base de datos, el actor debe ser el responsable temporal de la acción. El servidor debe estar configurado.
Poscondiciones	Se activó la replicación de datos para la escuela local.
Flujo de eventos.	
Flujo básico	
Acciones del actor	**Respuesta del sistema**
1. El actor selecciona la opción *Configure Replica* en el *Dashboard* del módulo Réplica	2. Si existe el fichero de configuración inicial en la dirección del SymmetricDS permite: ✓ *Activate*.
3. El actor selecciona la opción: ✓ *Activate*.	4. El sistema crea un fichero para facilitar el comando para inicializar el puerto para la réplica.
	5. El sistema muestra un mensaje informativo.

Objetivo	Configurar la réplica para la escuela local.
	6. El caso de uso termina.
Referencia	RF-2, 2.2

Tabla 8 CU Mostrar Últimas Sincronizaciones.

Objetivo	Listar últimas sincronizaciones.
Actores	Usuario(Administrador local, profesor, director).
Resumen	El caso de uso inicia cuando el actor entra al módulo Réplica y en el *Dashboard* en la página principal de este, selecciona la opción *Last synchronizations.* El sistema muestra un listado con las sincronizaciones efectuadas en el día hasta el momento en que seleccione la opción. El caso de uso termina.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. El servidor local debe estar configurado y la configuración estar completa. La replicación debe estar activada.
Poscondiciones	Lista de las últimas sincronizaciones realizadas.
Flujo de eventos.	
Flujo básico	
Acciones del actor	**Respuesta del sistema**
1. El actor selecciona la opción para entrar al módulo Réplica.	2. Si el actor es administrador local ver sección “Mostrar sincronizaciones

Objetivo	Listar últimas sincronizaciones.
	locales".
	3. Si el actor es administrador central el sistema muestra en el *Dashboard* debajo de la opción *Last synchronization* una lista con los nombres de las 7 últimas escuelas sincronizadas y permite: ✓ *Show all.*
4. El actor selecciona la opción "*Last synchronizations*" o "*Show all*" del menú en el *Dashboard* del módulo Réplica.	5. El sistema muestra una lista con los nombres de las escuelas que se han sincronizado, la fecha y la hora de la última sincronización ejecutada.
	6. El caso de uso termina.
Sección "Mostrar sincronizaciones locales"	
	1. El sistema muestra en el *Dashboard* debajo de la opción *Last synchronization* y permite: ✓ Show all.
2. El actor selecciona la opción "*Last synchronizations*" o "*Show all*" del menú en el *Dashboard* del módulo Réplica.	3. El sistema muestra una lista con el día, fecha y hora de las sincronizaciones realizadas de forma paginadas.
	4. El caso de uso termina.
Referencia	RF-5

Tabla 9 CU Mostrar Información General.

Objetivo	Mostrar la cantidad de escuelas configuradas, sin configurar y sincronizaciones.
Actores	Administrador Central.
Resumen	El caso de uso inicia cuando el actor entra al módulo Réplica y en el *Dashboard* el sistema muestra un listado con la cantidad de escuelas configuradas, sin configurar y sincronizaciones. El caso de uso termina.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado.
Poscondiciones	Mostrar la cantidad de escuelas configuradas, sin configurar y sincronizaciones.
Flujo de eventos.	
Flujo básico	
Acciones del actor	**Respuesta del sistema**
1. El actor selecciona la opción de entrar al módulo Réplica.	2. El sistema cuenta la cantidad de escuelas configuradas, las que no, y las sincronizaciones realizadas. Muestra un listado con cada una de estas cantidades y le permite al usuario: • *Configured*. (Ver CU Mostrar escuelas configuradas) • *Unconfigured*. (Ver CU Mostrar escuelas sin configurar) • *Syncrhonization*. (Ver CU

Objetivo	Mostrar la cantidad de escuelas configuradas, sin configurar y sincronizaciones.
	Mostrar últimas escuelas sincronizadas)
	3. El caso de uso termina.
Referencia	RF-6

Tabla 10 CU Mostrar Estado de Réplica.

Objetivo	Revisar el estado en que se encuentra la replicación a partir de una fecha de inicio y una de fin
Actores	Administrador central.
Resumen	El caso de uso inicia cuando el actor selecciona la opción "*Display Replica status*" del menú Réplica en la interfaz de administración. El sistema le permite filtrar por fechas las réplicas que el actor desea revisar y luego le muestra un reporte de replicación. Además, permite que el usuario guarde esta información en un archivo. El caso de uso termina.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. El servidor local debe estar configurado y la configuración estar completa. La replicación debe estar activada. Para ver el estado de la réplica, los elementos asociados deben estar seleccionados previamente.
Poscondiciones	El actor revisó el estado de la replicación.

	El actor guardó un fichero con los estados revisados.
Flujo de eventos.	
Flujo básico	
Acciones del actor	**Respuesta del sistema**
1. El actor selecciona la opción "*Display Replica status*" del menú Réplica interfaz de administración.	2. El sistema muestra dos calendarios, para que el actor seleccione el intervalo de tiempo del que quiere el reporte de replicación y un campo para que seleccione el nombre de la escuela: ✓ *Start Date.* ✓ *End date.* ✓ *Name school.*
3. El actor selecciona la fecha de inicio, la de fin, y el nombre de la escuela para filtrar el reporte de replicación.	4. El sistema comprueba si el intervalo de tiempo es válido. Si la fecha de inicio es menor o igual a la de fin, y si el campo del nombre de la escuela no está vacío Y permite: ✓ *Display status.*
5. El actor selecciona la opción: ✓ Display status.	6. El sistema busca y muestra la información de las réplicas ejecutadas para la escuela seleccionada en el intervalo de fechas seleccionado por el actor. Lista día, hora, fecha, nombre de la escuela, estado de cada réplica, clasificándolas en completada o fallida, y muestra una breve descripción de los errores producidos.

	Y permite: ✓ *Generate report.* ✓ *Cancel.*
7. El actor selecciona la opción: ✓ *Generate report.*	8. El sistema genera y guarda un archivo con la información del estado de la réplica mostrada.
	9. El caso de uso termina.
Flujo alterno	
4. a Intervalo de tiempo invalido.	4. a.1 El sistema comprueba si el intervalo de tiempo es válido. Si la fecha de inicio es mayor que la de fin y el campo del nombre de la escuela está vacío. Muestra un mensaje de error de color rojo sobre los calendarios.
	4. a.2 Regresa al paso 3.
Flujo alterno	
4.b El actor selecciona la opción *Cancel*.	4.b.1 El sistema borra los datos.
	4.b.2 Regresa al paso *DashBoard*.
Referencia	RF-3, 3.1, 3.2, 3.3.

Conclusiones Parciales

En este capítulo se describió la propuesta de solución al problema planteado, se abordaron los principales conceptos del dominio. Se especificaron los requisitos funcionales y no funcionales del sistema. Fueron definidas y descritas las principales funcionalidades del sistema. Teniendo en cuenta los resultados de este capítulo, se puede iniciar la elaboración de la propuesta de solución dada.

Capítulo III: Análisis y Diseño del sistema

Introducción

En el presente capítulo se expone lo concerniente al análisis y el diseño del sistema propuesto, enmarcándose en algunas de las actividades propuestas por la metodología RUP para el flujo de trabajo Análisis y Diseño, que tiene como objetivo traducir los requisitos a una especificación que describa cómo implementar el sistema. Se describen las clases del análisis y las de diseño, así como sus respectivos diagramas.

3.1 Análisis del sistema

Durante el análisis, son examinados los requerimientos descritos, los cuales son refinados y estructurados. Con el objetivo de conseguir una mejor comprensión y más precisa de estos, ayudando a que el sistema sea estructurado. Se realiza un análisis en profundidad de los requisitos utilizando lenguaje de los desarrolladores para describirlos. Consiste en obtener una visión del sistema con el objetivo de mostrar Qué hace, enmarcándose únicamente en los requerimientos funcionales.

Según Pressman, el propósito del análisis es definir las clases que son relevantes al problema a resolver, las operaciones y atributos asociados, las relaciones y comportamientos asociadas con ellas.

Este le facilita a los desarrolladores a comprender la estructura del sistema, como debe ser diseñado e implementado, además de servir como primera aproximación del diseño. Proporciona la estructura de la vista interna del sistema, estructurado por clases estereotipadas.

3.1.1 Clases del análisis.

Las clases del análisis se centran en el tratamiento de los requerimientos funcionales, estas representan conceptos y relaciones del dominio. Estas se conforman por atributos y las relaciones existentes entre ellas. Las mismas se clasifican en:

- ✓ **Interfaz:** estas clases modelan la interacción entre el sistema y sus actores. Las mismas son una representación de abstracciones de ventanas, paneles, formularios e interfaces de comunicación entre estos. Modelan las partes del sistema que depende de los actores.
- ✓ **Entidad:** modelan información que conservan sus datos y que a menudo es persistente. Generalmente son derivadas de alguna clase del negocio y suelen mostrar una estructura de datos lógica y contribuye a comprender de qué información depende el sistema.

- ✓ **Control:** representan coordinaciones, transacciones, y el control de los objetos, frecuentemente se utiliza para encapsular el control de un caso de uso en específico. Modelan los aspectos dinámicos de un sistema, pues manejan y coordinan las acciones y flujos de control principales, delegando trabajo a otros objetos.

Estas siempre se ajustan en uno de los estereotipos básicos, donde cada uno de estos involucra una semántica específica. Estos están estandarizados en el lenguaje UML, utilizados para ayudar a los desarrolladores a distinguir el ámbito de las diferentes clases.

3.1.2 Diagramas de Clases del Análisis.

En el diagrama de clases del análisis son representados los conceptos fundamentales del dominio del problema, representando las definiciones y relaciones entre las clases.

A continuación se muestra un Diagrama de Clases de Análisis, asociado al problema. Restantes diagramas en Anexo 1.

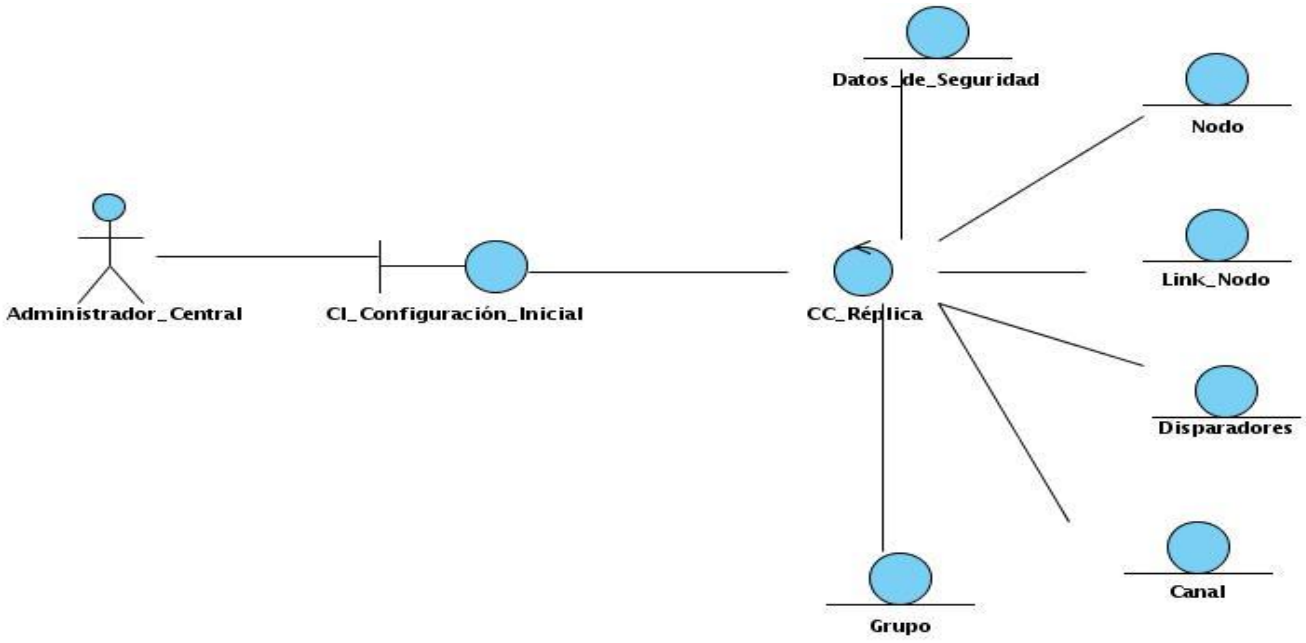


Figura 8 DCA[6] Configurar Ambiente Inicial

Diagramas de colaboración

[6] Diagrama de clases del análisis.

Un diagrama de colaboración muestra la interacción entre varios objetos y los enlaces que existen entre ellos, creando enlaces y añadiendo mensajes entre sí. El Diagrama de Colaboración se centra en estudiar todos los efectos de un objeto dado durante un escenario.

A continuación se muestran un diagrama de colaboración, asociado al problema. Restantes diagramas en Anexo 2.

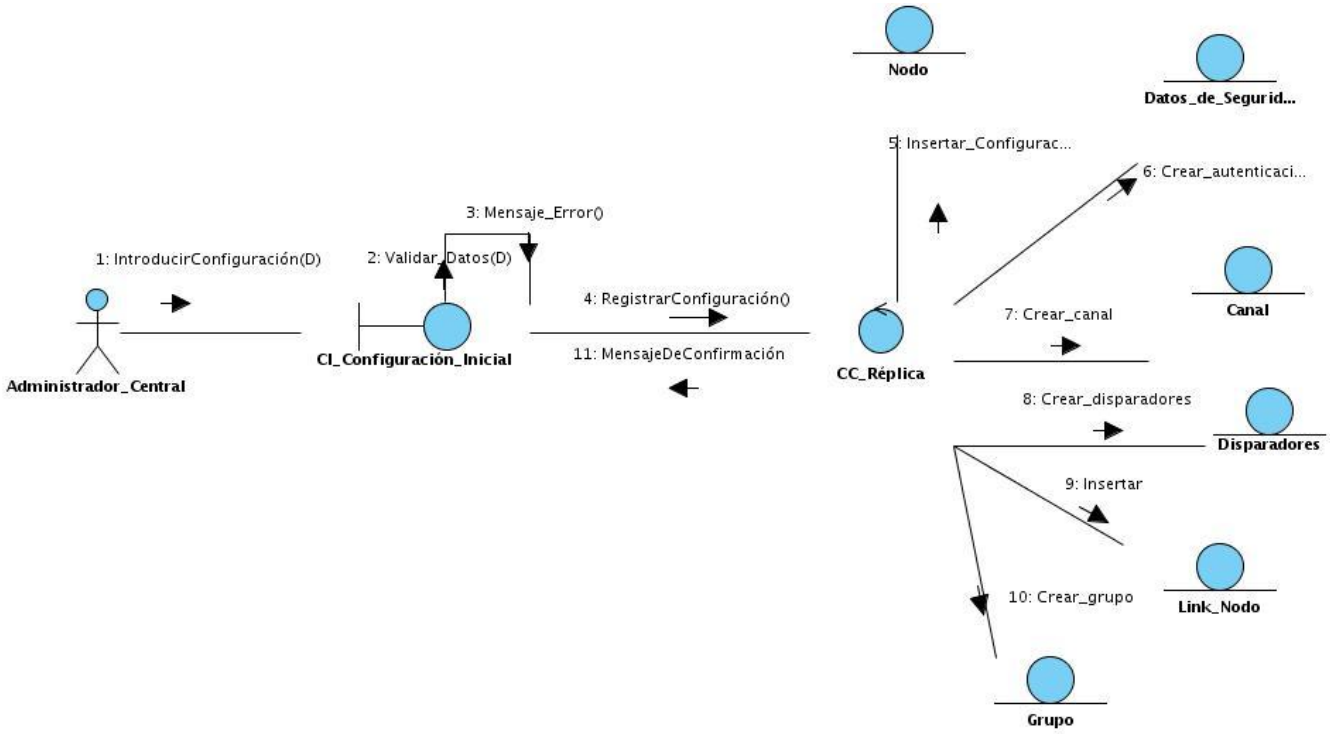


Figura 9 DC[7] Configurar Ambiente Inicial

3.2 Diseño Web

Durante el diseño se modela el sistema para que soporte los requerimientos que le suponen. El diseño consiste en el refinamiento de los Modelos de Análisis para crear especificaciones adicionales que enriquecen el Modelo de Análisis con detalles próximos a la implementación. Tiene como objetivos fundamentales crear una entrada apropiada y un punto de partida para las actividades de implementación

[7] Diagrama de colaboración.

y descomponer los trabajos de implementación en partes más manejables.

Las aplicaciones web tienen un estilo arquitectónico particular, por tanto, a la hora de diseñarlas, que es cuando se modela como debe ser construido el sistema internamente, deben tener en cuenta métodos diferentes para el desarrollo e implementación de estas.

3.2.1 Diagrama de Clases del Diseño

En el Diagrama de Clases del Diseño se exponen las clases que intervienen en la realización de los casos de uso y sus relaciones. Describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Son una representación más concreta y detallada que los diagramas de clases del análisis, aunque también representan la parte estática del sistema conteniendo las clases y sus relaciones.

A continuación se muestran un diagrama de clases del diseño, asociado al problema. Restantes diagramas en Anexo 3.

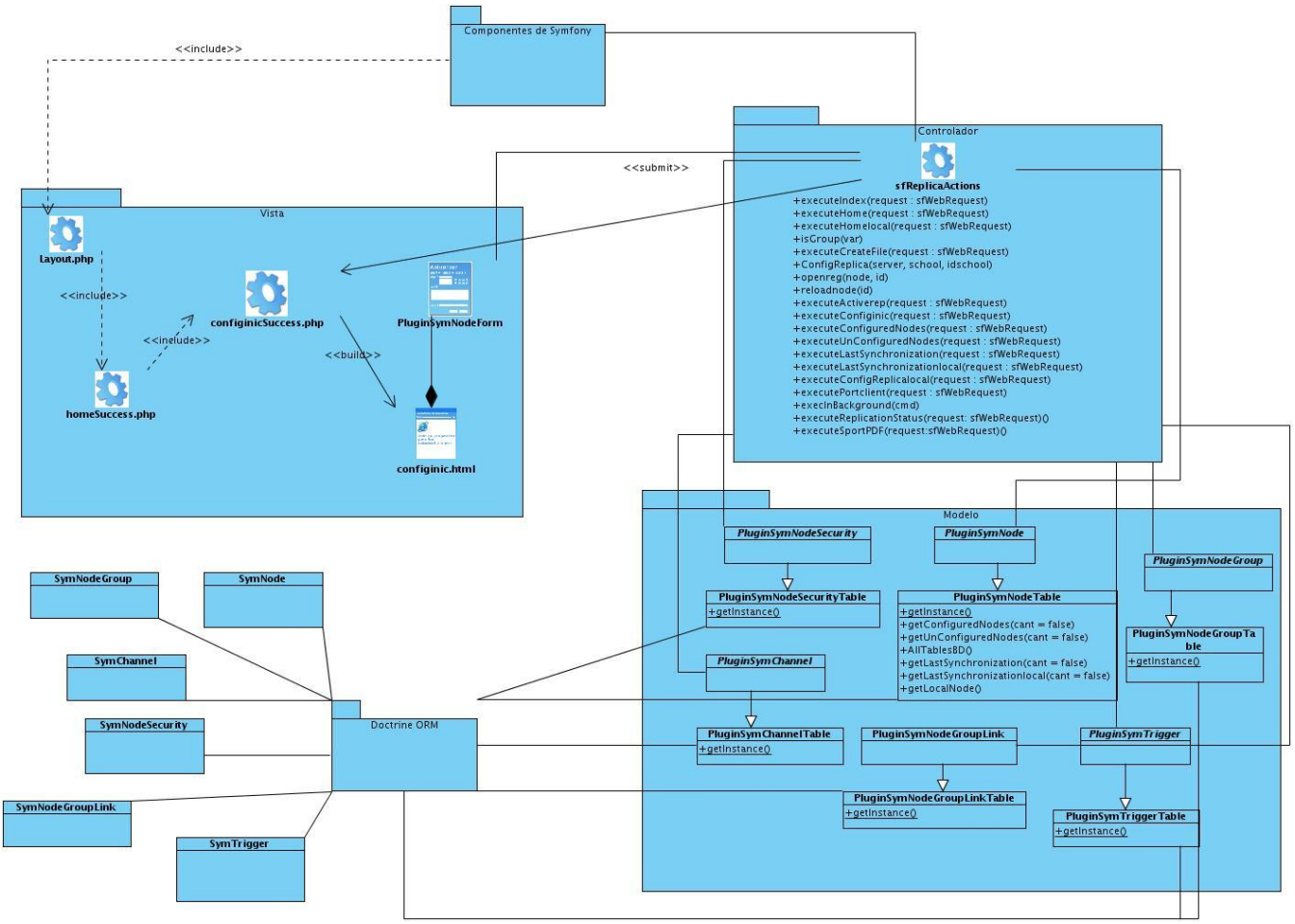


Figura 10 DD[8] Configurar Ambiente Inicial

8 Diagrama de clases del diseño.

Conclusiones parciales

En este capítulo se han modelado las funcionalidades que debe tener el sistema, mediante modelos que facilitan su entendimiento, brindando una visión de la solución para el desarrollo. Fueron definidas las clases del análisis y del diseño, así como las interacciones entre ellas, mostradas a través de los diagramas clases y análisis.

Capítulo IV: Validación de la solución propuesta.

Introducción

La calidad de un producto de software; conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia; se ha convertido en un elemento estratégico de las grandes organizaciones debido a su fuerte impacto en la competitividad de las empresas.

Durante el proceso de desarrollo de software las posibilidades de errores son múltiples, estas pueden aparecer desde la misma especificación de requisitos donde se define lo que el sistema debe hacer. Como elementos críticos aparecen entonces la prueba y la validación de los resultados, estas en lugar de efectuarse una vez desarrollado el software, se llevan a cabo en cada una de las etapas de desarrollo para detectar a tiempo las imperfecciones e irregularidades y proporcionar una visión objetiva de la madurez y calidad de los procesos asociados.

A continuación se evalúa el grado con que se le dio cumplimiento a las necesidades del cliente o usuario, para ello se definen inicialmente los tipos y métodos de prueba, y posteriormente se aplica a un requisito y procedimiento específico.

4.1 Pruebas de Software:

Al conjunto de técnicas experimentales para la búsqueda de fallos en los programas, que determinan en cierto grado la calidad de un producto, se les denomina pruebas de software.

La competencia mundial en el ámbito informático exige productos de calidad, de esta forma se muestra la necesidad de contar con pruebas de este tipo desde las primeras etapas de concepción de un proyecto.

La fase de pruebas es una de las más valiosas del ciclo de vida de un software y en ese sentido, deben evaluarse todos los artefactos generados, lo que incluye especificaciones de requisitos, diagramas de diversos tipos, el código fuente y el resto de productos que forman parte de la aplicación, ejemplo: la base de datos.

- **Niveles de Prueba:**

A la hora de evaluar dinámicamente un sistema se debe comenzar por los componentes más simples y pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto. Las pruebas se aplican en distintos niveles de trabajo, dentro de estos se distinguen:

Pruebas de Unidad:

Prueba individual a las unidades separadas de un sistema de software.

Pruebas de Integración:

Los componentes individuales son combinados con otros componentes para asegurar que la comunicación, enlaces y los datos compartidos ocurran apropiadamente.

Pruebas del Sistema:

Son usualmente conducidas para asegurar que todos los módulos trabajan como sistema sin error. Es similar a la prueba de integración pero con un alcance mucho más amplio.

Pruebas de Aceptación:

Son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales.

- **Métodos de Prueba:**

Enfoque sistemático, independiente del nivel en que se enmarque la prueba, que ayuda a encontrar buenos conjuntos de casos de prueba para detectar diferentes tipos de errores. (45)

Dos enfoques alternativos:

Caja Negra: Se comprueban las funcionalidades sin tener en cuenta la estructura interna.

Caja Blanca: Se comprueban los componentes internos.

4.1.1 Objetivo:

Conceptos como estabilidad, escalabilidad, eficiencia y seguridad se relacionan a la calidad de un producto bien desarrollado. El aspecto fundamental que rige esta etapa de pruebas es determinar cómo y en qué sentido el componente cumple con las expectativas del cliente, a partir de los requisitos establecidos y las restricciones impuestas. En ese ámbito se trazan un conjunto de objetivos dentro de los que se sitúan:

- Verificar la implementación del componente.
- Verificar la integración adecuada del componente.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar los errores y asegurar que estos sean corregidos de la mejor manera.

Con la idea de diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo. (45)

4.1.2 Alcance:

Las pruebas de software deben abarcar diversas ramas; desde la funcionalidad de los primeros prototipos, la estabilidad, cobertura y rendimiento de la arquitectura, hasta el producto final.

Cuando se ejecuta una prueba, se tienen en cuenta qué tanto los resultados obtenidos se asemejan a los resultados esperados. Si la salida no es la esperada, indica la ocurrencia de un error y en ese caso es preciso corregirlo, esto implica todo un proceso de depuración de errores a través de los casos de prueba.

Se debe tener en cuenta el costo de tiempo y esfuerzo que traen aparejado los errores, generalmente se presentan en situaciones complejas y en ocasiones las soluciones no las puede determinar el propio desarrollador.

4.2 Prueba de Software que será aplicada al módulo:

4.2.1 Descripción general de las pruebas para el nivel de Unidad:

Se le denomina Prueba de Unidad al proceso de separar y probar el correcto funcionamiento de las partes individuales de un programa para asegurar que cada uno de estos se ejecuta correctamente por separado.

Ventajas de usar este tipo de prueba:

1. Los errores son más fáciles de localizar.
2. Los errores están más acotados.
3. Se fomenta el cambio.
4. Se reducen los "efectos secundarios".
5. Se da más seguridad al programador.

"Antes de iniciar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del módulo. Si los datos no fluyen correctamente, todas las demás pruebas no tienen sentido." (45)

El equipo de desarrollo a partir de los conocimientos adquiridos durante la implementación del módulo se encargará de aplicar las pruebas para este nivel mediante el método de Caja negra.

4.2.1.2 Descripción y aplicación de la Prueba de Caja Negra o Funcional:

- **Descripción:**

A este tipo de prueba también se le conoce como Prueba de Caja Opaca o Inducida por los Datos.

Se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se ajusta a su especificación, esta prueba se limita a brindar solo datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo internamente, es decir, solo trabaja sobre su interfaz externa.

En esencia permite encontrar:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a las bases de datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y terminación.

Dentro de la prueba de caja negra se incluyen las Técnicas de Pruebas que serán descritas a continuación:

Partición de Equivalencia: Divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

Análisis de Valores Límites: Prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Grafos de Causa-Efecto: Permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

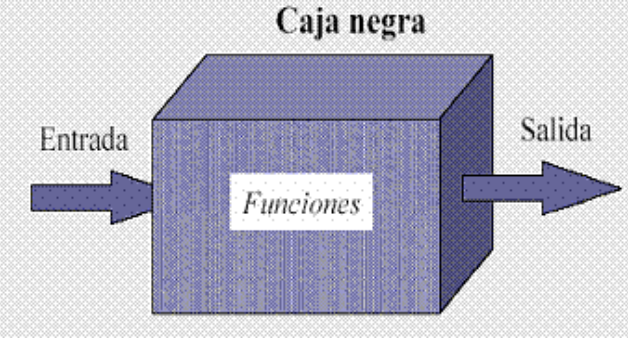


Figura 11 Caja negra

- **Aplicación:**

A continuación se aplica la prueba de partición de equivalencia como parte de la realización de la prueba de caja negra sobre la interfaz que responde al Caso de uso Configurar el ambiente inicial

La Partición de Equivalencia divide el dominio de entrada de un programa en un número finito de variables de equivalencia. Se definen dos tipos de variables de equivalencia, las válidas, que representan entradas válidas al programa, y las no válidas, que representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato.

ID del escenario	Escenario	Usuario de la BD	Contraseña	Nombre de la BD	Puerto	Respuesta del sistema	Flujo central
EC 1	Realizar configuración inicial correctamente	V(postgres)	V(postg8.)	V(Zer8)	V(5800)	Se configura la escuela y se elimina de la lista de Escuelas sin configurar.	Escritorio de trabajo/Replica/Uncofigured School
EC 2	Realizar configuración inicial con datos inválidos	V(Post5)	V(Pshd)	V(Zer8)	I(666666)	El sistema muestra un mensaje de error: Wrong IpServer or Port	Escritorio de trabajo/Replica/Uncofigured School/Configure
EC 3	Realizar configuración inicial dejando campos vacíos.	V(Ho87$)	I(Vacio)	V(H7$)	V(Ho87$)	El sistema muestra un mensaje de error: field not emptys.	Escritorio de trabajo/Replica/Uncofigured School/Configure
		I(Vacío)	V(Ot52)	V(o87$)	V(H7$)		
		V(Ot52)	V(Ot52)	I(Vacío)	V(Ot52)		
		V(Ot52)	V(Ot52)	V(H7$)	I(Vacío)		

Figura 12 SC Configurar ambiente inicial

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	*Usuario.*	Campo de texto	No	*Valores alfanuméricos y caracteres especiales. Ejemplo: pepeA65&*
2	*Contraseña.*	Campo de texto	No	*Valores alfanuméricos y caracteres especiales. Ejemplo: H&n2*
3	*Nombre de la BD.*	Campo de texto	No	*Valores alfanuméricos y caracteres especiales. Ejemplo: pepRa)*
4	*Puerto*	Campo de texto	No	*Valores numéricos, un número entero entre 0/65535. Ejemplo: 6*

Figura 13 Descripción de variable.

ID del escenario	Escenario	Respuesta del sistema	Flujo central
EC 1	Realizar configuración inicial correctamente.	Se muestra una interfaz con los siguientes datos: Usuario Contraseña Nombre de la Base de Datos Puerto	Se llenan correctamente los datos que se muestran en la interfaz. Se presiona el botón Guardar. Se elimina la escuela de la lista de Escuelas sin configurar.
EC2	Realizar configuración inicial con datos inválidos	Se muestra una interfaz con los siguientes datos: Usuario Contraseña Nombre de la Base de Datos Puerto	Se llenan incorrectamente los datos que se muestran en la interfaz. Se presiona el botón Guardar. El sistema muestra un mensaje de error.
EC3	Realizar configuración inicial dejando campos vacíos.	Se muestra una interfaz con los siguientes datos: Usuario Contraseña Nombre de la Base de Datos Puerto	Se dejan campos vacíos. Se presiona el botón Guardar. El sistema muestra un mensaje de error.
EC4	Cancelar	Se muestra una interfaz con los siguientes datos: Usuario Contraseña Nombre de la Base de Datos Puerto	Se presiona el botón Cancelar. Se vuelve a la interfaz Escuelas sin configurar.

Figura 14 SC Configurar ambiente inicial.

Conclusiones parciales

En el presente capítulo ha quedado especificada la estructura física del sistema, así como la forma en que puede intercambiar la información con los usuarios, además se han definido los diseños de casos de prueba para garantizar la correcta funcionalidad del sistema.

Conclusiones generales

Una vez concluida la investigación se arriba a las siguientes conclusiones:

- El estudio y análisis del estado del arte permitió seleccionar las herramientas, metodologías y tecnologías para el desarrollo de la investigación. Al respecto la bibliografía consultada permitió realizar, como valor agregado de la investigación un catálogo de referencias bibliográficas sobre el tema.
- RUP como metodología seleccionada justificó el análisis, diseño, implementación y prueba, así como la culminación en tiempo a partir del cronograma planificado para cumplir el objetivo de la investigación.
- La obtención del módulo Réplica demostró que la selección de las herramientas seleccionadas fue válida para su desarrollo.
- La validación realizada demostró el cumplimiento del objetivo propuesto con resultados favorables en los procesos de réplica y sincronización.

Recomendaciones

A partir de los resultados y beneficios tangibles e intangibles obtenidos en la investigación que se presenta el autor de la presente tesis propone las siguientes recomendaciones:

- A la dirección del proyecto, realizar al módulo Réplica la validación de los restantes niveles del ciclo de prueba, para detectar posibles errores en el desarrollo y ejecución del mismo.
- A la comunidad científica, realizar el despliegue del módulo Réplica en instituciones docentes, políticas y sociales que posean para el desarrollo de sus actividades sistemas de gestión de aprendizaje.
- A la dirección del proyecto realizar estudios que permitan la integración del módulo Réplica a otros sistemas de gestión de aprendizaje a partir de la reusabilidad de código que se exponen en la presente investigación.
- A la dirección del proyecto, documentar la implementación del módulo Réplica como parte de futuras investigaciones la cual no se tuvo en el alcance del presente estudio.

Bibliografía

1. **Valdés, Damián Pérez.** Pensamiento Imaginativo. [En línea] bligoo, 06 de 10 de 2008. [Citado el: 13 de enero de 2011.] http://manuelgross.bligoo.com/content/view/292173/Una-base-de-datos-es-mejor-que-una-planilla.html.

2. **Julio W. Russi Ed. CEO & Asociados.** Emprendedores UCU. [En línea] Universida Católica. [Citado el: 13 de enero de 2011.] http://www.emprendedoresucu.com/diccionario.htm.

3. **Departamento de O.E.I. - U.P.M.** *Diseño y Optimización de Bases de Datos: Bases de Datos Distribuidas.* 2010.

4. **Colima, Universidad de.** *Sistemas de Bases de Datos Distribuidas.* 2010.

5. **Preston, W. Curtis.** Tecnologia y Administracion de base de datos. [En línea] 2 de marzo de 2009. [Citado el: 15 de enero de 2011.] http://www.searchstorage.es/respaldo-de-datos/buenas-practicas-de-replicacion-de-datos-para-salvaguarda/.

6. **Monge, Profesor Raúl.** *Base de Datos Distribuidas: Replicación.* Valparaíso : s.n., 2005.

7. **MSDN.** MSDN. *MSDN.* [En línea] 2011. [Citado el: 15 de DICIEMBRE de 2010.] ttp://msdn.microsoft.com/es-es/library/ms151198%28v=SQL.100%29.aspx.

8. **CENTALAD.** *Panorámica de la replicación. Estrategia para su tratamiento en el Centro.* Ciudad Habana : UCI, 2010.

9. **FCLD.** GNU/LINUX. *Fundacion de código libre.* [En línea] FCLD, 2008. [Citado el: 14 de enero de 2011.] http://www.codigolibre.org/index.php?option=com_content&view=article&id=5441%3Apreguntas-generales-sobre-postgresql&catid=108%3Apostgresql&Itemid=60.

10. **Fajardo Vega, Norge.** *Sistema de réplica para base de datos distribuidas en PostgreSQL.* Ciudad Habana : UCI, 2007.

11. **EMC.** Where information lives. *Where information lives.* [En línea] [Citado el: 17 de enero de 2011.] http://argentina.emc.com/products/detail/software/mirrorview.htm.

12. —. Where information lives. *Where information lives.* [En línea] [Citado el: 17 de ENERO de 2011.] http://argentina.emc.com/products/detail/software/recoverpoint.htm .

13. **Pupo Polaco, Rosell y Gonzáles Pérez, Yenier.** *Implementación del componente réplica de base de datos para Akademos v2.0.* Ciudad Habana : UCI, 2009.

14. **Gálvez Velázquez, Yadisnel, y otros.** *CHRONOS: SISTEMA DE REPLICACIÓN ASÍNCRONA MULTIMAESTRO PARA POSTGRESQL.* Ciudad Habana : UCI, 2009.

15. **Pgfoundry.** Pyreplica. [En línea] http://pgfoundry.org/projects/pyreplica.

16. **Pgfundry.** Pgcluster. [En línea] http://pgfoundry.org/projects/pgcluster/.

17. **symmetricds.** symmetricds. *symmetricds.* [En línea] [Citado el: 20 de enero de 2011.] http://symmetricds.codehaus.org/.

18. **María A. Mendoza Sanchez.** Informatizate. *Informatizate.* [En línea] 7 de Junio de 2004. [Citado el: 8 de enero de 2011.] file:///G:/milo/Tesis/BIblio%20Usada/metodologias_de_desarrollo_de_software_07062004.html.

19. **RUP.** Proceso de desarrollo de sofware. *Proceso de desarrollo de sofware.* [En línea] [Citado el: 17 de ENERO de 2011.] http://www.innovavirtual.org/campus/mod/resource/view.php?inpopup=true&id=5219.

20. **G. Figueroa, Roberth, J. Solís, Camilo y A. Cabrera, Armando.** *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.* s.l. : Universidad Técnica Particular de Loja; Escuela de Ciencias en Computación.

21. **Romero, Oscar Casasola.** Programación en Castellano. *Programación en Castellano.* [En línea] [Citado el: 17 de ENERO de 2011.] http://www.programacion.com/articulo/introduccion_a_uml_181 .

22. **Libre Software, Descargas y.** Descargas y Software. [En línea] [Citado el: 11 de enero de 2011.] http://www.freedownloadmanager.org/es/downloads/paradigma_visual_uml_libre/.

23. **Software, Descargas y.** Descargas y Software. [En línea] [Citado el: 11 de enero de 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

24. **CACERES, ANA MERCEDES.** *Herramienta case " ArgoUML".* s.l. : UNIVERSIDAD DON BOSCO, 2006.

25. **mozilla Firefox.** Complementos para Firefox. *Complementos para Firefox.* [En línea] mozilla Firefox. [Citado el: 13 de enero de 2011.] https://addons.mozilla.org/es-es/firefox/addon/pencil/.

26. **Pérez, Javier Eguíluz.** *Introducción a XHTML.* 2008.

27. —. *CSS 2.1.* 2008.

28. —. *Introducción a JavaScript.* 2008.

29. —. *Introducción a Ajax.* 2008.

30. **HInostroza, R.R.** linuxcentro. *linuxcentro.* [En línea] 2005. [Citado el: 20 de enero de 2011.] http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP.

31. **lopezpino.** Servidores-web-mas-usados. [En línea] [Citado el: 16 de enero de 2011.] http://lopezpino.es/2010/07/30/servidores-web-mas-usados/).

32. **Fabien Potencier,Francois Zaninotto.** Libros web. *Libros web.* [En línea] http://www.librosweb.es.

33. **firebird.** firebirdsql. *firebirdsql.* [En línea] [Citado el: 18 de enero de 2011.] (http://www.firebirdsql.org/).

34. **postgresql.** postgresql. *postgresql.* [En línea] [Citado el: 17 de enero de 2011.] http://www.postgresql.org/about/press/presskit90.html.es.

35. **González Blanco, Rubén y Pérez Tobalina, Sergio.** ESSI. *ESSI.* [En línea] [Citado el: 1 de enero de 2011.] http://www.essi.upc.edu/~es-e/web/documents/lab/0304Q2/lessons/lese-2/LESE-2%20-%20Introduccion%20a%20Rational%20Rose.ppt.

36. **Jacobson, I y Booch, G.** *El lenguaje Unificado de Modelado.Manual de Referencia.* s.l. : Addison Wesley, 2000.

37. Patrones de diseño. [En línea] http://patronesdediseno.blogspot.com/2009/05/patron-facade.html.

38. **Ismael.** Debug_mode. *La red para los profesionales.* [En línea] 2009. http://es.debugmodeon.com/articulo/el-patron-mvc.

39. **Hista Internacional.** Hista Internacional. [En línea] [Citado el: 26 de noviembre de 2010.] http://www.histaintl.com/servicios/consulting/rup.php.

40. **Moscardó, Juan Morató.** *Dynamic System Development Method .* Valencia : UPV, 2003.

41. **Sierra, María.** *Trabajando con Visual Paradigm for UML.* s.l. : Univ. Cantabria – Fac. de Ciencias.

42. **EMC.** Where information lives. *Where information lives.* [En línea] [Citado el: 17 de ENERO de 2011.] http://argentina.emc.com/products/detail/software/replistor.htm .

43. **Mustain, A. Elein.** Reilly Database. *Reilly Database.* [En línea] 2004. [Citado el: 18 de noviembre de 2010.] http://onlamp.com/pub/a/onlamp/2004/11/18/slony.html.

44. **worldlingo.** worldlingo. *worldlingo.* [En línea] [Citado el: 18 de enero de 2011.] http://www.worldlingo.com/ma/enwiki/es/NetBeans.

45. **Ramírez, Jaime.** Unidad de Programación. Métodos de Prueba del Software. . [En línea]

Glosario de términos

Aplicación: Se refiere al programa que se está ejecutando y a los archivos y bases de datos con los que se trabaja.

Backus: **copia de seguridad** o **copia de respaldo,** se refiere a la copia de datos de tal forma que estas copias adicionales puedan restaurar un sistema después de una perdida de información.

Cluster: En términos de Slony-I, un cluster son dos bases de datos PostgreSQL que intervienen en la réplica.

Configuración: Es un conjunto de datos que determina el valor de algunas variables de un programa o sistema de software, dichas opciones generalmente son cargadas en su inicio y en algunos casos se deberá reiniciar para poder ver los cambios.

Consulta: En términos de base de datos, es un código donde se describe una acción a ejecutar.

Esquema: Un esquema es la definición de una estructura (generalmente relaciones o tablas de una base de datos), es decir, determina la identidad de la relación y qué tipo de información podrá ser almacenada dentro de ella; en otras palabras: son los metadatos de la relación.

Flujo de trabajo (workflow en inglés): Es el estudio de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan, cuál es su orden correlativo, cómo se sincronizan, cómo fluye la información que soporta las tareas y cómo se le hace seguimiento al cumplimiento de las tareas.

GNU: Proyecto con el objetivo de crear un sistema operativo completamente libre: el sistema GNU. Se establecieron algunas motivaciones para realizar el proyecto GNU, entre las que destaca "volver al espíritu de cooperación que prevaleció en los tiempos iníciales de la comunidad de usuarios de computadoras".

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

IDE: Un **entorno de desarrollo integrado o** en inglés *Integrated Development Environment ('**IDE')* es un programa compuesto por un conjunto de herramientas para un programador.

Identificador: Nombran entidades. El concepto es análogo al de "nombre". Los identificadores se usan en prácticamente todos los sistemas de procesamiento de la información. Nombrar las entidades hace posible referirse a las mismas, lo cual es esencial para cualquier tipo de procesamiento simbólico.

Internet: Es un conjunto descentralizado de redes de comunicación interconectadas, que utilizan la familia de protocolos TCP/IP, garantizando que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial.

IP (**El Protocolo de Internet, en inglés Internet Protocol):** Es un protocolo no orientado a conexión

usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados.

Nodo: En términos de la propuesta para replicar base de datos fragmentadas, son dos bases de datos.

Trigger: Es un evento que se ejecuta cuando se cumple una condición establecida al realizar una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE) en la base de datos.

Tupla: En la teoría de bases de datos, una tupla se define como una función finita que mapea (asocia unívocamente) los nombres con algunos valores.