UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 5

Centro de Consultoría y Desarrollo de Arquitecturas Empresariales

Sincronización de datos del proceso de réplica de estructura para el Replicador de datos REKO

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

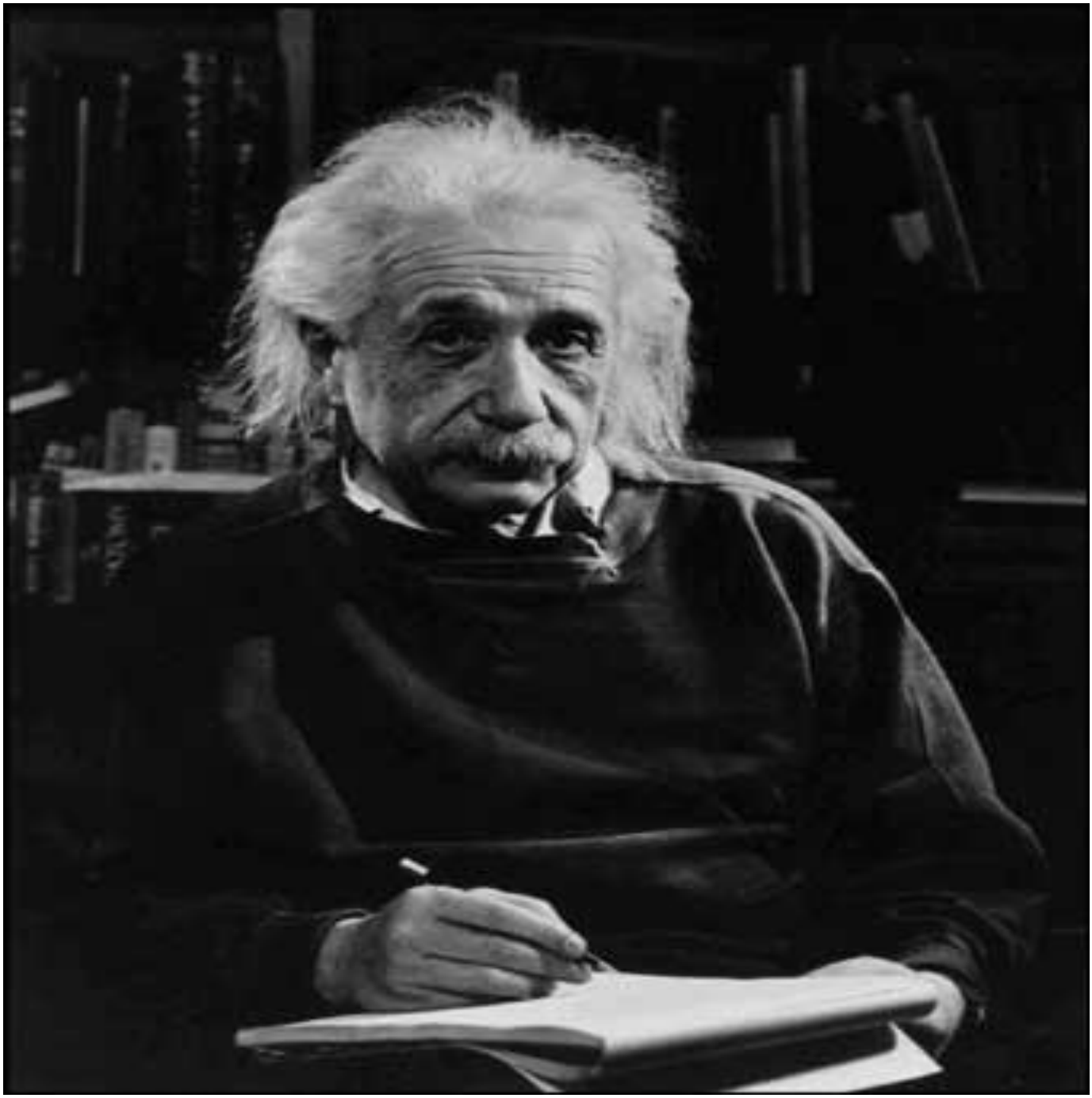
Autor (es): Enrique Basto Muguercia

Luis Angel Carbonel Hidalgo

Tutor (es): Ing. Gloria Raquel Leyva Jerez

La Habana, Junio del 2015

"Año 57 de la Revolución"



Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.

Albert Einstein.

AGRADECIMIENTOS

Luis Ángel:

A mis "primos-tíos" Yodeysi y Rafael por sus excelentes consejos por hacer más ameno el paso por la universidad.

A mis compañeros de aula por cantarme las felicidades en mis cumpleaños y en especial a mi compañero de tesis.

A mi tutora Gloria que es una profesional incansable, por su ayuda, dedicación y sus recomendaciones. Al equipo de REKO por su buen trabajo en los cortes.

A Lisi por brindarme su amor y su apoyo incondicional, por calmarme cuando los problemas rondan mi cabeza, por ser tan especial. Te quiero mucho Lisi, jamás te olvidaré.

Enrigue:

A mi mama y mi papá por todo el esfuerzo realizado para hacer que este sueño se convirtiera en realidad.

A mi novia Yuleidis que siempre ha estado conmigo en los momentos difíciles que he pasado a lo largo de toda la carrera.

A mi hermano Alexander por impulsarme a ser un ejemplo a seguir.

A mi abuelita querida que siempre me brinda su cariño y amor.

A mi tutora Gloria por dedicarnos una buena parte de su tiempo en ayudarnos a hacer posible que este sueño fuera realidad.

A mis compañeros de aula que se convirtieron en una familia para mí y me han brindado su afecto y cariño durante todos estos años que he pasado en la universidad.

A todos los que contribuyeron de una forma u otra a que pudiese llegar este momento.

DEDICATORIA

Luis Ángel:

A mi papá por todo su esfuerzo y dedicación, por ayudarme en los tiempos difíciles y apoyarme en el sueño de graduarme como ingeniero. Por brindarme su ejemplo en el afán de convertirme en un hombre independiente y luchar en lo que creo.

A mi mamá por su dedicación y cariño, por brindarme su comprensión de madre que es tan buena en los tiempos difíciles, para ella siempre soy su niño aunque ya tenga 24 años.

A mi hermano Jorge Daniel por hacer bien su papel de hijo cuando yo no estoy en casa y darle mucho cariño a nuestra madre por los dos.

A mi abuela Albertina que siempre me recibe con los brazos abiertos y le corren las lágrimas cuando salgo de viaje.

Enrique:

A mi mamá.

A mi papá.

A mi novia Yuleidis.

DECLARACIÓN JURADA DE AUTORÍA

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

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

_____
Luis Angel Carbonel Hidalgo

Autor

_____
Enrique Basto Muguercia

Autor

_____
Ing. Gloria Raquel Leyva Jerez

Tutora

RESUMEN

En la presente investigación se pretende mejorar la versión del Replicador REKO desarrollado en el Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE) ubicado en la Universidad de las Ciencias Informáticas (UCI).

El replicador anteriormente sólo era capaz de realizar sincronización de datos del proceso de réplica de datos sin contemplar la sincronización de estructura, por lo que se generaban errores que provocaban la pérdida de información e inconsistencias en las bases de datos distribuidas. El presente trabajo de diploma extiende las funcionalidades del módulo de sincronización del Replicador de datos REKO para realizar la sincronización de datos del proceso de réplica de estructura, que permite homogeneizar las bases de datos que intervienen en el proceso de réplica de estructura en ambientes distribuidos. La solución basa su funcionamiento en aplicar los cambios de estructuras encontrados durante la comparación de las bases de datos que intervienen en un proceso de réplica de estructura. Permite sincronizar de manera automática y manual las estructuras de las bases de datos involucradas en la réplica, evitando inconsistencias que impidan su correcto funcionamiento.

Se describe la problemática inicial que dio origen a la solución desarrollada, el estudio de otros replicadores de datos con características similares, así como el ambiente de desarrollo. Con el empleo de la metodología Proceso Unificado Ágil (AUP, *Agile Unified Process*, por sus siglas en inglés) se describe el diseño e implementación de las funcionalidades, el cual fue desarrollado completamente con herramientas *Open Source* y bibliotecas de clases con licencias gratuitas.

Palabras Claves: bases de datos distribuidas, réplica, sincronización.

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La tendencia actual a la globalización tecnológica exige de los sistemas y aplicaciones informáticas prestaciones que van más allá de lo alcanzable por cualquier ordenador aislado, por muy potente que este sea. Esto hace que las aplicaciones distribuidas se conviertan en un modelo generalizado desde las más extendidas arquitecturas cliente-servidor hasta los sistemas en clúster que ofrecen la imagen de una única máquina. Su uso ha hecho cada vez más común el empleo de sistemas de bases de datos distribuidas (BDD) que presentan como ventaja tener un procesamiento autónomo lo cual indica que pueden llevar a cabo operaciones locales o distribuidas.

La homogeneidad de los datos en un sistema de BDD es un aspecto de vital importancia que proporciona la igualdad entre las estructuras y datos de las base de datos (BD) que conforman los sistemas distribuidos. Para garantizarla se hace frecuente el uso de la sincronización de datos que utilizan los replicadores de datos.

REKO es un software de réplica de datos entre BDD desarrollado en la Universidad de las Ciencias Informáticas (UCI) por un grupo perteneciente al Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE), que pretende brindar una herramienta multiplataforma que le permita al usuario con el menor esfuerzo cubrir las necesidades fundamentales de replicación de datos. Cuenta con un módulo de sincronización que es capaz de realizar el intercambio de operaciones y la sincronización de datos entre las BD que intervienen en un proceso de réplica de datos. Permite además importar y exportar los datos de sincronización, garantizando la disponibilidad de los datos en caso de existir alguna BD remota sin conexión. Actualmente en este módulo al ejecutarse sentencias del tipo DML[1], el módulo es capaz de homogeneizar los datos almacenados en las estructuras existentes en las base de datos, teniendo en cuenta la configuración de réplica. Sin embargo si se ejecutan sentencias del tipo DDL[2] que modifican las estructuras de las bases de datos que se encuentran distribuidas, se pierde la homogeneidad entre ellas, debido a que estas acciones no se tienen en cuenta en el proceso de sincronización, provocando por ende la aparición de conflictos de estructura y datos, los cuales generan a su vez tablas o datos incompletos, inconsistencias entre las BD y la pérdida de la integridad de la información a replicar.

1 Lenguaje de manipulación de datos, Data Manipulation Language: insert, update y delete.

2 Lenguaje de definición de datos, Data Defination Languaje: create, alter y drop.

Partiendo de la problemática planteada anteriormente se formula el siguiente **problema a resolver**:

¿Cómo homogeneizar las bases de datos que intervienen en la réplica de estructura en ambientes distribuidos, para minimizar errores durante el proceso de réplica?

Para llevar a cabo la investigación se tiene como **objeto de estudio** la sincronización en replicadores de datos, enmarcando el **campo de acción** a la sincronización en el proceso de réplica de estructura del Replicador de datos REKO.

Para dar solución a la problemática descrita se define como **objetivo general**:

Extender las funcionalidades del módulo de sincronización del Replicador de datos REKO para realizar la sincronización de datos del proceso de réplica de estructura, que permita homogeneizar las bases de datos que intervienen en el proceso de réplica de estructura.

Para dar cumplimiento al objetivo general se definieron las siguientes **tareas de investigación**:

- Realización del marco conceptual para precisar los principales conceptos que se emplean en la investigación.
- Elaboración del estado del arte sobre las tendencias actuales en la utilización de sistemas de software, los métodos usados en el intercambio y la sincronización entre bases de datos para sintetizar los resultados alcanzados en la revisión bibliográfica e indagaciones realizadas relacionadas con el tema.
- Selección de metodologías a utilizar, para definir los métodos y técnicas necesarias que guiarán el desarrollo.
- Descripción de las herramientas, tecnologías y lenguajes a utilizar para definir el ambiente de desarrollo.
- Definición de los requisitos funcionales y no funcionales para identificar las capacidades que tienen que ser alcanzadas por el sistema para cumplir los objetivos trazados.
- Descripción de la arquitectura que soporta la implementación de las funcionalidades.
- Realización del análisis y diseño de las funcionalidades a extender en el módulo de sincronización para describir los artefactos.
- Implementación de las funcionalidades que dan cumplimiento a los requisitos identificados.

- Realización de pruebas a las funcionalidades extendidas en entornos de producción para valorar la calidad del producto implementado.
- Valoración de los resultados obtenidos a partir de las pruebas realizadas.

La idea a defender en la investigación se plantea de la siguiente manera: con la extensión de las funcionalidades del módulo de sincronización del Replicador de datos REKO para realizar la sincronización de datos del proceso de réplica de estructura, se logrará la homogeneidad en las bases de datos que intervienen en el proceso de réplica de estructura, minimizando errores del proceso de réplica.

Los métodos empleados en la presente investigación se describen a continuación:

Métodos teóricos

- **Analítico-Sintético:** se realizó un análisis sobre las diferentes teorías que existen en torno al concepto de sincronización del proceso de réplica de estructura, permitiendo sintetizar el conocimiento y dar solución al problema planteado.
- **Histórico-Lógico:** hizo posible que mediante un estudio cronológico sobre el estado actual del desarrollo de herramientas de réplica de estructuras, se obtuviera la información necesaria que se aplicaría en la solución del problema.

Métodos empíricos

- **Consulta de la información en todo tipo de base:** hizo posible que mediante una minuciosa revisión bibliográfica, se definieran los elementos esenciales que permitieran la elaboración de todos los aspectos teóricos y conceptuales de la investigación.
- **Nivel estadístico**: mediante la estadística descriptiva, con el empleo de gráficos, tablas, porcientos, hizo posible la organización y regularización de la información obtenida.

Con la presente investigación se espera que el módulo de sincronización del Replicador de datos REKO cuente con las funcionalidades necesarias para realizar la sincronización de datos del proceso de réplica de estructura, que permita homogeneizar las bases de datos que intervienen en la réplica.

El presente documento consta de tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos.

El **Capítulo 1. Fundamentación teórica** presenta un marco conceptual donde se describen los principales conceptos que se tratan en la investigación. Se realiza un análisis del estado del arte

sobre replicadores de datos. Se describen la metodología de desarrollo de software, herramientas y lenguajes a emplear en el desarrollo de la solución.

El **Capítulo 2. Propuesta de solución** incluye la descripción, análisis y diseño de la solución que se propone para darle respuesta a la problemática planteada. Se especifican los requisitos funcionales y los no funcionales. Se realiza la descripción de los requisitos, patrones de diseño aplicados, así como los diagramas de paquetes, los diagramas de clases del diseño y el modelo de despliegue.

El **Capítulo 3. Implementación y pruebas** muestra la implementación de las funcionalidades. Se describen los diagramas de componentes. Se muestra el código fuente de las principales clases y se aplican pruebas al sistema para demostrar la robustez del mismo.

Capítulo 1. Fundamentación teórica

A continuación se elabora el marco teórico, donde se abordan los principales conceptos relacionados con la sincronización de datos del proceso de réplica de estructura, así como un estudio de replicadores actuales que realizan este proceso. Se describen además las herramientas, tecnologías y metodologías utilizadas para la resolución del problema planteado.

1.1 Marco conceptual

En esta sección se ofrecen algunos de los principales conceptos que se consideran necesarios para una mejor comprensión de la investigación por parte del lector.

1.1.1 Base de Datos (BD)

El autor Randy Urbano[3] plantea que una base de datos es *una "colección compartida de datos relacionados desde el punto de vista lógico, junto con una descripción de esos datos (metadatos), diseñada para satisfacer las necesidades de información de una organización".* (1)

Por otra parte el autor R. Manuel Ignacio Marrero define a una base de datos como *"un sistema formado por un conjunto de datos almacenados en memorias de almacenamiento masivo (discos) que permiten el acceso directo a esos datos y un conjunto de programas que manipulan ese conjunto de datos".* (2)

A partir de lo anteriormente expuesto, los autores de la presente investigación consideran que una BD, es un conjunto de datos relacionados desde un punto de vista lógico, almacenados de tal forma que se pueden acceder a ellos desde cualquier aplicación externa, permitiendo modificar, insertar o eliminar los mismos.

1.1.2 Bases de datos distribuidas

Una Bases de Datos Distribuida (BDD) es definida como un conjunto de bases de datos lógicamente relacionadas, las cuales se ubican de forma distribuida en diferentes sitios interconectados por una red de comunicaciones, teniendo la capacidad de procesamiento autónomo lo cual posibilita la realización de operaciones locales o distribuidas. (3)

3 Randy Urbano, escritor técnico de Oracle.

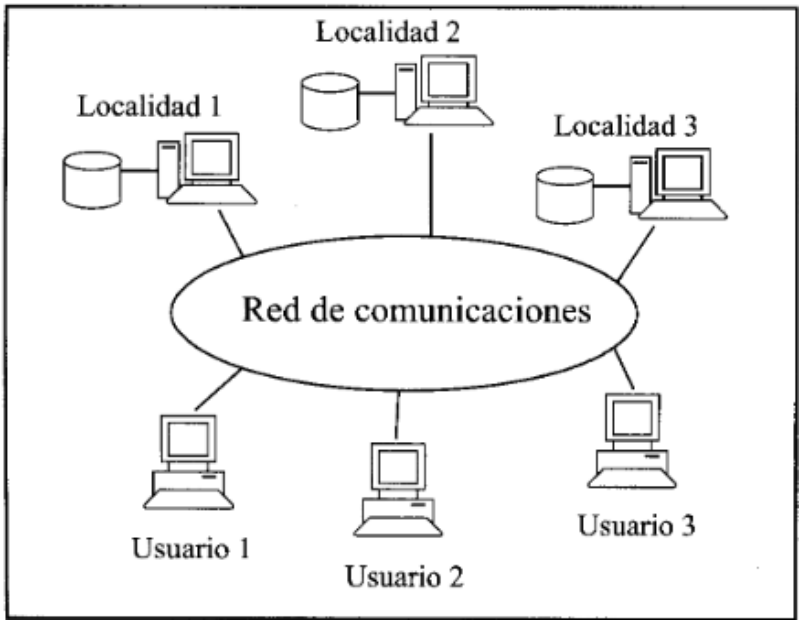


Figura 1 Ejemplo de una BDD
Fuente: (3)

1.1.3 Esquema o estructura de una BD

De forma general se puede decir que un esquema de base de datos determina la estructura de la misma teniendo en cuenta el tipo de gestor, donde define las tablas, campos de las tablas, así como la relación que existe entre cada campo y cada tabla. (4)

1.1.4 Replicación

La replicación es el proceso de copiar y mantener los objetos de base de datos, como tablas, en múltiples bases de datos que comprenden un sistema de base de datos distribuida. Cambios aplicados en un sitio se capturan y almacenan localmente antes de ser expedido y aplicado en cada uno de los lugares remotos. (1)

La replicación es el proceso de intercambio de información para garantizar la coherencia entre los recursos redundantes, tales como: componentes de software o hardware. Permite mejorar la confiabilidad, tolerancia a fallos y accesibilidad. (5)

La replicación es un procedimiento que consiste en copiar datos de un dispositivo de almacenamiento primario a un dispositivo secundario o de copia de seguridad. Si el dispositivo primario falla, los datos están disponibles en el secundario. La replicación de datos asegura alta disponibilidad y tolerancia ante errores graves del clúster. (6)

Los autores de la tesis coinciden con el concepto emitido por el autor Randy Urbano debido a que está acorde con los objetivos del trabajo, el cual converge con los emitidos por otros autores tales como: Marius Cristian Mazilu[4] y colectivo de autores de Oracle[5].

1.1.5 Sincronización

El colectivo de Microsoft Corporation en el año 2014 expone que: *"sincronizar es actualizar dos miembros de un conjunto de réplicas mediante el intercambio de todos los registros y objetos actualizados en cada miembro. Dos miembros de un conjunto de réplicas se encuentran sincronizados cuando los cambios de cada uno han sido aplicados al otro. Se pueden producir conflictos si han cambiado los datos de un mismo campo de los dos miembros del conjunto de réplicas que han sido sincronizados".* (7)

Los autores del Diccionario de Informática y Tecnología se refiere a la sincronización como**:** *"el proceso del establecimiento de consistencia entre datos en fuentes remotas y de la continua armonización de los datos en el tiempo"*. (8)

Por su parte el colectivo de SQL Server 2014 plantea que: "*la sincronización de los datos se refiere al proceso de propagación de los cambios en los datos y el esquema entre el publicador y los suscriptores después de haber aplicado la instantánea inicial en el suscriptor. La sincronización puede producirse: de forma continua, lo que es típico de la replicación transaccional; a petición, lo que es típico de la replicación de mezcla o según una programación, lo que es típico de la replicación de instantáneas"*. (9)

De los conceptos anteriormente expuestos, los autores de la presente investigación consideran que el concepto emitido por la organización Microsoft Corporation en el año 2014 es el que más se corresponde con los objetivo del trabajo.

4 Marius Cristian Mazilu, cibernético, Máster en el soporte de base de datos. Sus ámbitos de desarrollo son: desarrollo y gestión de aplicaciones de base de datos, desarrollo de aplicaciones web para empresas y marketing web

5 Oracle Corporation es una de las mayores compañías de software del mundo. Sus productos van desde bases de datos hasta sistemas de gestión.

1.1.6 Lenguaje de definición de datos

El Lenguaje de Definición de Datos se conoce genéricamente como DDL o *Data Definition Language*. Define y describe los objetos de la base de datos, su estructura, relaciones y restricciones. En la práctica puede consistir en un subconjunto de instrucciones de otro lenguaje informático. (10)

Posee tres operaciones básicas (10):

- *Create*: para crear nuevas tablas, campos e índices.
- *Drop*: para eliminar tablas e índices.
- *Alter*: para modificar las tablas agregando campos o cambiando la definición de los campos.

1.2 Soluciones homólogas

Con el objetivo de recopilar información para sintetizar los conocimientos necesarios y proponer una correcta solución a la presente investigación se realizó el análisis de algunos replicadores. La investigación se centra en las herramientas de replicación que realizan la sincronización de datos del proceso de réplica de estructura en un ambiente distribuido.

1.2.1 SymmetricDS 3.6.0

SymmetricDS es un software de código abierto escrito en Java bajo la GPL[6] de GNU para la replicación de base de datos. Soporta múltiples suscriptores con una dirección o la replicación de datos asincrónica bidireccional. Utiliza tecnologías web y bases de datos para replicar tablas entre bases de datos relacionales, casi en tiempo real, si lo desea. El software fue diseñado para escalar a un gran número de bases de datos, trabajar a través de conexiones de bajo ancho de banda, y soportar períodos de interrupción de la red. Además es capaz de realizar réplicas de datos entre los gestores MySQL, Oracle, SQL Server, SQL Server Azure, PostgreSQL, DB2, Informix, Interbase, Firebird, HSQLDB, H2, Apache Derby, Greenplum, y SQLite. (11)

SymetricDS permite además la sincronización entre dos o más niveles de nodos, se basa en 3 pasos fundamentales para realizar la sincronización de esquemas entre bases de datos relacionales:

6 Licencia Pública General de GNU.

1. Modificación de la base de datos: se ejecuta la sentencia DDL (c*reate, alter, drop*) que modificará la estructura en la base de dato central, pero no se lleva a cabo la actualización de los datos.
2. Sincronización con *triggers*[7]: se procede a la ejecución del proceso "*Sync Triggers*" que es el encargado de detectar los cambios de estructuras que se realizaron mediante el comando "*symadmin sync-triggers*".
3. Enviar esquema: se envían los cambios capturados por el proceso "*Sys Triggers*" a los nodos remotos. Este comando se encarga de detectar que cambios se tienen que aplicar en la estructuras de las bases de datos remotas.

1.2.2 Daffodil Replicator 2.1

Daffodil Replicator es una herramienta implementada sobre Java, de código abierto para la integración, migración y protección de datos en tiempo real. Permite bidireccionar datos de replicación y sincronización entre bases de datos homogéneas y heterogéneas, incluyendo Oracle, MySQL, Daffodil DB, PostgreSQL, entre otras. (12)

1.2.3 Oracle Streams 11g

Oracle Streams es una herramienta que basa su funcionamiento mediante el intercambio de información entre base de datos, específicamente a través de mensajes que contienen LCR[8] que almacenan los cambios ocurridos en las bases de datos. Los mensajes son propagados y las LCR aplicadas respectivamente de una base de datos a otra. El resultado es una característica que proporciona mayor funcionalidad y flexibilidad. Los mensajes pueden ser originados, transformados y/o enviados por el gestor de Oracle u otras herramientas externas que interactúen con Oracle Streams, lo cual brinda una alta flexibilidad y variadas funcionalidades. Cuenta dentro de sus funcionalidades con la sincronización de datos y de estructuras.(13)

7 Los Triggers o Disparadores son objetos que se asocian con tablas y se almacenan en la base de datos. Se ejecutan cuando sucede algún evento sobre las tablas a las que se encuentra asociado. Los eventos que hacen que se ejecute un trigger son las operaciones de inserción (INSERT), borrado (DELETE) o actualización (UPDATE), ya que modifican los datos de una tabla.

8 Unidad básica de Captura de Cambios, del inglés *Logical Change Records.*

1.2.4 Replicador de datos REKO 4.0

REKO es un sistema de replicación de bases de datos distribuidas desarrollado en la UCI con el objetivo de cubrir las principales necesidades relacionadas con la distribución de datos en los gestores de bases de datos más populares tales como: la protección, recuperación, sincronización y transferencia de datos entre diferentes localidades o la centralización de la información en una localidad. (14)

El mismo ha sido desarrollado sobre la plataforma JEE[9] dándole como característica especial que sea un software multiplataforma garantizando su ejecución en sistemas operativos como Linux y Windows. Se utilizó un diseño basado en componentes que permite la incorporación de funcionalidades de forma de forma sencilla. (15)

Principales funcionalidades que brinda el software REKO en su versión 4.0 (15):

- Soporte para réplica de datos en ambientes con conexión y sin conexión.
- Soporte para réplica entre bases de datos con diferentes estructuras.
- Soporte para réplica de ficheros externos a la base de datos.
- Soporte para la sincronización de datos en ambientes con conexión y sin conexión.
- Selección de los datos de réplica ajustada por filtros.
- Soporte para réplica de datos entre los gestores PostgreSQL, Oracle, MySQL, y Microsoft SQL Server.
- Soporte para resolución de conflictos de forma automática y manual.
- Monitoreo en tiempo real de los datos de réplica.
- Los archivos de gran tamaño son enviados por FTP[10] permitiendo resumir el envío y el recibo de los mismos en caso de interrupciones en la red.
- Soporte para programación del momento de captura y envío de los datos de réplica.
- Seguridad de los datos que se envían con encriptación SSL[11].

9 Plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java, del inglés Java Enterprise Edition.

10 Protocolo de Transferencia de Ficheros, del inglés File Transference Protocol.

11 Capa de conexión segura, del inglés Secure Sockets Layer.

Cuenta con un módulo de sincronización que permite tener sincronizadas las bases de datos que intervienen en la réplica de datos. Este módulo permite realizar la sincronización de dos formas diferentes:

- Conectado: cuando existe conexión entre los nodos implicados en el proceso de réplica.
- Desconectado: cuando no existe conexión entre los nodos implicados en el proceso de réplica.

1.2.5 Resultados

Al realizar un análisis de las funcionalidades de los replicadores anteriormente expuestos en busca de una solución homóloga, se arrojó como resultado que a pesar de que los mismos poseen características tentadoras, no cumplen con los requisitos necesarios para ser tomados como solución de la presente investigación.

SymetricDS se ve limitado en materia de usabilidad puesto que está destinado a usuarios con avanzados conocimientos en la gestión de BD, debido a que las operaciones que realiza son mediante comandos en consola. Daffodil Replicator no es capaz de replicar en ambiente multimaestro. Durante el estudio también se profundizó en las herramientas privativas tales como: Oracle Streams, Oracle Advanced Replication y SQL Server; las cuales por su alto costo en licencias y soporte, así como el no acceso a su código fuente fueron descartadas como solución factible. Por lo que se decidió llevar a cabo la implementación de una solución propia teniendo en cuenta los conocimientos sintetizados durante el estudio, haciendo uso de los componentes ya desarrollados en el software de réplica REKO contribuyendo a la independización tecnológica del país.

1.3 Metodologías de desarrollo

Las metodologías para el desarrollo del software imponen un proceso disciplinado con el fin de hacerlo más predecible y eficiente. Una metodología define una representación que permite facilitar la manipulación de modelos, y la comunicación e intercambio de información entre todas las partes involucradas en la construcción de un sistema.

El ciclo de vida del proceso de desarrollo de software de la solución estará guiado por la metodología Proceso Unificado Ágil (AUP, Agile Unified Process, por sus siglas en inglés)

debido a que es la designada por la universidad para la actividad productiva, específicamente para los proyectos de desarrollo.

1.3.1 Metodología AUP

Describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. Ha sido adaptada a las necesidades de desarrollo de software en la Universidad de las Ciencias informáticas. (16)

En la variación efectuada se proponen 3 fases para guiar el proceso de desarrollo de los proyectos en la universidad:

Inicio: durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

Ejecución: en esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.

Cierre: en esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.4 Herramientas y lenguajes para el desarrollo

Para la determinación de las herramientas y lenguajes que se utilizarían en el desarrollo de la solución, se tuvo en cuenta el análisis de las utilizadas durante el desarrollo del software REKO. Las mismas se mantendrán en la implementación de la solución de la presente investigación, debido a que permite disminuir la curva de aprendizaje y el tiempo empleado en el desarrollo de la solución.

Visual Paradigm para UML 8.0: herramienta multiplataforma diseñada para soportar el ciclo de vida completo del proceso de desarrollo del software, a través de la representación de todo tipo de diagramas. Con capacidad de ingeniería directa e inversa enfocada al negocio que generan un software de mayor calidad.(17)

JEE 1.7.71: es una plataforma que encierra una colección de especificaciones que definen una infraestructura para desarrollar aplicaciones distribuidas multicapa que facilita el desarrollo de aplicaciones distribuidas en Java, ofrece un marco y una serie de convenciones, junto un conjunto de servicios sobre los cuales desarrollar aplicaciones multicapa. (18)

Java: es un lenguaje de programación de alto nivel que aplica un modelo de programación orientada a objetos, que se puede utilizar para crear aplicaciones completas que pueden ejecutarse en un único ordenador o ser distribuidos entre servidores y clientes en una red. (19)

UML 2.0: es un lenguaje de modelado estándar para el desarrollo de sistemas de software orientado a partir del paradigma de Programación Orientada a Objetos, permitiendo especificar, visualizar, construir y documentar todos los elementos que conforman dichos sistemas. (20)

Spring 2.5: contenedor y framework, basado en inyección de dependencias, promoviendo el bajo acoplamiento pues los objetos reciben pasivamente sus dependencias en lugar de crearlas o buscar los objetos dependientes por sí mismos. Brinda facilidades para desarrollar una aplicación empresarial en JEE. (21)

JMS: es una API[12] Java que permite a las aplicaciones basadas en la plataforma JEE crear, enviar, recibir y leer mensajes. Hace posible la comunicación confiable de manera síncrona y asíncrona.(22)

Apache ActiveMQ 5.9.0: es un poderoso servidor de mensajería que garantiza la comunicación entre clientes a través de varios protocolos con el empleo de lenguajes de programación así como son: C, C ++, C #, Ruby, Perl, Python, PHP. (23)

Eclipse STS 3.6.1: Spring Tool Suite (STS) es una herramienta libre desarrollada por Spring Source, basada actualmente en Eclipse 3.x, para construir aplicaciones empresariales enriquecidas por los proyectos de Spring Source. Una de las principales razones para emplear STS es que incluye herramientas para el desarrollo en lenguaje Java, para Spring.(24)

JDBC[13] 4.0: es un componente de software que permite a las aplicaciones escritas en Java interactuar con una base de datos a través del dialecto SQL sin importar el proveedor de base de datos que se utilice ni el sistema operativo donde se ejecute.(25)

12 API del inglés: *Application Programming Interface,* es el conjunto de subrutinas, funciones que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

13 Java Database Connectivity

PostgreSQL 9.1: es un sistema de gestión de bases de datos objeto relacional, distribuido bajo licencia BSD[14] y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado.(26)

1.5 Consideraciones parciales del capítulo

- Para la comprensión del trabajo se profundizó en el conocimiento de algunos conceptos fundamentales.
- Se definieron y justificaron las herramientas y metodología a utilizar para dar solución al problema planteado.
- Tomar otros replicadores como solución del problema no es factible debido a las limitantes que presentan en los mismos.

14 Es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution).

Capítulo 2. Características y diseño del sistema

Concluida la descripción de los principales conceptos de la investigación se procede a diseñar y caracterizar el sistema, creando las bases necesarias para ejecutar el desarrollo de la solución y tomando como guía para el proceso de desarrollo la metodología AUP.

2.1 Modelamiento del sistema

2.1.1 Funcionamiento del software de replicación REKO

Cuando se inicia el proceso de sincronización, el "Capturador de Cambios" de REKO crea un grupo replicable que puede ser instancia de las clases **`ReplicableGroup` o `ReplicableStructureGroupo`,** en dependencia del tipo de réplica que se esté realizando. En el grupo replicable se almacenan un conjunto de acciones replicables o estructuras replicables contienen la información necesaria para realizar el proceso de réplica. A partir de estos objetos, las configuraciones registradas y los filtros asociados a estas, se determinan los destinos hacia donde se enviará el grupo replicable. De la propagación de los grupos replicables se encarga el "Distribuidor de Datos" empleando el servidor Apache ActiveMQ.

En cada nodo destino, una vez recibido un grupo replicable, el "Aplicador de Cambios" ejecuta en la BD las acciones que contiene.

Existen mecanismos de chequeo de envío, confirmación de recepción y registro en una BD local de los grupos replicables, que garantizan la integridad referencial y persistencia de los datos que se replican ante posibles eventualidades. (27)

2.1.2 Descripción del proceso a automatizar

REKO cuenta con un módulo de sincronización capaz de homogeneizar los datos de las BD que intervienen en un proceso de réplica. Actualmente en este módulo si se ejecutan sentencias del tipo DDL que modifican las estructuras de las BD, se pierde la homogeneidad de las BDD. Debido a que estas acciones no se tienen en cuenta durante el proceso de sincronización, generando por ende errores durante el proceso de réplica. Se propone dadas estas limitantes extender las funcionalidades del módulo de sincronización, de tal manera que se logre homogeneizar las estructuras de las BD evitando la generación de conflictos.

2.1.3 Modelo de dominio

Un modelo de dominio es una representación visual de las clases conceptuales u objetos del mundo real que se definen en un dominio de interés, permitiendo establecer un lenguaje común entre los usuarios, clientes, desarrolladores y demás interesados. (28)

La realización de este modelo permite además delimitar el alcance del domino del problema, añadir precisión y enfoque para la discusión entre el equipo de técnicos y de negocio.

Diagrama de clases del modelo de dominio

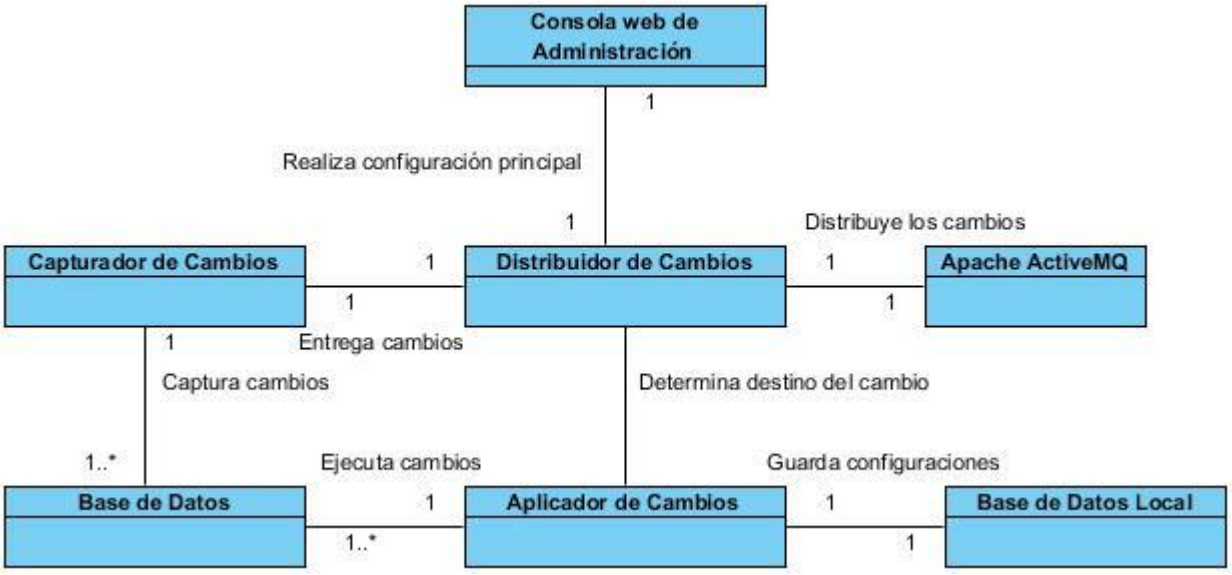


Figura 2 Modelo de dominio

Descripción de las clases del modelo de dominio

Capturador de Cambios: captura los cambios que se realizan sobre la BD y se los entrega al Distribuidor de Cambios.

Aplicador de Cambios: ejecuta en la base de datos los cambios que son enviados hacia él desde otro nodo de réplica.

Distribuidor de Cambios: determina para dónde debe ser enviado cada cambio realizado en la BD, los envía y se responsabiliza de que cada cambio llegue a su destino.

BD Local: es utilizada para guardar las configuraciones propias de la réplica, las acciones sobre la BD que han dado conflicto al aplicarse y las acciones o transacciones que no han podido llegar a su destino.

Consola Web de Administración: representa la interfaz del replicador. Permite realizar las configuraciones principales del software como el registro de nodos, configuración de las tablas a replicar, datos a replicar y el monitoreo del funcionamiento del sistema.

Servidor Apache ActiveMQ: servidor de mensajería bajo la especificación *Java Message Service*, es utilizado como punto intermedio en la distribución de la información enviada bajo JMS.

Base de Datos: es la BD que se está replicando, el software de réplica envía los cambios que se realizan sobre ella y aplica los cambios que provienen de otros nodos de réplica.

2.1.4 Especificación de requisitos de software

Según la IEEE[15] *"es un conjunto de recomendaciones para la especificación de los requerimientos o requisitos de software el cual tiene como producto final la documentación de los acuerdos entre el cliente y el grupo de desarrollo para así cumplir con la totalidad de exigencias estipuladas"*. (29)

Requisitos funcionales

Describen como debe comportarse el sistema ante un determinado estímulo. Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones específicas. (30)

Los requisitos funcionales identificados se describen a continuación:

RF1 Guardar configuración general de sincronización de estructura.

RF2 Sincronizar de forma automática.

RF3 Sincronizar de forma manual.

RF4 Construir acciones DDL.

RF5 Comparar acciones DDL.

RF6 Aplicar cambios de sincronización.

RF7 Resolver conflictos según la configuración de sincronización de estructura.

RF8 Actualizar configuración de réplica.

RF9 Enviar notificaciones.

15 Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, por sus siglas en ingles).

RF10 Monitorizar acciones de sincronización.

RF11 Exportar datos de sincronización de estructura.

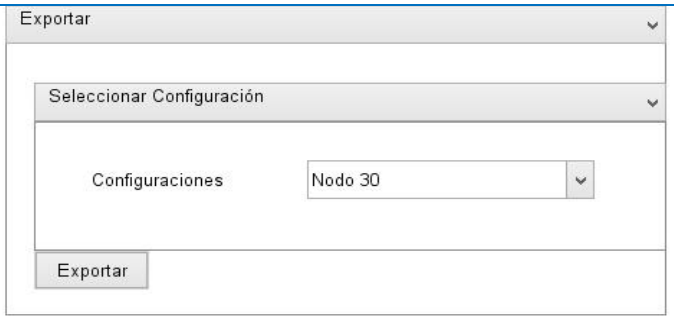
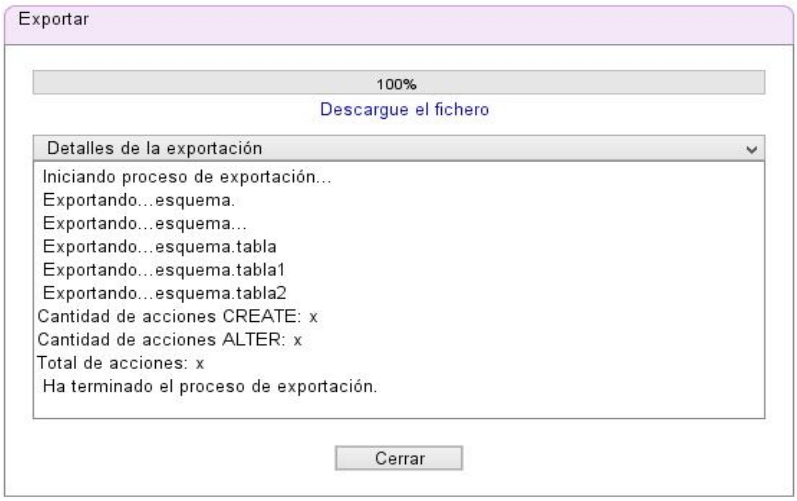
RF12 Importar datos de sincronización de estructura.

Requisitos no funcionales

Imponen restricciones de diseño las cuales son propiedades o cualidades que el producto debe tener. Restringen los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, el tipo de proceso de desarrollo a utilizar, fiabilidad, tiempo de respuesta y capacidad de almacenamiento. (30)

Tabla 1 Requisitos no funcionales

Atributo de Calidad	Usabilidad.
Sub-atributos/Sub-característica	Operabilidad.
Objetivo	Capacidad que debe presentar el producto que le permite al usuario operarlo y controlarlo con facilidad.
Origen	Proveedor de requisitos.
Artefacto	Interfaz de sincronización en modo “Desconectado”.

	 
Entorno	El sistema desplegado.
Estímulo **<<1>>. <<a>>< Número de pasos a ejecutar para exportar la sincronización de estructura >**	**Respuesta: Flujo de eventos (Escenarios)**
El usuario escoge la opción de sincronizar en modo “Desconectado” y va a la pestaña de exportar la configuración de sincronización y selecciona el nodo con configuración y hace clic en el botón “Exportar”.	1. El sistema crea un archivo con la configuración de sincronización y le brinda al usuario la opción de descargarlo.
Medida de respuesta	

Navegar en el sistema.

Fuente: elaboración propia

Atributo de Calidad	Usabilidad.
Sub-atributos/Sub-característica	Protección contra errores de usuarios.
Objetivo	Capacidad del producto para proteger a los usuarios de cometer errores.
Origen	Proveedor de requisitos.
Artefacto	Las alertas para prevenir al usuario ante el mal llenado de los datos.
Entorno	El sistema desplegado.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
<<1>>. <<a>>< Capacidad del producto para proteger a los usuarios de cometer errores al exportar la sincronización. >	
El usuario intenta exportar la sincronización sin escoger el nodo con configuración de réplica con quien desea sincronizar y hace clic en el botón “Exportar”.	1. El sistema muestra el mensaje de alerta en la lista desplegable: "El valor especificado no es válido.".
Medida de respuesta	
Navegar en el sistema.	

Fuente: elaboración propia

Atributo de Calidad	Mantenibilidad.
Sub-atributos/Sub-característica	Modificabilidad.

Objetivo	Capacidad del producto que permite ser modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.
Origen	Arquitecto de software.
Artefacto	Configuración general del sistema.
Entorno	El ambiente de desarrollo del sistema.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
<<1>>. <<a>>< Capacidad de modificación del sistema>	
El usuario cambia en la configuración general los datos referentes a la sincronización de estructura.	1. El sistema realiza la sincronización de acuerdo a la configuración de sincronización de estructura definida en la configuración general. • Sincroniza de forma “Manual”. • Sincroniza de forma “Automática”.
Medida de respuesta	
Introducir una modificación al sistema.	

Fuente: elaboración propia

Para consultar los restantes requisitos no funcionales de atributos de calidad aplicados en la solución ver (Anexo 20).

2.1.5 Propuesta del sistema

Para realizar el desarrollo de la solución se extendieron los componentes Administrador, Distribuidor de Cambios, Capturador de Cambios y Aplicador de Cambios del software REKO.

Al realizar la sincronización de estructura se debe comprobar que exista configuración de réplica, a partir de esta se construyen los grupos de estructura replicables (RSG, por sus siglas en inglés), cada RSG contiene una lista de las acciones de estructura replicables (RSA, por sus siglas en inglés) que contienen las tablas de los esquemas almacenados en la configuración de réplica. Una vez concluido este proceso, se envían los grupos a través del distribuidor de

cambios que es el encargado de determinar el destino y además se responsabiliza de la propagación de estos empleando el servidor Apache ActiveMQ.

Recibidos los grupos por el nodo remoto, se deben extraer las RSA y se procede a compararlas con las RSA locales que almacenan las tablas de los esquemas de la base de datos local, finalizado el proceso de comparación se entregan las diferencias encontradas al Aplicador de Cambios que se encargará de ejecutarlas en la BD.

2.1.6 Historias de usuarios

Entre las técnicas ágiles que utiliza AUP se encuentra el Modelado ágil que permite encapsular los requisitos funcionales en Historias de Usuarios (HU), descripción de requisitos por procesos o en por Casos de Uso. El proyecto Replicador de datos REKO en su versión 4.0 realiza la descripción de las HU. Por tanto, se hará uso de esta técnica para encapsular los requisitos funcionales.

Las HU son descripciones cortas y simples de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea dicha funcionalidad, poseen una descripción escrita que será utilizada para planificar y posteriormente disgregar los detalles con el dueño del producto. (31)

A continuación se muestra las Historias de Usuario del RF2, RF11, RF12 respectivamente:

Tabla 2 Historias de usuario del RF2

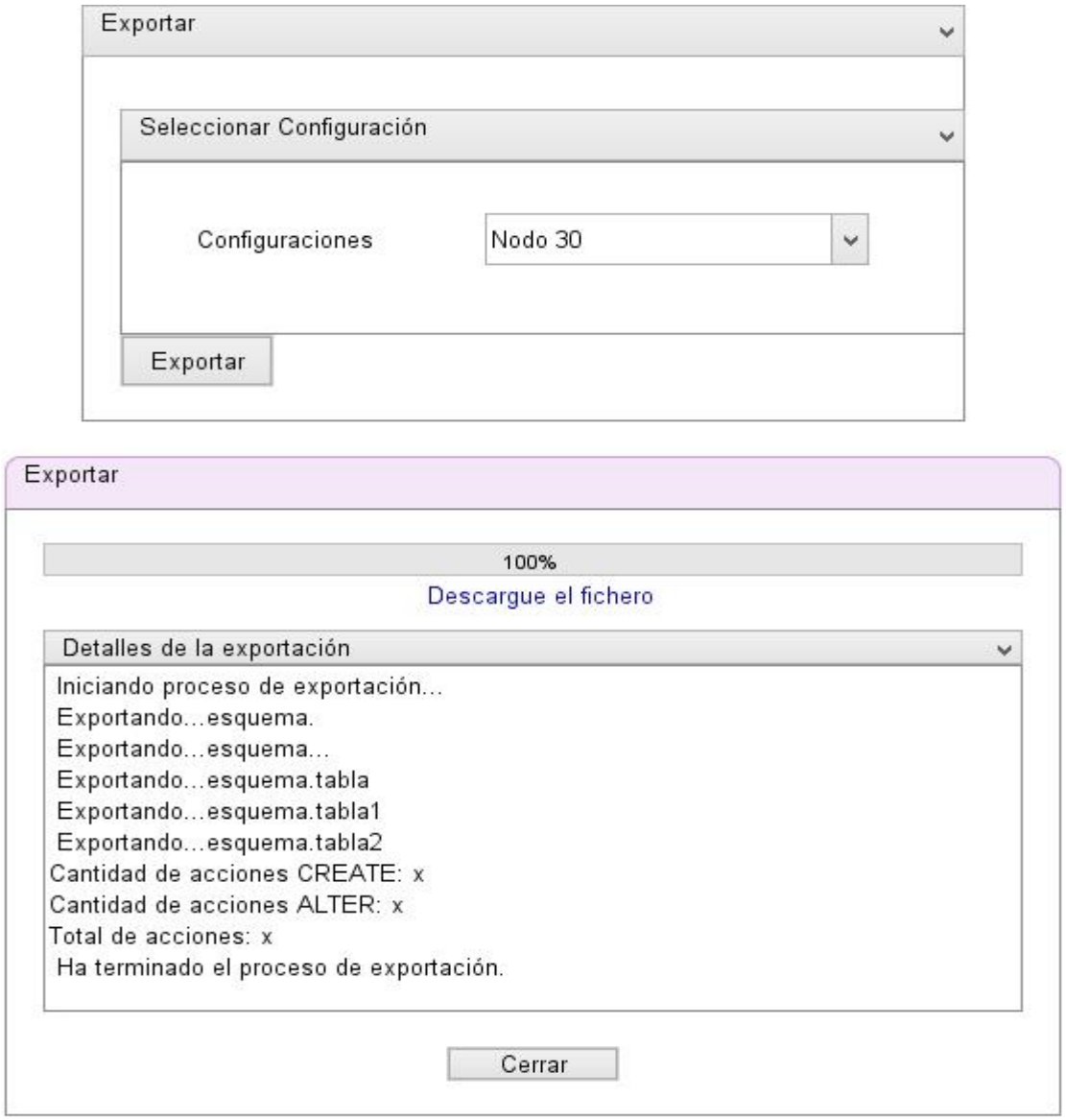
Historias de usuario	
Número:HU2	**Nombre del requisito:** Sincronizar de forma automática.
Programador: Enrique Basto Muguercia	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 3 semanas
Riesgo en Desarrollo: • Rotura de alguna estación de trabajo perteneciente al proyecto. • Problemas eléctricos.	**Tiempo Real:** 120 horas
Descripción: El en módulo “Sincronizar” en la pestaña “Conectado” en la sección “Exportar”, el sistema debe mostrar los nodos con configuración de réplica brindando la opción ”Sincronizar”. Cuando se ejecute la opción “Sincronizar”, el sistema debe realizar el proceso de sincronización de forma automática. Para realizar el proceso de sincronización el sistema debe construir RSG y enviar estos al nodo destino. El sistema debe comparar los RSG con la estructura local, aplicar los cambios en la base de datos y actualizar la configuración de réplica. En caso de existir conflictos debe resolverlos según la configuración definida para conflictos durante la sincronización de estructura. Una vez terminado el proceso de sincronización el sistema debe notificar la culminación del proceso al nodo remoto.	
Observaciones: El sistema debe cambiar los estados de los nodos según la fase que se encuentre el proceso, los estados son: • “Sincronizar” • “Esperando Autorización” • “Sincronizando” • “Cancelado” • “Sincronizado”	
Prototipo de interfaz:	

Fuente: elaboración propia

Tabla 3 HU del RF11

Historias de usuario	
Número:HU11	**Nombre del requisito:** Exportar datos de sincronización de estructura
Programador: Enrique Basto Muguercia	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 2 semanas
Riesgo en Desarrollo: • Rotura de alguna estación de trabajo perteneciente al proyecto. • Problemas Eléctricos.	**Tiempo Real**: 80 horas
Descripción: En el módulo “Sincronizar” en la pestaña “Desconectado” en la sección “Exportar”, el sistema debe mostrar una lista desplegable. Cuando el usuario haga clic en la lista desplegable, el sistema debe mostrar los identificadores de los nodos con configuración de réplica. Cuando se seleccione un identificador, el sistema debe habilitar el botón “Exportar”. Cuando se presione el botón “Exportar” el sistema debe comenzar el proceso de exportación de los datos de sincronización.	

Para realizar la exportación de datos del proceso de sincronización de estructura, el sistema debe construir RSG para almacenar las estructuras de los esquemas definidos en la configuración de réplica. Debe crear un fichero serializado con la configuración de réplica, escribir en un fichero XML los RSG y crear un fichero log donde se almacenen datos referentes a la cantidad de acciones y grupos que contienen el fichero XML. El sistema debe mostrar en la interfaz “Exportar” los datos referentes al proceso de sincronización. El sistema debe empaquetar los ficheros en un archivo de extensión (.rk) y encriptarlos con una contraseña propia del sistema para garantizar la integridad de la información. Debe almacenarlo en una dirección temporal habilitando en la interfaz “Exportar” un enlace de descarga. Cuando se ejecute el enlace “Descargue el fichero”, el sistema debe abrir una ventana donde brinda la posibilidad al usuario de seleccionar la ruta de descarga. Cuando se haga clic en “Aceptar”, el sistema debe descargar el archivo a la ruta seleccionada.
Observaciones: El comportamiento de la ventana de descarga es según el navegador y la configuración que tenga este para realizar este proceso.
Prototipo de interfaz:



Fuente: elaboración propia

Tabla 4 HU del RF12

Historias de usuario	
Número:HU12	**Nombre del requisito:** Importar datos de sincronización de estructura.
Programador: Luis Ángel Carbonel Hidalgo y Enrique Basto Muguercia.	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 2 semanas

Riesgo en Desarrollo:	**Tiempo Real:** 80 horas
• Rotura de alguna estación de trabajo perteneciente al proyecto. • Problemas Eléctricos.	

Descripción:

En el módulo “Sincronizar” en la pestaña “Desconectado” en la sección “Importar”, el sistema debe mostrar un formulario con la posibilidad de importar un archivo.

Cuando el usuario haga clic en la opción “…”, el sistema debe abrir la interfaz “Directorio Seleccionado” brindando la posibilidad de seleccionar el fichero de configuración. Cuando se haga clic en el botón “Aceptar”, el sistema debe mostrar la dirección del fichero en el campo “Directorio local”. Cuando se haga clic en el botón “Subir”, el sistema debe validar el archivo de configuración, en caso de ser correcto, debe desplegar la sección “Detalles de importación”, escribir un log con información referente al archivo de configuración y habilitar el botón “Importar”. En caso de existir algún error el sistema debe informar en la sección “Detalles de importación” el tipo de error. Cuando se haga clic en el botón “Importar” el sistema debe comenzar el proceso de importación.

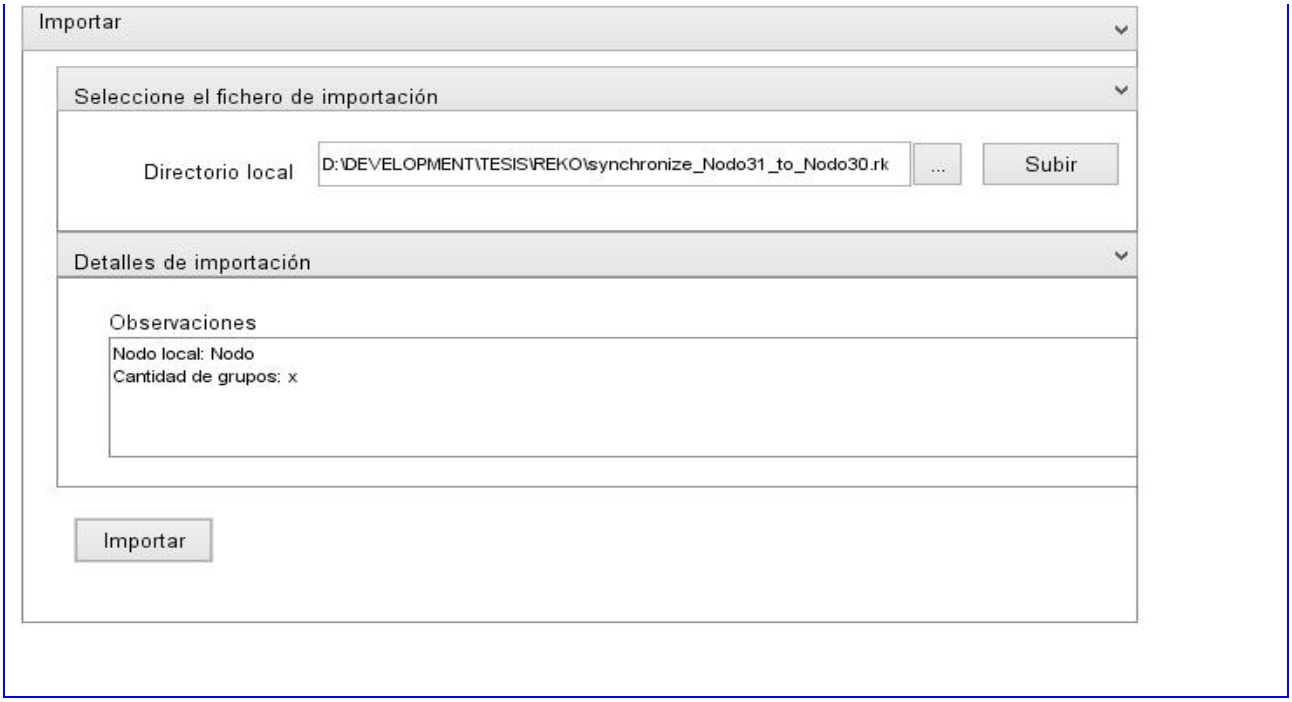
Para realizar la importación, el sistema debe copiar el archivo de configuración a un directorio temporal, desencriptarlo y extraer los ficheros. El sistema debe leer el fichero XML y a partir de este construir RSG que contiene la estructura de la base de datos remota, deserializar el archivo de configuración de réplica y proceder a realizar el proceso de sincronización.

Observaciones:

El sistema puede mostrar varios tipos de errores tales como:

- “Solo archivos .rk”
- “No se encuentra registrado el nodo.”
- “La importación no corresponde a este nodo.”
- “El archivo esta corrupto”

Prototipo de interfaz:



Fuente: elaboración propia

2.2 Descripción de la arquitectura del software REKO

Según David Garlan la *"Arquitectura de Software constituye un puente entre el requerimiento y el código, ocupando el lugar que en los gráficos antiguos se reservaba para el diseño." ((32)*

Por otra parte la IEEE 1471-2000 define a la arquitectura de software como: *"La organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente, y los principios que orientan su diseño y evolución." (33)*

A partir de lo anteriormente expuesto se puede concluir que la concepción de una arquitectura en el desarrollo de software brinda una vista general del sistema, fijando la relación de los principales puntos de diseños a tratar, lo que facilita el desarrollo simultaneo de componentes y la detección de errores de diseño desde fases tempranas. Además fomenta la reutilización y hacer evolucionar el sistema.

Una buena práctica en la definición de la arquitectura del sistema es el uso de patrones. Un patrón según lo que expresa Christopher Alexander es *"una descripción de un problema que ocurre una y otra vez en nuestro entorno, así como la solución a ese problema, de tal modo que se pueda aplicar esta solución un millón de veces, sin hacer lo mismo dos veces". (34)* Por lo que se puede concluir que un patrón es la base para la búsqueda de soluciones a problemas comunes.

Según la escala o nivel de abstracción los patrones son clasificados como:

- **Patrones de arquitectura:** aquellos que expresan un esquema organizativo estructural fundamental para sistemas de software.
- **Patrones de diseño:** aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software.
- **Dialectos:** patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto. (35)

El replicador de datos REKO define su arquitectura basada en componentes ya que sus partes encapsulan un conjunto de comportamientos que pueden ser reemplazados por otros.

Una arquitectura basada en componentes es un acercamiento basado en la reutilización para definir, implementar, y componer componentes débilmente acoplados en sistemas. Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado. (36)

Los principales componentes que se encuentran en el software son:

Capturador_de_Cambios: encargado de capturar los cambios que se realizan en la base de datos y entregarlos al distribuidor.

Distribuidor: determina el destino de cada cambio realizado en la base de datos, los envía y se responsabiliza de su llegada.

Aplicador: ejecuta en la base de datos los cambios que son enviados hacia él desde otro nodo de réplica.

Administrador: permite realizar las configuraciones principales del software como el registro de nodos, configuración de las tablas a replicar y el monitoreo del funcionamiento.

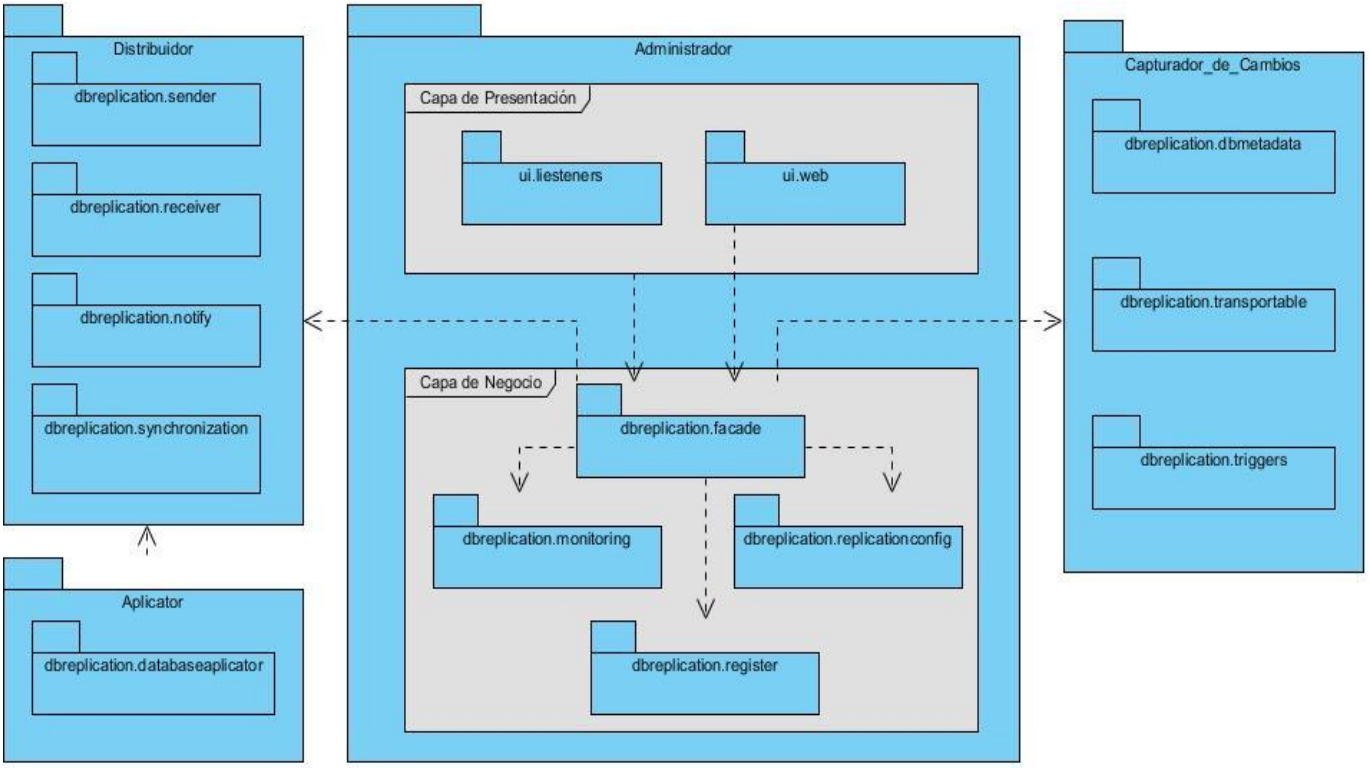


Figura 3 Arquitectura del sistema

A pesar de que la arquitectura es basada en componentes se puede apreciar que el componente **Administración** responde a un modelo multicapas, donde cada capa encapsula un comportamiento y objetivos precisos, permitiendo así que su implementación sea desacoplada con respecto a otra capa y el intercambio de información se realice a través de interfaces. Además de estar separadas lógica y estructuralmente, las capas se encuentran separadas de manera física.

La capa de **presentación** proporciona al usuario una interfaz gráfica con la que puede introducir información al sistema y recibir las notificaciones del mismo. Esta capa solo se comunica con la capa de **negocio**, donde se reciben las peticiones del usuario y se procesan. También es conocida como capa de lógica de negocio debido a que aquí es donde ese establece las reglas que deben cumplirse.

2.2.1 Patrones de diseño

Los patrones de diseño son soluciones bien documentadas que los desarrolladores emplean para dar solución a nuevos problemas apoyados en la experiencia de haberlas utilizado con éxito en el pasado. Provee además un esquema para refinar componentes de un sistema de software y la forma en que se relacionan entre sí. Describe una estructura generalmente

recurrente de comunicación de componentes que resuelve un problema de diseño general dentro de un contexto particular. (37)

Patrones GRASP

- **Creador:** este patrón es el encargado de que una clase B cree una instancia de una clase A, siempre que:
    - La clase B contenga a la clase A.
    - B sea una agregación (o composición) de A.
    - B almacene a A.
    - B tenga los datos de inicialización de A.
    - B use a A.

El patrón se refleja en la implementación de la clase *Table*. La clase *Table* contiene los objetos de la clase Columna, tiene los datos de inicialización que serán transmitidos a Columna cuando este objeto sea creado.

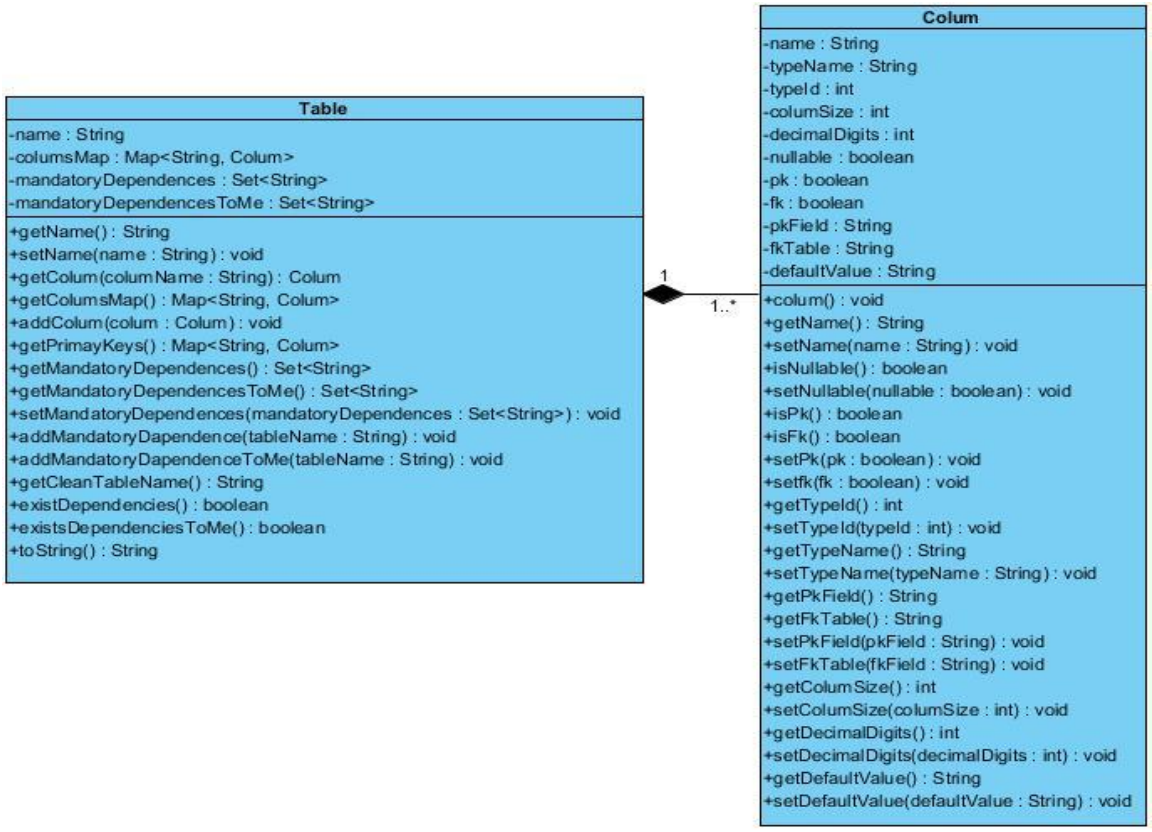


Figura 4 Ejemplo del patrón creador

- **Experto:** se basa en que se le asigne la responsabilidad de la creación de un objeto o la implementación de un método a la clase que conoce toda la información necesaria para crearlo. Por ejemplo la clase *SchemaComparison* es la que se especializa en la comparación de los elementos que componen los esquemas, mientras que la clase *AplicatorSynchronizationDao* se especializa en aplicar cambios en la base de datos.
- **Controlador:** el patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal manera que es el que recibe todas las peticiones y ejecuta en las diferentes clases el método solicitado. Por ejemplo se utilizan clases controladoras tales como: *ConfigurationManager*, *SenderManager*, *ReciverManager*, *RegistrationManager*, *SyncManager* encargadas de desencadenar eventos según las peticiones que se realicen al sistema, separando la capa de presentación de la lógica de negocio lo cual permite una mayor reutilización del código.
- **Alta cohesión:** el patrón define que la información que almacena una clase debe de ser coherente y estar en mayor medida relacionada con la clase.
- **Bajo acoplamiento:** el patrón fue utilizado con el objetivo de tener las clases del sistema con la menor dependencia entre ellas. De tal manera que en caso de producirse alguna modificación, el impacto sea el mínimo posible en el resto de clases, potenciando la reutilización.

Patrones GOF

- **Fachada:** proporciona una interfaz unificada simple que permite agrupar las interfaces de un subsistema permitiendo reducir la dependencia entre las clases. Por ejemplo, el sistema replicador de datos REKO cuenta con la clase *ReplicationFacade* perteneciente al componente *facade* que ofrece un punto de acceso para iniciar el proceso de sincronización.

2.3 Modelo de diseño

Muestra los objetos o clases en un sistema y, donde sea apropiado, los diferentes tipos de relaciones entre estas entidades. Es el puente en los requerimientos y la implementación del sistema. (38)

2.3.1 Diagramas de paquetes

Un diagrama de paquetes es un diagrama UML que permite organizar los elementos del modelo de un sistema y muestra cómo el mismo está dividido en agrupaciones lógicas mostrando las dependencias entre estas, las cuales pueden indicar el orden de desarrollo requerido, además representa una visión general de la arquitectura suministrando una descomposición lógica del sistema. (39)

A continuación se describen algunos de los diagramas de paquetes, para los restantes ver (Anexos 1-9):

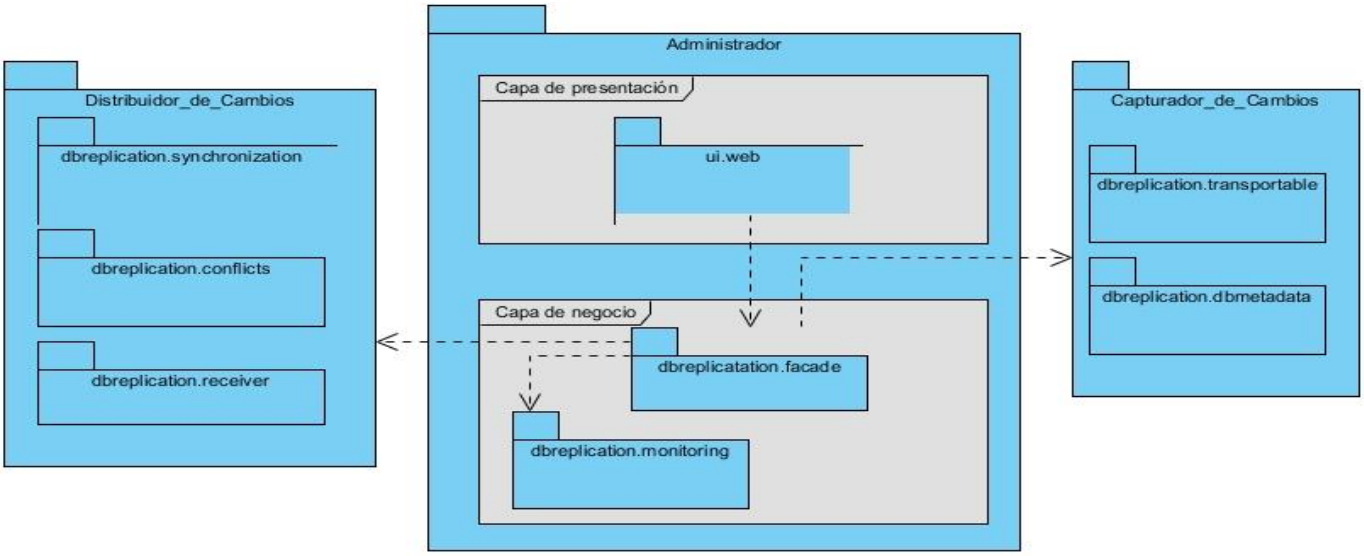


Figura 5 DP Sincronizar de forma automática

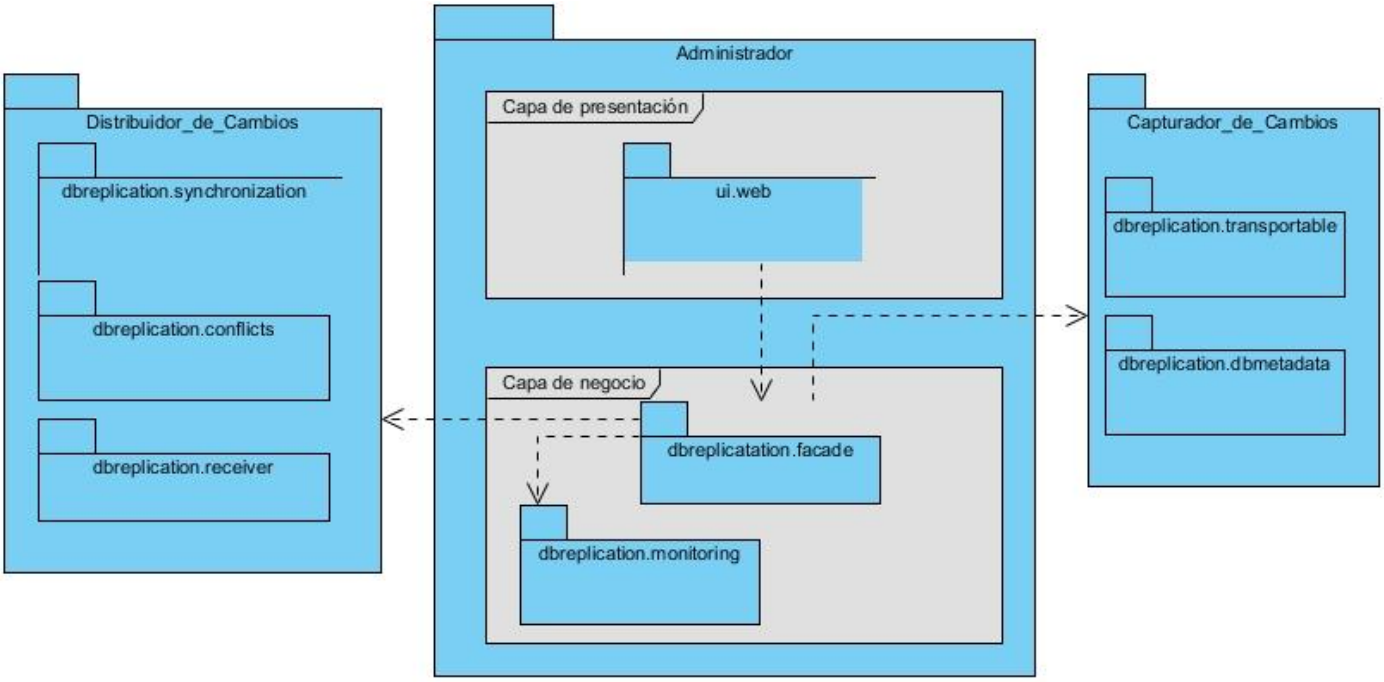


Figura 6 DP Importar datos de sincronización de estructura

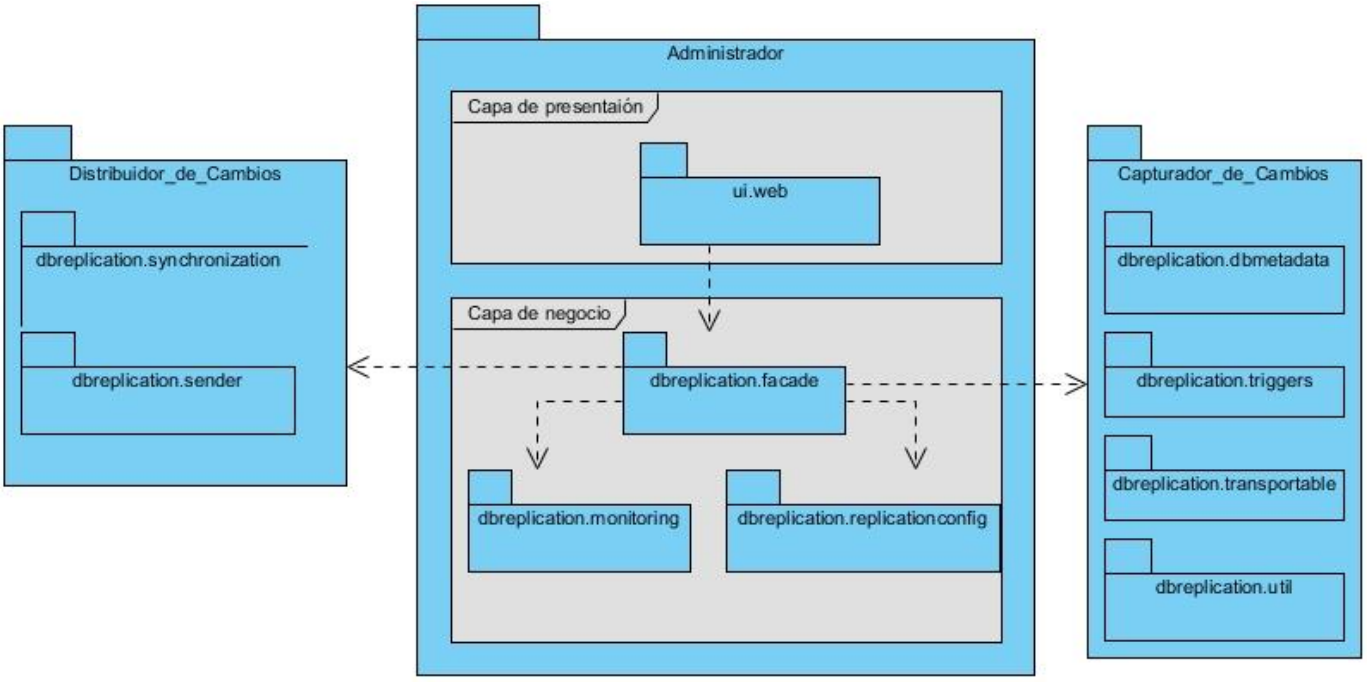


Figura 7 DP Exportar datos de sincronización de estructura

2.3.2 Diagramas de clases del diseño

A continuación se muestran algunos de los diagramas de clases (DC) correspondientes a los diagramas de paquetes (DP) de la solución, para los restantes ver (Anexos 10-12):

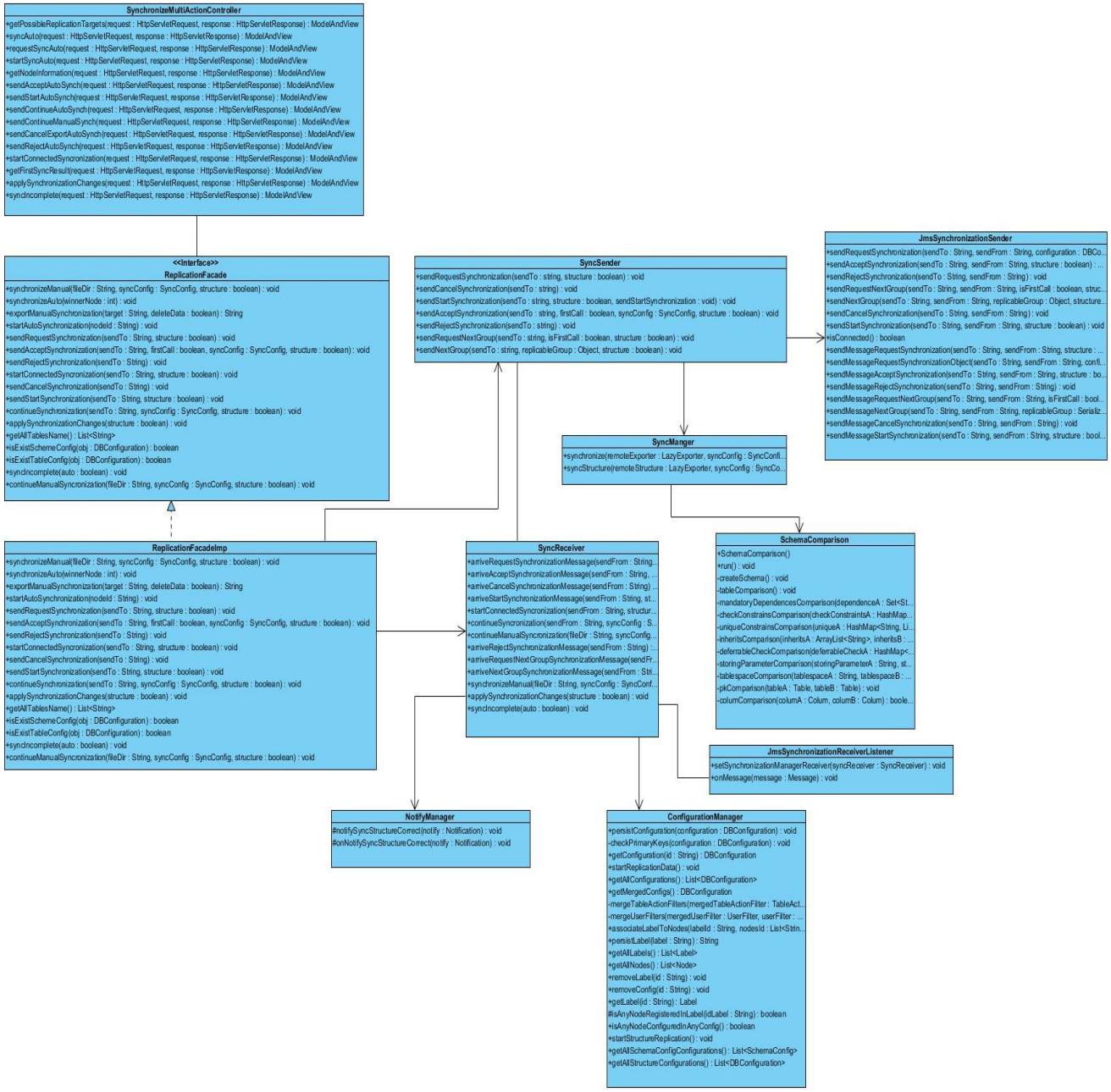


Figura 8 DC correspondiente al DP Sincronización automática

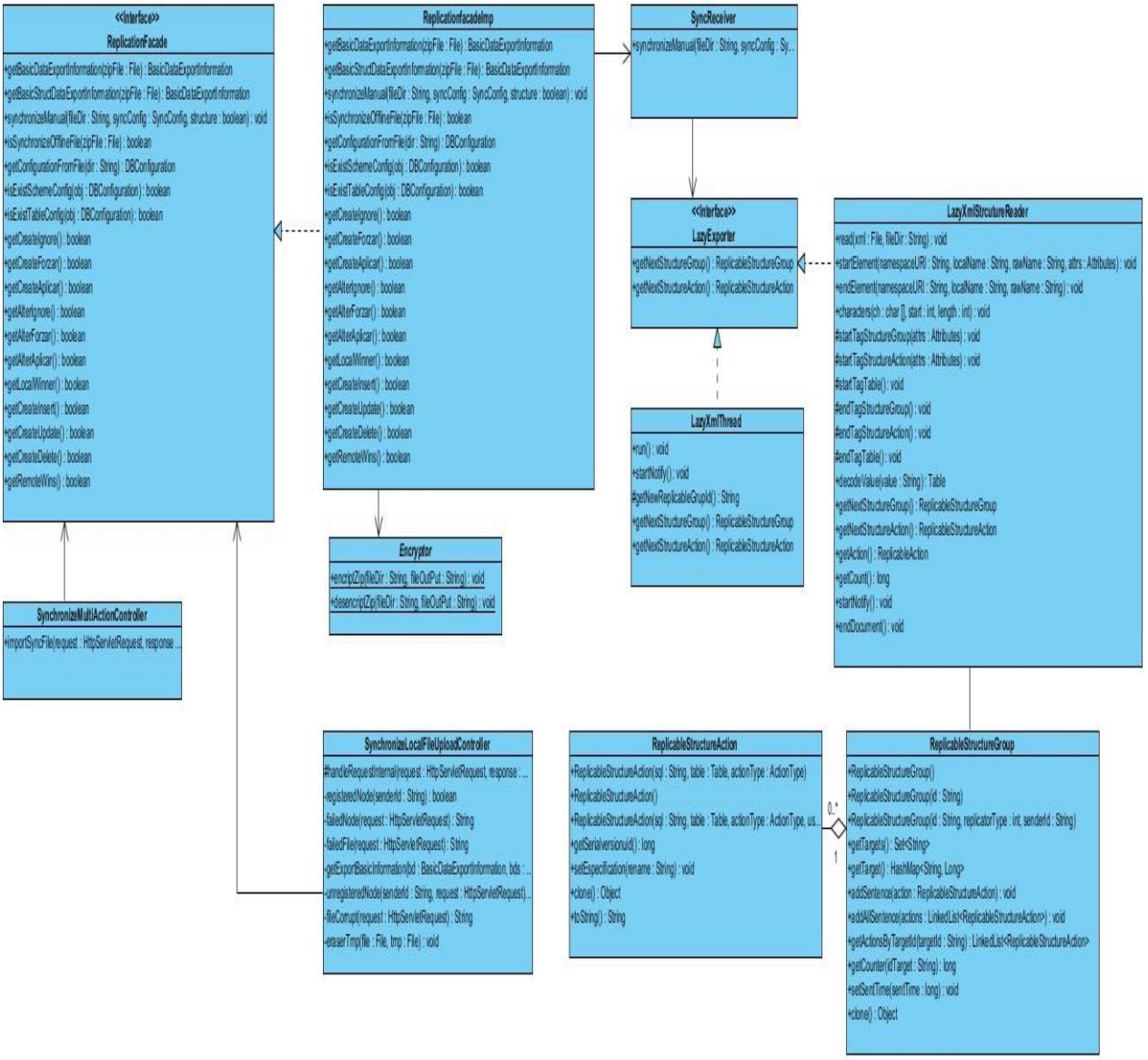


Figura 9 DC correspondiente al DP Importar datos de sincronización de estructura

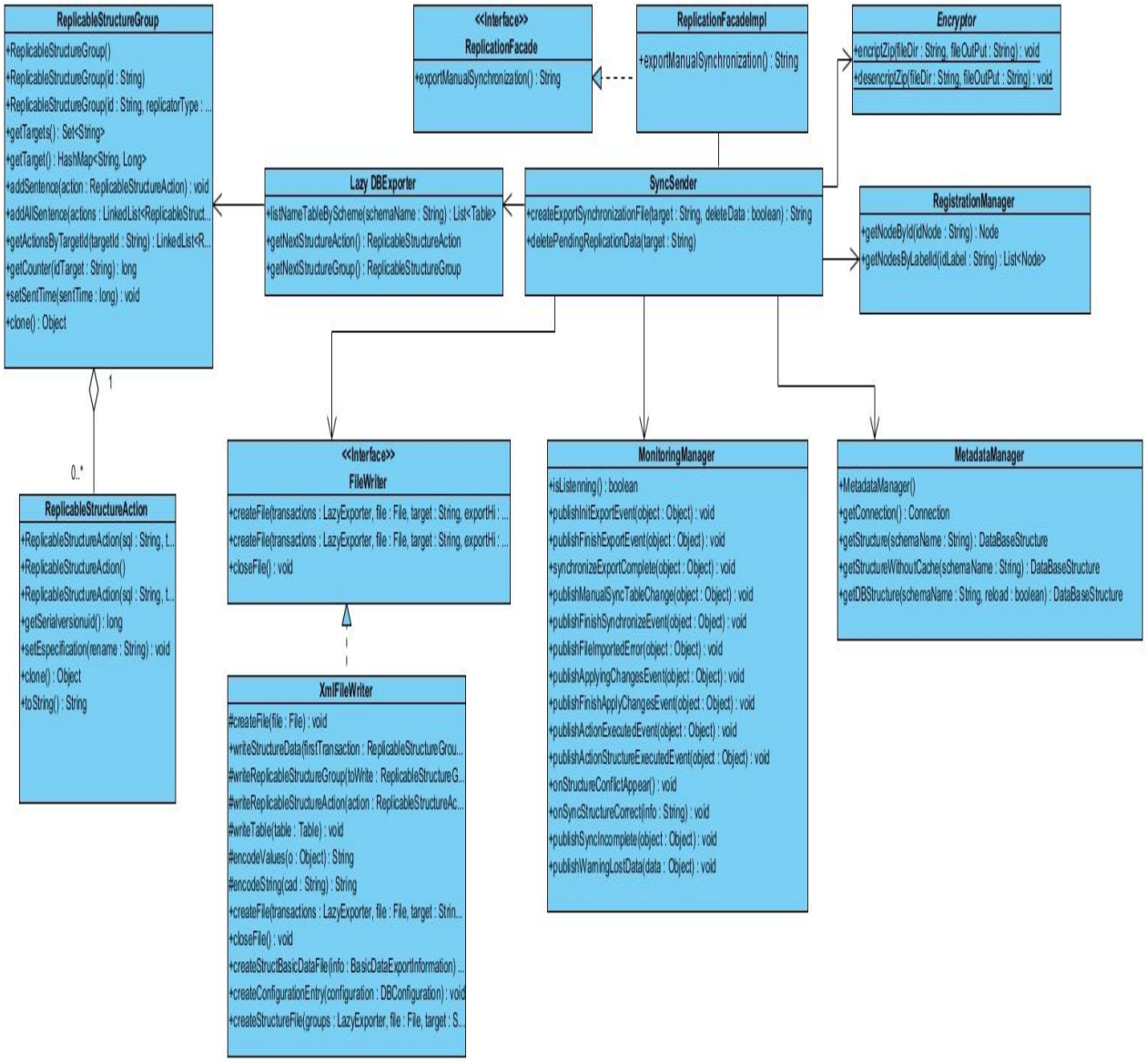


Figura 10 DC correspondiente al DP Exportar datos de sincronización de estructura

2.3.3 Descripción de las clases

A continuación se realiza una descripción de algunas de las clases de la solución, para las restantes ver (Anexos 13-19).

Tabla 5 Descripción de la clase SyncManager

Descripción de la clase SyncManager	
Nombre: SyncManager	
Tipo de clase: Controladora	
Atributos	**Tipos**

metadataManager	private
triggerReplicationManager	private
monitoringManager	private
synchronizationDao	private
transformation	private
Finish	private
Apply	private
remoteId	private
Responsabilidades	
Nombre:	syncStructure(LazyExporter remoteStructure,SyncConfig syncConfig):void
Descripción:	Sincroniza las estructuras locales y remotas.

Fuente: elaboración propia

Tabla 6 Descripción de la case SchemaComparison

Descripción de la clase SchemaComparison	
Nombre: SchemaComparison	
Tipo de clase: Modelo	
Atributos	**Tipos**
tablesA	private
tablesB	private
tablesResult	private
identColum	private
diferentColum	private
createSchema	private
ocurenceDiference	private
existFK	private
schemaName	private
Actions	private
metadataManager	private
syncManager	private
Dialect	private
actionCount	private
Responsabilidades	
Nombre:	createSchema():void

Descripción:	Crea la sentencia sql para crear esquema.
Nombre:	tableComparison():void
Descripción:	Compara las estructuras de dos tablas
Nombre:	columComparison(cu.uci.dbreplication.dbmetadata.Colum columA, cu.uci.dbreplication.dbmetadata.Colum columB): boolean
Descripción:	Compara las columnas de las tablas.
Nombre:	getTablesA():LinkedList<Table>
Descripción:	Obtiene el valor del atributo tablesA.
Nombre:	getTablesB():LinkedList<Table>
Descripción:	Obtiene el valor del atributo tablesB.
Nombre:	setTablesA(LinkedList<cu.uci.dbreplication.dbmetadata.Table> tablesA): void
Descripción:	Cambia la tabla A por la especificada por parámetro.
Nombre:	setTablesB(LinkedList<cu.uci.dbreplication.dbmetadata.Table> tablesB): void
Descripción:	Cambia la tabla B por la especificada por parámetro.
Nombre:	getMetadataManager ():*cu.uci.dbreplication.dbmetadata.MetadataManager*
Descripción:	Obtiene el metadataManager.

Fuente: elaboración propia

Tabla 7 Descripción de la clase ApplicatorSynchronizationDao

Descripción de la clase ApplicatorSynchronizationDao	
Nombre: ApplicatorSynchronizationDao	
Tipo de clase: Modelo	
Atributos	**Tipos**
Con	private
monitoringManager	private
actionForRemove	private
inputStreamToClose	private
filesTempsToDelete	private
Responsabilidades	
Nombre:	*getConnection():java.sql.Connection*
Descripción:	Obtiene la conexión de la base de datos.
Nombre:	openConnection():void
Descripción:	Abre la conexión de la base de datos.
Nombre:	execute(cu.uci.dbreplication.transportable.ReplicableStructureGroup group):

	void
Descripción:	Ejecuta los cambios de estructura en la base de datos.
Nombre:	shorStructureAction (LinkedList<cu.uci.dbreplication.transportable.ReplicableStructureAction >actions): LinkedList< cu.uci.dbreplication.transportable.ReplicableStructureAction >
Descripción:	Ordena las acciones que van a ser ejecutadas.
Nombre:	getDependencia(Table table): Set<String>
Descripción:	Obtiene las dependencias de la tabla especificada por parámetro.

Fuente: elaboración propia

Tabla 8 Descripción de la clase ReplicableStructureGroup

Descripción de la clase ReplicableStructureGroup	
Nombre: ReplicableStructureGroup.	
Tipo de clase: Modelo.	
Atributos	**Tipos**
Id	private
senderId	private
sentTime	private
replicatorType	private
REPLICATOR_TYPE_COMPARATION=0	public
REPLICATOR_TYPE_TRIGGERS=1	public
REPLICATOR_WITH_FUNCTION=2	public
actions	private
Targets	private
Responsabilidades	
Nombre:	addSentence(cu.uci.dbreplication.transportable.ReplicableStructureAction action): void
Descripción:	Adiciona a la lista de acciones de estructuras replicables la acción especificada por parámetro.
Nombre:	addAllSentence(LinkedList<cu.uci.dbreplication.transportable.ReplicableStructureAction > actions): void
Descripción:	Adiciona a la lista de acciones de estructuras replicables la lista de acciones especificada por parámetro.
Nombre:	getActions():LinkedList<cu.uci.dbreplication.transportable.ReplicableStructureAction>

Descripción:	Obtiene la lista de acciones de estructuras replicables.
Nombre:	setActions(LinkedList<cu.uci.dbreplication.transportable.ReplicableStructureAction> actions): void
Descripción:	Cambia la lista de acciones de estructuras replicables por la especificada por parámetro.

Fuente: elaboración propia

Tabla 9 Descripción de la clase ReplicableStructureAction

Descripción de la clase ReplicableStructureAction	
Nombre: ReplicationStructureAction	
Tipo de clase: Modelo	
Atributos	**Tipos**
Table	private
Colum	private
schemaName	private
Sql	private
tableName	private
column_name	private
Destination	private
executedByUser	private
otherEspecification	private
User	private
actionType	private
Replace	private
Responsabilidades	
Nombre:	getTable():cu.uci.dbreplication.dbmetadata.Table
Descripción:	Obtiene la tabla almacenada
Nombre:	setTable(table: cu.uci.dbreplication.dbmetadata.Table)
Descripción:	Cambia la tabla almacenada por la especificada por parámetro.
Nombre:	getColumnName():String
Descripción:	Obtiene el nombre de la columna almacenada.
Nombre:	setColumnName(columnName: String)
Descripción:	Cambia el nombre de la columna almacenada por la especificada por parámetro.

Nombre:	getActionType(): int
Descripción:	Obtiene el tipo de acción.
Nombre:	setActionType(actionType: int)
Descripción:	Modifica el tipo de acción almacenada.
Nombre:	getColum():cu.uci.dbreplication.dbmetadata.Colum
Descripción:	Devuelve la columna almacenada
Nombre:	setColum(cu.uci.dbreplication.dbmetadata.Colum)
Descripción:	Cambia la columna almacenada por la especificada por parámetro.
Nombre:	getSchemaName():String
Descripción:	Obtiene el nombre del esquema almacenado.
Nombre:	setSchemaName(String schemaName)
Descripción:	Cambia el esquema por el especificado por parámetro.
Nombre:	getTableName():String
Descripción:	Obtiene el nombre de la tabla almacenada.
Nombre:	setTableName(String tableName)
Descripción:	Cambia el nombre de la tabla por el especificado por parámetro.
Nombre:	getSql():String
Descripción:	Obtiene el sql de la acción.
Nombre:	setSql(String sql)
Descripción:	Cambia el sql por el especificado por parámetro.

Fuente: elaboración propia

2.4 Modelo de despliegue

Los diagramas de despliegue muestran la disposición física de los distintos nodos que intervienen en la composición de un sistema y el reparto de los programas ejecutables sobre estos nodos.

En la figura 6 se muestra el diagrama de despliegue del software REKO:

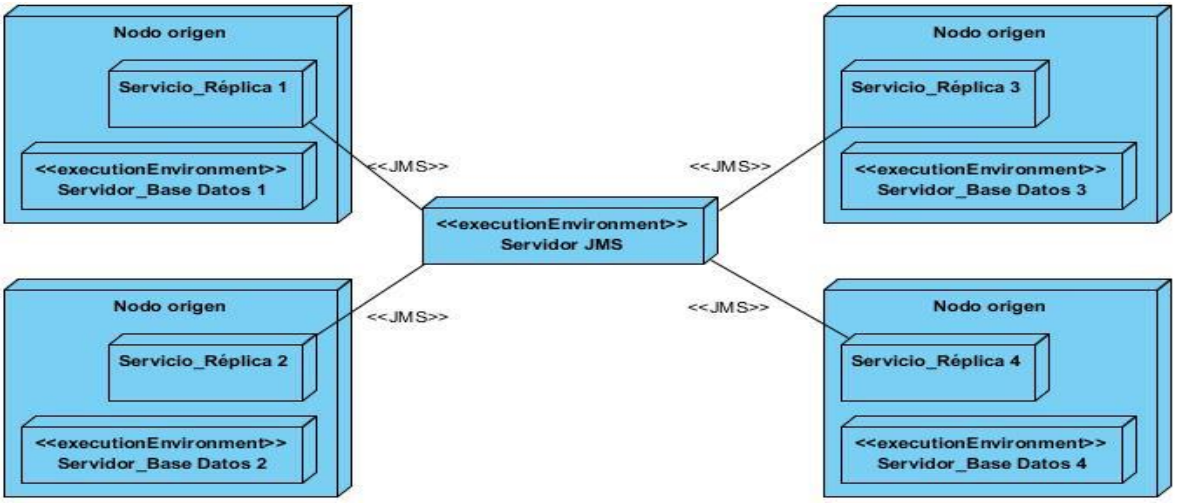


Figura 11 Diagrama de despliegue

2.5 Consideraciones parciales del capítulo

- Con la implementación de los requisitos funcionales se logrará un sistema capaz de dar solución al problema planteado.
- Se hace imprescindible extender las funcionalidades de los componentes Capturador, Distribuidor y Aplicador de cambios, principalmente el paquete *synchonization* pues hasta el momento su funcionamiento se basa en la sincronización de datos y ahora se extiende a la sincronización de estructuras.

Capítulo 3. Implementación y pruebas

A continuación se describen los diagramas de componentes, se muestran ejemplos de la implementación de las funcionalidades, así como el estilo de codificación empleados en la solución de estas y se realizan pruebas al sistema para demostrar la valides de la implementación.

3.1 Modelo de implementación

El modelo de implementación describe cómo los elementos de diseño se implementan en componentes (ficheros de código fuente, de código binario, ejecutable, scripts). Toma el resultado del modelo de diseño para generar el código final del sistema. (40; 41)

3.1.1 Diagramas de componentes

Un diagrama de componente describe la descomposición física del sistema de software en componentes (código fuente, binario, ejecutable, así como librerías dinámicas, páginas web, interfaces, un conjunto de relaciones de dependencia, generalización, asociación y realización), así como la relación que existen entre ellos en el sistema. (42)

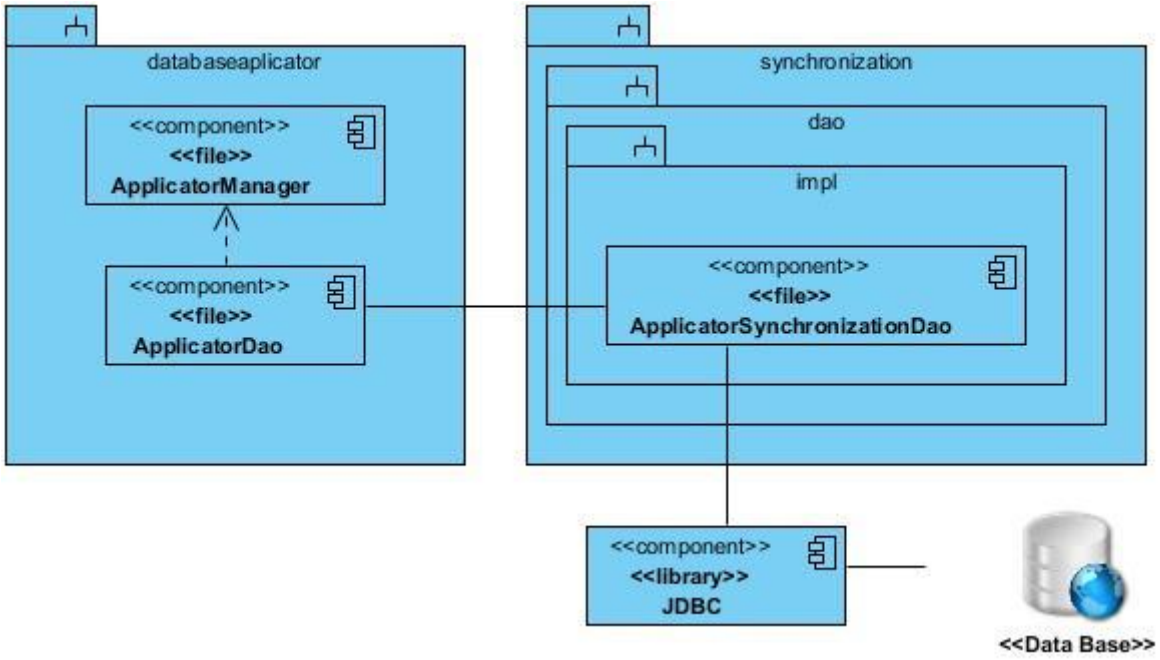


Figura 12 Diagrama de componente del subsistema aplicador de cambios

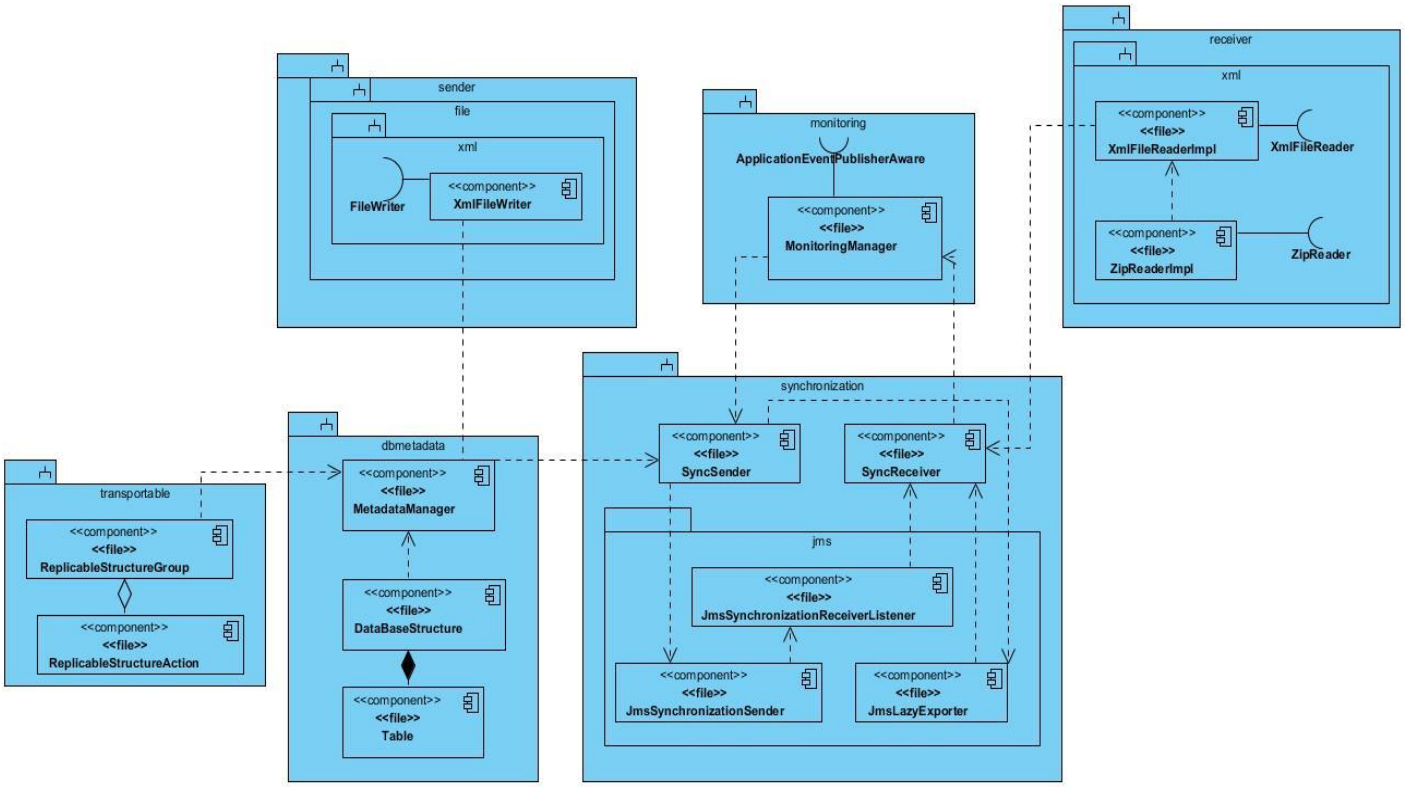


Figura 13 Diagrama de componente del subsistema distribuidor de cambios

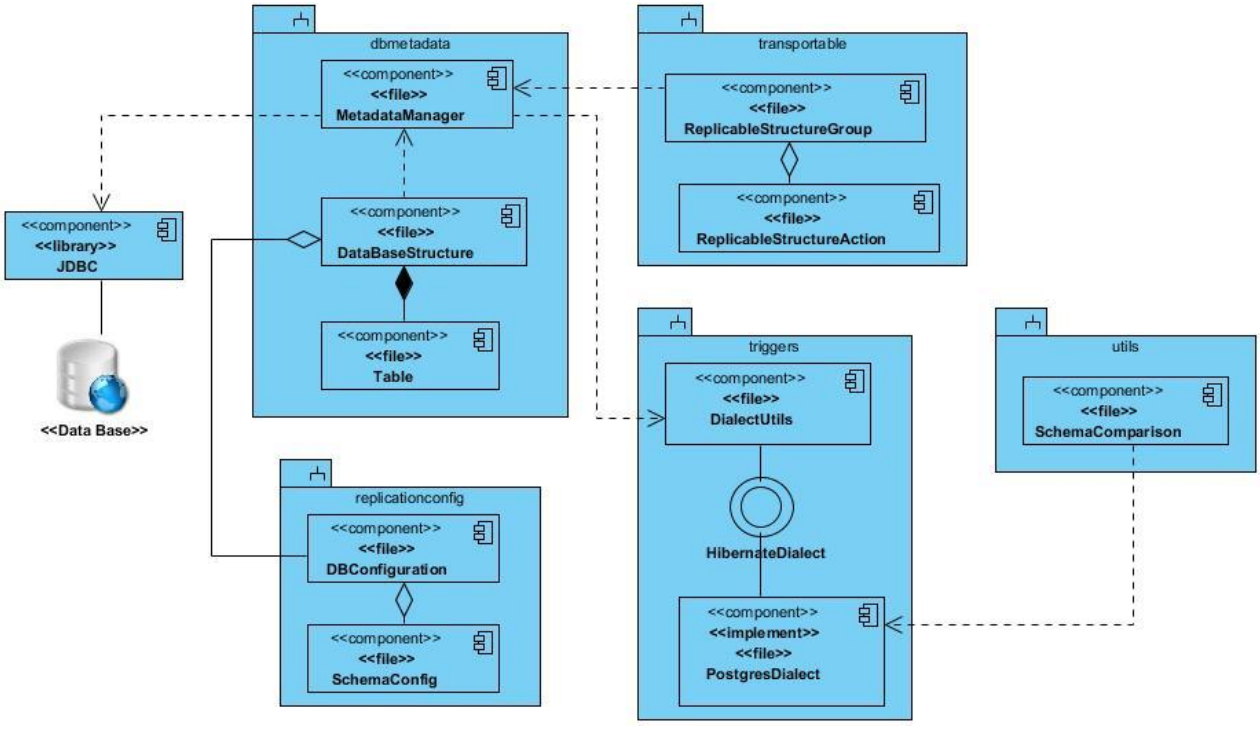


Figura 14 Diagrama de componente del subsistema capturador de cambios

3.2 Código fuente

A continuación se muestra un fragmento de código perteneciente a la clase *LazyDBExporter*, el código fuente corresponde al método *getNextStructureGroup*, que se encarga de crear los RSG que serán enviados hacia el nodo destino y el método *getNextStructureAction* se encarga de crear las RSA que serán almacenadas en los RSG.

Tabla 10 Código fuente del método que construye los RSG y el que construye los RSA

LazyDBExporter

```
public ReplicableStructureGroup getNextStructureGroup(){
513         ReplicableStructureGroup replicableStructureGroup = new ReplicableStructureGroup(getNewReplicableGrupId(),
514                 ReplicableStructureGroup.REPLICATOR_TYPE_COMPARATION, from);
515
516         //Capturando los esquemas configurados
517         Map<String, SchemaConfig> map = dbConfiguration.getSchemaConfigMap();
518         List<SchemaConfig> map1 = new LinkedList<>();
519         for(Entry<String,SchemaConfig> entry: map.entrySet()){
520             map1.add(entry.getValue());
521         }
522         //Construyendo RSG
523         if(countSchema < map1.size()){
524             List<Table> auxNameTableByScheme = listNameTableByScheme(map1.get(countSchema).getSchemaName());
525
526             for(int i1 = 0; i1 < MAX_STRUCT_ACTIONS_BY_GROUP && (countTables < auxNameTableByScheme.size());i1++){
527                 actualStructureTable = auxNameTableByScheme.get(countTables++);
528                 if(!actualStructureTable.getName().contains("reko_")) {
529                     ReplicableStructureAction structureAction = getNextStructureAction();
530                     structureAction.setSchemaName(map1.get(countSchema).getSchemaName());
531                     replicableStructureGroup.addSentence(structureAction);
532                     structureGroupCount++;
533                 }
534             }
535             if(countTables == auxNameTableByScheme.size()){
536                 countTables =0;
537                 countSchema++;
538             }
539             return replicableStructureGroup;
540         }
541         return null;
542
543     }
```

```
224     public ReplicableStructureAction getNextStructureAction() {
225
226         ReplicableStructureAction structureAction= null;
227
228         String sqlSelect = "";
229         try {
230             sqlSelect = DialectUtils.sentenceCreateTable(dialect, actualStructureTable);
231
232         } catch (Exception e) {
233             System.out.println("No se pudo tomar el SQL CREATE de la TABLA: "+actualStructureTable.getName());
234             // TODO: handle exception
235         }
236
237         structureAction = new ReplicableStructureAction(sqlSelect, actualStructureTable, ActionType.create_table);
238         structureActionCount ++;
239         JSONObject initExportObject = new JSONObject();
240         initExportObject.put("table", actualStructureTable.getName());
241         monitoringManager.publishInitExportEvent(initExportObject);
242
243         return structureAction;
244
245     }
```

Fuente: elaboración propia

El método *arriveRequestNextGroupSynchronizationMessage* de la clase *SyncReceiver* es el encargado de enviar los RSG solicitadas por el nodo remoto.

Tabla 11 Código fuente del método que envía los RSG

SyncReceiver

```
public void arriveRequestNextGroupSynchronizationMessage(String sendFrom,boolean isFirstCall,
                                                         boolean structure) throws DataAccessException, Excep
        System.out.println("Recibiendo peticion de grupo desde " + sendFrom);
        if (isFirstCall) {
            {
                LazyDBExporter exporter=null;
                syncSender.stopReplication(sendFrom);
                syncSender.deletePendingReplicationData(sendFrom);
                syncSender.resetLocalCount(sendFrom);
                Node node = registrationManager.getNodeById(sendFrom);
                DBConfiguration configuration = configurationManager
                        .getConfiguration((node.getLabel() == null) ? sendFrom
                                : node.getLabel().getId());
                if(!structure) {
                    exporter = new LazyDBExporter(DialectFactory
                            .getDialect(metadataManager), metadataManager,
                            triggerReplicationManager, configuration, sendFrom,
                            GeneralConfig.getNodeId(), monitoringManager,
                            transformationManager, true);
                }else{
                    exporter= new LazyDBExporter(DialectFactory
                            .getDialect(metadataManager), metadataManager,
                            triggerReplicationManager, configuration, sendFrom,
                            GeneralConfig.getNodeId(), monitoringManager,
                            transformationManager);
                }
                lazyDbExporters.put(sendFrom, exporter);
            }
        }
        {
            LazyDBExporter exporter = lazyDbExporters.get(sendFrom);
            Object group;

            if(!structure){
                group = exporter.getNextGroup();

                if (group == null ) {

                    System.out.println("null el grupo");
                    JSONObject exportInformation = new JSONObject();
                    exportInformation.put("insertActions", exporter
                            .getInsertActions());
                    exportInformation.put("deleteActions", 0);
                    exportInformation.put("groupsCount", exporter.getGroupsCount());
                    exportInformation.put("totalActions", exporter
                            .getInsertActions());
                    exportInformation.put("structure",false);
                    exportInformation.put("file", false);
                    lazyDbExporters.remove(sendFrom);
                    monitoringManager.synchronizeExportComplete(exportInformation);
                    syncSender.startReplication(sendFrom);
                }
                syncSender.sendNextGroup(sendFrom,group,false);
```

Fuente: elaboración propia

3.3 Pruebas al sistema

Las pruebas de software son un elemento crítico para la garantía de la calidad del sistema y representa una revisión final de las especificaciones, diseños y codificación. Las pruebas suelen ser básicamente un conjunto de actividades dentro del ciclo de vida del desarrollo de software y según el tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento del proceso de desarrollo. (43)

Con el objetivo de validar el correcto funcionamiento de las funcionalidades implementadas y la robustez de la solución, se realizaron pruebas de aceptación empleando del método caja negra, el método partición equivalente.

3.3.1 Pruebas de aceptación

Las pruebas de aceptación del usuario es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.(44)

Pruebas de caja negra

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, con el objetivo de demostrar que las funciones del sistema son operativas, que los datos válidos introducidos son aceptados y que las salidas que generan estos son correctas. Se enfocan más en los requisitos funcionales del software, permitiendo crear un conjunto de condiciones de entradas que ejercitarán por completo los requisitos funcionales del sistema. (42)

Partición equivalente

La partición equivalente es un método de prueba de caja negra que se basa en dividir el dominio de entrada de un programa en clases de datos a partir de los cuales pueden surgir casos de pruebas. Se centra en definir casos de prueba que descubran ciertas clases de errores, reduciendo así la cantidad total de casos de pruebas que hay que desarrollar. (42)

A continuación se muestra el caso de prueba aplicado al RF11, para los restantes ver (Anexos 27-30):

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Exportar configuración de sincronización con actualización.	El usuario realiza el proceso de exportar la configuración de sincronización con actualización de la BD correctamente.	2. El sistema muestra una lista de los nodos con configuración de réplicas. 4. El sistema habilita el botón "Exportar". 6. El sistema muestra la interfaz "Eliminar Información" con la interrogante "¿Desea eliminar los datos de réplica pendientes para el nodo seleccionado?", donde el usuario tendrá dos opciones con dos botones para su selección: "Si" y "No". 8. El sistema elimina los datos de réplica pendientes y muestra la interfaz "Exportar" donde se tiene el enlace "Descargue el fichero" y sección "Detalles de la exportación". 10. Según la configuración del navegador web, el sistema muestra una interfaz que permite guardar el fichero de configuración de sincronización en el directorio deseado del gestor de archivos o lo descarga automáticamente en una dirección predefinida. 12. El sistema descarga el	1. El usuario hace clic en la lista desplegable de configuraciones. 3. El usuario selecciona el nodo con configuración que se desea exportar. 5. El usuario hace clic en el botón "Exportar". 7. El usuario hace clic en el botón "Sí". 9. El usuario hace clic en el enlace "Descargue el fichero". 11. El usuario hace clic en el botón "Guardar" o "Aceptar" en dependencia de la interfaz del navegador. 13. El usuario hace clic en el botón "Cerrar".

		fichero de configuración de sincronización en el directorio seleccionado. 14. El sistema cierra la ventana y termina el proceso de exportación.	
EC 1.2 Exportar configuración de sincronización sin actualización.	El usuario realiza el proceso de exportar la configuración de sincronización sin actualización de la BD correctamente.	2. El sistema muestra una lista de los nodos con configuración de réplicas. 4. El sistema habilita el botón "Exportar". 6. El sistema muestra la interfaz "Eliminar Información" con la interrogante "¿Desea eliminar los datos de réplica pendientes para el nodo seleccionado?", donde el usuario tendrá dos opciones con dos botones para su selección: "Si" y "No". 8. El sistema no elimina los datos de réplica pendientes y muestra la interfaz "Exportar" donde se tiene el enlace "Descargue el fichero" y sección "Detalles de la exportación".	1. El usuario hace clic en la lista desplegable de configuraciones. 3. El usuario selecciona el nodo con configuración que se desea exportar. 5. El usuario hace clic en el botón "Exportar". 7. El usuario hace clic en el botón "No". 9. El usuario hace clic en el enlace "Descargue el fichero". 11. El usuario hace clic en el botón "Guardar" o "Aceptar" en dependencia de la interfaz del navegador.

		10. Según la configuración del navegador web, el sistema muestra una interfaz que permite guardar el fichero de configuración de sincronización en el directorio deseado del gestor de archivos o lo descarga automáticamente en una dirección predefinida. 12. El sistema descarga el archivo de configuración de sincronización en el directorio seleccionado. 13. El sistema cierra la ventana y termina el proceso de exportación.	13. El usuario hace clic en el botón "Cerrar".
EC 1.3 Exportar configuración de sincronización incorrectamente.	El usuario realiza el proceso de exportar la configuración de sincronización incorrectamente.	2. El sistema muestra la alerta "El valor especificado no es válido".	1. El usuario introduce datos en la lista desplegable.
EC 1.4 Cancelar la exportación de la configuración de sincronización.	El usuario cancela la interfaz de "Exportar" del proceso de exportar la configuración de sincronización.	2. El sistema muestra una lista de los nodos con configuración de réplicas. 4. El sistema habilita el botón "Exportar". 6. El sistema muestra la interfaz "Eliminar Información" con la interrogante "¿Desea eliminar los	1. El usuario hace clic en la lista desplegable de configuraciones. 3. El usuario selecciona la configuración que se desea exportar. 5. El usuario hace clic en

		datos de réplica pendientes para el nodo seleccionado?", donde el usuario tendrá dos opciones con dos botones para su selección: "Si" y "No". 8. El sistema elimina los datos de réplica pendientes y muestra la interfaz "Exportar" donde se tiene el enlace "Descargue el fichero" y el sección "Detalles de la exportación". 10. El sistema cierra la interfaz "Exportar" y termina el proceso de exportación.	el botón "Exportar". 7. El usuario hace clic en el botón "Sí"/"No". *9*. El usuario hace clic en el enlace "Cerrar".
EC 1.5 Cancelar la descarga del fichero de configuración de sincronización.	El usuario cancela la selección del directorio de descargar el fichero de exportación de la configuración de sincronización.	2. El sistema muestra una lista de los nodos con configuración de réplicas. 4. El sistema habilita el botón "Exportar". 6. El sistema muestra la interfaz "Eliminar Información" con la interrogante "¿Desea eliminar los datos de réplica pendientes para el nodo seleccionado?", donde el usuario tendrá dos botones para su selección: "Si" y "No". 8. El sistema elimina los datos de réplica pendientes y muestra la interfaz "Exportar" donde se tiene el enlace "Descargue el fichero" y la sección "Detalles de	1. El usuario hace clic en la lista desplegable de configuraciones. 3. El usuario selecciona la configuración que se desea exportar. 5. El usuario hace clic en el botón "Exportar". 7. El usuario hace clic en el botón "Sí"/"No". *9*. El usuario hace clic en el enlace "Descargue el fichero". 11. El usuario hace clic en el botón "Cerrar".

		la exportación". 10. Según la configuración del navegador web, el sistema muestra una interfaz que permite guardar el fichero de configuración de sincronización en el directorio deseado del gestor de archivos o lo descarga automáticamente en una dirección predefinida. 12. El sistema cierra la ventana y mantiene la interfaz que contiene el link "Descargar fichero".	
EC 1.6 Cancelar la eliminación de los datos pendientes de réplica en el proceso de exportar la configuración de sincronización.	El usuario cancela la interfaz "Eliminar Información" que elimina o no los datos de réplica pendientes referente del proceso de exportar la configuración de sincronización.	2. El sistema muestra una lista de los nodos con configuración de réplicas. 4. El sistema habilita el botón "Exportar". 6. El sistema muestra la interfaz "Eliminar Información" con la interrogante "¿Desea eliminar los datos de réplica pendientes para el nodo seleccionado?", donde el usuario tendrá dos botones para su selección: "Si" y "No". 8.El sistema cierra la interfaz "Eliminar Información" y se mantiene habilitado el botón "Exportar"	1. El usuario hace clic en la lista desplegable de configuraciones. 3. El usuario selecciona la configuración que se desea exportar. 5. El usuario hace clic en el botón "Exportar". 7. El usuario hace clic en el botón "Cancelar".

Resultados de las pruebas

Durante el proceso de prueba se llevaron a cabo tres iteraciones donde se probaron seis casos de pruebas. En una primera iteración se probaron todos los casos de pruebas centrando estas en validar el correcto funcionamiento de las funcionalidades implementadas.

En la tabla 12 se pueden observar los escenarios de pruebas realizados así como las no conformidades (3) detectadas durante la aplicación de las pruebas.

Tabla 12 Escenarios de pruebas

Descripción del escenario	Resultado esperado	Resultado obtenido	NC
Se realiza el proceso de sincronización y se encuentran en la BD 20 tablas.	El sistema construye un RSG con 20 RSA.	El sistema construyó un RSG con 15 RSA.	El sistema no construye satisfactoriamente los RSG.
Se realiza el proceso de sincronización y se encuentran en la BD 20 tablas incluyendo tres tablas de control del sistema.	El sistema construye un RSG con 17 RSA sin incluir las referentes a las tablas de control del sistema.	El sistema construyó un RSG con 20 RSA e incluyó las referentes a las tablas de control.	El sistema construye las RSA de las tablas de control al construir los RSG.
Se realiza el proceso de sincronización y se introdujeron 25 cambios de estructuras en la BD.	El sistema detecta los 25 cambios introducidos en la BD.	El sistema detectó 15 de los 25 cambios realizados en la BD.	El sistema no realiza el proceso de comparación de forma correcta.
Se realiza el proceso de sincronización y comparación, detectándose 25 cambios de estructuras en la BD.	El sistema aplica en la BD 25 cambios de estructuras.	El sistema aplicó en la BD 10 cambios de estructuras.	El sistema no es capaz de aplicar todos los cambios detectados en la BD correctamente.
Se realiza el proceso de exportar configuración de sincronización existiendo 20	El sistema escribe en el fichero de configuración los	El sistema escribe en el fichero de	El sistema no escribe correctamente los

tablas en la BD	20 RSG a exportar	configuración 10 RSG.	RSG en el fichero de configuración.
Se realiza el proceso de importación de configuración de sincronización de un fichero corrupto.	El sistema notifica que el archivo a importar se encuentra corrupto.	El sistema importa el archivo de configuración.	El sistema no valida correctamente la importación del archivo de configuración de sincronización.
Se realiza el proceso de sincronización exportando 20 tablas y se ejecuta la funcionalidad ver progreso.	El sistema muestra en la interfaz "Exportar" los datos de las estructuras que se están exportando en ese momento.	El sistema no muestra datos en la interfaz "Exportar".	El sistema no monitoriza correctamente las acciones de exportación.
Se realiza una configuración general para la sincronización de estructura.	El sistema guarda en el archivo de configuración general los datos configurados.	El sistema no guarda los datos configurados en el archivo de configuración general.	El sistema no es capaz de guardar una configuración general correctamente.

Fuente: elaboración propia

En la segunda iteración se volvieron a probar los seis casos de pruebas para verificar la correcta solución de las NC detectadas en la iteración anterior y se realizaron pruebas centradas en las interfaces de usuarios, durante esta iteración se detectaron 15 NC referentes a errores de redacción, faltas de ortografías y diseño de las interfaces las cuales fueron solucionadas de forma inmediata.

Al realizar la tercera iteración se probaron todos los casos de pruebas confeccionados y no se detectaron NC validando así el correcto funcionamiento de las funcionalidades de la solución desarrollada.

3.4 Resumen de pruebas

Durante el desarrollo de las pruebas a las funcionalidades de la solución desarrollada se detectaron en general 23 NC distribuidas en: 8 de funcionamiento, 5 de redacción y las restantes de diseño, las cuales fueron resueltas durante cada una de las iteraciones de pruebas realizadas. Con las pruebas se comprobó que el sistema es capaz de realizar el proceso de sincronización de estructura, validando la robustez de la solución desarrollada, así como el cumpliendo con los requerimientos planteados.

3.5 Consideraciones parciales del capítulo

- Las funcionalidades extendidas garantizan que en el módulo de sincronización se realice una correcta sincronización de datos del proceso de réplica de estructura que permita la homogeneidad entre las BD que intervienen en el proceso de réplica.
- La realización de las pruebas validaron las funcionalidades de la solución implementada estando acordes a los requerimientos planteados.
- Los resultados arrojados por las pruebas reflejan la robustez de la solución desarrollada.

CONCLUSIONES GENERALES

- La elaboración del marco teórico fue la fuente para recopilar información y sintetizar los conocimientos necesarios que garantizaron dar inicio a la investigación y proponer una correcta solución.
- La solución desarrollada garantiza la sincronización de datos del proceso de réplica de estructura de forma manual y automática en un ambiente distribuido, con lo cual minimiza la generación de errores que detienen el proceso de réplica.
- Las pruebas realizadas a la solución validaron su correcto funcionamiento y su aceptación por parte del cliente.

RECOMENDACIONES

- Agregar la posibilidad al sistema de sincronizar los datos del proceso de réplica de estructura entre bases de datos MySQL, Oracle y Microsoft SQL Server.

BIBLIOGRAFÍA REFERENCIADA

1. URBANO, R. *Oracle® Database Advanced Replication 11g Release 2 (11.2)* [Consultado el: 2 de diciembre de 2014]. Disponible en: https://docs.oracle.com/cd/E11882_01/server.112/e10706.pdf.

2. RIVERA, M. I. M. *Implementación de Bases de Datos Empresariales* [Consultado el: 2 de diciembre del 2014 Disponible en: http://www.grin.com/es/e-book/230205/implementacion-de-bases-de-datos-empresariales.

3. ARLEY, R. C. Fragmentación de datos en base de datos distribuidas. 7 de febrero 2013, vol. 16, nº Disponible en: http://ccp.ucr.ac.cr/bvp/pdf/desarrollohumano/bd_distribuidas-rca.pdf.

4. VELASCO, M. C. J. E. P. *Taller de Base de Datos* [en linea], [Consultado el: 6 de diciembre de 2014]. Disponible en: http://www.prograweb.com.mx/tallerBD/0201Esquema.html.

5. MAZILU, M. C. *Database Systems Journal.* 2010, Disponible en: http://www.dbjournal.ro/archive/2/2.pdf#page=34. ISBN 2069–3230

6. ORACLE. *Guía de administración del sistema de Oracle Solaris Cluster* 2010, Disponible en: http://docs.oracle.com/cd/E22734_01/html/821-2812/datarep.html.

7. CORPORATION, M. *Sincronizar y resolver conflictos en un conjunto de réplicas (MDB)* [Consultado el: 3 de diciembre de 2014]. Disponible en: http://office.microsoft.com/es-mx/access-help/sincronizar-y-resolver-conflictos-en-un-conjunto-de-replicas-mdb-HP003069834.aspx.

8. TECNOLOGÍA, D. D. I. Y. *Sincronización de Datos* [en línea], [Consultado el: 15 de enero de 2015]. Disponible en: http://www.alegsa.com.ar/Dic/sincronizar%20datos.php.

9. 2014, S. S. *Sincronizar Datos* [en línea], [Consultado el: 15 de enero de 2015]. Disponible en: http://msdn.microsoft.com/es-es/library/ms151793.aspx.

10. AUTORES, C. D. *DDL y DML* [en linea], [Consultado el: 3 diciembre de 2014]. Disponible en: http://www.ub.edu.ar/catedras/ingenieria/Datos/capitulo4/cap42.htm.

11. SYMMETRISDB. *SymmetrisDB* [en linea], [Consultado el: 3 diciembre de 2014]. Disponible en: http://www.symmetricds.org/about/overview.

12. CEPEDA. *Replicadores* [en linea], [Consultado el: 3 diciembre de 2014]. Disponible en: www.buenastareas.com/ensayos/Replicadores/3124122.html.

13. AUTORES, C. D. *Oracle Streams* [Consultado el: 25 de junio de 2015]. Disponible en: http://docs.oracle.com/cd/B28359_01/server.111/b28321/strms_over.htm#strms_overview.

14. COMPANIONI SARDIÑA, Y. A. R., ALBIN; HERNÁNDEZ SUAREZ,EYVIS; VELAZQUEZ VIZCAY,ADRIEL; NUÑEZ RIOS,MADIELENNIS; PÉREZ MATOS, LEISER *Reko: una solución de réplica para sistemas de bases de datos relacionales distribuidos.* Cuba: IV Simposio Informática y Comunidad, 2012.

15. PIMENTEL GONZÁLEZ, L. A. H. S., EIVYS; BERMÚDEZ PEÑA,ROLANDO; PÉREZ ALFONSO,DAMIÁN; MENA RODRIGUEZ,LUIS EDGARDO; ALFONSO VALDÉS,JAVIER; GOMEZ PEÑA, ANGELA GLORIA *Reko Replicador.* Cuba: Universidad de las Ciencias Informáticas, 2009.

16. AUTORES, C. D. *Metodología de desarrollo para la Actividad productiva de laUCI.* 2014.

17. ---. *Visual Paradigm* Última actualización: 27 de abril 2015. [Consultado el: 27 de junio de 2015]. Disponible en: http://www.visual-paradigm.com/.

18. ORACLE. *JEE* [en linea], [Consultado el: 3 de diciembre de 2014]. Disponible en: http://www.oracle.com/technetwork/java/javaee/overview/index.html.

19. *Wath is Java ?* [en linea], [Consultado el: 3 de diciembre de 2014]. Disponible en: http://searchsoa.techtarget.com/definition/Java.

20. RUSS MILES, K. H. *Learning UML 2.0.* Editado por: "O'reilly Media, I., 2006. 2006, ISBN 0596555229, 9780596555221.

21. WALLS, C. *Spring in Action.* Manning Publications, 2011. vol. 3, 426 p. ISBN 9781935182351.

22. *JMS*. [en linea], [Consultado el: 3 de diciembre 2014 de 2014]. Disponible en: http://docs.oracle.com/javaee/6/tutorial/doc/bncdr.html.

23. *ActiveMQ*. [en linea], [Consultado el: 3 diciembre de 2014]. Disponible en: http://activemq.apache.org/.

24. PARI, J. *Spring Tool Suit Instalacion* [en linea], [Consultado el: 3 de diciembre de 2014]. Disponible en: http://www.slideshare.net/juliopari/spring-tool-suite-instalacion.

25. *JDBC*. [en linea], [Consultado el: 3 de diciembre de 2014]. Disponible en: http://www.alegsa.com.ar/.

26. *Sobre Postgresql*. [Consultado el: 3 de diciembre de 2014]. Disponible en: http://www.postgresql.org.es/sobre_postgresql.

27. EIVYS HERNÁNDEZ SUAREZ , Y. C. S., LUIS E. MENA RODRÍGUEZ ,ALBIN AMAT REYES *REKO : Herramienta de réplica de datos para sistemas distribuidos. .* Disponible en: http://publicaciones.uci.cu/index.php/SC.

28. CRAIG LARMAN, P. H. *MODELO DEL DOMINIO.* 2003,

29. IEEE-STD-830-1998. *ESPECIFICACIONES DE LOS REQUISITOS DEL*

SOFTWARE. 1998, Disponible en: http://www.ctr.unican.es/asignaturas/is1/IEEE830_esp.pdf.

30. FUENTES, M. D. C. G. MATERIAL DIDÁCTICO NOTAS DEL CURSO ANÁLISIS DE REQUERIMIENTOS. 2011, nº Disponible en: http://www.cua.uam.mx/pdfs/conoce/libroselec/Notas_Analisis_Requerimiento.pdf. ISSN 978-607-477-442-9.

31. SUAZA, K. V. *Definición de equivalencias entre historias de usuario y especificaciones en UN-LENCEP*

para el desarrollo ágil de software. 2013, Disponible en: http://www.bdigital.unal.edu.co/11631/1/1128431389.2014.pdf.

32. GARLAN, D. *ICSE '00 Proceedings of the Conference on The Future of Software Engineering* ACM New York, NY, USA ©200, 2000, 91-100 p. ISBN 1-58113-253-0.

33. ENGINEERS, T. I. O. E. A. E. *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems* [en linea],[Consultado el :20 de febrero del 2015], publicado el: 9 Octùbre de 2000, última actualización: 9 Octùbre. Disponible en: http://cabibbo.dia.uniroma3.it/ids/altrui/ieee1471.pdf. ISBN 7381-2519-9.

34. CHRISTOPHER ALEXANDER, M. J., MURRAY SILVERSTEIN,SARA ISHIKAWA. *A Pattern Language* [Consultado el: 21 de febrero 2015 ISBN 978-0195019193

35. WILSON, J. M. A. Patrones de Arquitectura. 2011, nº [Consultado el: 21 de febrero 2015]. Disponible en: http://jmaw.blogspot.com/2011/04/h2-margin-bottom-0.html.

36. *Ingeniería de Software Basada en Componentes*. [Consultado el: 21 de febrero 2015 de 2015]. Disponible en: https://sites.google.com/site/lawebdelsoftware/ingenieria-de-software-1/unidad-vi.

37. ADRIANA SANDRA ALMEIRA, V. P. C. *Arquitectura de Software:Estilos y Patrones* marzo 2007, publicado el: 21 de febrero 2015 de 2007, última actualización: 21 de febrero 2015. Disponible en: http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_1/Arquitectura_de_Software/Tesina_Arquitectura_de_Soft.pdf.

38. SOMMERVILLE. *Ingeniería del Software.* 2005. ISBN 84-7829-074-5.

39. GUTIERREZ, D. *UML Diagramas de Paquetes.* 2011, Disponible en: http://www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf.

40. SANZ, M. L. *Tema 4g:Proceso Unificado:Implementación.* 2010, Disponible en: http://www.kybele.etsii.urjc.es/docencia/IS_LADE/2010-2011/Material/%5BIS-LADE-2010-11%5DTema4g.PUD.Implementacion.pdf.

41. BUEMO, A. W. R. Y. S. G. *Ingeniería de software: el proceso para el desarrollo de software.* 2008, Disponible en: ftp://ftp.itam.mx/pub/alfredo/PAPERS/WeitzenfeldGuardatiComputacion2008.pdf.

42. PRESSMAN, R. S. *Ingería del software Un enfoque práctico.* 6ta ed. 2014, ISBN 8448132149.

43. KRAFFCZYK, J. F. F. Realidad virtual aplicada al tratamiento del trastorno de lateralidad y ubicación espacial 2003, nº Disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/fuentes_k_jf/capitulo4.pdf.

44. PRESSMAN, R. *Ingeniería de Software, un enfoque práctico.* 2002. vol. Quinta edición, ISBN 8448132149.