Facultad 4

“Desarrollo de una herramienta para la replicación de datos de la colección El Navegante”

Trabajo para optar por el título de Ingeniero en Ciencias Informáticas

Autores:

Lecsy Olivera López

Ramiro Manuel Vázquez Villar

Tutores:

Ing. Angel Alberto Vazquez Sánchez

Ing. Rafael Jacobo Hidalgo Urbino

Co-Tutores:

Ing. José Ernesto Lara Rodríguez

Ing. Jorge Martínez García

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmamos la presente a los ___ días del mes de junio del año 2011.

_____
Lecsy Olivera López

Autor

_____
Ramiro Manuel Vázquez Villar

Autor

_____
Rafael Jacobo Hidalgo Urbino

Tutor

_____
Ing. Angel Alberto Vazquez Sanchez

Tutor

Resumen

Las tecnologías de la información y las comunicaciones (TIC) han permitido grandes avances en las diferentes esferas de la vida cotidiana. El proceso enseñanza-aprendizaje se ha nutrido de las posibilidades y los beneficios que brindan las TIC, lo que ha permitido mejorar la calidad en la obtención del conocimiento.

El e-learning o aprendizaje electrónico, es el resultado de la aparición de tecnologías en los procesos de enseñanza, su perfeccionamiento constituye objeto de investigación constante, con el objetivo de mejorar sus funcionalidades y brindar mayor interacción entre profesores, estudiantes y contenido.

Con el presente trabajo se persigue el desarrollo de una herramienta encargada de replicar y controlar los datos generados por estudiantes y profesores, que forme parte de la colección El Navegante desarrollada por profesores y estudiantes de la Universidad de las Ciencias Informáticas. Para llevar a cabo su realización fue necesario un estudio de las soluciones similares existentes tanto en nuestro país como fuera del mismo, así como de las tendencias y tecnologías más usadas en la actualidad. Este análisis arrojó las características fundamentales de 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. Como parte de la investigación se le realizan pruebas a la herramienta desarrollada, lo que la hace más robusta y fiable.

Índice de contenido

Índice de contenido

Índice de tablas

Índice de figuras

Introducción

Las tecnologías de la información y las comunicaciones (TIC) conforman un conjunto de recursos que facilitan la transmisión de la información, el manejo de los servicios, las herramientas, las tecnologías y la educación, ocupando un lugar muy importante en la vida humana contemporánea. El uso de las TIC no para de crecer y de extenderse, particularmente en la esfera de la educación, ofreciendo las herramientas necesarias para el fortalecimiento de los procesos de enseñanza y aprendizaje.

El desarrollo de las TIC abrió nuevas posibilidades individuales e institucionales, propiciando un avance en el sector educacional para una expansión sin precedentes: el aprendizaje electrónico, ampliando el proceso de enseñanza-aprendizaje. Este tipo de aprendizaje permite que los estudiantes puedan continuar con sus estudios apoyándose en alguna TIC.

Cuba como potencia educacional cuenta con varios centros de desarrollo y especialistas capacitados en los avances tecnológicos, lo que ha permitido obtener buenos resultados en este sector. La industria del software en el país es relativamente nueva, por lo que han surgido numerosos proyectos e iniciativas para fortalecer la misma, un ejemplo de ello es la Universidad de las Ciencias Informáticas (UCI).

La UCI es un centro de educación superior que tiene como misión formar profesionales altamente calificados en la rama de la informática que a través del vínculo estudio-trabajo logran crear software y servicios informáticos de alta calidad, por lo que se ha convertido a lo largo de los años en el motor impulsor dentro del ámbito informático en nuestro país contribuyendo a su economía.

El Ministerio de Educación cubano desarrolló una colección llamada El Navegante, la cual había sido implementada con tecnologías propietarias y era una solución escasamente configurable y extensible. Al ser una aplicación de escritorio creada sólo para el sistema operativo Windows, requería de mucho esfuerzo para su despliegue puesto que debía ser instalada y configurada en cada estación de trabajo. Estos problemas trajeron consigo que se le diera la tarea a la UCI de desarrollar una nueva versión de esta colección que tuviera como característica fundamental que fuera multiplataforma y desarrollada con herramientas completamente libres. Esta colección está compuesta por 10 productos y estos a su vez se dividen en 6 módulos, los que se relacionan con las diferentes asignaturas impartidas en las escuelas secundarias. El módulo Resultados contenido en estos productos, permite a los usuarios ver las trazas y realizar análisis sobre los datos generados por los estudiantes en los diferentes productos a partir de su

interacción con los mismos, aunque los productos sean instalados de manera local en entornos móviles o de conectividad limitada. Para mejorar la usabilidad del módulo Resultados, es necesario que cada entorno mantenga conectividad permanente con un servidor encargado de centralizar el registro de los estudiantes, y de esta forma se podrán obtener diferentes reportes generados por el módulo. En muchos lugares se dificulta tener conectividad, cuando esto ocurre las operaciones de los estudiantes se guardan de forma local, lo que trae como consecuencia que cuando se quiera revisar las estadísticas de los estudiantes estas pueden estar incompletas y no existe la posibilidad de recuperarlas.

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

¿Cómo centralizar la información del módulo Resultados que se almacena en los diferentes entornos hacia el servidor central de la escuela?

Objetivo general:

Desarrollar una herramienta que permita la centralización de la información para el módulo Resultados de la colección El Navegante a partir de los datos generados del trabajo sin conexión de los estudiantes de la enseñanza secundaria.

Objeto de Estudio:

Herramientas para la replicación de datos.

Campo de Acción:

Técnicas y herramientas que permiten la sincronización de datos basadas en arquitecturas cliente servidor.

Idea a defender:

Si se desarrolla una herramienta para la replicación de datos se permitirá centralizar la información generada por la interacción de los estudiantes y profesores con los productos de la colección hacia un único servidor.

Objetivos específicos:

- ✓ Analizar las técnicas, herramientas y lenguajes de programación abordados por los autores en el tema sobre replicación de datos.
- ✓ Realizar el análisis y el diseño de la herramienta.
- ✓ Implementar la herramienta.
- ✓ Probar la solución implementada.

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

1. Realización del estudio del estado del arte de herramientas y técnicas que se utilicen para las tareas de replicación de datos.
2. Estudio de la arquitectura de la colección El Navegante y del módulo Resultados.
3. Identificación de las herramientas de replicación de datos y lenguajes de programación que se vinculen con las mismas.
4. Definición de los requerimientos funcionales y no funcionales de la aplicación.
5. Realización del análisis y diseño de la herramienta.
6. Implementación de la herramienta de replicación de datos.
7. Realización de pruebas a la aplicación implementada.

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.

A continuación se explican las características de estructura capitular de la presente investigación:

Capítulo I: Fundamentación teórica

En este capítulo se 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

En este capítulo se describe las características de la herramienta propuesta 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 sistema de réplica.

En este capítulo se agrupa la propuesta de análisis y el diseño del sistema de 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: Implementación y pruebas del sistema de réplica.

En este capítulo se describen aspectos relacionados con la construcción de la solución propuesta a través de los diagramas de despliegue y colaboración. Concentra los aspectos de las pruebas que se le realizan a través de los casos de prueba con lo que se probará la solución.

Capítulo I: Fundamentación teórica

Introducción

El objetivo que se quiere lograr con este capítulo es la selección de las herramientas a utilizar en el desarrollo de la propuesta de una herramienta para la replicación de datos de la colección El Navegante. Se tratan los principales conceptos relacionados con bases de datos, procesos de réplica y sincronización de datos. Se aborda el estado del arte de las herramientas de réplica con el objetivo de obtener un acercamiento a ellas, las que serán sometidas a un proceso de estudio y análisis, para seleccionar la más apropiada a utilizar en la solución.

1.1 Base de datos

Con el aumento de los datos almacenados por las aplicaciones que existen en las empresas a nivel mundial aparece la tendencia a la pérdida y falta de organización de los datos y fallo en las búsquedas relacionadas con los mismos. Una de las soluciones a este problema es el empleo de bases de datos (BD).

Existen diferentes definiciones de este concepto como son:

“Una BD se puede definir como un conjunto de información relacionada que se encuentra agrupada o estructurada.” (1)

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

Partiendo de estos conceptos se puede definir una base de datos como un conjunto de datos persistentes que son utilizados por aplicaciones de alguna empresa para almacenar grandes cantidades de información con el objetivo de facilitar la organización y búsqueda de los datos. El término empresa convenientemente se refiere a una organización. Una empresa pudiera ser un individuo (con una pequeña base de datos), una corporación o similar (con una gran base de datos compartida). (3)

Se dice que los datos en una base de datos "persisten" porque, una vez que han sido aceptados por el sistema de base de datos para entrar en la base de datos, estos pueden ser borrados de la misma solamente por una petición explícita del sistema de base de datos. (3)

Las BD se pueden clasificar según la variabilidad de los datos almacenados como estáticas o dinámicas. Las estáticas son de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones. En las dinámicas la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización, borrado y adición de datos, además de las operaciones fundamentales de consulta.

1.1.1 Sistema de bases de datos distribuidos (SBDD)

Un modelo de bases de datos distribuidas consiste en un conjunto de localidades, cada una de las cuales mantiene un sistema de bases de datos local. Cada localidad puede procesar transacciones locales, o bien transacciones globales entre varias localidades, requiriendo para eso comunicación entre ellas. Cada uno de los ordenadores que integran el sistema es llamado nodo o emplazamiento. La idea de un control centralizado es mucho menos acentuada que en un sistema de bases de datos centralizada. Esto depende también de la arquitectura, es posible identificar una estructura de control jerárquico sobre la base de una administración de base de datos mundial. Existe un administrador central responsable de los datos a nivel global, y un administrador local por cada emplazamiento con un nivel de autonomía local diferente.

En la Figura 1, la cual muestra un sistema de bases de datos distribuidas, se puede observar que cuando un nodo falla no afecta los demás y si los datos se encuentran en varios emplazamientos pueden ser extraídos de otra computadora.

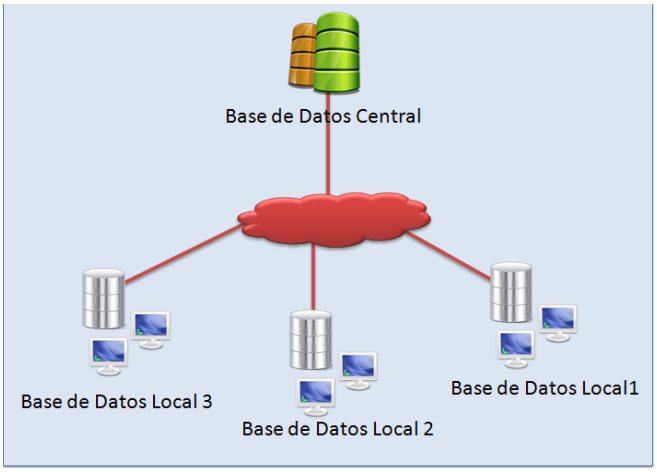

Figura 1: Sistema de bases de datos distribuidas.

El tamaño del sistema puede variar fácilmente, ya que se pueden agregar computadoras adicionales a la red conforme aumentan el número de usuarios. (4)

1.2 Réplica de base de datos

La replicación es el proceso de copiar y de mantener los objetos de la BD en los múltiples nodos que incorporan un sistema de base de datos distribuida. (5)

La réplica de datos consiste en el transporte de datos entre dos o más servidores, permitiendo que ciertos datos de la base de datos estén almacenados en más de un sitio, y así aumentar la disponibilidad de los mismos y mejorar el rendimiento de las consultas globales. (6) (7)

La replicación de datos es el proceso de compartir objetos y datos de una base de datos a múltiples bases de datos, en diferentes localizaciones. (8)

En la Figura 2 se muestra el proceso de intercambio de datos, el cual consiste en capturar la modificación, y transmitirla a todas las instancias de la base de datos, para luego aplicarla en cada una de ellas.

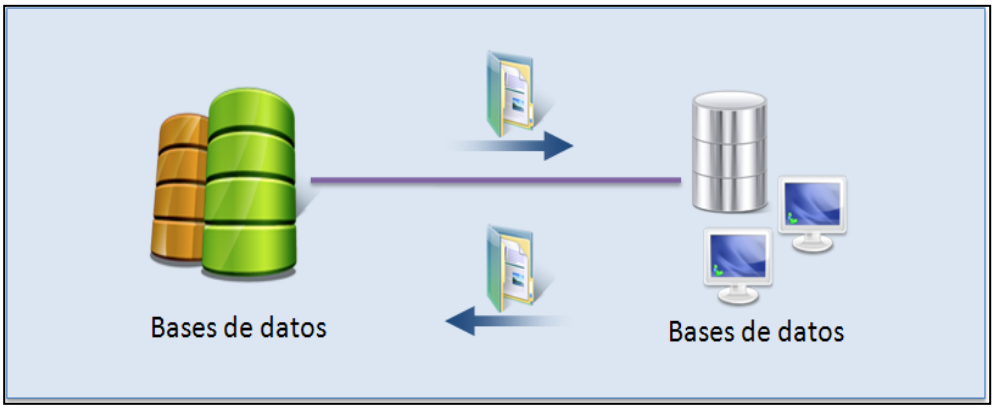

Figura 2: Replicación de datos.

“La réplica es el proceso de crear y mantener versiones duplicadas de objetos de bases de datos, por ejemplo tablas, en un sistema de base de datos distribuida”. (9)

“Técnica que permite copiar y distribuir idénticamente las tablas de una base de datos en múltiples bases de datos ubicadas en diferentes nodos de la red. La replicación asegura que los datos correctos estén siempre disponibles en el momento y en el lugar necesario”. (10)

Al replicar los datos de varias bases de datos locales a una central se evita que al ocurrir un fallo en alguna de las bases de datos locales no se pierde la información almacenada en esta, además existe un lugar donde se puede encontrar los datos centralizados, los cuales pueden ser accedidos desde cualquier lugar y permite realizar acciones que impliquen los datos reunidos.

1.2.1 Entornos de réplica

Existen distintos entornos o ambientes de replicación como son:

- ✓ Maestro-Esclavo (master-slave): Es un entorno de replicación de datos que permite que los datos sean almacenados por un grupo donde un solo miembro del grupo es designado como el "maestro" de una pieza dada de datos y es el único nodo que puede modificar ese elemento de datos, si otros miembros desean modificar el primer elemento de datos deben comunicarse con el nodo maestro, permitiendo lograr la coherencia entre los miembros del grupo. La principal desventaja es que si ocurre un cambio en alguno de los miembros del grupo no se podrá actualizar ningún otro nodo. (11)

La siguiente figura muestra un ejemplo de entorno maestro-esclavo donde se puede apreciar que la réplica se realiza en un solo sentido.

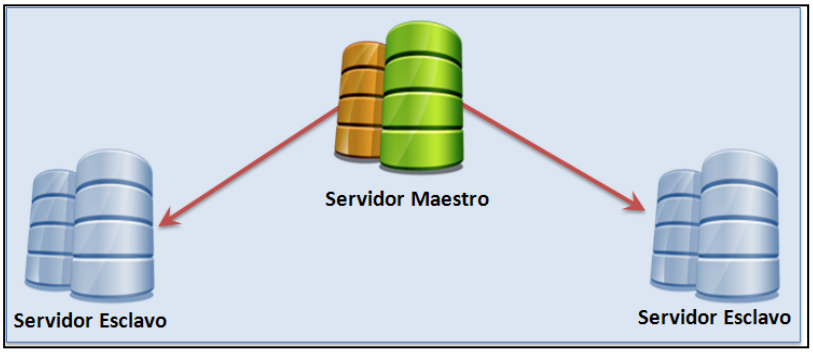

Figura 3: Entorno maestro-esclavo.

- ✓ Maestro-Maestro (multi-master): Consiste en varias bases de datos que interactúan como pares iguales para participar en un modelo de actualización en cualquier lugar. Las actualizaciones realizadas en un sitio maestro individual se propagan a otro sitio maestro participante (12). Según (13) es un entorno de replicación de datos que permite que los datos sean almacenados por un grupo de computadores y actualizado por cualquier miembro del grupo. El sistema de replicación multi-master es responsable de propagar las modificaciones de datos realizadas por cada miembro con el resto del

grupo. Entre las ventajas fundamentales se encuentra que cada miembro del grupo puede estar ubicado en cualquier lugar y permite leer o escribir consultas que serán enviadas a varias computadoras replicadas, aumentando el rendimiento mediante la sincronización de los cambios entre servidores.

La siguiente figura muestra gráficamente como quedaría la réplica donde se utilice el entorno maestro-maestro.

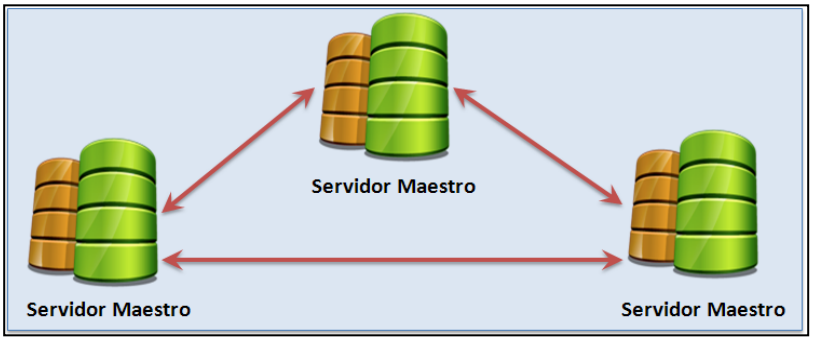

Figura 4: Entorno maestro-maestro.

1.2.2 Modelos de distribución de datos

Los SBDD tienen dos modelos de distribución que se aplican a cada uno de los ambientes de replicación vistos, los cuales se basan en la medida de latencia para llevar a cabo la sincronización de bases de datos. La medida de latencia es la cantidad de tiempo que una réplica puede estar inconsistente hasta llegar a estar consistente con la fuente primaria designada (8). La sincronización es la actualización de las bases de datos en un ambiente de replicación mediante el intercambio de la información actualizada de cada nodo.

Los modelos de distribución son:

- Asincrónica: es llamada de consistencia débil respecto a los datos almacenados, ya que existe una latencia, es decir una desactualización de la copia replicada respecto a la original. La replicación asincrónica permite un alto nivel de autonomía en los sitios ya que un usuario puede trabajar sin estar conectado a la red, estas modificaciones se guardan en una bitácora local y una vez conectados a la red se actualizarán en el siguiente período de replicación. (14)

- Sincrónica: también llamada replicación en tiempo real por su rapidez con que actualiza los cambios, la desactualización es casi nula y el tiempo de latencia casi es cero. La replicación sincrónica genera un alto nivel de sobrecarga en la red y no permite que los usuarios puedan trabajar desconectados. Un sistema que trabaje con este modelo solo soporta clientes tradicionales[1], ya que no permite manejar clientes móviles[2]. (14)

1.3 Análisis de otras soluciones existentes

En Cuba, uno de los lugares donde se han desarrollado soluciones para réplica de datos es en la UCI a partir de la creación del Centro de Tecnologías de Gestión de Datos (DATEC), dentro de esta institución resaltan las siguientes soluciones:

1. **Réplica bidireccional basada en control de cambios:** fue propuesta por el proyecto Identidad de la Facultad 1 de la UCI, se caracteriza por ser bidireccional y se basa en el control de cambios, lo que evita las transmisiones redundantes. Soporta ambientes heterogéneos y no tiene en cuenta el

[1]Clientes tradicionales: Computadores que mantienen conectividad de forma permanente.
[2]Clientes móviles: Computadores que no mantienen conectividad de forma permanente, entre los que se pueden encontrar las laptops.

gestor de base de datos, además permite particionamiento horizontal de los datos y garantiza su integridad, 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. (15)

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. (15)

Las soluciones antes mencionadas no satisfacen todas las necesidades ya que:

- ✓ La solución para la réplica bidireccional basada en el control de cambios del proyecto Identidad a pesar de basarse en el control de cambios, de no tener en cuenta el gestor de base de datos y de no soporta el entorno de replicación maestro-esclavo, tiene como principal desventaja que está implementado bajo la plataforma .NET utilizando un lenguaje de programación propietario.
- ✓ La solución para la réplica de datos del proyecto Prisiones está hecha en lenguaje Java y utiliza protocolos que se encuentran entre los más populares de internet como FTP y HTTP, pero presenta incompatibilidad con el sistema gestor de base de datos PostgreSQL.

1.4 Herramientas de replicación

Luego de realizar un estudio del estado del arte se confirmó la existencia de soluciones que se le han dado a problemas con puntos de contacto con el planteado anteriormente. A continuación se muestran algunos ejemplos existentes a nivel internacional que son privativas:

1.4.1 EMC MirrorView

- Replicación remota basada en arreglos.
- Espejeado de datos independiente del host.

- Síncrono y Asíncrono.
- Administración centralizada. (16)

1.4.2 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. (17)

EMC MirrorView y RecoverPoint a pesar de permitir la replicación asincrónica y proteger los datos, están implementadas sobre software propietario, esto no cumple con las necesidades de la colección de utilizarse sobre software libre, por lo que es necesario buscar herramientas que no sean propietarias. Entre las herramientas libres se encontraron:

1.4.2 Pyreplica

Pyreplica es una herramienta desarrollada en el lenguaje Python, permite réplicas maestro-esclavo y multi-maestro limitado, funciona de forma asincrónica y especialmente para el gestor PostgreSQL. Se caracteriza por ser fácil de usar, no se necesita aprender nuevos comandos para su administración, es muy fácil de adaptar manualmente, es multiplataforma además de que permite:

- ✓ La replicación condicional.
- ✓ La detección de conflictos.
- ✓ El monitoreo de las conexiones.
- ✓ Notificaciones vía email.
- ✓ La replicación de valores devueltos por funciones de fecha, aleatorias, secuencias. (18)

1.4.3 PgCluster

Es un sistema de replicación sincrónico multi-maestro para PostgreSQL. Contiene tres tipos de servidores para realizar la replicación, un servidor para balance de carga, un clúster de base de datos y un servidor de réplica. Es compatible con los sistemas operativos Linux, Solaris y FreeBSD, pero no lo es con Windows. Este tiene dos funciones principales:

- ✓ Función de compartición de carga: la carga de la sesión de las demandas es distribuida. Es efectivo en aplicaciones web donde existe gran demanda por el número de peticiones.
- ✓ Función de alta disponibilidad: cuando ocurre un fallo en el clúster de base de datos, el servidor de balance de carga y el de replicación separan el fallo del sistema, y continúa el servicio.

PgCluster tiene una compleja instalación y configuración y puede requerir configuraciones avanzadas de hardware, realiza la replicación a nivel de sentencias, por lo que en algunos casos no se replican correctamente. El clúster de base de datos cuando es reparado puede restaurarse dinámicamente a un sistema, sin detener el servicio. Los datos son copiados automáticamente a la BD restaurada o añadidos desde otra BD. (19)

1.4.4 SymmetricDS

Es un software desarrollado en Java que permite la replicación de datos de forma asincrónica multi-maestro y sincronización bidireccional la cual se define a nivel de tablas. El software está diseñado para replicar una gran cantidad de bases de datos, trabajar con conexiones de bajo ancho de banda y resistir a períodos de inoperatividad de la red. SymmetricDS soporta la sincronización entre diferentes plataformas de base de datos (MySQL, Oracle, SQL Server, PostgreSql, DB2, Firebird, HSQLDB y H2) y puede ser utilizado tanto en el sistema operativo Linux como en Windows. Su uso está basado en HTTP o HTTPS, por lo que resulta ligero y fácil de manejar. Esta herramienta cuenta con una aplicación del lado del servidor y otra del lado del cliente cliente, las cuales interactúan entre sí para realizar la réplica. Está liberado bajo los términos de software libre[3]. (20)

Dentro de las herramientas libres encontradas, se desecha Pyreplica ya que soporta réplicas maestro-esclavo y maestro-maestro limitado, y es imprescindible una herramienta que soporte réplicas multi-maestro sin restricciones, PgCluster no se escoge ya que soporta como modelo de distribución de datos el

[3]Software libre: se refiere a la libertad que tienen los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.

sincrónico. Por todo lo explicado anteriormente se selecciona la herramienta SymmetricDS porque soporta el ambiente de replicación maestro-maestro asíncrono, es multiplataforma y funciona con un gran número de gestores de BD incluyendo PostgreSQL.

1.5 Metodología de desarrollo de software

Las metodologías de desarrollo de software son las guías encargadas de definir qué se debe hacer en cada momento del proceso. En la actualidad se pueden clasificar en dos grandes grupos: las ágiles y las tradicionales o pesadas. Las primeras permiten gran libertad a los integrantes del equipo de desarrollo y se basan en el intercambio constante entre clientes y desarrolladores, lo que permite variabilidad entre los requisitos con gran calidad de solución ante estos cambios. Las tradicionales en cambio se centran en el control del proceso donde se establecen las responsabilidades y tareas que se deben llevar a cabo. Con el uso de estas se genera gran cantidad de documentos y artefactos. Entre las metodologías pesadas encontramos el Proceso Unificado de Rational (RUP) (21).

Esta metodología es la utilizada en el proyecto encargado de desarrollar la colección, por lo que es necesario adoptarla como metodología de desarrollo para conseguir uniformidad en la documentación generada.

1.5.1 Proceso Unificado de Rational (RUP)

RUP es una metodología pesada que define quién debe hacer qué, cuándo y cómo. Esta tiene tres características fundamentales: dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Es dirigido por casos de uso porque estos se especifican, se diseñan, enlazan los flujos de trabajo y los casos de uso finales son la fuente a partir de la cual los ingenieros de prueba construyen sus casos de prueba. Centrado en la arquitectura por ser esta el eje central del desarrollo de un software y esta nos permite obtener los casos de uso, correctos y de manera económica. Es iterativo e incremental ya que se divide el producto en mini proyectos donde cada uno representa una iteración que resulta en un incremento.

RUP se compone de 4 fases, estas son:

- ✓ Inicio: durante la fase de inicio las iteraciones ponen mayor énfasis en las actividades de modelado del negocio y requisitos.
- ✓ Elaboración: se determina la arquitectura óptima para el producto.
- ✓ Construcción: se lleva a cabo la construcción del producto por medio de una serie de iteraciones.

- ✓ Transición: en la fase de transición se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios.

Además cuenta con nueve flujos de trabajo, de los cuales los seis primeros se conocen como flujos de ingeniería y los tres restantes de apoyo, estos son: Modelación del negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba, Despliegue, Administración del proyecto, Administración de configuración y cambios, Ambiente (21) (22).

RUP propone como herramienta de modelado visual UML ya que con ella se consigue la integración con entornos de programación y asegurarnos que el modelo y la implementación siempre son consistentes.

1.5.2 Lenguaje Unificado de Modelado (UML)

UML (Unified Modeling Language) por sus siglas en inglés está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo, mediante el mismo es posible establecer las estructuras necesarias para modelar un sistema de software previo al proceso intensivo de escribir el código. Este es un lenguaje para visualizar un sistema a través de gráficos, especificar las características que debe tener el sistema para que los integrantes de un equipo multidisciplinario participen y se intercomuniquen fácilmente.

El uso de este es independiente del lenguaje de programación y las características del proyecto. UML permite la modificación de todos sus miembros mediante estereotipos y restricciones. Un estereotipo nos permite indicar especificaciones del lenguaje al que se refiere el diagrama de UML. Una restricción identifica un comportamiento forzado de una clase o relación, es decir, mediante la restricción estamos forzando el comportamiento que debe tener el objeto al que se le aplica (23).

1.6 Herramientas case

Proviene de las siglas que significan ingeniería del software asistida por computadora, en inglés Computer Aided Software Engineering (CASE).

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. (24)

1.6.1 Visual Paradigm para UML

Es una herramienta CASE profesional que soporta el lenguaje de modelado UML y el ciclo de vida completo del proyecto con licencia gratuita y comercial. Es multiplataforma y admite el modelado de los requerimientos, procesos de negocio y modelado de bases de datos, además que proporciona la generación automática de documentos y código a partir de los diagramas en diferentes formatos. Presenta además:

- ✓ Entorno de creación de diagramas.
- ✓ Diseño centrado en casos de uso y enfocado al negocio.
- ✓ Uso del lenguaje UML común a todo el equipo de desarrollo para facilitar la comunicación.
- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- ✓ Disponibilidad de integrarse con los principales IDEs (Entorno de desarrollo integrado). (25)

1.7 Tecnologías y lenguajes del lado del cliente

En este epígrafe se describen los lenguajes del lado del cliente que basan su procesamiento en el cliente web, es decir, que se ejecutan en el navegador web del usuario.

1.7.1 Lenguaje Extensible de Marcado de Hipertexto (XHTML)[4]

Es una versión avanzada del Lenguaje de Marcado de Hipertexto (HTML) que nace para remplazar a HTML ante su limitación de uso con las herramientas basadas en XML.

El propio W3C[5] define el HTML como "un lenguaje reconocido universalmente y que permite publicar información de forma global". Este lenguaje es sencillo y permite describir hipertexto, el texto se presenta de forma estructurada y agradable, no necesita de grandes conocimientos cuando se cuenta con un editor de páginas web, los archivos son pequeños y es admitido por todos los navegadores. (26)

[4] En inglés: eXtensible Hypertext Markup Language.
[5] El World Wide Web Consortium es una comunidad internacional que desarrolla estándares que aseguran el crecimiento de la Web a largo plazo

1.7.2 Hojas de Estilos en Cascadas (CSS)[6]

Las Hojas de Estilo en Cascadas como bien su nombre lo indican se utilizan para ayudar a embellecer y controlar el aspecto de forma general de los documentos electrónicos definidos con HTML y XHTML. Es la mejor forma para separar los contenidos de la presentación y es imprescindible para crear páginas web complejas.

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

1.7.3 JavaScript

Este es un lenguaje interpretado por la mayoría de los navegadores web, por lo que no necesita ser compilado, que permite la creación de interfaces de usuario mejoradas y 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. Es un lenguaje con una elevada curva de aprendizaje. (28)

1.8 Lenguajes de programación del lado del servidor

En este epígrafe se describen los lenguajes de programación del lado del servidor los cuales se procesan en el lado del servidor y generan la página antes de enviarla al cliente. Este proceso consiste en el procesamiento de una petición de un usuario mediante la interpretación de un script en el servidor web para generar páginas XHTML dinámicamente como respuesta.

1.8.1 Pre-procesador de hipertexto (PHP)

PHP es un lenguaje interpretado que puede ser desplegado en la mayoría de los servidores web y casi todos los sistemas operativos. Está diseñado para la construcción de páginas web y se encuentra entre los más utilizados del mundo en este aspecto. Requiere poco tiempo de estudio para su comprensión y manipulación. Permite la conexión a diferentes tipos de servidores de bases de datos, tales como: MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite. Este puede ser embebido dentro del código HTML. Entre sus principales ventajas se encuentran:

[6] En inglés: Cascading Style Sheets

- ✓ Es multiplataforma
- ✓ Está orientado a la creación de páginas web con acceso a información almacenada en bases de datos.
- ✓ 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.
- ✓ Permite técnicas de programación con el paradigma Orientado a Objetos (29).

1.9 Servidores web

El servidor Web es un programa que corre sobre el servidor, que escucha las peticiones HTTP que le llegan y las satisface. Dependiendo del tipo de la petición, el servidor Web buscará una página Web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realizó la petición.

El servidor Web va a ser fundamental en el desarrollo de las aplicaciones del lado del servidor, que vayamos a construir, ya que se ejecutarán en él. (30)

1.9.1 Apache

Hoy en día se puede decir que Apache es uno de los servidores web más utilizados a nivel mundial, esto se debe a sus características entre las que se mencionan las principales a continuación:

- Corre sobre varias plataformas y sistemas operativos.
- Es un servidor que ofrece facilidades de configuración y de diseño modular, capaz de ampliar su funcionalidad.
- Permite emplear gran cantidad de lenguajes de programación interpretados como PHP, Perl, Java y otros lenguajes de script.
- Se puede configurar cada uno de los mensajes de error que se pueden producir por su mal utilización.

Otra ventaja de Apache es que al ser tan popular y utilizado es posible encontrar gran cantidad de documentos, ejemplos y ayuda en todos los idiomas. (31)

1.10 Frameworks para el desarrollo

Un framework o marco de trabajo es un software que se puede personalizar e intercambiar para el desarrollo de una aplicación. También, se puede considerar como otra aplicación incompleta y configurable a la que se le pueden añadir últimas piezas para construir una aplicación (21).

Existe gran cantidad de frameworks para facilitar el desarrollo de las aplicaciones, estos generalmente dependen del lenguaje que se desee utilizar. Para trabajar con el lenguaje del lado del cliente JavaScript se utilizará el framework jQuery mientras que para el lenguaje del lado del servidor PHP se utilizará el framework Symfony.

1.10.1 jQuery 1.5.1

jQuery es un framework hecho para JavaScript con el objetivo de facilitar el desarrollo de páginas web dinámicas. Este admite un fácil manejo de los elementos que componen el HTML y permite manejar eventos, animaciones e interactuar con la tecnología AJAX (Asynchronous JavaScript And XML).

Según (32) entre las ventajas que se tiene con su utilización están:

- Ahorro de líneas de código.
- Es soportada por cualquiera de los principales navegadores web como Internet Explorer, Firefox, Opera, Safari y Chrome.
- Proporciona un conjunto de funciones para animar el contenido de la página.
- Integra funcionalidades para trabajar con AJAX.
- Tiene gran aceptación por parte del grupo de desarrollo ya que es una librería basada en JavaScript, del cual se tiene un conocimiento amplio.
- Posee una amplia comunidad creadora de componentes o plugins, lo que proporciona una manera más fácil de encontrar soluciones ya creadas para desarrollar elementos de interfaz de usuario, galerías de imágenes y validadores.

1.10.2 Symfony 1.4.3

Symfony es un framework utilizado para facilitar el trabajo con el lenguaje del lado del servidor PHP, ayuda a mejorar la legibilidad del código, la calidad de las aplicaciones web y la utilización de patrones de diseño. Entre sus principales características está que es multiplataforma, utiliza el patrón de diseño Modelo-Vista-Controlador y aplica buenas prácticas de desarrollo de aplicaciones web (21). Symfony además tiene como características:

- Fácil de instalar y configurar en la mayoría de las 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 los casos, pero lo suficientemente flexible como para adaptarse a los 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. (33)

1.11 Sistema Gestor de Base de Datos (SGBD)

Un gestor de base de datos o sistema de gestión de base de datos es un software que permite introducir, organizar y recuperar la información de las bases de datos. Ayuda a realizar acciones como: definición, mantenimiento de la integridad, control de la seguridad, privacidad y manipulación de los datos (34).

1.11.2 PostgreSQL 8.4

Es un SGBD relacional orientado a objetos, de software libre, liberado bajo la licencia BSD (Berkeley System Distribution) lo que implica que se pueda distribuir y modificar con la restricción de mantener el derecho de autor a sus verdaderos autores. Este cuenta con más de 15 años de desarrollo, por lo que ha ganado en mayor seguridad, integridad y corrección de los datos. Funciona en una gran cantidad de sistemas operativos, entre los que se encuentran Linux y Windows. Soporta gran cantidad de tipos de datos entre los que se encuentran los binarios, donde se incluyen imágenes, sonidos o videos. Posibilita la creación de disparadores (Tigger), vistas (Views) y funciones. Permite la replicación de sus datos a partir de herramientas elaboradas para esta función. Mediante el sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe

en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Tiene gran soporte del SQL estándar. (35)

1.12 Entorno de desarrollo integrado

Un entorno de desarrollo integrado, llamado también IDE (siglas en inglés de Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien poder utilizarse para varios.

1.12.1 Netbeans IDE

NetBeans es un entorno de desarrollo integrado libre, hecho inicialmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. Es un producto sin restricciones de uso. Entre los lenguajes que soporta además de Java están XHTML, CSS, JavaScript y PHP. Soporta integración con varios frameworks, entre los que se encuentran Symfony y jQuery. Algunas de las posibilidades que brinda el IDE hasta el momento son:

- Ejecución de comando Symfony.
- Limpiar la caché.
- Ayuda para encontrar comandos y para conocer los parámetros.
- Configuración de atajos del teclado.

1.13 Propuesta de solución

A partir de los planteamientos antes expuestos se observa que la herramienta SymmetricDS brinda gran cantidad de funcionalidades que permiten un adecuado control de la réplica, es por esto que se propone que la herramienta para dar solución al problema queda como sigue:

1. En el servidor central de cada escuela, junto al servidor de bases de datos, se tendrá instalada la aplicación servidor SymmetricDS, la que contará con las configuraciones necesarias para replicar los datos hacia los entornos móviles.
2. En cada entorno móvil se tendrá instalada la aplicación cliente SymmetricDS, la que contará con las configuraciones necesarias para replicar los datos hacia el servidor central de cada escuela.

3. La colección El Navegante será instalada tanto en el servidor central de cada escuela como en los entornos móviles, dentro de la misma se encontrará el módulo Resultados que contendrá, entre otros aspectos, el control de la réplica que se realiza.

1.14 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 han 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.

Dado las anteriores definiciones del entorno de replicación se considera que la más adecuada para darle respuesta a la problemática es maestro-maestro, ya que permite obtener mejoras en el rendimiento de la replicación y que esta se realice en ambos sentidos, logrando replicar la información desde el servidor central al local o viceversa.

Para la replicación de las bases de datos de la colección El Navegante se empleará el modelo de distribución asíncrono, con el fin de mantener coherentes y actualizados los datos entre los servidores del mismo. 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.

Para el desarrollo de la réplica 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.

Como metodología de desarrollo seleccionada está RUP, como herramienta de modelación se utilizará Visual Paradigm para UML, como lenguaje del lado del servidor se usará PHP y como frameworks Symfony y jQuery. Como sistema de gestión de bases de datos se utilizará PostgreSQL y como servidor web Apache. No es objetivo de este trabajo la justificación de estas herramientas y lenguajes, ya que han sido definidas en estudio previo para ser utilizadas en la arquitectura del proyecto El Navegante.

1.15 Conclusiones

Luego de haber realizado el estudio para dar solución al problema planteado anteriormente, enfocado fundamentalmente en los conceptos mencionados en el objeto de estudio, se obtuvo como resultado la necesidad de elaboración de una herramienta completamente libre, la cual estará compuesta por la aplicación SymmetricDS, integrada a la colección El Navegante y desarrollada con los lenguajes antes mencionados, encargada de intercambiar los datos entre diferentes computadores y mantener el control sobre los mismos.

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 especifica el modelo de dominio, las características del sistema planteadas en requisitos funcionales, no funcionales y descripción de los casos de uso.

2.1 Modelo de Dominio

Al analizar la propuesta de solución del problema se comprueba que los procesos del negocio no están bien delimitados, que poseen un bajo nivel de estructuración, por lo que se considera dirigir los procesos de réplica y sincronización por medio de un modelo de dominio. Este modelo se utilizará como un medio para comprender el sector de negocios al cual el sistema va a servir. El Modelo de Dominio se centra en una parte del negocio, la relacionada con el ámbito del proyecto. En este contexto el término "dominio" representa una parte del "negocio". El modelo identifica las relaciones entre todas las entidades importantes dentro del sistema e identifica generalmente sus métodos y cualidades importantes. El modelo de dominio puede ser tomado como el punto de partida para el diseño del sistema.

2.1.1 Análisis de las clases conceptuales del Dominio

Clases conceptuales	Descripción
Colección_El_Navegante	Conjunto de Software que representan hiperentornos de enseñanza-aprendizaje.
Módulo	Conjunto de componentes fundamentales que forman el producto.
Juegos	Módulo encargado de desarrollar habilidades y conocimientos en los estudiantes a través de juegos divertidos.
Mediateca	Módulo encargado de gestionar todo lo referente a los recursos multimedia.
Ejercicios	Módulo encargado de controlar el aprendizaje de los

	estudiantes a través de preguntas.
Resultados	Módulo encargado de controlar y almacenar las trazas del estudiante para visualizar su comportamiento durante su interacción con cada producto de la Colección.
Contenido	Módulo encargado de representar los contenidos de las asignaturas o temas correspondientes a cada producto.
Sección	Conjunto de opciones que se puede seleccionar en el módulo Resultados.
Trayectoria del estudiante	Contiene información resumida de la interacción del estudiante con el producto.
Nodo_central	Representa al servidor central.
Nodo_móvil	Representa los entornos móviles.
SymmetricDS Cliente	Representa la herramienta SymmetrisDS en su versión cliente.
SymmetricDS Servidor	Representa la herramienta SymmetrisDS en su versión servidor.

Tabla 1 Descripción de los conceptos de dominio.

Estas clases se relacionan y dan lugar al siguiente modelo de dominio:

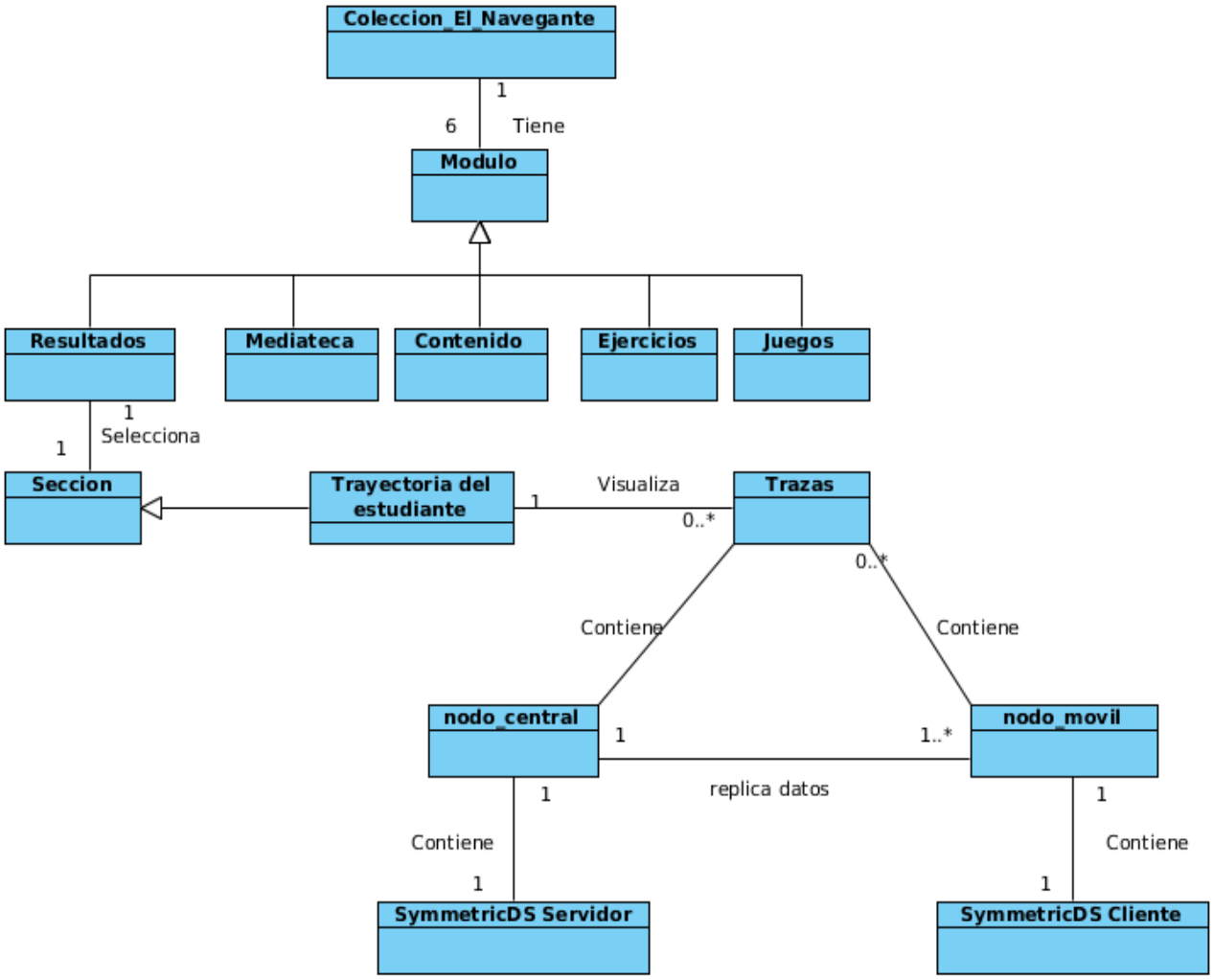

Figura 5 Diagrama del modelo de dominio.

2.2 Especificación de los requisitos de software

La especificación de los requisitos del software es la descripción de un sistema informático a desarrollar. Los requisitos son las necesidades del producto que se debe desarrollar en cualquier proyecto de software, por lo que es importante que un requisito deba ser especificado por escrito, sea posible de probar o verificar para poder comprobar si se cumplió con él o no; sea consistente, es decir, que no entre en contradicción con otros requisitos y conciso, o sea, fácil de leer y entender.

La ingeniería de requisitos proporciona el mecanismo apropiado para entender lo que el cliente quiere, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación, y administrar los requisitos conforme estos se trasformen en un sistema operacional. (36)

La herramienta para la replicación de datos de la colección El Navegante 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.

2.2.1 Requisitos funcionales

Los requisitos funcionales son los que definen el comportamiento interno del sistema, o sea, son las condiciones o capacidades que el sistema debe cumplir.

A continuación se muestran los requisitos funcionales correspondientes a la herramienta de réplica de la colección El navegante:

RF-1: Replicar datos del servidor hacia los entornos móviles.

RF-1.1: Replicar noticias hacia los entornos móviles.

Este requisito consiste en que el sistema debe permitir enviar las nuevas noticias insertadas por los maestros en el servidor hacia los entornos móviles.

Salida:

En cada entorno móvil se podrán observar las noticias de forma actualizada.

RF-1.2: Replicar artículos hacia los entornos móviles.

Este requisito consiste en que el sistema debe permitir enviar los nuevos artículos insertadas por los maestros en el servidor hacia los entornos móviles.

Salida:

En cada entorno móvil se podrán observar los artículos de forma actualizada.

RF-1.3: Replicar efemérides hacia los entornos móviles.

Este requisito consiste en que el sistema debe permitir enviar las nuevas efemérides insertadas por los maestros en el servidor hacia los entornos móviles.

Salida:

En cada entorno móvil se podrán observar las efemérides de forma actualizada.

RF-1.4: Replicar ejercicios hacia los entornos móviles.

Este requisito consiste en que el sistema debe permitir enviar los nuevos ejercicios insertadas por los maestros en el servidor hacia los entornos móviles.

Salida:

En cada entorno móvil se podrán observar los ejercicios de forma actualizada.

RF-2: Replicar datos de los entornos móviles hacia el servidor.

RF-2.1: Replicar los datos generados por los estudiantes en el módulo Contenido.

Este requisito consiste en que el sistema debe permitir enviar los datos generados en el módulo Contenido de los entornos móviles hacia el servidor.

Salida:

En el módulo Resultados se podrán visualizar los datos del estudiante actualizados.

RF-2.2: Replicar los datos generados por los estudiantes en el módulo Ejercicios.

Este requisito consiste en que el sistema debe permitir enviar los datos generados en el módulo Ejercicios de los entornos móviles hacia el servidor.

Salida:

En el módulo Resultados se podrán visualizar los datos del estudiante actualizados.

RF-2.3: Replicar los datos generados por los estudiantes en el módulo Juegos.

Este requisito consiste en que el sistema debe permitir enviar los datos generados en el módulo Juegos de los entornos móviles hacia el servidor.

Salida:

En el módulo Resultados se podrán visualizar los datos del estudiante actualizados.

RF-2.4: Replicar los datos generados por los estudiantes en el módulo Mediateca.

Este requisito consiste en que el sistema debe permitir enviar los datos generados en el módulo Mediateca de los entornos móviles hacia el servidor.

Salida:

En el módulo Resultados se podrán visualizar los datos del estudiante actualizados.

RF-3. Replicar los datos de manera automática.

El sistema debe replicar los datos apenas encuentre red la PC en la que se tiene instalado el producto.

Salida:

En el módulo Resultados se podrán visualizar de forma casi inmediata los datos del estudiante actualizados.

RF-4. Mostrar pantalla principal de la replicación de los datos.

El sistema debe mostrar al usuario la pantalla principal de la replicación de los datos.

Salida:

Muestra la pantalla principal de la replicación de los datos dando la posibilidad de seleccionar la opción que desee.

RF- 5. Mostrar estado de la réplica.

El sistema muestra la posibilidad de conocer si la réplica está activa o no.

Salida:

Muestra si la réplica se encuentra activa o no.

RF-6. Activar/Desactivar la réplica.

El sistema debe permitir desactivar la réplica de forma manual en caso de que esté activada, de lo contrario debe permitir activarla.

Salida:

Muestra un mensaje afirmando que se realizó la operación correctamente.

RF-7. Mostrar características de la réplica.

El sistema permite seleccionar las fechas de inicio y fin para mostrar cómo se ha comportado la réplica en ese intervalo de tiempo.

Salida:

Muestra la información de las réplicas ejecutadas en el intervalo de fechas seleccionadas. Lista día, hora, fecha, canal, nodo, estado de cada réplica, clasificándolas en completada o fallida.

RF- 8. Escoger datos a replicar.

El sistema permite seleccionar los datos, las cuales el usuario desea replicar y las que no.

Salida:

Muestra un mensaje afirmando que se han escogido los datos correctamente.

2.2.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el sistema debe tener, los cuales deben verse como las características que hacen al producto atractivo, usable, rápido y confiable.

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema. (37)

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

RNF-1 Restricciones en el Diseño y la Implementación:

- ✓ Como lenguaje de programación se tiene PHP.
- ✓ Como framework para PHP se tiene Symfony.
- ✓ Como lenguaje del lado del cliente se tiene XHTML, CCS y JavaScript.
- ✓ Como framework para JavaScript se tiene jQuery.
- ✓ Las herramientas para el modelado debe ser Visual Paradigm.
- ✓ Se utilizará como metodología de desarrollo de software RUP, usando el lenguaje de modelación UML.
- ✓ Como servidor web se tiene Apache.
- ✓ Como sistema gestor de base de datos se tiene PostgreSQL.

RNF-2 Requisitos de apariencia o interfaz externa: Es necesario que tenga una apariencia amigable y fácil de usar, además se debe corresponder con la utilizada en el módulo Resultados.

RNF-3 Requisitos de software: Computadora Personal con navegador Mozilla Firefox 3.0 o superior.

RNF-4 Requisitos de soporte:

- ✓ Se realizará transferencia tecnológica de la colección a los clientes.
- ✓ Se impartirán clases a los profesores para explicar el funcionamiento y utilidad del producto.

2.3 Definición y descripción de los casos de uso del sistema

Un caso de uso (CU) proporciona un medio intuitivo y sistemático para capturar los requisitos funcionales con un énfasis especial en el valor añadido para cada usuario individual o para cada sistema externo. Mediante la utilización de los casos de uso, los analistas se ven obligados a pensar en términos de quienes son los usuarios y que necesidades u objetivos de la empresa pueden cumplir. (38)

En la siguiente tabla se muestra los CU del sistema:

Casos de uso	Nombre
CU-1	Mostrar Pantalla Principal de la Réplica.
CU-2	Replicar Datos de Estudiantes.
CU-3	Activar/Desactivar Réplica.
CU-4	Replicar Datos Automáticamente.
CU-5	Replicar Datos hacia los Computadores Locales.
CU-6	Mostrar Estado de la réplica.
CU-7	Mostrar características de la réplica.
CU-8	Escoger datos a replicar.

Tabla 2 Definición de los casos de uso.

Un actor es una idealización de una persona externa, de un proceso, o de una cosa que interactúa con un sistema, un subsistema, o una clase. Un actor caracteriza las interacciones que los usuarios exteriores pueden tener con el sistema. En tiempo de ejecución, un usuario físico puede estar limitado a los actores múltiples dentro del sistema. Diferentes usuarios pueden estar ligados al mismo actor y por lo tanto pueden representar casos múltiples de la misma definición de actor. Cada actor participa en uno o más

casos de uso. Interactúa con el caso de uso (y por lo tanto con el sistema o la clase que posee el caso de uso), intercambiando mensajes. (39)

En la siguiente tabla se definen y describen los actores del sistema:

Actor	Descripción
SymmetricDS Cliente	Se encarga de replicar los datos de los estudiantes en todos los módulos hacia el servidor central.
SymmetricDS Servidor	Se encarga de replicar los datos hacia los computadores locales.
Usuario	Es el encargado de mostrar la pantalla principal de la réplica, además de mostrar el estado actual de la réplica y de activar o desactivar la réplica de forma manual.
Servicio de conexión	Es el encargado de verificar la existencia de red en las computadoras

Tabla 3 Descripción de los actores.

El diagrama de casos de uso del sistema es un conjunto de acciones que un sistema ejecuta y que produce un resultado observable para un actor. Es decir, son funcionalidades que el sistema brinda a los actores que interactúan con el mismo. *(36)*

En la figura 6 se muestra el diagrama de CU del sistema, donde quedan representadas las funcionalidades del mismo y su interacción con los actores.

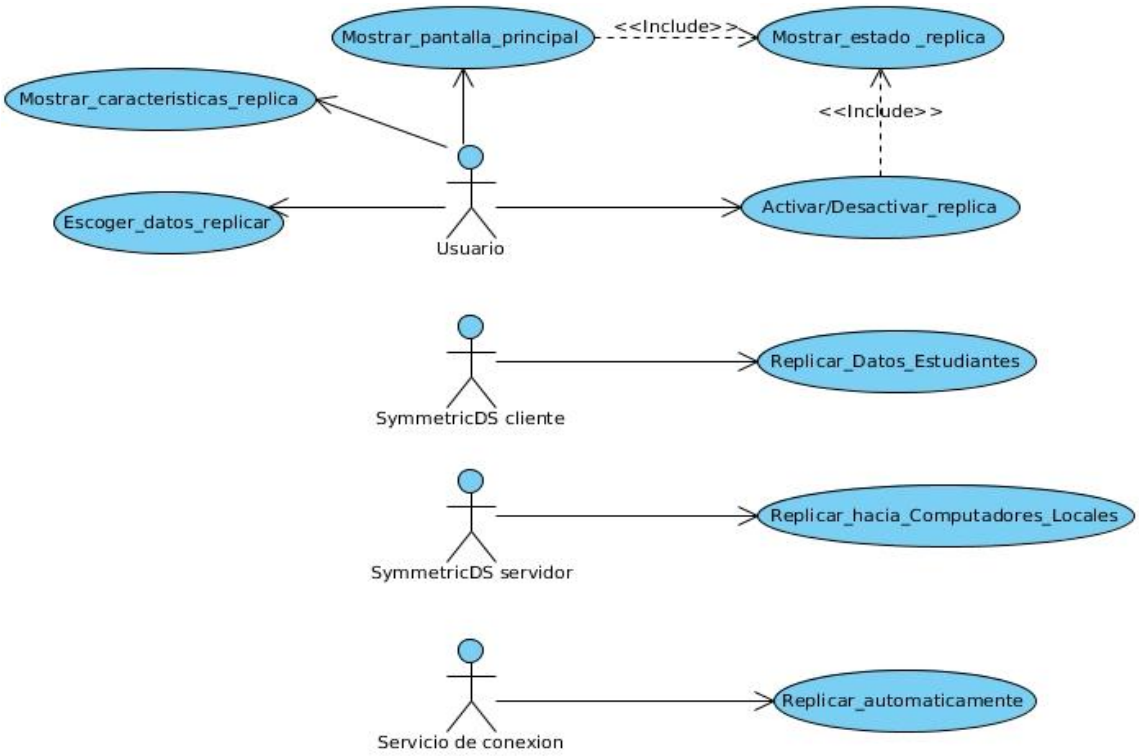

Figura 6: Descripción de Casos de Uso.

A continuación se realiza una descripción de los CU representados en el diagrama anterior, donde se define: el objetivo general, el actor que lo inicia, un resumen de su funcionamiento, así como las condiciones para que el CU pueda realizarse.

Objetivo	Replicar hacia Computadores Locales.
Actores	SymmetricDS_Servidor
Resumen	El caso de uso se inicia cuando el actor SymmetricDS_Servidor comienza a enviar los nuevos datos hacia los computadores locales.
Precondiciones	Debe existir red de manera estable tanto en el computador local como en el servidor.
Poscondiciones	Se podrán visualizar los datos actualizados en los computadores

	locales.
Referencia	RF-1

Tabla 4 CU Replicar hacia Computadores Locales.

Objetivo	Replicar datos de estudiantes.
Actores	SymmetricDS_Cliente
Resumen	El caso de uso se inicia cuando el actor SymmetricDS_Cliente comienza a enviar los datos generados por los estudiantes, hacia el servidor central, terminando así el caso de uso.
Precondiciones	Debe existir red de manera estable tanto en el computador local como en el servidor.
Poscondiciones	Se podrán visualizar, en el módulo Resultados, los datos delos estudiantes actualizados.
Referencia	RF-2

Tabla 5 CU Replicar Datos de Estudiantes.

Objetivo	Replicar los datos de manera automática.
Actores	Servicio de conexión.
Resumen	El caso de uso se inicia cuando el actor Servicio de Conexión verifica la existencia de red en las computadoras para poder replicar automáticamente, terminando así el caso de uso.
Precondiciones	Debe existir red tanto en el servidor como en los entornos móviles.
Poscondiciones	Se mostró la pantalla principal de la réplica de datos.
Referencia	RF-3

Tabla 6 CU Replicar Datos Automáticamente.

Objetivo	Mostrar la pantalla principal de la réplica de datos

Actores	Usuario.
Resumen	El caso de uso se inicia cuando el actor Usuario selecciona la opción *Réplica de datos* que le permite mostrar la pantalla inicial para la réplica de la base de datos. El actor del sistema puede observar el estado de la réplica, seleccionar *Desactivar réplica, Mostrar características de la réplica o Escoger datos a replicar* terminando así el caso de uso.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. El servidor central debe estar configurado.
Poscondiciones	Se mostró la pantalla principal de la réplica de datos.
Referencia	RF-4

Tabla 7: CU Mostrar Pantalla Principal de la Réplica.

Objetivo	Mostrar el estado de la réplica.
Actores	Usuario
Resumen	El caso de uso se inicia cuando el actor Usuario selecciona la opción *"Mostrar Estado de la réplica"* en la interfaz de Réplica de datos. El sistema permite filtrar por fechas las réplicas que el actor desea revisar y luego le muestra un reporte de replicación, terminando así el caso de uso.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. El servidor local debe estar configurado y la configuración estar completa.
Poscondiciones	Se mostró el estado de la réplica.
Referencia	RF-5

Tabla 8 CU Mostrar estado de la réplica.

Objetivo	Activar la réplica de los datos de manera manual.
Actores	Usuario registrado.
Resumen	El caso de uso se inicia cuando el actor Usuario selecciona la opción *Activar réplica* que le permite activar de manera manual la réplica de la base de datos, terminando así el caso de uso.
Precondiciones	Debe haberse desactivado la réplica.
Poscondiciones	Se activó de manera manual la réplica de datos.
Referencia	RF-6

Tabla 9 CU Activar Réplica Manualmente.

Objetivo	Mostrar características de la réplica.
Actores	Usuario
Resumen	El caso de uso se inicia cuando el actor Usuario selecciona la opción *"Mostrar características de la réplica"* en la interfaz de Réplica de datos. El sistema permite filtrar por fechas las réplicas que el actor desea revisar y luego le muestra un reporte de replicación, terminando así el caso de uso.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. El servidor local debe estar configurado y la configuración estar completa.
Poscondiciones	Se mostró el estado de la réplica.
Referencia	RF-7

Tabla 10 CU Mostrar características de la réplica.

Objetivo	Escoger datos a replicar.

Actores	Usuario
Resumen	El caso de uso se inicia cuando el actor Usuario selecciona la opción *"Escoger datos a replicar"* en la interfaz de Réplica de datos. El sistema permite escoger los datos de los diferentes módulos que desea replicar o no, en ese momento, luego le muestra un mensaje de aceptación, terminando así el caso de uso.
Precondiciones	Debe haberse generado el escritorio de trabajo del usuario autenticado. El servidor local debe estar configurado y la configuración estar completa.
Poscondiciones	Se escogió los datos que se desean replicar.
Referencia	RF-8

Tabla 11 CU Escoger datos a replicar.

2.4 Conclusiones

Con un correcto entendimiento de los requisitos se lograron especificar cada uno de estos, logrando definir las principales funcionalidades del sistema y asociándolas a los diferentes actores que las inicializan. Teniendo en cuenta los resultados de este capítulo, se puede iniciar la elaboración de la propuesta de solución anteriormente planteada.

Capítulo III: Análisis y Diseño del sistema

Introducción

En el presente capítulo se muestra lo referente al análisis y el diseño del sistema propuesto, lo que será de gran ayuda para la comprensión de la realización del sistema. Se describen las clases del análisis y las del diseño, así como sus correspondientes diagramas.

3.1 Análisis del sistema

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

El análisis de requisitos le proporciona al diseñador de software una representación de información, función y comportamiento que puede que puede trasladar a diseños arquitectónicos de interfaz y en el nivel de componentes. (36)

3.1.1 Modelo de análisis.

El modelo de análisis nos permite razonar sobre los aspectos internos del sistema, incluidos sus recursos compartidos internos. Además nos ofrece un mayor poder expresivo y una mayor formalización, como por ejemplo la que nos ofrece los diagramas de interacción. El modelo de análisis también nos ayuda a estructurar los requisitos y nos proporciona una estructura centrada en el mantenimiento, en aspectos tales como la flexibilidad ante los cambios y la reutilización.

3.1.2 Clases del análisis.

El Modelo de Análisis contiene el resultado de los casos de uso del análisis traducido en Clases del Análisis. Sin embargo el Modelo de Análisis es un artefacto opcional. Normalmente las clases del análisis evolucionan directamente en elementos del modelo de diseño: algunas se convierten en clases de diseño, otras se convierten en subsistemas de diseño. La meta del Análisis es identificar un mapeo preliminar de conducta requerida dentro de elementos de modelado en el sistema. La meta del diseño es transformar este mapeo preliminar dentro de elementos de modelo que pueden ser implementados. Como resultado hay un refinamiento en el detalle y la precisión cuando se muda del análisis hacia el diseño. Como

consecuencia, las "clases de análisis" son a menudo muy fluidas, cambiables, y evolucionan grandemente antes de consolidarse en las actividades de diseño. (40)

Una clase de análisis se centra en el tratamiento de los requisitos funcionales y pospone los no funcionales, denominándolos requisitos especiales, hasta llegar a las actividades de diseño e implementación subsiguientes. Estas se conforman por atributos y las relaciones existentes entre ellas. Las mismas se clasifican en:

- ✓ **Interfaz**: se utilizan para modelar la interacción entre el sistema y sus actores. Las clases de interfaz representan a menudo abstracciones de ventanas, formularios, paneles, interfaces de comunicaciones, interfaces de impresoras, sensores, entre otros.
- ✓ **Entidad**: se utilizan para modelar información que posee una vida larga y que es a menudo persistente**.** Las clases de entidad modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un objeto del mundo real, o un suceso del mundo real. En la mayoría de los casos, las clases de entidad se derivan directamente de una clase de entidad del negocio o de una clase del dominio. Las clases de entidad suelen mostrar una estructura de datos lógica y contribuyen a comprender de que información depende el sistema.
- ✓ **Control**: representan la coordinación, secuencia, gestión de transacciones y control de otros objetos. Usualmente se usan para encapsular el control relacionado con un caso de uso específico. La dinámica del sistema se modela en una clase controladora, que se encarga de delegar trabajo a otras clases. (41)

3.1.3 Diagramas de Clases del Análisis.

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

A continuación se muestran los diagramas de clases de análisis, asociados al problema.

Figura 7 DCA[7] del CU Mostrar estado de réplica.

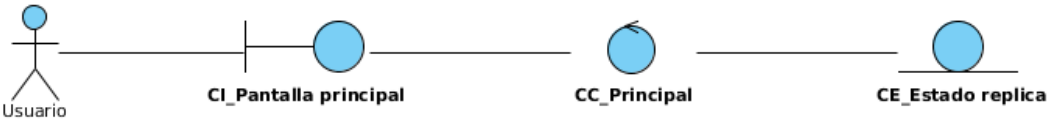

Figura 8 DCA del CU Mostrar pantalla principal.

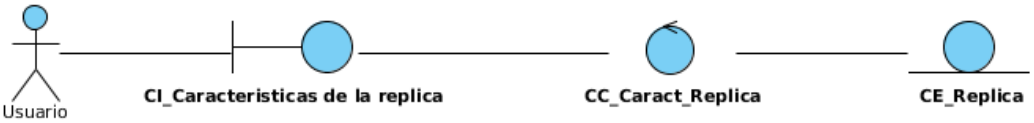

Figura 9 DCA del CU Mostrar características de la réplica.

Figura 10 DCA del CU Escoger datos a replicar

Un diagrama de colaboración se utiliza para ilustrar la realización de un Caso de Uso. Muestra como los objetos interactúan para lograr el comportamiento de un CU o parte de este y de esta forma definen los roles de los mismos. A diferencia de los diagramas de secuencia su principal objetivo es mostrar la relación entre dichos objetos. Estos diagramas tienen una mayor utilidad cuando se utilizan en interacciones entre un número no muy grande de objetos, pues en caso contrario el número de mensajes entre estos crece y el diagrama se hace difícil de entender; en estos casos los diagramas de secuencia son una mejor elección.

A continuación se muestra un diagrama de colaboración, asociado al problema.

[7] Diagrama de clases del análisis.

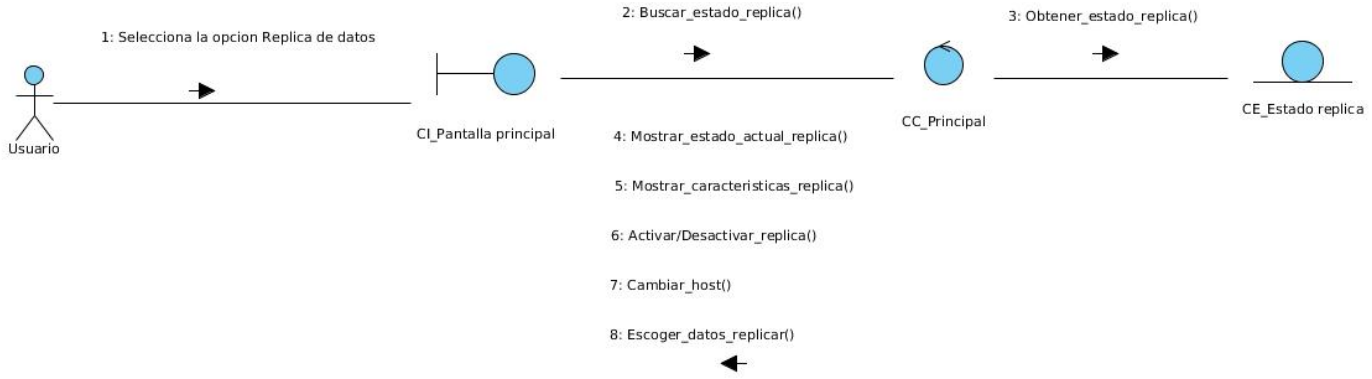

Figura 11 DC[8] del CU Mostrar pantalla principal.

3.2 Patrón arquitectónico Modelo-Vista-Controlador

El patrón conocido como Modelo-Vista-Controlador (MVC) separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes:

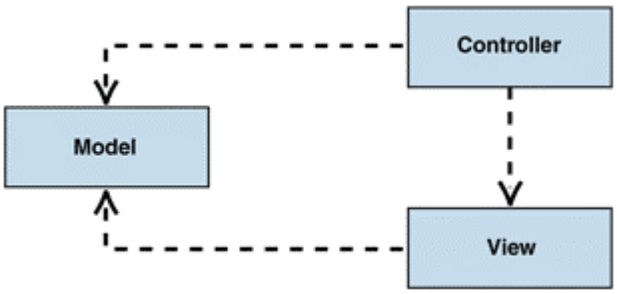

Figura 12 Patrón Modelo-Vista-Controlador.

- ✓ Modelo: el modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).
- ✓ Vista: maneja la visualización de la información.
- ✓ Controlador: interpreta las acciones del ratón y el teclado, informando al modelo y a la vista para que cambien según resulte apropiado.

[8] Diagrama de colaboración.

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual. La separación entre vista y controlador puede ser secundaria en aplicaciones de clientes ricos y, de hecho, muchos frameworks de interfaz implementan ambos roles en un solo objeto. En aplicaciones de Web, por otra parte, la separación entre la vista (el browser) y el controlador (los componentes del lado del servidor que manejan los requerimientos de HTTP) está mucho más taxativamente definida. (42)

3.3 Patrones de diseño

Los patrones son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Sus soluciones están basadas en los problemas del diseño que se repiten y que se presentan en situaciones particulares. (42)

Para el desarrollo de la herramienta de replicación se utilizaron los patrones patrones de diseño orientado a objeto: GRASP y GOF que emplea el framework Symfony para darle solución a problemas específicos.

3.3.1 Patrones GRASP

GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades. Los patrones GRASP describen principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Existen varios tipos de patrones GRASP como: Experto, Creador, Alta Cohesión, Bajo Acoplamiento, Controlador, entre otros. (43)

Experto

Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen. (43)

Este patrón se utiliza en la capa de abstracción del modelo, las clases creadas contienen información y varias funcionalidades relacionadas directamente con la entidad que representan en la base de datos.

Creador

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. (43)

Dentro de la aplicación este patrón se observa en las acciones del controlador, las cuales crean objetos del modelo que representan las entidades.

Bajo acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras y lo contrario para un acoplamiento alto. (43)

Este patrón es implementado por el framework Symfony al no asociar las clases del modelo con las de la vista o el controlador, la dependencia entre las clases en este caso se mantiene baja.

Alta cohesión

En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen mucho trabajo. (43)

Este patrón es utilizado en las clases del modelo, donde cada una tiene las responsabilidades moderadas y colabora con las otras para llevar a cabo las tareas.

Polimorfismo

El polimorfismo tiene varios sentidos. Dentro de este contexto significa asignar el mismo nombre a servicios en varios objetos, cuando los servicios se parecen o estén relacionados entre ellos. Los tipos de objetos suelen estar relacionados en una jerarquía con una superclase común. (43)

Este patrón es de los más utilizados en la aplicación, ya que las clases del modelo y las clases de las acciones tienen cada una clases de la cual heredan y a pesar de esto mantienen las responsabilidades desde su misma implementación.

3.3.2 Patrones GOF

GOF es un acrónimo que significa Gang of Four (“Banda de los cuatro”) que hace referencia a los autores del libro Design Patterns donde definieron un catálogo con 23 patrones básicos.

Patrón Mediador

Define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.

Coordina las relaciones entre sus asociados. Permite la interacción de varios objetos, sin generar acoples fuertes en esas relaciones.

Aplicabilidad

Usar el patrón Mediator cuando:

- Un conjunto grande de objetos se comunica de una forma bien definida, pero compleja.
- Usar un objeto se hace difícil porque se relaciona con muchos objetos.
- El comportamiento de muchos objetos que está distribuido entre varias clases, puede resumirse en una o varias.

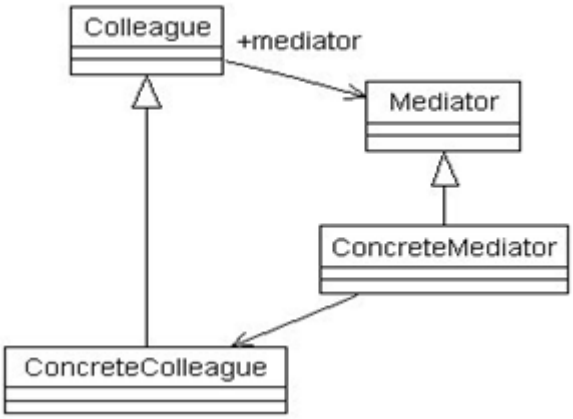

Figura 13 Ejemplo del patrón Mediator.

Patrón Decorator

Añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.

El patrón Decorator responde a la necesidad de añadir dinámicamente funcionalidad a un Objeto. Esto nos permite no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera.

Aplicabilidad

- Añadir objetos individuales de forma dinámica y transparente.
- Responsabilidades de un objeto pueden ser retiradas.
- Cuando la extensión mediante la herencia no es viable.
- Hay una necesidad de extender la funcionalidad de una clase, pero no hay razones para extenderlo a través de la herencia.
- Hay la necesidad de extender dinámicamente la funcionalidad de un objeto y quizás quitar la funcionalidad extendida. (42)

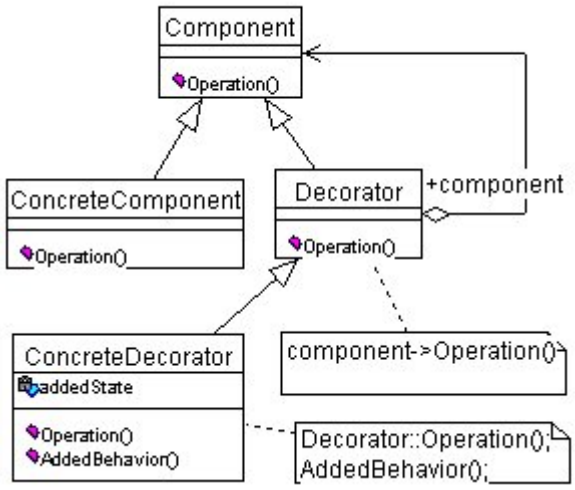

Figura 14 Ejemplo del patrón Decorator.

3.4 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además el modelo de diseño sirve de abstracción de la implementación del sistema y es, de ese modo, utilizado como una entrada fundamental de las actividades de implementación. (38)

3.4.1 Diagramas de clases del diseño.

Clases	Esteriotipos	Función
Server Page[9]		Representa la página web que tiene código y que se ejecuta en el servidor.
Client Page[10]		Representa una página web, con formato HTML. Son interpretadas por el navegador, su función es visualizar, interactuar y mostrar lo que el usuario necesita.
Form[11]		Colección de elementos de entrada que son parte de una página cliente. Sus atributos son los elementos de entrada del formulario. Su principal función es enviar los datos entrados por el usuario a la Página servidora.

Tabla 12 Clases del diseño.

A continuación se presentan los diagramas de clases del diseño para cada uno de los CU del sistema.

[9] En español Página Servidora.
[10] En español Pagina Cliente.
[11] En español Formulario.

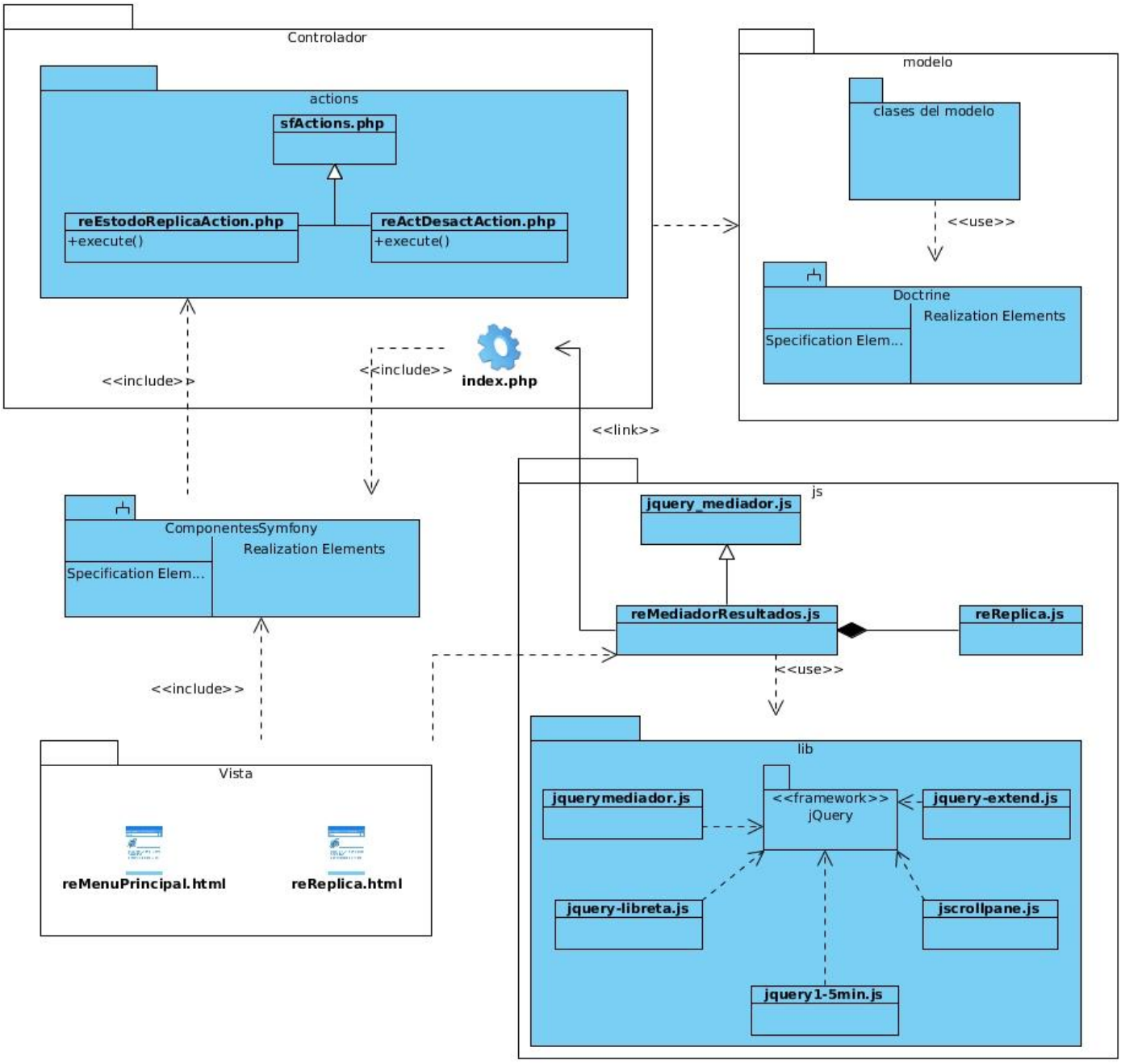

Figura 15 Diagrama de clases del diseño CU Mostrar pantalla principal.

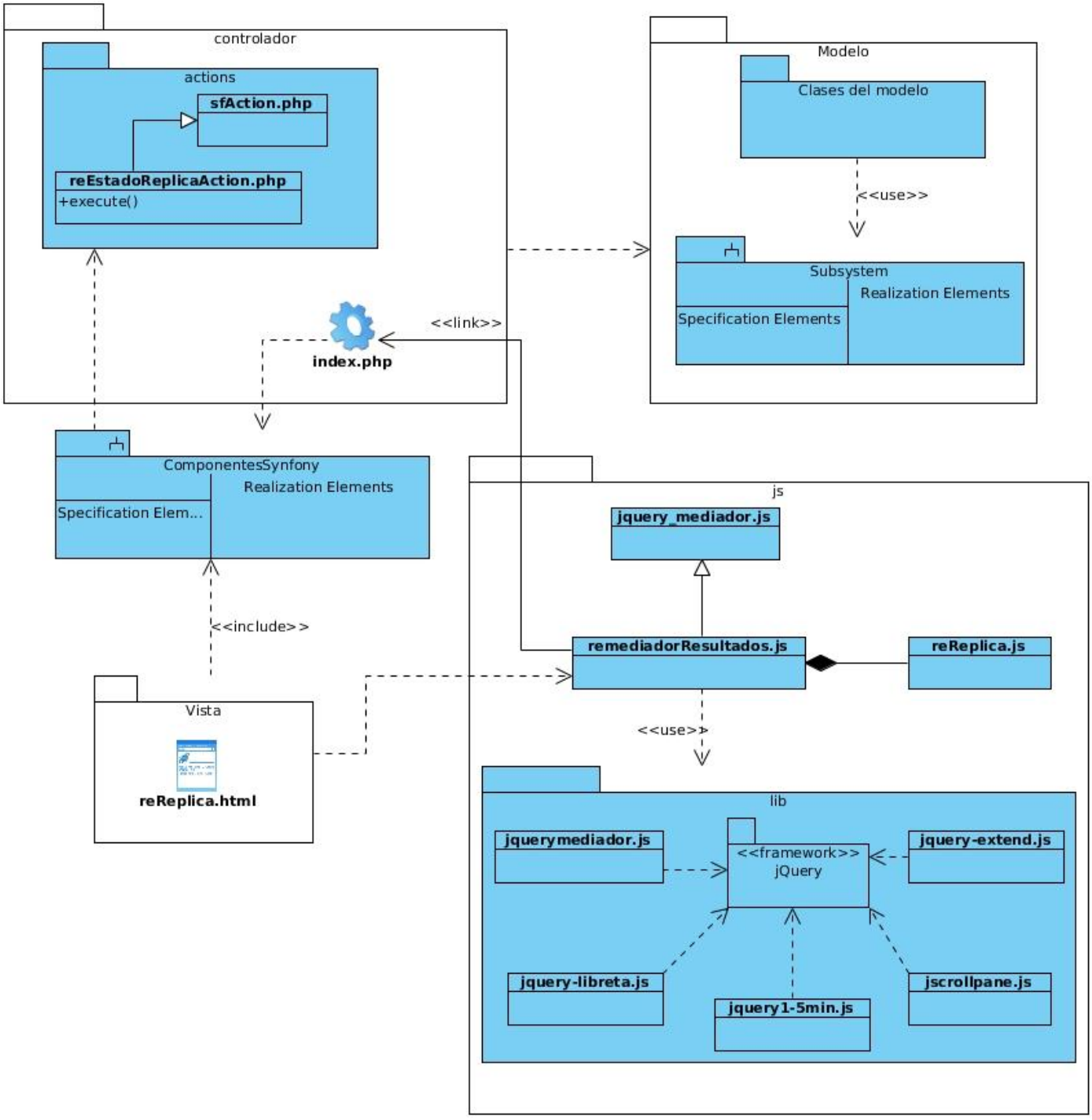

Figura 16 Diagrama de clases del diseño CU Mostrar estado de la réplica.

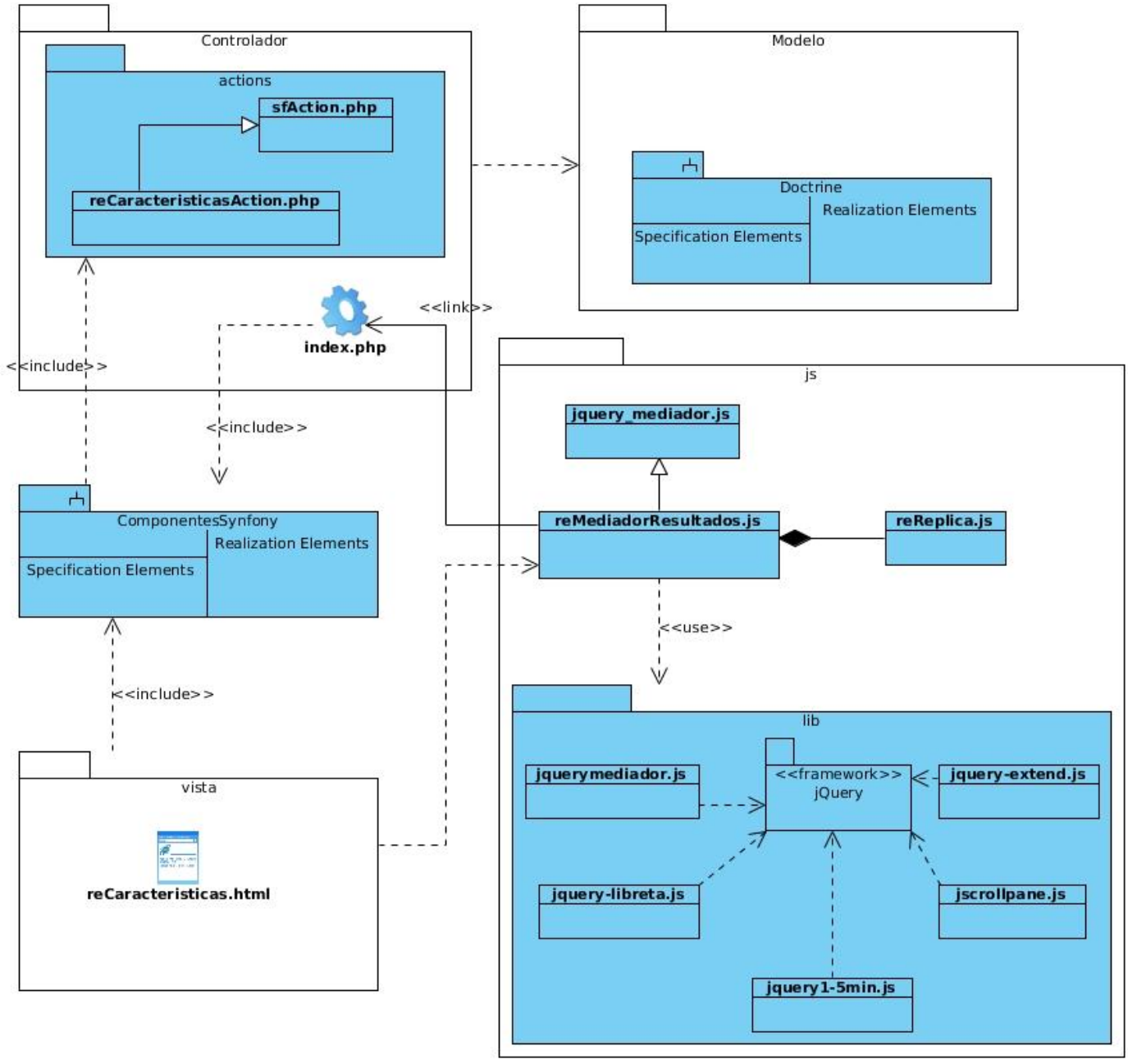

Figura 17 Diagrama de clases del diseño CU Mostrar características de la réplica.

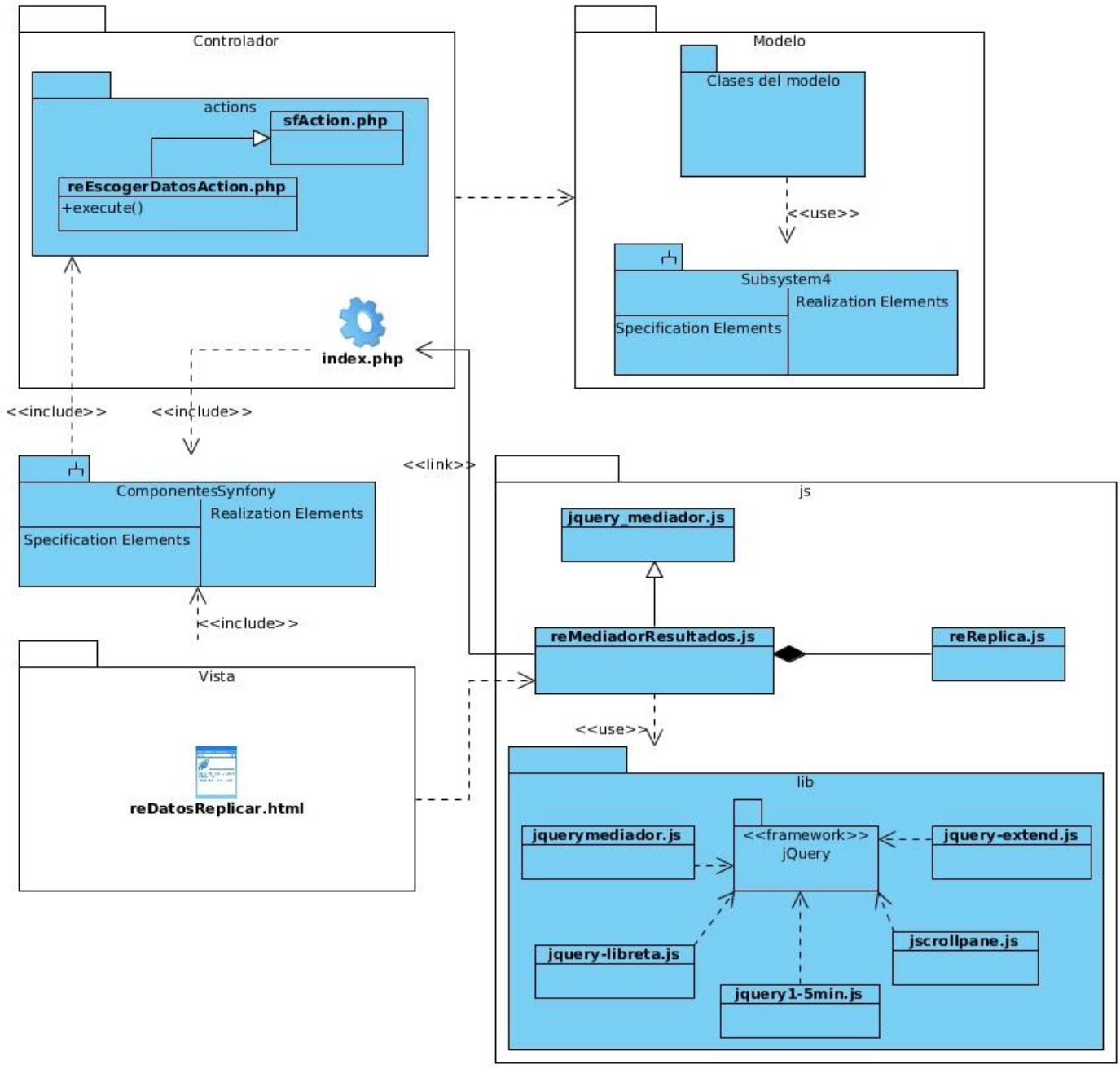

Figura 18 Diagrama de clases del diseño CU Escoger datos a replicar.

3.5 Diseño de la base de datos

Las bases de datos necesitan de una definición de su estructura que le permita almacenar datos, reconocer el contenido, y recuperar la información. La estructura tiene que ser desarrollada para satisfacer las necesidades de las aplicaciones que la usarán.

3.5.1 Modelo Entidad-Relación

Un Diagrama o Modelo Entidad Relación (a veces denominado por su siglas, E-R "Entity Relationship", o, "DER" Diagrama Entidad Relación), es una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información así como sus interrelaciones y propiedades.

La base de datos que se representa se vincula con las tablas que: agrega la herramienta SymmetricDS y las de la colección. Con el objetivo de observar de manera ampliada las tablas se dividió el Modelo Entidad-Relación en dos: en la figura 19 se pueden observar las tablas relacionadas con la herramienta SymmetricDS y en la figura 20 las que se relacionan con la colección.

A continuación se muestra el Modelo Entidad-Relación referente a la base de datos del sistema a desarrollar.

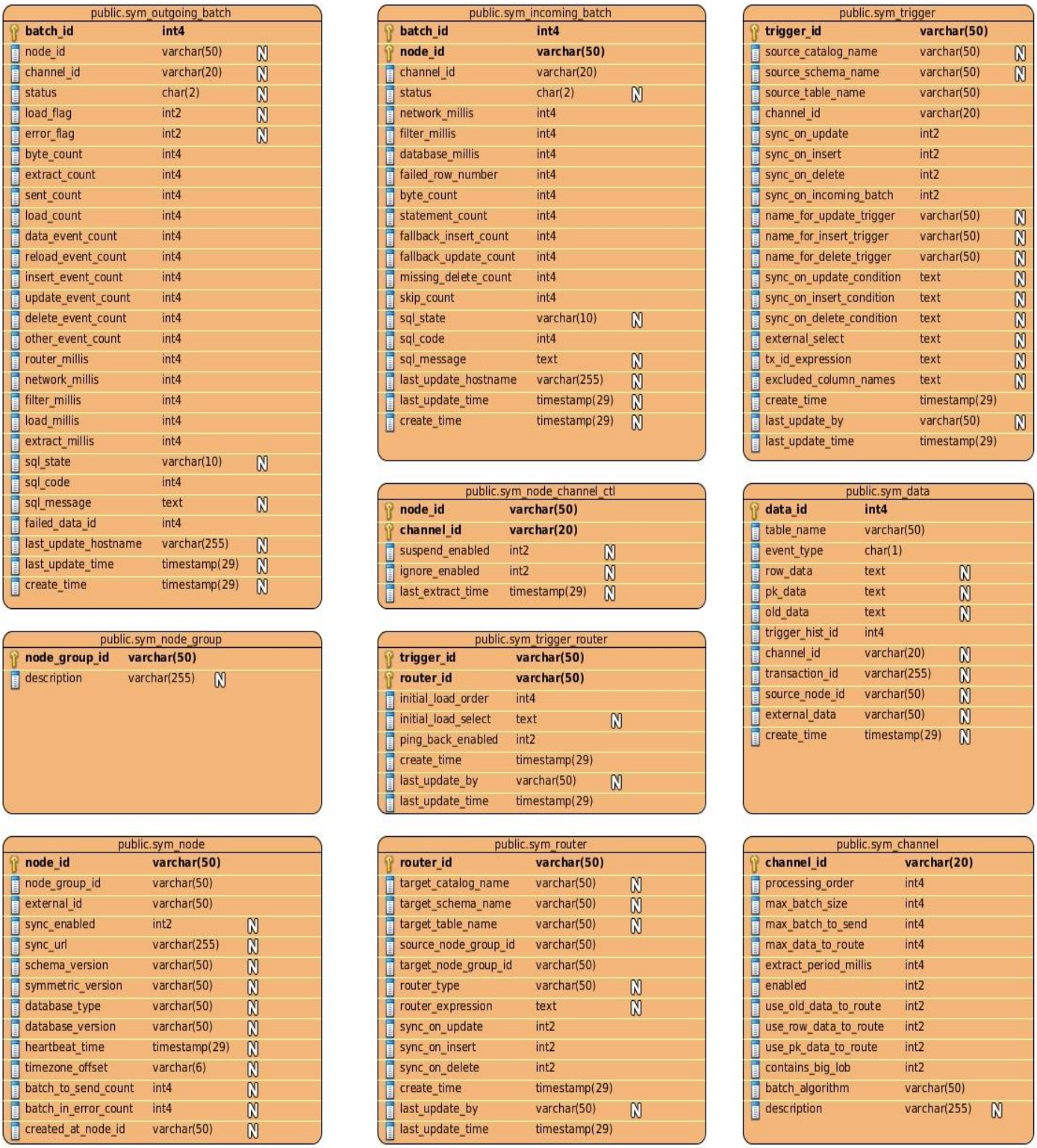

Figura 19 Modelo entidad-relación de las tablas que agrega SymmetricDS.

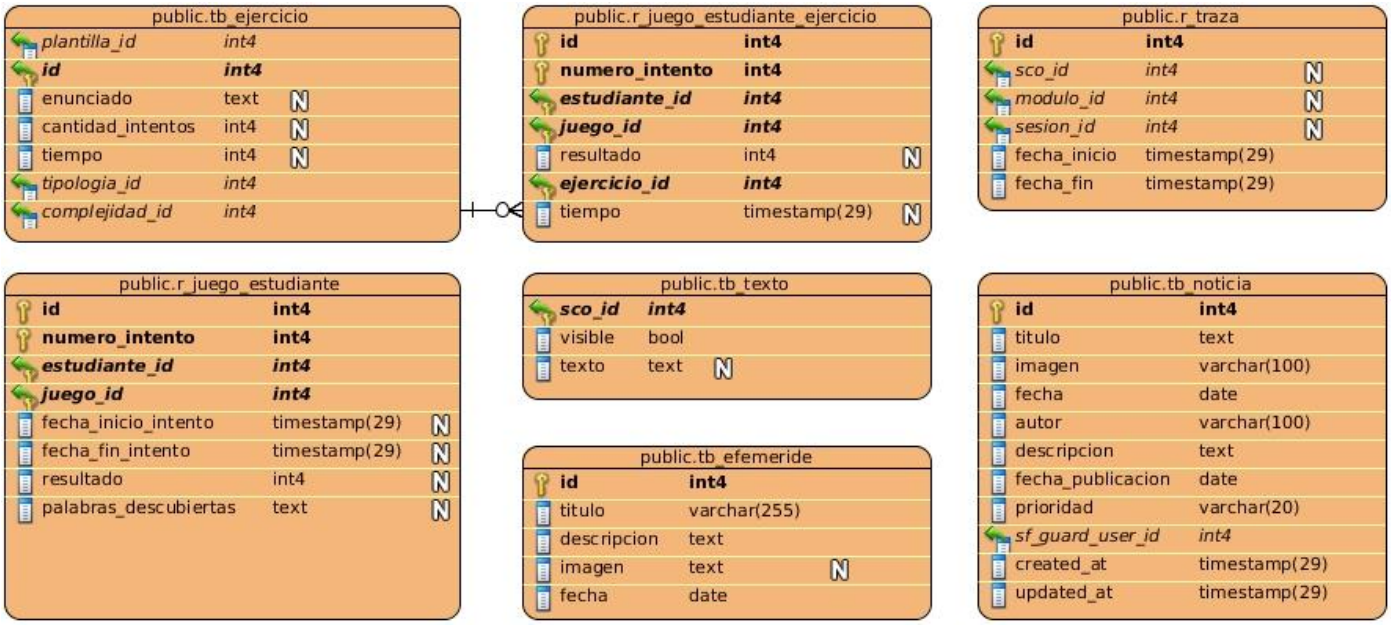

Figura 20 Modelo entidad-relación de las tablas de la colección.

3.6 Conclusiones

En el capítulo, como resultado del flujo de trabajo análisis y diseño, se representaron los diagramas de clases del análisis, el modelo de diseño para cada caso de uso de la especificado anteriormente y el modelo entidad-relación. El framework Symfony brinda la posibilidad de utilizar el patrón arquitectónico MVC y varios patrones de diseño, enfocados fundamentalmente a la reutilización del código.

Capítulo IV: Implementación y pruebas.

Introducción:

En el presente capítulo se describen las características del funcionamiento de la herramienta de replicación SymmetricDS, además se muestra cómo va a ser desarrollada la aplicación, la cual se inicia con los artefactos derivados del análisis y diseño y continúa con la construcción del sistema en términos de componentes, obteniéndose los diagramas de despliegue y de componentes; también se presenta la fase de pruebas del producto final, en la cual se diseñan las pruebas de caja negra a través de la confección de los casos de prueba para garantizar la calidad del producto.

4.1 Funcionamiento del SymmetricDS

Para llevar a cabo la configuración de SymmetricDS se deben tener en cuenta algunos conceptos de vital importancia para su compresión. Primeramente se debe identificar los nodos implicados, que no son más que los servidores de base de datos, donde se ejecutarán instancias de la herramienta SymmetricDS. En el negocio de réplica se cuenta con varios nodos, de ellos, uno central donde serán guardados los datos de los estudiantes. Es importante y necesario 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 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 trasmisión, además puede especificar cambios definidos como: actualizar, eliminar o simples inserciones. Se definen tantos disparadores como tabla a replicar.

Después de conocer algunos de los conceptos fundamentales de SymmetricDS, a continuación se hace un análisis más profundo de cada uno de los archivos que deben ser cambiados para asegurar una configuración exitosa.

En primer lugar se deben cambiar los archivos:

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

La herramienta SymmetriDS cuenta con varias tablas de configuración, en las que se definen los detalles, las más importantes serán explicadas de manera que se pueda realizar una correcta configuración para su entorno de réplica.

Las tablas de configuración de SymmetricDS serán creadas inicialmente vacías y la configuración se puede agregar en cada una de las tablas correspondientes para esta operación, además se puede configurar a través de un script o archivo nombrado insert_sample.sql.

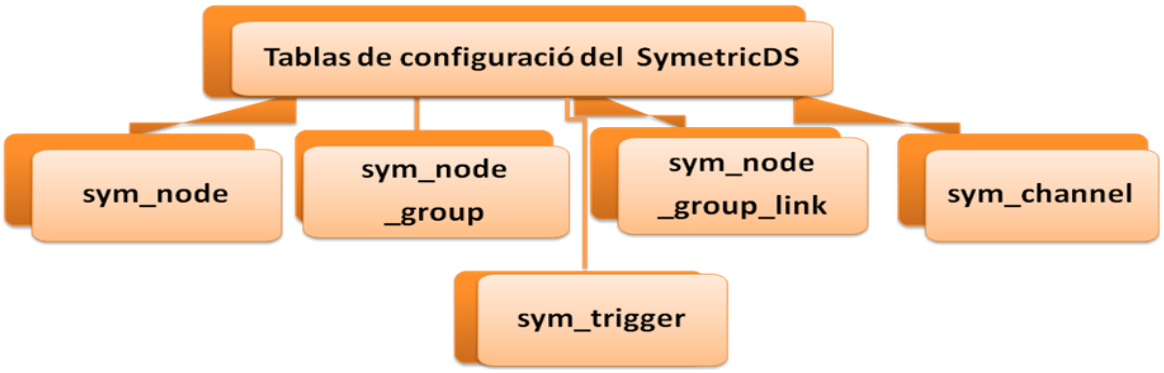

Figura 21 Principales tablas de configuración de SymmetricDS.

Para realizar la instalación después de haber configurado los archivos correspondientes se debe:

Paso 1: Necesita crear las tablas de SymmetricDS en la BD, estas guardarán la configuración para la sincronización. Debe entrar por consola hasta la carpeta sample y luego ejecutar los siguientes comandos.

```
../bin/sym -p centro.properties --auto-create
```

Al ejecutar estos se crean todas las tablas necesarias de SymmetricDS.

Paso 2: Aplicar la configuración realizada del nodo raíz en la cual se especifican los nodos, enlaces, canales, trigger de cada una de las tablas que se van a replicar. Se carga la configuración ejecutando los siguientes comandos.

```
../bin/sym -p centro.properties --run-sql insert_sample.sql
```

Paso 3: En cada uno de los clientes se debe crear las tablas de la base de datos clientes o institución, corriendo el script de la BD.

Paso 4: Se deben iniciar los servidores después de ubicarse en la consola en la carpeta sample, para el caso del nodo raíz debe ejecutar los siguientes comandos.

```
../bin/sym -p centro.properties --port 5800 –server
```

Y para iniciar el servidor de la institución debe ejecutar los siguientes comandos

```
../bin/sym -p int.properties --port 5800 –server
```

Se puede inicializar por otros puertos.

Paso 5: Cada uno de los nodos asociados al nodo central debe ser registrado para lograr recibir su carga inicial de forma que se pueda recibir y enviar datos al mismo. Para registrar la institución 1 se debe ejecutar la siguiente instrucción.

```
../bin/sym -p centro.properties --open-registration "int_1, 1"
```

Y de esta manera para cada uno de las instituciones asociadas al nodo central, especificando el grupo de nodo y el identificador externo.

4.2 Estándar de codificación

Los estándares de codificación son pautas que se utilizan en la escritura del código fuente, las mismas aseguran de que todos los programadores del proyecto mantengan un vocabulario común. La utilización de los mismos es muy ventajosa, ya que permite asegurar la legibilidad del código entre distintos programadores, le provee una guía para el encargado del mantenimiento o actualización del sistema, con código claro y bien documentado y además facilita la portabilidad entre plataformas y aplicaciones.

4.2.1 Convenciones de nombres (Funciones, Variables Globales, Clases)

Los nombres que se utilizan son cortos, y de forma abreviada, siempre que su contexto sea específicamente local y su lectura sea intuitiva. Para los nombres se establecen las siguientes reglas:

- ✓ **Funciones y métodos:** se utiliza la notación Camel que consiste en escribir los identificadores con la primera letra de cada palabra en mayúsculas y el resto en minúscula: EndOfFile. Las funciones y métodos se nombran usando el formato CamelCase (notación camello) que es una variante de la notación Camel donde la primera letra es minúscula.
- ✓ **Clases:** Los nombres de las clases comienzan con el sufijo del módulo en este caso del módulo Resultados seguidos del formato CamelCase ejemplo reMenuPrincipal.

  Las clases que se generan en el Modelo-Entidad-Relación comienzan con:

  - Tb_: las clases de entidad.
  - Nom_: las clases de nomencladores.
  - R_: las clases que representan relaciones muchos-a-muchos.

4.2.2 Indentación y espacios en blanco

- ✓ **Alineación y longitud de líneas**

Indentar con dos espacios, sin tabulador, para que cualquier editor de texto reconozca correctamente la indentación. Utilizar líneas entre 75-80 caracteres de longitud, de esta manera se manejan mejor algunas herramientas y terminales.

- ✓ **Líneas plegadas**

Cuando una expresión no cabe en una línea simple debido a su extensión se divide en más de una línea, siguiendo las siguientes precisiones:

- Dividir después de una coma.
- Dividir después de un operador.
- Alinear la nueva línea al inicio de la expresión en el mismo nivel que la línea anterior.

4.2.3 Comentarios

El uso de comentarios en línea se utiliza para facilitar la comprensión del código, sobre todo en procedimientos complejos. Los comentarios pueden ser con fin documental o para ayudar a la memoria del programador .Se recomienda utilizar el estilo (/* */).

Ejemplo de comentario:

```
/*
        *Comentario
 */
```

4.3 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, entre otros. El modelo de implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y como dependen los componentes unos de otros. El modelo de implementación define una jerarquía, tal y como se ilustra en la figura. (38)

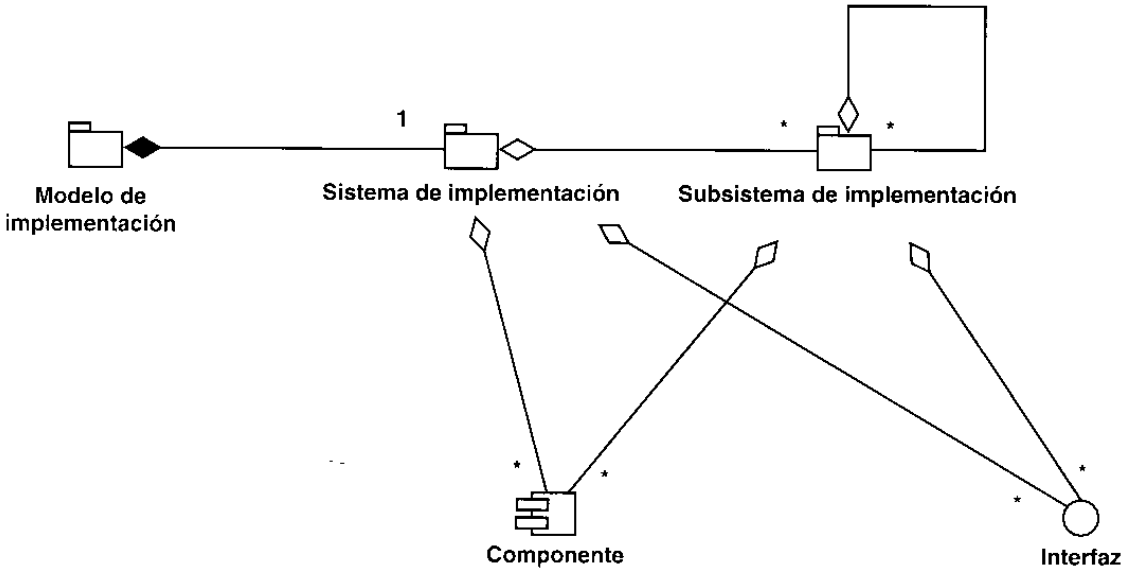

Figura 22 El modelo de implementación.

4.3.1 Diagrama de despliegue

Un diagrama de despliegue representa las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución. Se puede decir además que, un diagrama de despliegue es una colección de nodos y arcos; donde cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware similar. Se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño. (37)

A continuación se muestra como está distribuido los nodos para este producto:

El diagrama de despliegue está compuesto por dos nodos, donde en el nodo servidor se tiene una computadora como servidor central donde reside el servidor web en conjunto con el de base de datos y también va a estar instalada la herramienta de replicación servidor de SymmetricDS. En el nodo PC_Cliente va a estar instalada la herramienta de replicación cliente de SymmetricDS y la aplicación para que el usuario pueda interactuar con ella.

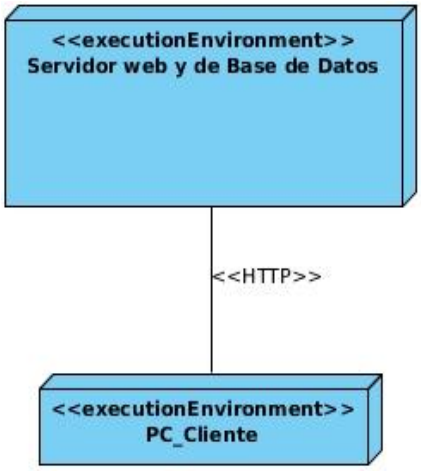

Figura 23 Diagrama de despliegue.

4.3.2 Diagrama de componente

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable.

Los componentes representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes, bibliotecas cargadas dinámicamente,

entre otros. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente. (44)

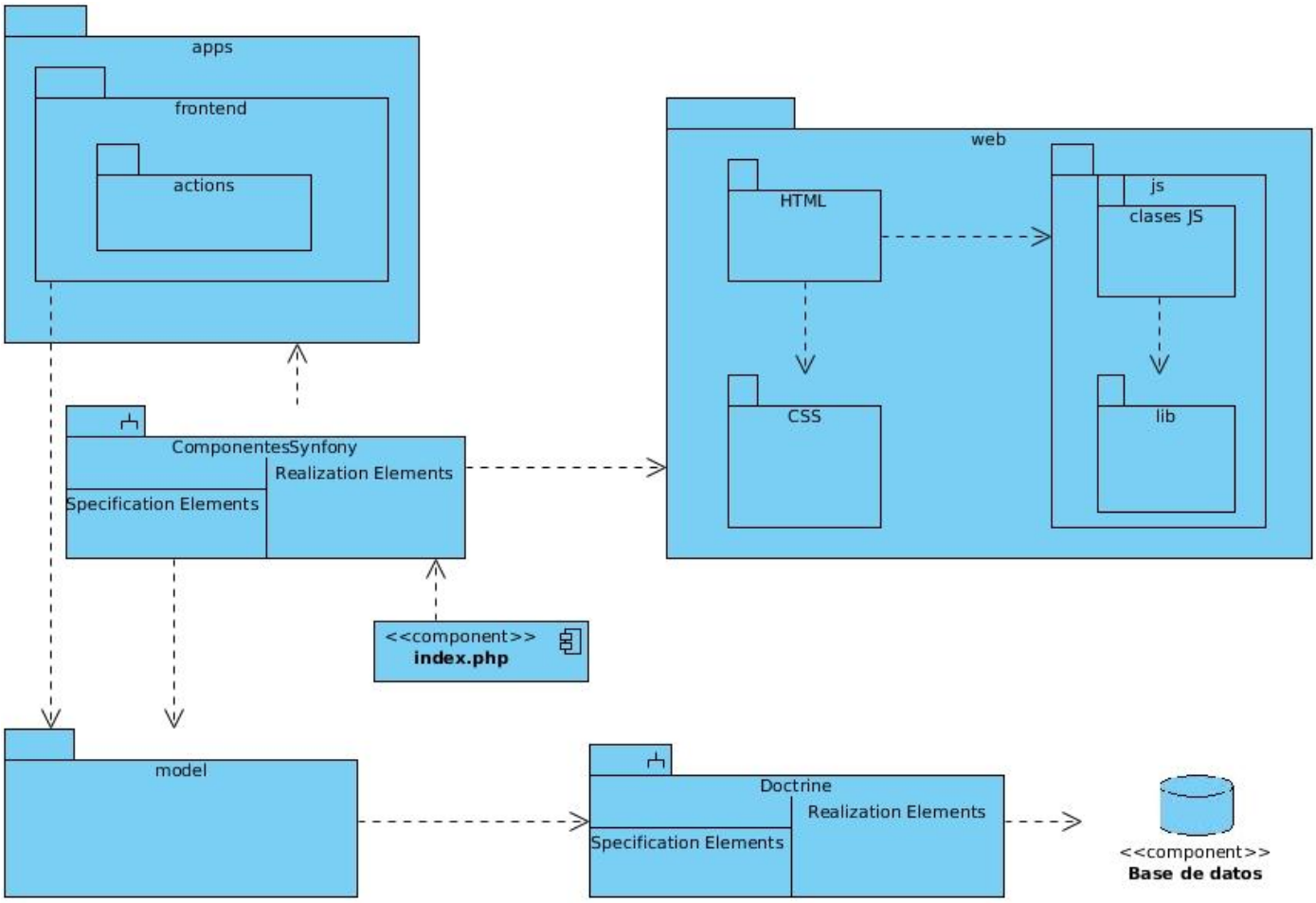

Figura 24 Diagrama de componentes del sistema réplica.

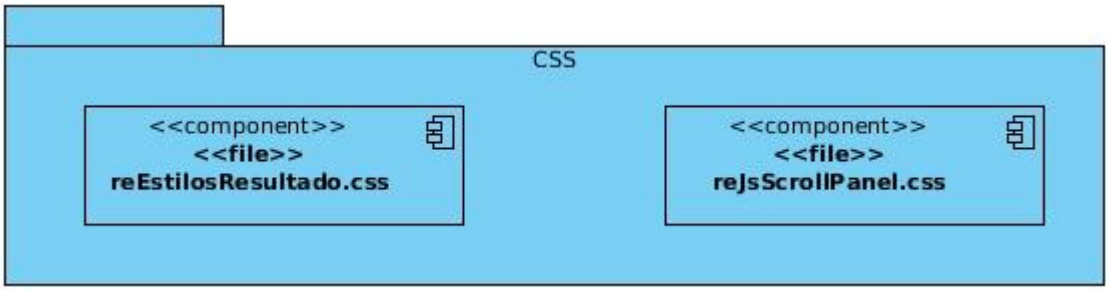

Figura 25 Diagrama de componentes del paquete CSS.

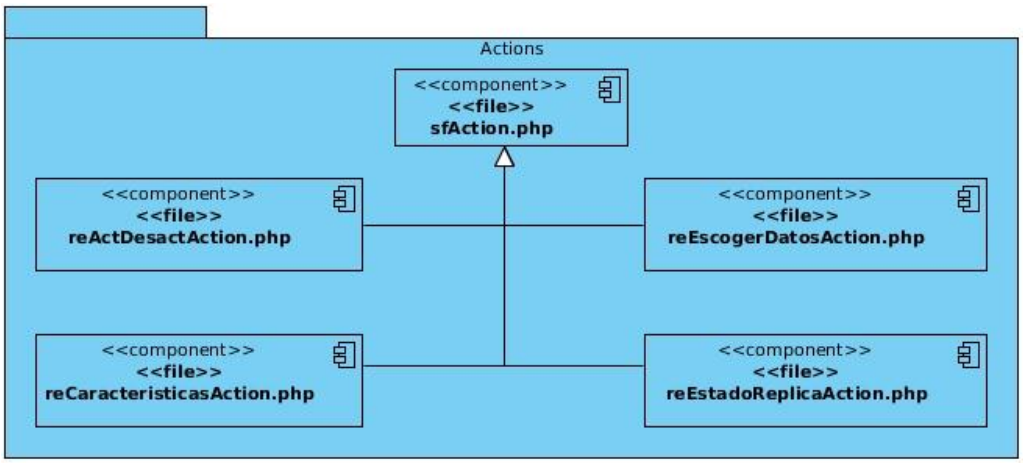

Figura 26 Diagrama de componentes del paquete Actions.

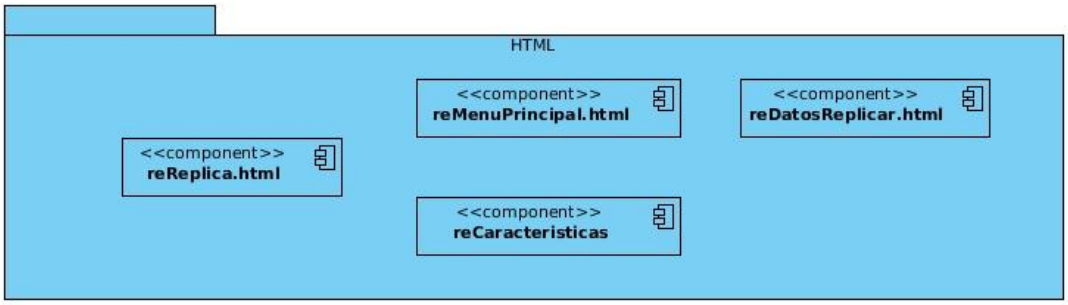

Figura 27 Diagrama de componentes del paquete HTML.

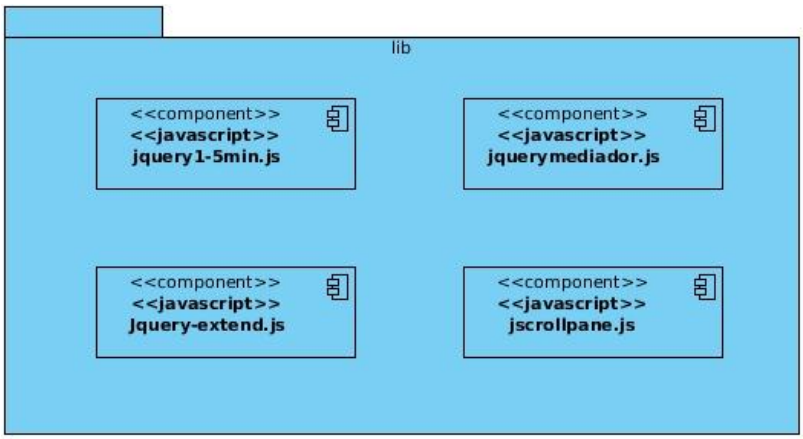

Figura 28 Diagrama de componentes del paquete Lib.

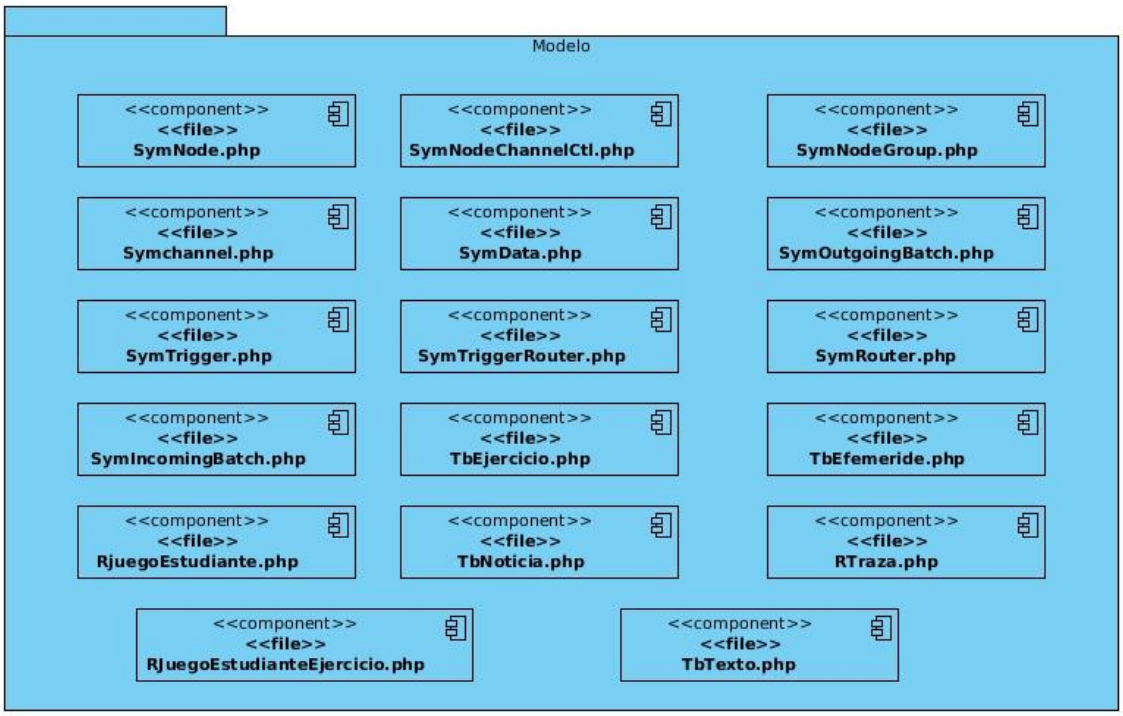

Figura 29 Diagrama de componentes del paquete Modelo.

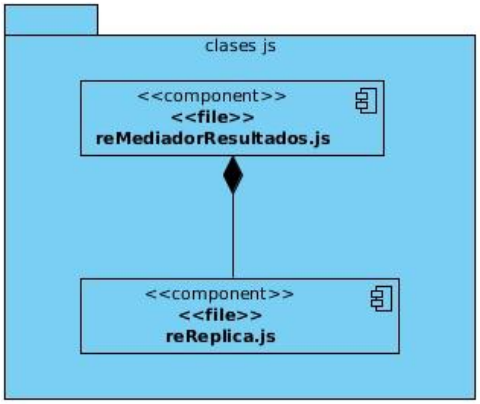

Figura 30 Diagrama de componentes del paquete vista

4.4 Pruebas de software

Una vez generado el código fuente, el software como producto puede tener defectos o fallos, por lo que es necesario probarlo para descubrir y corregir la mayor cantidad de errores posibles, para así garantizar un producto con calidad.

Existen diferentes tipos de prueba de software, entre las que se encuentran:

- ✓ Pruebas unitarias: La prueba de unidad se centra en el módulo, usando la descripción del diseño detallado como guía, se prueban los caminos de control importantes con el fin de descubrir errores dentro del ámbito del módulo.
- ✓ Pruebas de requerimientos: Un asunto de gran importancia es asegurar la corrección, coherencia y exactitud de los requisitos. Se revisará el documento de especificación de requerimientos, con la lista de chequeo general del documento y la lista de chequeo de requerimientos.
- ✓ Pruebas de integración: Las pruebas de integración se llevan a cabo durante la construcción del sistema, involucran a un número creciente de módulos y terminan probando el sistema como conjunto.
- ✓ Pruebas de aceptación: Las pruebas de aceptación, son las que se hacen con los clientes y define su aceptación del sistema. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario.
- ✓ Pruebas de Sistema: En el diseño de sistemas, estos no se toman como sistemas completos ni se prueban como sistemas únicos, razón por la cual deben hacerse pruebas parciales y del sistema en su totalidad.
- ✓ Pruebas Funcionales: Se asegura el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados.

Para probar la solución se escogen las pruebas funcionales, ya que este tipo de pruebas están basadas en técnicas de caja negra, la cual verifica la aplicación (y sus procesos internos) mediante la interacción con la aplicación y analizar la salida. (45)

Pruebas de caja negra

Permiten al ingeniero de software derivar conjuntos de condiciones de entrada que ejercitan por completo todos los requisitos funcionales de un programa.

Las pruebas de caja negra tratan de encontrar errores en las siguientes categorías:

1. Funciones incorrectas o faltantes.
2. Errores de interfaz.
3. Errores en estructura de datos o en acceso a bases de datos externas.
4. Errores de comportamiento o desempeño.

5. Errores de inicialización y término. (36)

Para la solución implementada se definen casos de prueba para cada caso de uso, para comprobar que el sistema cumple con las funcionalidades descritas.

En las pruebas funcionales realizadas a la herramienta se encontraron un total de 15 no conformidades de las tres iteraciones realizadas, las cuales fueron resueltas para brindarle mayor calidad al producto. Los detalles del resultado se muestran en la siguiente tabla:

Tipos No conformidades	1ra iteración	2da iteración	3ra iteración
Formato		1	
Error técnico			
Ortografía	1		
Redacción	4		
Validación	1		
Opciones que no funcionan			
Error de interfaz		1	1
Error de idioma			
Funcionalidad			
Excepciones	3	2	
Otros			1
Recomendación			
Total	9	4	2
Alta	3	2	0
Media	1	1	1
Baja	5	1	1

Tabla 13 Resumen de no conformidades.

4.5 Conclusiones

El presente capítulo es la culminación del desarrollo de la herramienta de réplica para la colección El Navegante, en el mismo se implementó la propuesta anteriormente planteada y luego se probó la solución implementada, esto arrojó como resultado que se obtuviera un producto de calidad que cumple con los requerimientos.

Conclusiones generales

Como resultado del trabajo desarrollado se arribaron a las siguientes conclusiones:

- ✓ De las soluciones similares existentes estudiadas se determinó que SymmetricDS cumple con el entorno de réplica y el modelo de distribución de datos necesarios para darle solución al problema, y cumplen con los requerimientos de licencia definidos para la colección, por lo que se escogió como herramienta que realizará la réplica de los datos. Además se utilizaron los lenguajes definidos anteriormente para mantener un control de los datos de forma sencilla y agradable a la vista del usuario.
- ✓ A partir de la integración de SymmetricDS con El Navegante se logró replicar los datos de la colección, permitiendo activar o desactivar la réplica, ver el estado de los datos replicados y manejar los posibles datos a replicar.
- ✓ Se logró centralizar la información generada por la interacción de los estudiantes con los productos hacia un único servidor, facilitando el trabajo de los maestros en la generación de reportes, el manejo de las estadísticas y la revisión de los registros de los estudiantes en la aplicación.
- ✓ Mediante el análisis, diseño e implementación de la herramienta para la replicación de datos, fue posible obtener una aplicación que replica los datos generados por la interacción con la colección y permite manejar los datos replicados y los posibles datos a replicar, facilitando el trabajo de los profesores en la generación de reportes y el manejo de las estadísticas de los estudiantes.
- ✓ La solución propuesta fue probada utilizando pruebas de caja negra, basadas en los casos de prueba, aportando una mayor calidad al producto. Esto permitió erradicar deficiencias encontradas y entregar una herramienta que cumple con las funcionalidades requeridas por el cliente.

Recomendaciones

A los desarrolladores de la colección El Navegante se les recomienda:

Realizar mejoras mejoras como extender las funcionalidades de la herramienta para obtener un mayor control sobre la réplica de los datos.

Además se recomienda hacer de esta herramienta un plugin que permita instalarse de forma rápida y sin necesidad de realizar gran cantidad de configuraciones para posibilitar su reusabilidad en posibles futuros proyectos.

Realizar un estudio del tiempo de reacción de la herramienta SymmetricDS bajo determinadas condiciones, lo cual será un medidor más preciso para controlar el tiempo de réplica.

Referencias bibliográficas

1. **Valdes, Damián Pérez.** Pensamiento imaginativo. [En línea] 2008 de 10 de 6. 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] www.emprendedoresucu.com/diccionario.htm.

3. **Date, C. J.** *Introduction to Database Systems.* 2003.

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

5. **y Baños Fernández, Kenny y Rodríguez García, Eddy Ernesto.** *Sistema de réplica para la intranet Corporativa y el sitio en Intranet de PDVSA.* Cuidad de La Habana : s.n., 2009.

6. **De Ávila León, Ramón Ernesto y Rodríguez Barani, Lorián.** *Diseño e implementación del módulo de réplica de datos para "GeForza".* 2010.

7. **Instituto Superior de Calkini.** ITESCAM. [En línea] http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r11294.PDF..

8. **Buretta, M.** *Data Replication Tools and Techniques for managing distributed information.* s.l. : Wiley, 1997.

9. **Monge, Raúl.** *Base de datos Distribuidas.* Valparaiso : s.n., 2005.

10. **Nodalis Ricelda Nodal, González Dunia Ortiz Nodal.** *Paquete de instalación de réplicas en servidores PostgreSQL.* Habana : s.n., 2009.

11. **Searchnetworking.** Searchnetworking. *master/slave.* [En línea] 2006. http://searchnetworking.techtarget.com/definition/master-slave.

12. **Oracle Documentation.** Oracle. [En línea] http://docs.oracle.com/cd/E16338_01/server.112/e10706/repmaster.htm#BGBHCCDJ.

13. **Keating, Brian.** DatabaseSpecialists. [En línea] 2001. http://www.dbspecialists.com/files/presentations/mm_replication.html.

14. **Huacuja, Héctor Juaquín Fraire.** *Automatización del Diseño de la Fragmentación Horizontal en Bases de Datos Distribuidas.*

15. **Pupo Polaco, Rosell y González Pérez, Yenier.** *Implementación del componente réplica de base se datos para akademos V2.0.* Cuidad de la Habana : UCI, 2009.

16. **EMC.** EMC. [En línea] http://argentina.emc.com/products/detail/software/mirrorview.htm..

17. —. EMC. [En línea] http://argentina.emc.com/products/detail/software/recoverpoint.htm .

18. **Ppgfoundry.** Ppgfoundry. *Pyreplica.* [En línea] http://pgfoundry.org/projects/pyreplica..

19. **PgFoundry.** PgFoundry. *PgCluster.* [En línea] http://pgfoundry.org/projects/pgcluster/.

20. **Symmetricds.** [En línea] http://symmetricds.codehaus.org/..

21. **Yadima Morales Hernández, Joel Tamayo Vega.** *Desarrollo del módulo General de la colección El Navegante.* Habana : s.n., 2011.

22. **Departamento de Sistemas Informáticos y Computación Universidad Politécnica de Valencia.** *Rational Unified Process.* Valencia : s.n.

23. **Osmosis Latina.** Osmosis Latina. [En línea] http://www.osmosislatina.com/lenguajes/uml/.

24. **Morales, Freddy.** Slideshare. [En línea] Instituto Nacional de Estadistica e Informática, 2012. http://www.slideshare.net/FreddyMorales1/herramientas-case-11216676.

25. **Software y Descargas.** Free Download manager. *Software y Descargas.* [En línea] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p.

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

27. **Pérez, Javier Eguiluz.** *CSS 2.1.* 2008.

28. **Pérez, Javier Eguiluz.** *Introducción a JavaScript.* 2008.

29. **Hinostroza, R. R.** linuxcentro. [En línea] http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP.

30. **Vegas, Jesús.** Departamento de informática. [En línea] Universidad de Valladolid. http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node20.html.

31. **Pino, José Luis López.** lopezpino. *Servidores web más usados.* [En línea] http://lopezpino.es/2010/07/30/servidores-web-mas-usados/.

32. **No hay Limites.** No hay Limites. [En línea] http://www.nohaylimites.com/?p=141.

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

34. Computer Audio Video Systems Integrator. [En línea] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/..

35. **Postgresql.** Postgresql. [En línea] http://www.postgresql.org/about/press/presskit90.html.es..

36. **Pressman, Roger S.** *Ingeniaria del Software.*

37. **Sommerville, Ian.** *Ingenieria del Software.* s.l. : Pearson Education, 2005.

38. **Ivar Jacobson, Grady Booch,James Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Addison Wesley, 2000.

39. **Ivar Jacobsion, James Rumbaugh, Grady Booch.** *El lenguaje unificado de modelado. Manuel de referencia.* s.l. : Addison Wesley.

40. **Dofactory.** Dofactory. [En línea] 2012. http://www.dofactory.com/Patterns/Patterns.aspx.

41. **Scribd.** Scribd. [En línea] http://www.scribd.com/doc/7930106/Modelo-de-Analisis-Saul-Cuzcano-Quintin.

42. **Reynoso, Carlos Billy.** *Introducción a la Arquitectura de Software.* Buenos Aires : s.n., 2004.

43. **Mühlrad, Daniel.** *Patrones de diseño.* 2008.

44. **Craig, Larman.** *UML y patrones.* s.l. : Prentice Hall, 2004. ISBN 978-8420534381.

45. **Torres, Paticio Letelier.** Scribd. [En línea] Universidad Politécnica de Valencia. http://es.scribd.com/doc/19317161/57/Diagrama-de-Componentes.

46. **Jiménez, Arianna Rodríguez.** *Proceso de evaluación de la documentación para el servicio de Pruebas Funcionales a nivel de Sistema que ofrece el Grupo de Calidad del centro FORTES.* Habana : s.n., 2011.

Bibliografía

1. **Valdes, Damián Pérez.** Pensamiento imaginativo. [En línea] 2008 de 10 de 6. 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] www.emprendedoresucu.com/diccionario.htm.

3. **Date, C. J.** *Introduction to Database Systems.* 2003.

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

5. **y Baños Fernández, Kenny y Rodríguez García, Eddy Ernesto.** *Sistema de réplica para la intranet Corporativa y el sitio en Intranet de PDVSA.* Cuidad de La Habana : s.n., 2009.

6. **De Ávila León, Ramón Ernesto y Rodríguez Barani, Lorián.** *Diseño e implementación del módulo de réplica de datos para "GeForza".* 2010.

7. **Instituto Superior de Calkini.** ITESCAM. [En línea] http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r11294.PDF..

8. **Buretta, M.** *Data Replication Tools and Techniques for managing distributed information.* s.l. : Wiley, 1997.

9. **Monge, Raúl.** *Base de datos Distribuidas.* Valparaiso : s.n., 2005.

10. **Nodalis Ricelda Nodal, González Dunia Ortiz Nodal.** *Paquete de instalación de réplicas en servidores PostgreSQL.* Habana : s.n., 2009.

11. **Searchnetworking.** Searchnetworking. *master/slave.* [En línea] 2006. http://searchnetworking.techtarget.com/definition/master-slave.

12. **Oracle Documentation.** Oracle. [En línea] http://docs.oracle.com/cd/E16338_01/server.112/e10706/repmaster.htm#BGBHCCDJ.

13. **Keating, Brian.** DatabaseSpecialists. [En línea] 2001. http://www.dbspecialists.com/files/presentations/mm_replication.html.

14. **Huacuja, Héctor Juaquín Fraire.** *Automatización del Diseño de la Fragmentación Horizontal en Bases de Datos Distribuidas.*

15. **Pupo Polaco, Rosell y González Pérez, Yenier.** *Implementación del componente réplica de base se datos para akademos V2.0.* Cuidad de la Habana : UCI, 2009.

16. **EMC.** EMC. [En línea] http://argentina.emc.com/products/detail/software/mirrorview.htm..

17. —. EMC. [En línea] http://argentina.emc.com/products/detail/software/recoverpoint.htm .

18. **Ppgfoundry.** Ppgfoundry. *Pyreplica.* [En línea] http://pgfoundry.org/projects/pyreplica..

19. **PgFoundry.** PgFoundry. *PgCluster.* [En línea] http://pgfoundry.org/projects/pgcluster/.

20. **Symmetricds.** [En línea] http://symmetricds.codehaus.org/..

21. **Yadima Morales Hernández, Joel Tamayo Vega.** *Desarrollo del módulo General de la colección El Navegante.* Habana : s.n., 2011.

22. **Departamento de Sistemas Informáticos y Computación Universidad Politécnica de Valencia.** *Rational Unified Process.* Valencia : s.n.

23. **Osmosis Latina.** Osmosis Latina. [En línea] http://www.osmosislatina.com/lenguajes/uml/.

24. **Morales, Freddy.** Slideshare. [En línea] Instituto Nacional de Estadistica e Informática, 2012. http://www.slideshare.net/FreddyMorales1/herramientas-case-11216676.

25. **Software y Descargas.** Free Download manager. *Software y Descargas.* [En línea] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p.

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

27. **Pérez, Javier Eguiluz.** *CSS 2.1.* 2008.

28. **Pérez, Javier Eguiluz.** *Introducción a JavaScript.* 2008.

29. **Hinostroza, R. R.** linuxcentro. [En línea] http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP.

30. **Vegas, Jesús.** Departamento de informática. [En línea] Universidad de Valladolid. http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node20.html.

31. **Pino, José Luis López.** lopezpino. *Servidores web más usados.* [En línea] http://lopezpino.es/2010/07/30/servidores-web-mas-usados/.

32. **No hay Limites.** No hay Limites. [En línea] http://www.nohaylimites.com/?p=141.

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

34. Computer Audio Video Systems Integrator. [En línea] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/..

35. **Postgresql.** Postgresql. [En línea] http://www.postgresql.org/about/press/presskit90.html.es..

36. **Pressman, Roger S.** *Ingeniaria del Software.*

37. **Sommerville, Ian.** *Ingenieria del Software.* s.l. : Pearson Education, 2005.

38. **Ivar Jacobson, Grady Booch,James Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Addison Wesley, 2000.

39. **Ivar Jacobson, James Rumbaugh, Grady Booch.** *El lenguaje unificado de modelado. Manuel de referencia.* s.l. : Addison Wesley.

40. **Dofactory.** Dofactory. [En línea] 2012. http://www.dofactory.com/Patterns/Patterns.aspx.

41. **Scribd.** Scribd. [En línea] http://www.scribd.com/doc/7930106/Modelo-de-Analisis-Saul-Cuzcano-Quintin.

42. **Reynoso, Carlos Billy.** *Introducción a la Arquitectura de Software.* Buenos Aires : s.n., 2004.

43. **Mühlrad, Daniel.** *Patrones de diseño.* 2008.

44. **Craig, Larman.** *UML y patrones.* s.l. : Prentice Hall, 2004. ISBN 978-8420534381.

45. **Torres, Paticio Letelier.** Scribd. [En línea] Universidad Politécnica de Valencia. http://es.scribd.com/doc/19317161/57/Diagrama-de-Componentes.

46. **Jiménez, Arianna Rodríguez.** *Proceso de evaluación de la documentación para el servicio de Pruebas Funcionales a nivel de Sistema que ofrece el Grupo de Calidad del centro FORTES.* Habana : s.n., 2011.

47. **Rolando Alfredo Hernández León, Sayda Coello González.** *El proceso de investigación científica.* Ciudad Habana : Universitaria, 2011.

48. **Biblioteca Digital.** Biblioteca Digital. [En línea] 2011. http://bdigital.bnjm.cu/bases.html.

49. **Bases de datos.** Bases de datos. [En línea] 2010. http://www.basesdedatos.org/.

50. **Dolores Lankenau, Cleopatra Garza.** Sistema Tecnológico de Monterrey. [En línea] Dirección de Desarrollo Académico, ITESM Campus Monterrey, 2012. http://www.mty.itesm.mx/rectoria/dda/usols/creacion.htm.

51. **Microsoft.** Microsoft SQL Server. [En línea] 2008. http://technet.microsoft.com/es-es/library/ms152485%28v=sql.90%29.aspx.

52. **BD1 Informática.** BD1 Informática Software y Consultoría. [En línea] 2010. http://www.db1.com.br/novo/esp/replica-datos.php.

53. **Code.** Code. [En línea] 2010. http://www.hostingcode.net/e/foro/mysql/51-replicacion-maestro-esclavo-en-mysql.html.

54. **O´Reilly.** *Head First Design Patterns.* 2009.

55. —. *Head First HTML with CSS and XHTML.* 2009.

56. —. *Head First JavaScript.* 2009.

57. —. *Head First PHP and MySQL.* 2009.

58. —. *Head First SQL.* 2009.

59. **Borges, Rasiel Aponcio.** *Solución de réplica de base de datos multi-maestra asincrónica.* Habana : s.n., 2012.

60. **Openalfa.** Openalfa. [En línea] 2012. http://www.openalfa.com/index.php/symmetricds/symmetricds-introduccion.

61. **Framework Studio.** Vico.org. [En línea] 2000. http://www.vico.org/pages/PatronsDisseny.html.

62. **Bear Bibeault, Yehuda Katz.** *jQuery in Action.* 2010.