Facultad 4

Centro de Consultoría y Desarrollo de Arquitecturas Empresariales

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

Proceso de convergencia entre mecanismos de réplica para el Replicador de datos REKO

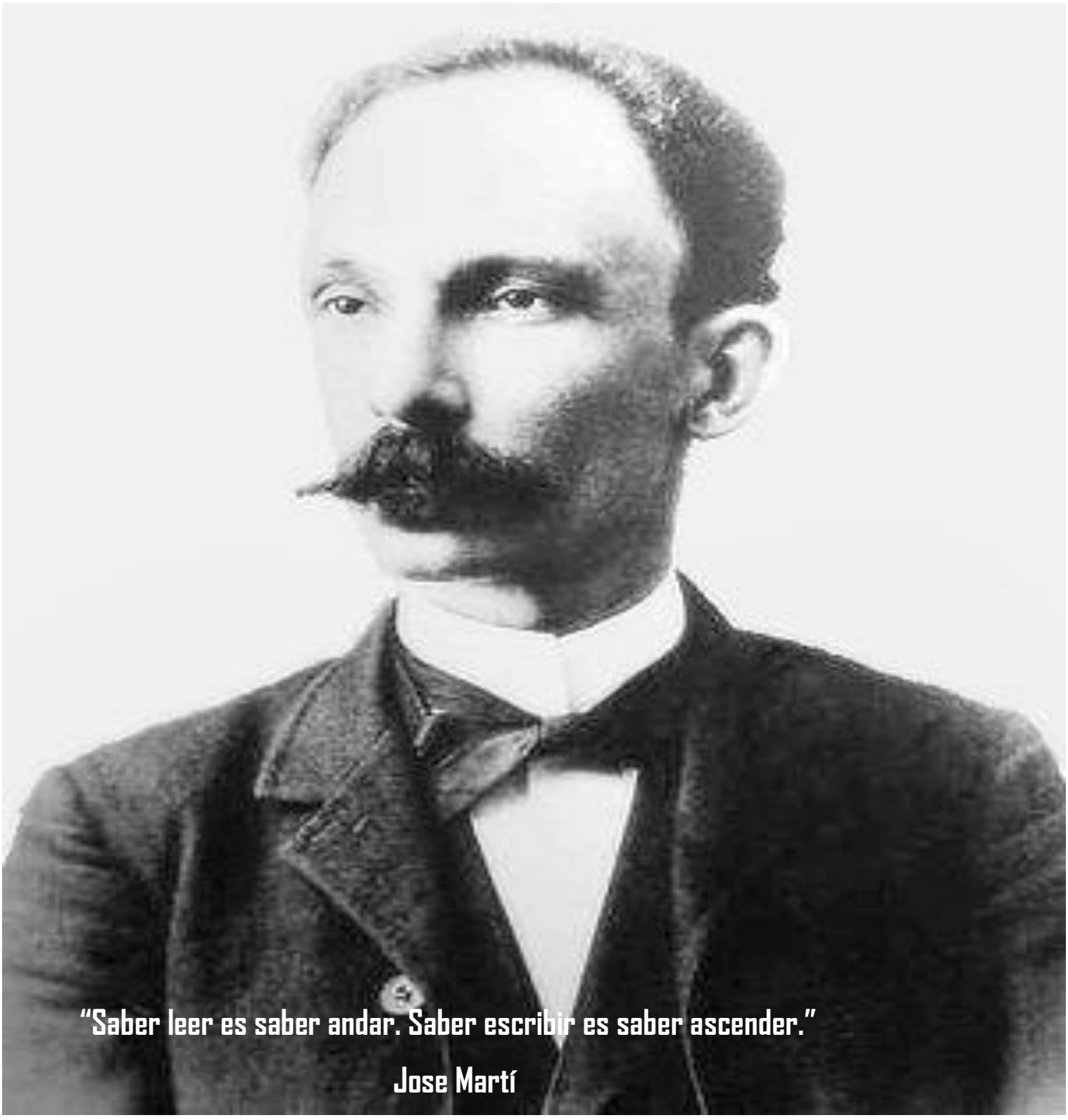
Autor: Jessilié Paz Lara

Tutor(es): Ing. Gloria Raquel Leyva Jerez, Ing. Nayibi Martín Peña

Co-tutor: Ing. Yoan Céspedes Williams

La Habana, junio
"2018"

PENSAMIENTO



DECLARACIÓN DE AUTORÍA

Declaro ser autor del presente trabajo de diploma y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste, firmo la presente a los __ días del mes de _____ del año _____.

Firma del autor:

_____

Jessilié Paz Lara

Firma de los tutores:

_____ _____

Ing. Gloria Raquel Leyva Jerez Ing. Nayibi Martín Peña

AGRADECIMIENTOS

Quiero agradecerle en primer lugar a mi mamá por la confianza y por siempre creer en mí, porque ese voto que tuviste conmigo me ha permitido explorar y aventurarme para cumplir mis metas.

A mis abuelos gracias por sus consejos y palabras de aliento por su apoyo incondicional.

A mis tíos y primos que a pesar de la distancia y no poder compartir en este día tan especial con ustedes, los amo, por su ayuda, cariño y comprensión han sido parte fundamental de mi vida.

A mi hermanita bella que es la personita que más amo en esta vida, gracias por estar ahí y aguantarme cuando peleo.

A mi tío José Luis por su apoyo, cariño y por siempre estar ahí a mi lado.

A mi abuela Yuya y mis tías por apoyarme y siempre estar preocupadas por mí.

En general a toda mi familia gracias.

A mi novio gracias por entrar a mi vida por hacerme tan feliz y llenarme de alegría por compartir tu tiempo conmigo por que durante estos años de carrera has sabido apoyarme para continuar y nunca renunciar, gracias por tu amor incondicional.

A Dalquerine, Shirley, Yailyn, Daylén y Lamis por compartir grandes momentos y estar siempre a mi lado.

A mi amigo Lijandy por compartir momentos de alegría, por todo el apoyo, el cariño y demostrarme que siempre podré contar con él.

A mi suegra y cuñado por su cariño y confianza.

A mis tutoras y co-tutor por su valiosa guía y asesoramiento en la realización de la tesis.

En fin gracias a todas las personas que ayudaron directa e indirectamente en la realización de mi tesis.

DEDICATORIA

Le dedico este gran logro a las personas que más amo en esta vida a mi mamá, a mi tía Gisela, mis tíos Gerardo y José Luis, a mis abuelos, a mi hermana y a mi novio, por haberme apoyado en todo momento, por sus consejos, sus valores, por la motivación constante que me ha permitido ser una persona de bien, pero más que nada, por su amor.

RESUMEN

El Replicador de datos REKO es un producto totalmente nacional e independiente con capacidad autónoma de soporte técnico, desarrollado sobre tecnologías libres que le permite al usuario, con el menor esfuerzo, cubrir las necesidades fundamentales de replicación, tales como: sincronización, transferencia de datos entre diversas localizaciones, la centralización de la información en una única localización, protección y recuperación. Aunque el Replicador de datos REKO posee un mecanismo de captura de acciones DML (Lenguaje de Manipulación de Datos) y DDL (Lenguaje de Definición de Datos), la herramienta no reconoce automáticamente los cambios estructurales dentro de sus configuraciones, por tanto, no se mantiene una correspondencia entre la estructura existente de la Base de Datos (BD) y la estructura mostrada en la configuración de réplica. La ejecución simultánea de los procesos de réplica de datos y réplica de estructura generan conflictos e inconsistencias en las BD debido a que no está definido un orden de ejecución para evitar que los cambios DML arriben primero que los cambios DDL. Se empleó la metodología AUP (Proceso Unificado Ágil) en su variante UCI (Universidad de las Ciencias Informáticas) para guiar el desarrollo del Proceso de convergencia entre mecanismos de réplica para el Replicador de datos REKO, apoyándose en los artefactos que propone. Como resultado se obtiene una solución informática que permite la convergencia de los mecanismos de réplica, además de permitirle al usuario actualizar las configuraciones de réplica para cada cambio de estructura existente.

Palabras claves: Base de datos, configuración, convergencia, réplica, cambio de estructuras.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La gestión en las bases de datos ha evolucionado desde una aplicación informática especializada, hasta una parte esencial de un entorno informático moderno y como resultado, el conocimiento acerca de los sistemas de base de datos se ha convertido en una parte esencial en la enseñanza de la informática 1).

Con el surgimiento de los Sistemas Gestores de Base Datos (SGBD) se ha logrado un avance significativo para el almacenamiento de la información. La posibilidad de compartir datos, eliminar inconsistencias, mejorar la seguridad; ha transformado a los SGBD en la principal fuente de almacenamiento en el campo informático.

La distribución geográfica de las organizaciones y entidades ha impulsado el empleo de los Sistemas de Bases de Datos Distribuidas (SBDD).

Los SBDD proporcionan servicios de archivo remoto a clientes, de forma transparente, estando los clientes, servidores, y dispositivos de almacenamiento entre las máquinas de un sistema distribuido, permitiendo la compartición de datos entre los usuarios. Además, un sistema de archivos distribuido busca una serie de objetivos, como es la disponibilidad, el rendimiento, la transparencia o la tolerancia a fallos 2).

En la actualidad, la movilidad de los usuarios es cada vez mayor y las empresas tienden a dispersarse mediante pequeñas oficinas, bien sean en oficinas fijas o mediante oficinas de obra, lo que genera una gran cantidad de puntos de generación de datos. Todos estos datos han de protegerse, y la implementación de sistemas de backup fiables no suele ser sencilla, dado que por norma general los sistemas completos de backup son complejos, además de necesitar personal preparado para su configuración, puesta en marcha y mantenimiento. Una de las soluciones es recopilar los datos y centralizarlos en un punto, donde se protegerán mediante los diversos sistemas de backup. La recopilación de datos, se puede realizar mediante sistemas de replicación, desde sistemas sencillos de copia programada, hasta sistemas más complejos de réplica en tiempo real 2).

Actualmente Cuba incursiona en esta rama de las tecnologías y la gestión de datos, fortaleciendo los conocimientos y aplicaciones existentes en pos de alcanzar un desarrollo tecnológico superior. Entre las entidades pioneras del desarrollo de las Tecnologías de la Información y las comunicaciones (TIC) en el país se encuentra la Universidad de las Ciencias

Informáticas (UCI), la cual cuenta con una alta responsabilidad en el desarrollo socioeconómico del país y en la informatización de la sociedad cubana.

En el Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE) perteneciente a la UCI se desarrolla el Replicador de datos REKO, producto totalmente nacional e independiente con capacidad autónoma de soporte técnico que permite ser desplegado en entornos empresariales, el cual constituye una solución de réplica en ambientes distribuidos que tiene como objetivo, brindar una herramienta multiplataforma que le permita al usuario, con el menor esfuerzo, cubrir las necesidades fundamentales de replicación, tales como: sincronización, transferencia de datos entre diversas localizaciones y la centralización de la información en una única localización, protección y recuperación, así como satisfacer otras necesidades relacionadas con la distribución de datos entre los gestores más populares, dígase PostgreSQL, MySQL, Oracle y Microsoft SQL Server.

Aunque el Replicador de datos REKO cuenta con un mecanismo de captura de acciones DML y DDL, la herramienta no reconoce automáticamente los cambios estructurales dentro de sus configuraciones, por tanto, no se mantiene una correspondencia entre la estructura existente en la base de datos y la estructura mostrada en la configuración de réplica. Por lo que, resulta necesario reiniciar el sistema para que este actualice sus metadatos y por ende los cambios sean visibles por el software en el módulo Configuración de réplica.

Sumado a esto se encuentra el hecho de que el Replicador de datos REKO realiza el proceso de réplica de datos por la actualización de cambios tras la ejecución de sentencias DML y réplica de estructura por la actualización de cambios tras la ejecución de sentencias DDL, la ejecución de ambos procesos es programada por el usuario lo cual puede generar conflictos.

Por tanto, la ejecución simultánea de ambos procesos también desencadena conflictos, que generan a su vez inconsistencias precisamente porque no hay definido un orden de ejecución y el tiempo de captura de acciones DDL puede ser mayor que el de DML. Por lo que, puede darse el caso de que el cambio sobre los datos arribe primero que el estructural.

A partir lo anteriormente expuesto, se plantea como **problema de la investigación:** ¿Cómo evitar las incoherencias en los datos al detectar cambios estructurales en los procesos de réplica en el Replicador de datos REKO?

Teniendo como **objeto de estudio:** proceso de réplica en sistemas de base de datos distribuidos, enmarcado en el **campo de acción:** proceso de convergencia de réplica en replicadores.

Se define como **objetivo general:** desarrollar un mecanismo de ejecución simultánea de procesos de réplica para el Replicador de datos REKO, que permita evitar las incoherencias en los datos al detectar cambios estructurales.

Para darle cumplimiento al objetivo general planteado se definen los siguientes **objetivos específicos:**

1. Elaborar el marco teórico referencial de la investigación para organizar los conocimientos científicos adquiridos.
2. Análisis y diseño de la solución aplicando la metodología y herramientas seleccionadas para lograr un producto acorde a las necesidades del cliente.
3. Implementación y validación de la solución propuesta mediante la aplicación de pruebas, para el aseguramiento de la calidad del producto.

A continuación, se desagrega por cada objetivo las siguientes **tareas de investigación**:

Tareas relacionadas con el objetivo 1:

- Realización del marco conceptual para precisar los principales conceptos que se emplean en la investigación.
- Elaboración del estado del arte, para sintetizar los resultados alcanzados en la revisión bibliográfica e indagaciones realizadas relacionadas con el tema.
- Análisis del entorno y dominio del Replicador de datos REKO.
- Selección de la metodología 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.

Tareas relacionadas con el objetivo 2:

- Definición de los requisitos funcionales y no funcionales para cumplir los objetivos trazados.

- Diseño del proceso para detallar las historias de usuarios, diagramas de clases y diagramas de paquetes con sus respectivas descripciones.

- Descripción de la arquitectura para la implementación de la propuesta de solución.

Tareas relacionadas con el objetivo 3:

- Implementación de las funcionalidades que dan cumplimiento a los requisitos identificados.
- Realización de pruebas para valorar la calidad del producto implementado.
- Valoración de los resultados obtenidos a partir de las pruebas aplicadas y objetivos planteados.

Teniendo en cuenta el objetivo general se plantea como **idea a defender** que, con el desarrollo de un proceso de convergencia entre mecanismos de réplica, se evitará las incoherencias en los datos al detectar cambios estructurales en los procesos de réplica en el Replicador de datos REKO.

Como parte del cumplimiento de las tareas definidas se emplearon los siguientes métodos científicos:

Métodos Teóricos

Histórico-Lógico: Se estableció un estudio bibliográfico y una base sólida de fundamentos teóricos que permitió relacionar el análisis documental y estado del arte para conocer con precisión cómo han ido evolucionando los sistemas de réplica, investigando como es tratado el tema en la actualidad, con el objetivo de detectar aspectos positivos a tener en cuenta en la solución que se propone desarrollar.

Analítico-Sintético: Fue utilizado para profundizar en el tema que se desarrolla. Permitió concretar los elementos más importantes relacionados con los sistemas de réplica, posibilitando descubrir sus características generales y las relaciones esenciales mediante el análisis.

Métodos Empíricos:

Observación: Se utilizó durante toda la etapa de identificación y análisis de recopilación de datos e información que fundamenten su contribución en el análisis del campo de acción y del objetivo de estudio, y para analizar cómo funciona el Replicador de datos REKO, y la información que se genera.

Análisis de documentos: Se estudiaron los documentos relacionados con la metodología y los métodos a implementar.

El presente trabajo se encuentra estructurado en 3 capítulos:

Capítulo 1. FUNDAMENTACIÓN TEÓRICA este capítulo constituye la base teórica de la investigación, se abordan los principales conceptos a tratar durante su desarrollo, así como la metodología utilizada y el análisis de las herramientas para el desarrollo de la aplicación.

Capítulo 2. PROPUESTA DE SOLUCIÓN incluye la descripción, análisis y diseño de la solución que se propone para darle respuesta al problema planteado, se describen los requisitos funcionales y no funcionales, las historias de usuarios, además de los diagramas de clases y los patrones de diseño empleados.

Capítulo 3. IMPLEMENTACIÓN Y PRUEBAS AL SISTEMA muestra la implementación de las funcionalidades, se realiza el diseño de los diagramas de componentes, se describen las pruebas realizadas al sistema con el objetivo de validar que la solución funcione de manera correcta.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

La periódica distribución y recopilación de la información ubicada geográficamente distante resulta una engorrosa tarea a realizar. Con el uso de los sistemas de réplica de datos bien configurado y que explote al máximo sus potencialidades se puede automatizar y lograr buenos resultados en este sentido. En este capítulo se tratan los conceptos fundamentales respecto a los sistemas de replicación. Se caracterizan los sistemas de replicación más usados. Además, se describe y explica la selección de las tecnologías usadas para la implementación del mecanismo de convergencia de los procesos.

1.1 Conceptos asociados al objeto de estudio del problema

A continuación, se expone un grupo de conceptos y terminologías de todos aquellos elementos que intervienen en el proceso de la investigación a partir de la revisión de publicaciones de varios autores.

1.1.1 Base de datos

El autor Manuel Sierra[1] plantea: “Una base de datos es un sistema informático a modo de almacén. En este almacén se guardan grandes volúmenes de información. Permite automatizar el acceso a la información y poder acceder a esta de forma rápida y fácil, además de poder efectuar cambios de manera más eficiente”□3).

Rosa M. Mato[2] declara:“ Una base de datos es un conjunto de datos interrelacionados, almacenados con carácter más o menos permanente en la computadora, o sea, una colección de datos variables en el tiempo” □4).

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 Manuel Sierra, analista y programador informático con experiencia en la tecnología JAVA, PHP y C#.

2 Rosa M. Matos, Licenciada en Cibernética Matemática, Profesora Auxiliar, del Centro de Estudios de Ingeniería de Sistemas (CEIS), del Instituto Superior Politécnico José Antonio Echeverría (ISPJAE), Ciudad de La Habana

1.1.2 Sistemas distribuidos

El autor Fredy Carranza[3] considera que: “Una Base de Datos Distribuida es un conjunto de múltiples bases de datos lógicamente relacionadas las cuales se encuentran distribuidas entre diferentes sitios interconectados por una red de comunicaciones, los cuales tienen la capacidad de procesamiento autónomo lo cual indica que puede realizar operaciones locales o distribuidas” 5).

1.1.3 DDL (*Data Definition Languaje* -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 6).

Posee tres operaciones básicas:

- **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.

1.1.4 DML (*Data Manipulation Language*- Lenguaje de Manipulación de Datos)

Un Lenguaje de Manipulación de Datos es un lenguaje proporcionado por los SGBD que permite a los usuarios llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado 6).

Los comandos DML son:

- **Insert**: una sentencia INSERT de SQL agrega uno o más registros a una tabla en una BD relacional.
- **Update**: una sentencia UPDATE de SQL es utilizada para modificar los valores de un conjunto de registros existentes en una tabla.

3 Fredy Carranza Athó, Ingeniero Informático, Licenciado y Máster en Ciencias de la Computación.

- **Delete**: una sentencia DELETE de SQL borra uno o más registros existentes en una tabla.

Si en un sistema de información, durante la generación de reportes, se ejecuta un comando DML sobre la base de datos, se modifican los datos almacenados. La actualización de estos datos es posible utilizando un replicador de datos para que replique los cambios. Por el contrario, si se ejecuta un comando DDL sobre la base de datos, la estructura de la base de datos es quien cambia instantáneamente. Estas diferencias de estructuras, que se generan con respecto al resto de las bases de datos distribuidas, es posible actualizarlas con un componente que permita replicar los cambios de estructura o con un replicador de datos que posea o adquiera esta funcionalidad 6).

1.1.5 Replicación

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

Randy Urbano[4] en el año 2010 expresó que: “la replicación en el contexto de los SGBD, se entiende como el proceso de copiar y mantener objetos de una base de datos, entiéndase tablas, secuencias y/o *triggers*[5], etcétera, en múltiples bases de datos, de tal forma que los cambios realizados localmente, sean enviados a las bases de datos remotas y luego sean aplicados” 8).

1.1.6 Coherencia

Para la autora Dharma Rojas[6] la coherencia de los datos:“…está referida a la idoneidad de los datos para ser cambiados de diferentes maneras y para diferentes usos” 9).

José Antonio Ocampo[7] expresa que: “…es la idoneidad de los datos para ser combinados en forma fiable de diferentes maneras y distintos usos, procedan de una fuente única o investigaciones estadísticas de diversas naturalezas” 10).

[4] Escritor técnico de Oracle.

[5] Conocido como desencadenador. Procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación *DML (Lenguaje de Modelado de Datos)*.

6 Dharma Rojas, Universidad de Los Andes (Venezuela), Administración y contaduría

1.1.7 Convergencia

La Real Academia Española (2000) define el concepto de convergencia en: unirse o tender a unirse dos o más cosas en un mismo punto 11).

1.1.8 Configuración

Es lo que determine cómo, a través de qué medios y con qué recursos funcionará un elemento. Se utiliza para adaptar una aplicación o un elemento al resto de los elementos del entorno y a las necesidades específicas del usuario. Es una tarea esencial antes de trabajar con cualquier nuevo elemento 12).

1.2 Replicadores homólogos

En el siguiente acápite se realizó un estudio de replicadores homólogos para conocer como realizan los procesos de réplica y sus configuraciones, en busca de una posible propuesta de solución para el mecanismo de convergencia de los procesos de réplica en el Replicador de datos REKO.

1.2.1 DBMoto Cloud Edition

DBMoto realiza la actualización y la replicación de datos en tiempo real para su servidor empresarial y las necesidades de replicación de escritorio. Con DBMoto, los usuarios pueden definir las replicaciones y transformaciones de los datos aplicando reglas de negocios y mapeo. Las reglas de negocio pueden ser aplicadas a través de *scripts*[8], los cuales son usados para filtrar datos o para agregar lógica de negocios a las replicaciones. Estos son generalmente implementados a través de una generación automática de eventos durante la replicación con DBMoto. Los *scripts* pueden configurar las vistas de las tablas de origen y destino, añadir actividad a los *logs* y actualizar las tablas de destino. También incluye un poderoso generador de expresiones para la transformación de datos 12).

---

7 José Antonio Ocampo, Secretario General de las Naciones Unidas para Asuntos Económicos y Sociales, y ex Secretario Ejecutivo de la CEPAL.

[8] Son un conjunto de instrucciones generalmente almacenadas en un archivo de texto que deben ser interpretados línea a línea en tiempo real para su ejecución; esto los distingue de los programas (compilados), pues estos deben ser convertidos a un archivo binario ejecutable.

Fue especialmente diseñado para replicar y capturar datos en modo "*Refresh*" y "*Mirroring*".

- Modo ***Refresh*** (*Snapshot*[9]): lee los datos, aplica las reglas de mapeo definidas por el administrador y escribe los resultados en la BD de destino.
- Modo ***Mirroring*** (Captura de datos): replicación desde el origen al destino en tiempo real, basado en los *log/journal*[10] transaccionales de las BD y aplicando captura de datos para minimizar el tráfico de datos 12).

Algunas de sus características son:

- Soporta gestores de BD como: Oracle, Microsoft SQL Server, MySQL, Server Enterprise, PostgreSQL, y MS Access.
- Modos de replicación y captura de datos: *Refresh* y *Mirroring*.
- Replicación de datos desde origen al destino, usando la nube (*cloud*) y *web service* para transferencia de datos.
- Soporte para múltiples BD (origen y destino).
- No requiere programación en las plataformas de BD de origen y destino.
- Habilidad para leer *logs* de transacciones de BD (para hacer más eficiente el acceso a los datos).
- Completo *log* de reportes y accesibilidad.
- Poderosa herramienta visual con información sobre el estatus de la replicación.
- Creación automática de tablas de destino 12).

1.2.2 SymmetricDS

SymmetricDS es un software de replicación asíncrona[11] soporta múltiples suscripciones y de sincronización bidireccional, usa tecnologías web y de bases de datos para replicar tablas entre bases de datos relacionales 14).

[9] Copia instantánea de volumen.

[10] Mecanismo por el cual un sistema informático puede implementar transacciones.

[11] Tecnología de replicación que proporciona tolerancia a fallos en servidores y almacenamiento en red. Trabaja capturando los cambios en los ficheros en el nivel del sistema operativo. Una vez que los datos han sido escritos en el sitio de almacenamiento primario, nuevas escrituras a ese sitio pueden ser aceptadas, sin tener que esperar que el sitio de almacenamiento secundario o remoto también termine su escritura.

A pesar de contar con grandes potencialidades para ser desplegado en diferentes ambientes de replicación resulta un poco engorrosa la administración y control del SymmetricDS, debido a que no cuenta con una interfaz que posibilite la administración visual de todos los objetos y componentes que contiene el mecanismo. La selección del mecanismo de replicación adecuado para poder montar un ambiente de replicación genérico es compleja, pues cada mecanismo posee características enfocadas a resolver un problema en específico 14).

SymmetricDS utiliza los términos nodo registrador y nodo hijo para su configuración. Esto implica que el nodo registrador posee toda la configuración del mecanismo de replicación, la cual es enviada automáticamente a todos los nodos hijos que son registrados al mismo. Por su parte los nodos hijos deben ser registrados a un nodo registrador, para cargar la configuración del ambiente de réplica almacenada en el nodo registrador, además de realizar la cargada inicial de los datos para la sincronización entre los nodos SymmetricDS involucrados en un ambiente de réplica. El mecanismo de replicación crea un conjunto de tablas de metadatos en la base de datos de la réplica, sobre las que se almacena la configuración de los componentes de SymmetricDS, para posteriormente ser enviada esta configuración de los nodos registradores a los nodos hijos (14).

Algunas de sus características son:

- Licencia de código abierto GPL[12](Licencia Pública General, en inglés *General Public License*), multiplataforma.
- Pensado para tener nodos desconectados por largos periodos de tiempo.
- Es capaz de replicar los cambios estructurales de la base de datos que puedan surgir.
- Flexibilidad a la hora de declarar las reglas de replicación de los datos.
- Facilidad de configuración, basado en *triggers*.
- Alto rendimiento en ambientes con bajo ancho de banda y problemas de conexión 14).

1.2.3 Oracle Streams

Oracle Streams basa su funcionamiento en mensajes, unidad básica de la información que comparte. Los mensajes pueden ser originados, transformados o enviados por el gestor de Oracle u otras herramientas externas que interactúen con Oracle Streams, lo cual brinda una

[12] Es una licencia de software libre, los usuarios de un programa con licencia GPL son libres para usarlo, acceder al código fuente, modificarlo y distribuir los cambios; siempre que redistribuyan el programa completo.

alta flexibilidad y variadas funcionalidades. La unidad básica para capturar los cambios se denomina LCR[13] (Unidad básica de Captura de Cambios, en inglés *Logical Change Records*), de la cual existen dos tipos: *row* LCR relativa a los cambios por operaciones DML y DDL LCR relativa a los de operaciones de DDL, como su nombre lo indica. Oracle Streams ofrece transformaciones basadas en reglas, que son aplicados a estos mensajes, para la replicación entre bases de datos con estructuras diferentes. Existen dos tipos de transformaciones 15):

Para realizar la réplica se basa en tres componentes:

- **Captura**: los cambios por acciones DDL o DML son capturados del registro de rehacer y luego son empaquetados en LCR. Los datos y los eventos pueden ser cambiados o formateados por un conjunto predefinido de reglas antes de ser empaquetados en un LCR.
- **Puesta en escena**: los LCR se almacenan en el entorno de ensayo hasta que un abonado los recoge para ser utilizado o consumido. El abonado puede ser otro entorno de ensayo o una aplicación de usuario.
- **Consumo**: durante el consumo, los LCR se recogen y se aplican en una base de datos. Se modifican los LCR antes de ser aplicados en la base de datos 15).

Las reglas declarativas abarcan transformación para *row* LCR, incluyendo el cambio de nombre de un esquema, cambiar el nombre de una tabla, agregar una columna, cambiar el nombre de una columna y supresión de una columna. Oracle Streams realiza transformaciones declarativas internamente, sin necesidad de invocar PL/SQL[14](Lenguaje de Procedimientos/ Lenguaje de Consulta Estructurado, en inglés *Procedural Language/Structured Query Language)*. Las reglas personalizadas requieren una función definida por el usuario PL/SQL para realizar la transformación que es invocada por Oracle Streams. Una regla personalizada basada en reglas de transformación puede modificar cualquiera de LCR o mensajes de usuario como el tipo de datos de una columna concreta de una LCR 15).

1.2.4 Pglogical

[13] **Son cargas útiles de mensajes que contienen información sobre cambios en una base de datos.**

[14] **Es un lenguaje de programación incrustado en Oracle**

Pglogical es un sistema de replicación lógico implementado completamente como una extensión de PostgreSQL. Totalmente integrado, no requiere desencadenantes ni programas externos. Esta alternativa a la replicación física es un método altamente eficiente para replicar datos utilizando un modelo de publicación / suscripción para la replicación selectiva. Utiliza las características de Decodificación Lógica agregadas por 2ndQuadrant (y disponibles desde PostgreSQL 9.4). Funciona aún más rápido con PostgreSQL 9.5 y versiones posteriores, con una baja sobrecarga tanto para el proveedor como para el suscriptor 16).

Pglogical se basa en gran medida en las características introducidas como parte del desarrollo de BDR[15](Replicación Bidireccional), que incluye 16):

- Decodificación lógica.
- Ranuras de replicación.
- Trabajadores de fondo estático.
- Origen de la replicación.
- Cometer sellos de tiempo.
- Mensajes lógicos WAL.

Beneficios de usar Pglogical 16)

- Replicación sincrónica.
- Resolución de conflictos configurable.
- Posibilidad de convertir el modo de espera físico en replicación lógica.
- Puede publicar datos de PostgreSQL a un suscriptor Postgres-XL.
- Las secuencias pueden ser replicadas.
- Sin desencadenantes significa carga de escritura reducida en el proveedor.
- Ninguna re-ejecución de SQL significa una sobrecarga y latencia reducidas para el suscriptor.
- El suscriptor no se encuentra en la recuperación en espera activa, por lo que puede usar tablas temporales, no registradas o normales.
- No es necesario cancelar consultas para permitir que la réplica continúe la reproducción.

[15] **Es un sistema de replicación asíncrono multi-maestro para PostgreSQL.**

- El suscriptor puede tener diferentes usuarios y seguridad, diferentes índices, diferentes configuraciones de parámetros.
- Replicar solo una base de datos, o un subconjunto de tablas, conocidas como Conjuntos de Replicación.
- Replicar en todas las versiones o arquitecturas de PostgreSQL, permitiendo actualizaciones de tiempo de inactividad bajo o nulo.

1.2.5 Microsoft SQL Server

En Microsoft SQL Server se admite la replicación, la captura de datos modificados (CDC) y el seguimiento de cambios (CT) de Grupos de disponibilidad AlwaysOn. Microsoft SQL Server proporciona los siguientes tipos de replicación para usarlos en las aplicaciones distribuidas 17):

Replicación transaccional

La replicación transaccional se implementa con el Agente de instantáneas, el Agente de registro del LOG y el Agente de distribución de SQL Server. El Agente de instantáneas prepara archivos de instantáneas que contienen esquemas y datos de las tablas y objetos de base de datos publicados, almacena los archivos en la carpeta de instantáneas y registra los trabajos de sincronización en la base de datos de distribución del distribuidor. El Agente de registro del LOG supervisa el registro de transacciones de cada base de datos configurada para la replicación transaccional y copia las transacciones marcadas para ser replicadas desde el registro de transacciones a la base de datos de distribución, que actúa como una cola de almacenamiento y reenvío confiable. El Agente de distribución copia los archivos de instantáneas iniciales de la carpeta de instantáneas y las transacciones almacenadas en las tablas de la base de datos de distribución a los suscriptores 17).

Replicación de mezcla

La replicación de mezcla inicia con una instantánea de los objetos y datos de una base de datos de publicaciones. Los cambios de datos y las modificaciones de esquema posteriores que se lleven a cabo en el publicador y en los suscriptores se controlan mediante desencadenadores. El suscriptor se sincroniza con el publicador cuando están conectados a la red e intercambian todas las filas que han cambiado entre el publicador y el suscriptor desde la última vez que se produjo la sincronización 17).

Replicación de instantáneas

La replicación de instantáneas distribuye los datos exactamente como aparecen en un momento específico en el tiempo y no supervisa las actualizaciones de los datos. Cuando se

produce la sincronización, se genera la instantánea completa y se envía a los suscriptores. No se realiza ningún seguimiento de los cambios incrementales. No obstante, si el conjunto de datos que se está replicando es muy grande, será necesaria una cantidad importante de recursos para generar y aplicar la instantánea 17).

Captura de datos modificados (CDC)

Las bases de datos habilitadas para la captura de datos modificados (CDC) pueden aprovechar las ventajas de Grupos de disponibilidad AlwaysOn para asegurarse de que no solo la base de datos sigue estando disponible en caso de error, sino que los cambios en las tablas de la base de datos se siguen supervisando y depositando en las tablas de cambios de CDC. El orden en que se configuran CDC y Grupos de disponibilidad AlwaysOn no es importante. Las bases de datos habilitadas para CDC se pueden agregar a Grupos de disponibilidad AlwaysOny las bases de datos que son miembros de un grupo de disponibilidad AlwaysOn se pueden habilitar para CDC. Sin embargo, en ambos casos, la configuración de CDC se realiza siempre en la réplica principal prevista o actual 17).

Seguimiento de cambios (CT)

Una base de datos habilitada para el seguimiento de cambios (CT) puede formar parte de un grupo de disponibilidad AlwaysOn. No se necesita ninguna configuración adicional. Las aplicaciones cliente de seguimiento de cambios que usan las funciones con valores de tabla (TVF) de CDC para tener acceso a los datos modificados también necesitarán poder encontrar la réplica principal después de la conmutación por error. Si la aplicación cliente se conecta mediante el nombre de agente de escucha del grupo de disponibilidad, las solicitudes de conexión siempre se dirigirán correctamente a la réplica principal actual 17).

1.2.6 Replicador de datos REKO

Es una herramienta de réplica de datos entre BD. Fue diseñado e implementado completamente usando herramientas Open Source y librerías de clases con licencia gratuita. Permite que dos BD puedan replicar entre sí aunque se encuentren en subredes distintas y utilicen protocolos de transmisión diferentes, la única restricción es que todos los nodos deben estar conectados a un servidor central JMS (Servicio de Mensajería de Java, en inglés *Java Message Service*) o en caso de un *clúster*[16] de servidores JMS. Con el empleo de herramientas libres y la metodología de desarrollo AUP (Proceso Unificado Ágil) 18).

Permite la creación de configuraciones para las acciones DDL donde se define la captura de los cambios de estructuras sobre toda la información. Cuando en la BD se genera cualquier cambio de estructura sobre la acción que es especificada en la configuración, todos los

[16] Grupo de múltiples ordenadores unidos mediante una red de alta velocidad, de tal forma que el conjunto es visto como un único ordenador, más potente que los comunes de escritorio.

cambios son replicados, por lo que los cambios no deseados por el cliente son enviados hacia las instancias que intervienen en el proceso de réplica de estructura 19).

El Replicador de datos REKO cuenta con el módulo de Configuración de Réplica donde permite la creación de configuraciones para las acciones DDL (Lenguaje de Definición de Datos, del inglés *Data Definition Language*), donde se define la captura de los cambios de estructura sobre toda la información.

En el módulo Configuración de Réplica por cada configuración, se registran las configuraciones independientes de los esquemas que se desean replicar. Cada esquema puede ser configurada para replicar según la acción que se efectúe sobre el: creación, modificación y/o eliminación.

- **Creación:** para especificar que se desean replicar todas las tablas sobre el esquema seleccionado.

- **Modificación:** especifica que se replicarán los cambios referentes a la ejecución de una acción ALTER sobre las tablas de la base de datos.

- **Eliminación:** se replicarán todos los cambios de eliminación de tabla realizados en la base de datos.

Principales funcionalidades que ofrece el Replicador de datos REKO en su versión 4.0 18).

- Soporta la réplica de acciones a través de *triggers*.
- Soporte para réplica de datos en ambientes con conexión y sin conexión.
- Soporte para réplica de datos entre los gestores: PostgreSQL, Oracle, MySQL y Microsoft SQL Server.
- Soporte para réplica entre bases de datos con diferentes estructuras.
- Monitoreo en tiempo real de los datos de réplica.
- Réplica de estructura programada para PostgreSQL, se utiliza para iniciar la captura y envío de los cambios de esquemas.
- Seguridad de los datos que se envían con encriptación SSL[17](Capa de Conexión Segura, en inglés *Secure Sockets Layer*).

[17] Son los protocolos de seguridad de uso común que establecen un canal seguro entre dos ordenadores conectados a través de Internet o de una red interna.

- Interfaz de administración de réplica.

1.3 Resultados de la investigación

Teniendo en cuenta las características que presentan los sistemas homólogos anteriormente estudiados y la forma en que se realizan los procesos de réplica se concluye que estos no satisfacen en su totalidad los requerimientos básicos trazados, puesto que:

- Para el caso de Microsoft SQL Server el uso de software privativo no está alineado a las políticas del uso de software libre que presenta el país, puesto a que la adquisición de este tipo de licencia resulta difícil y costosa para Cuba, además como es una herramienta de Microsoft SQL Server, la réplica está definida únicamente para sistemas operativos de Microsoft.
- Las herramientas como DBMoto Cloud Edition, Pglogical y Oracle Streams, por sí solas no cuentan con un proceso o solución que pueda ser integrado al Replicador de datos REKO para establecer un mecanismo de convergencia en los procesos de réplicas.
- Para trabajar con SymmetricDS resulta necesario que los usuarios posean un nivel avanzado sobre el tema debido a que las configuraciones se realizan mediante comandos en la consola de administración.
- No se posee el código de la implementación en algunas de las fuentes consultadas.
- La información que se ofrece sobre estos sistemas no es abundante a excepción de algunos como SymmetricDS, por lo que resulta engorroso conocer a ciencia exacta el funcionamiento interno de los procesos que se realizan.

Por las limitantes antes expuestas se hace necesario dedicar esfuerzo en el desarrollo de una solución para el Replicador de datos REKO, que independice al país de soluciones propietarias y contribuya a la independencia tecnológica.

1.4 Metodología de desarrollo

Para guiar el proceso de desarrollo de la solución planteada se empleará la metodología AUP (Proceso Unificado Ágil, en inglés *Agile Unified Process*) adaptada al ciclo de vida definido para la actividad productiva, debido a que es la metodología establecida por la universidad para los proyectos de desarrollo como es el caso del Replicador de datos REKO, proyecto al cual se le integrará la solución propuesta en su próxima versión. Por ende, el proyecto requiere que el flujo de trabajo y artefactos generados como parte de la solución, se correspondan al desarrollo del proyecto 20).

1.4.1 Proceso Unificado Ágil (AUP)

El Proceso Unificado Ágil de Scott Ambler del inglés *Agile Unified Process* es una versión simplificada de RUP (Proceso Unificado de Desarrollo, del inglés *Rational Unified Process*). Esta metodología 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. Además, aplica técnicas ágiles incluyendo TDD (Desarrollo Dirigido por Pruebas, del inglés *Test Driven Development*), modelado ágil, gestión de cambios ágil, y refactorización de BD para mejorar la productividad. En la variación de AUP para la UCI se definen las siguientes fases, las cuales se aplicarán para guiar el desarrollo del presente trabajo 21).

- **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 21).
- **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 elabora la arquitectura, 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 21).
- **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto 21).

1.5 Herramientas, tecnologías y lenguajes de desarrollo

En este acápite se tratan los conceptos relacionados con las herramientas y tecnologías del entorno de réplica del Replicador de datos REKO que han sido empleadas desde versiones anteriores, debido que utilizarlas provoca que se disminuya considerablemente la curva de aprendizaje y el tiempo de desarrollo. Previendo problemas de compatibilidad e integración si se emplean otras herramientas.

1.5.1 Lenguajes de programación

Java

Lenguaje de programación orientado a objeto (*OOP*, por sus siglas en inglés), robusto, de código abierto e independiente de la plataforma. Es capaz de gestionar la memoria automáticamente, posee mecanismos de seguridad incorporados, los cuales limitan el acceso a recursos de las máquinas donde se ejecuta e incorpora herramientas de documentación. Es multihilos lo que permite dividir el trabajo de un programa en diferentes hilos de ejecución 22).

1.5.2 Herramientas de programación

Eclipse KEPLER

Es una plataforma de desarrollo de código abierto basada en *Java*, tiene un conjunto de complementos, incluidas las Herramientas de Desarrollo de *Java* (JDT, por sus siglas en inglés). Su uso no se limita al lenguaje *Java*, incluye soporte para los lenguajes de programación como C/C++ y COBOL 24).

1.5.3 Sistema Gestor de Base de Datos

PostgreSQL v 9.5 es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD (Distribución de Software Berkeley, en inglés *Berkeley Software Distribution*) y con su código fuente disponible libremente. Es el SGBD de código abierto más potente del mercado. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. Cuenta con un rico conjunto de tipos de datos, permitiendo además su extensión mediante tipos y operadores definidos y programados por el usuario 25 26).

1.5.4 Tecnologías de desarrollo

Spring v 2.5

Es un framework [18]que proporciona un modelo de programación y configuración completo para aplicaciones de empresa moderna basada en Java en cualquier tipo de plataforma de despliegue. Framework Open Source que proporciona un marco de trabajo para el desarrollo

[18] Es una estructura real o conceptual destinada a servir de soporte o guía para la construcción de algo que expande la estructura en algo útil.

de aplicaciones J2EE[19](*Java Enterprise Edition*). Está basado en el uso de ficheros planos JavaBeans para la lógica de aplicación y archivos XML[20] (Lenguaje de Marcas Extensible, en inglés *eXtensible Markup Language*) para la configuración. Se utiliza para la inyección de dependencias, o sea, para establecer la comunicación entre la capa de presentación y la capa modelo a través de parámetros. También se basa en un conjunto de módulos que proporcionan todo lo necesario para desarrollar una aplicación empresarial 27).

PrimeFaces 6.0

Es una librería de componentes para JavaServer Faces (JSF) de código abierto que cuenta con un conjunto de componentes enriquecidos que facilitan la creación de las aplicaciones web. PrimeFaces está bajo la licencia de Apache License V2. Una de las ventajas de utilizar PrimeFaces, es que permite la integración con otros componentes 28).

Las principales características de Primefaces son:

- Soporte nativo de Ajax, incluyendo Push/Comet.
- Útil para crear aplicaciones web para móviles.
- Compatible con otras librerías de componentes, como JBoss RichFaces
- Uso de javascript no intrusivo (no aparece en línea dentro de los elementos, sino dentro de un bloque <script>) (28).

1.5.5 Herramientas de modelado

Visual Paradigm v 8.0 es una herramienta CASE (Ingeniería de Software Asistida por Computadora, en inglés *Computer Aided Software Engineering*) que propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades, aunque también existe una alternativa libre y gratuita de este software. La versión Visual Paradigm UML (Lenguaje de Modelado Unificado, del inglés *Unified Modeling Language*) 6.4 Community Edition ha sido diseñado para una

[19] Es una plataforma para el cómputo empresarial a partir de la cual es posible el desarrollo profesional de aplicaciones empresariales distribuidas sobre una arquitectura multicapa.

[20] Es un subconjunto de SGML (Estándar Generalised Mark-up Language), simplificado y adaptado a Internet.

amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos 29).

1.5.6 Lenguaje de modelado

Lenguaje Unificado de Modelado v 2.0 (*UML* por sus siglas en inglés) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar elementos conceptuales como lo son procesos de negocio y funciones de sistema, además de sucesos concretos como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables 30).

1.5.7 Servidor de mensajería

Apache ActiveMQ v 5.9.0 es un popular, potente y rápido servidor de mensajería de código abierto que se encuentra distribuido bajo la licencia de Apache y soporta gran variedad de lenguajes de programación como Java, C[21] , C++, C#[22] , Python[23] , entre otros 31).

1.5.8 Servidor de aplicación

Apache Tomcat v 7.0.47 es un servidor web multiplataforma que funciona como contenedor de servlets[24]. Dicho servidor es mantenido y desarrollado por miembros de la fundación y voluntarios independientes, los cuales tienen libre acceso al código fuente bajo los términos establecidos por la Fundación de Software Apache, del inglés *Apache Software Foundation* 32).

Servicio de Mensajería Java 1.0

La API[25](*Application Programming Interface*) *Java Message Service* (JMS por sus siglas en inglés) es el estándar de mensajería que permite a los componentes de aplicaciones basados

[21] Lenguaje de programación orientado a la implementación de sistemas operativos.

[22] Lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft.

[23] Lenguaje de programación interpretado y multiparadigma.

[24] Programa Java que se ejecuta dentro de un servidor Web. Reciben y atienden las solicitudes de los clientes web.

[25] Es el conjunto de subrutinas, funciones que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

en la plataforma JEE crear, enviar, recibir y leer mensajes. Hace posible la comunicación confiable de manera síncrona y asíncrona 33).

1.6 Consideraciones parciales del capítulo

Una vez realizada la investigación científica y haber analizado los principales aspectos de este capítulo, se puede concluir que:

- Algunos de los principales sistemas similares al Replicador de datos REKO presentan como limitación para nuestra economía: que pertenecen a compañías privadas, con licencias del mismo tipo, por lo que el costo de adquisición es muy elevado.
- Para el desarrollo de la solución se utilizaron las herramientas, tecnologías y lenguajes con los que se ha trabajado en el sistema desde su versión inicial, lo que trajo consigo que se disminuya considerablemente la curva de aprendizaje y el tiempo de desarrollo.

CAPÍTULO 2. PROPUESTA DE SOLUCIÓN

En el presente capítulo se realiza la descripción de la propuesta de solución, además se especifican los requisitos funcionales y no funcionales, se describen las historias de usuarios, los patrones arquitectónicos, patrones de diseño aplicados, los diagramas de clases del diseño y el modelo de despliegue.

2.1 Descripción detallada del proceso a desarrollar

Actualmente el Replicador de datos REKO 5.0 cuenta con el módulo de Configuración de réplica, el cual permite la creación de configuraciones para las acciones DML. Utilizando un mecanismo de captura de cambios basado en el uso de funciones disparadoras o triggers. Estas se disparan y ejecutan un procedimiento en el momento que suceden las acciones para las cuales están destinadas.

La versión 5.0 del Replicador de datos REKO, no brinda la posibilidad de crear las configuraciones para las acciones de tipo DDL, donde se define la captura de los cambios de estructura, a su vez el sistema no reconoce automáticamente los cambios estructurales dentro de sus configuraciones de réplica.

Dada estas limitaciones se propone extender el módulo de Configuración de réplica, adicionándole la capacidad de que se actualicen de forma automática los cambios de estructura en las configuraciones de réplica, además de un mecanismo de convergencia para los cambios de tipo DDL y DML, que permita realizar estos procesos de manera concurrente para evitar posibles conflictos durante el proceso de réplica.

2.2 Modelo de dominio

Un modelo de dominio es una representación de las clases conceptuales del mundo real, no de componentes de software 34).
La realización de este modelo permite además aumentar la comprensión del problema y contribuir a esclarecer la terminología o nomenclatura del dominio.

Diagrama de clases del modelo de dominio

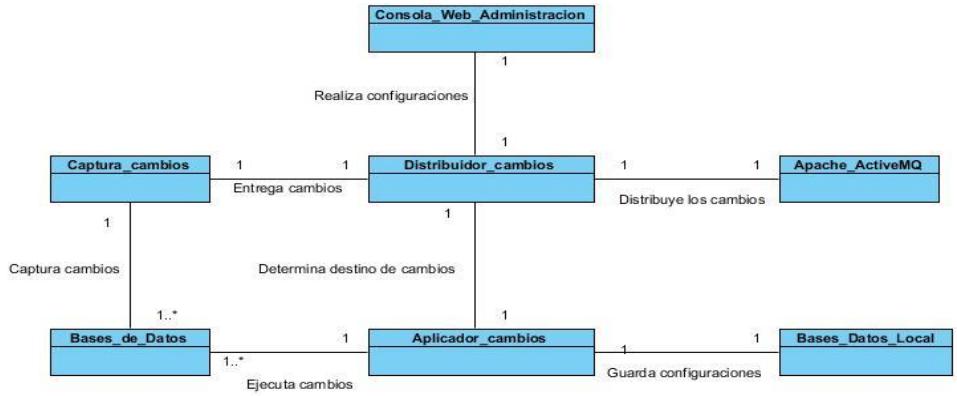


Figura 1 Modelo de dominio

Descripción de las clases del modelo de dominio

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

Distribuidor_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.

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

Base_Datos_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_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.

Apache_ActiveMQ: servidor de mensajería bajo la especificación *Java Message Service* (Servicio de Mensajería Java por sus siglas en inglés JMS), es utilizado como punto intermedio en la distribución de la información enviada bajo JMS.

Base_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.

2.3 Requisitos funcionales

Los requerimientos funcionales son los que definen las funciones que el sistema será capaz de realizar, describen las transformaciones que el sistema realiza sobre las entradas para producir salidas 35).

A continuación se muestran los requisitos funcionales obtenidos aplicacando la técnica de tormenta de ideas:

RF1 Adicionar configuración de réplica de estructura

RF1.1 Listar esquemas

RF1.2 Adicionar filtros de usuarios

RF1.3 Buscar esquemas

RF2 Editar configuración de réplica de estructura

RF3 Eliminar configuración de réplica de estructura

RF4 Establecer un orden en el proceso de réplica

RF5 Actualizar automáticamente configuración de réplica a partir de cambios de estructura

2.3.1 Especificación de requisitos

La especificación de requisitos del software es una descripción completa del comportamiento del sistema software a desarrollar. Incluye la descripción de todas las interacciones que se prevén que los usuarios tendrán con el software 36).

En la siguiente tabla se muestra la especificación de los requisitos funcionales.

Tabla 1 Especificación de requisitos

No	Nombre	Descripción	Prioridad	Complejidad	Historias de Usuarios
RF1	Adicionar configuración de réplica de estructuras	Permite al usuario registrar una nueva configuración de réplica de estructura	Alta	Media	HU 1
RF1.1	Listar esquemas	Muestra el listado de los esquemas de la	Alta	Media	HU 2

		base de datos			
RF1.2	Adicionar filtros de usuarios	Permite registrar la configuración de los filtros de usuario en la medida que se configuren las tablas a replicar	Alta	Baja	HU 3
RF1.3	Buscar esquemas	Permite una búsqueda por el nombre de esquema	Media	Baja	HU 4
RF2	Editar configuración de réplica de estructura	Permite modificar los parámetros almacenados en una configuración de réplica de estructura	Alta	Media	HU 5
RF3	Eliminar configuración de réplica de estructura	Permite eliminar la configuración de una réplica de estructura especificada por el usuario	Alta	Media	HU 6
RF4	Establecer un orden en el proceso de réplica	Permite establecer un orden de prioridad para la ejecución de la réplica	Alta	Alta	HU 7

RF5	Actualizar automáticamente configuración de réplica a partir de cambios de estructura	Permite actualizar las configuraciones de réplica almacenas cuando se produzca un cambio estructural sobre tablas con configuración de réplica	Alta	Alta	HU 8

2.4 Requisitos no funcionales

Son los requisitos que imponen restricciones al diseño o funcionamiento del sistema software (tal como requisitos de funcionamiento, estándares de calidad, o requisitos del diseño) □36).

Los requisitos no funcionales de la solución a desarrollar se representan en la siguiente tabla:

Tabla 2 Requisitos no funcionales

No.	**Atributo de calidad**	**Sub-Atributo**	**Objetivo**
RnF1	**Fiabilidad**	Tolerancia a fallos	Capacidad del producto para operar según lo previsto en presencia de fallos o conflictos generados por configuraciones erróneas o agentes externos al sistema.
		Recuperabilidad	Capacidad del producto para recuperar los datos directamente afectados y reestablecer el estado deseado del sistema en caso de interrupción o fallos.
RnF2	**Mantenibilidad**	Analizabilidad	Facilidad con la que se puede evaluar el impacto de un determinando cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a

			modificar.
		Modificabilidad	Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.
		Capacidad para ser probado	Capacidad que presenta el producto de facilitar el establecimiento de criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.
RnF3	**Portabilidad**	Adaptabilidad	Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.
RnF4	**Adecuación funcional**	Integridad funcional	Grado en el que el conjunto de funciones cubre todas las tareas y objetivos del usuario especificados.
		Corrección funcional	Grado en que un producto o sistema proporciona los resultados correctos con el grado necesario de precisión.
RnF5	**Seguridad**	No repudio	Grado en que las acciones o eventos pueden ser probados a haber tenido lugar, por lo que los eventos o acciones no pueden ser repudiados más tarde.

Para un mejor entendimiento de los requisitos no funcionales y sus atributos de calidad ver Anexo del 1 al 8.

2.5 Propuesta de solución

Para garantizar la disponibilidad y el acceso a la información como si se tratase de un único nodo, es fundamental que el replicador de datos REKO mantenga actualizadas las configuraciones de réplica que tiene almacenada. En la versión anterior esto no sucedía puesto que, se realizaba de forma manual la configuración de réplica de estructura y para que el

sistema reconociera los cambios en sus configuraciones era necesario reiniciarlo. En la nueva versión del sistema solo se podía adicionar la configuración de réplica de datos para las acciones DML. Al adicionar la funcionalidad configurar réplica de estructuras para acciones DDL, se podrán establecer filtros de usuario para cada tipo de acción (creación, modificación, eliminación) que se realizan exclusivamente sobre los objetos que pertenecen a los esquemas previamente seleccionados. También, replicar la creación de un nuevo esquema, replicar la modificación del nombre de algún esquema y replicar la eliminación de algún esquema para los esquemas existentes en la BD, además de automatizar el proceso de captura de cambios en las configuraciones de réplica mediante la utilización de una tabla de control para almacenar los cambios que ocurran y para la realización del mecanismo de convergencia entre procesos de réplica, se adiciona a la tabla de control una columna que indica el momento en que se capturó el cambio ocurrido.

La configuración del proceso de réplica de estructura en la nueva versión del Replicador de datos REKO, actualiza sus configuraciones de réplica almacenadas cada vez que se desee adicionar una nueva configuración de réplica, el mismo consiste en verificar si se han capturado cambios estructurales sobre las tablas que ya contienen configuración de réplica de datos. De ser así, el sistema analiza el tipo de cambio estructural almacenado en la tabla de control dado que, si el tipo de acción DDL es eliminar esquema, de todas las tablas que pertenecen al esquema eliminado, se borran todas las funciones disparadoras de captura de datos teniendo en cuenta el tipo de acción DML y todos los residuos de configuraciones a replicar sobre las tablas. Si el tipo de acción DDL es un eliminar tabla, el sistema borra la tabla en cuestión, así como las funciones disparadoras de captura de datos por los tipos de acciones DML y los residuos de configuraciones que existen sobre la tabla. Mientras que el tipo de acción DDL no sea crear tabla o crear esquema, el sistema tratará el resto de los cambios como acciones DDL de tipo modificación y automáticamente elimina las funciones disparadoras asociadas a la configuración de réplica por tipo de acciones DML que posee la tabla, también borra todos los residuos de configuraciones que se refieran a la tabla sobre el cual se realiza la modificación. Luego actualiza la configuración creando las funciones disparadoras para acciones de tipo DML.

Para la realización conjunta de los procesos de réplica fue necesario crear un mecanismo de convergencia que realizara de forma simultánea ambos procesos. Partiendo de que todas las configuraciones de cambios DML y DDL que han sido capturadas se encuentran almacenadas

en una sola tabla de control, el mecanismo trabaja obteniendo todas las tuplas de la tabla de control y analizando por cada tupla el tipo de cambio, conformando el tipo de acción replicable para cada tupla del mismo tipo y agregándola al Grupo replicable o Grupo de estructura replicable que corresponda, luego se analiza el tipo de cambio de la tupla siguiente. En caso de coincidir, se continúa adicionando al grupo y en caso de que no coincidan, se envía el grupo replicable o el grupo de estructura replicable con las acciones que se han adicionado. Se repite el proceso hasta que no queden tuplas por analizar.

2.6 Historias de Usuario

Entre los artefactos que genera la metodología AUP se encuentran las Historias de Usuarios (HU) que permiten una descripción de los requisitos, en el Replicador de datos REKO se realizan las descripciones de las HU. Por tanto, se hará uso de esta técnica para encapsular los requisitos funcionales.

Las “Historias de usuarios” sustituyen a los documentos de especificación funcional, y a los “casos de uso”. La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios 37).

2.6.1 Descripción de las HU

A continuación, se muestra las Historias de Usuario del RF 4, RF 5 respectivamente, para un mejor entendimiento de las Historias de Usuarios ver Anexo del 9 al 14.

Tabla 3 Historia de Usuario Establecer un orden en el proceso de réplica

Número: HU 7	**Nombre del requisito:** Establecer un orden en el proceso de réplica
Programador: Jessilié Paz Lara	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 30 días
Riesgo en Desarrollo: Fallos eléctricos	**Tiempo Real:** 240 horas

Descripción: El sistema debe permitir establecer un orden de prioridad para la ejecución de la réplica. A medida que los cambios sean capturados, estos se van almacenando ordenados ascendentemente en la tabla de control del sistema. Se debe verificar que existan elementos en la tabla de control para iniciar la réplica, luego se crean las instancias de los grupos replicables y los grupos de estructura replicables respectivamente. De esta forma a medida que se analice el tipo de configuración de réplica que posee cada tupla en la tabla de control se irán adicionando en el grupo al cual correspondan, siempre que la siguiente tupla sea del mismo tipo, se mantendrá adicionando configuraciones de la tabla de control del sistema al grupo que corresponda, cuando la tupla siguiente sea de otro tipo de configuración, se persiste el grupo que contiene las acciones replicables y se envía. Posteriormente el sistema deberá continuar con el proceso hasta que no queden más tuplas por analizar.
Observaciones: NA
Prototipo de interfaz: NA

Tabla 4 Historia de Usuario Actualizar automáticamente configuración de réplica a partir de cambios de estructura

Número: HU 8	**Nombre del requisito:** Actualizar automáticamente configuración de réplica a partir de cambios de estructura
Programador: Jessilié Paz Lara	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 10 días
Riesgo en Desarrollo: Fallos eléctricos	**Tiempo Real:** 80 horas
Descripción: El sistema debe permitir actualizar las configuraciones de réplica que están almacenadas, permitiendo que estas se actualicen en dependencia del cambio estructural que sufran las tablas que poseen configuración de réplica de dato. Por ejemplo, si existe una tabla que tiene configuración de réplica de datos por inserción y desde la base de datos se elimina esta tabla, el sistema deberá eliminar los triggers que se crearon cuando se le adicionó la configuración de réplica de datos a la tabla. Para cada cambio estructural el sistema deberá actualizar las vistas en la configuración de réplica respectivamente.	
Observaciones: NA	
Prototipo de interfaz: NA	

2.7 Descripción de la arquitectura del Replicador de datos REKO

Según Clements[26] la Arquitectura de Software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones 37).

Según David Garlan[27] 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 37).

Según la IEEE[28] Std 1471-2000 afirma que la Arquitectura de Software es 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 37).

El Replicador de datos REKO define una arquitectura basada en capas, donde cada capa se define como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Además de estar separadas lógica y estructuralmente, las capas se encuentran separadas de manera física.

La arquitectura basada en capas se enfoca en la distribución de roles y responsabilidades de forma jerárquica suministrando una forma muy efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada 40).

BUSINESS LOGIC, PRESENTATION, CORE y el DATA ACCESS son las capas presentes en el sistema.

CORE: se encarga de inicializar el sistema.

PRESENTATION: contiene todos los módulos de la capa de presentación del sistema.

---

[26] Paul Clements (Arquitecto).

[27] David Garlan es profesor en la Facultad de Ciencias de la Computación de la Universidad Carnegie Mellon, donde dirige varios proyectos de investigación.

[28] El Instituto de Ingeniería Eléctrica y Electrónica.

BUSINESS LOGIC: contiene las funciones lógicas del sistema.

DATA ACCESS: se encarga de persistir en la base de datos embebida las configuraciones del sistema.

En el paquete **configuration**, correspondiente a la capa BUSINESS LOGIC, se le realizarán modificaciones, pues hasta el momento su funcionamiento se basa en las configuraciones de réplica de datos, es decir, cambios de tipo DML (inserción, actualización, eliminación), ahora se extenderá su funcionamiento para las configuraciones de réplica de tipo DDL (creación, modificación, eliminación), la capacidad de actualizar las configuraciones de réplica automáticamente y el estableciemiento de un mecanismo que permitirá realizar estas operaciones de manera concurrente.

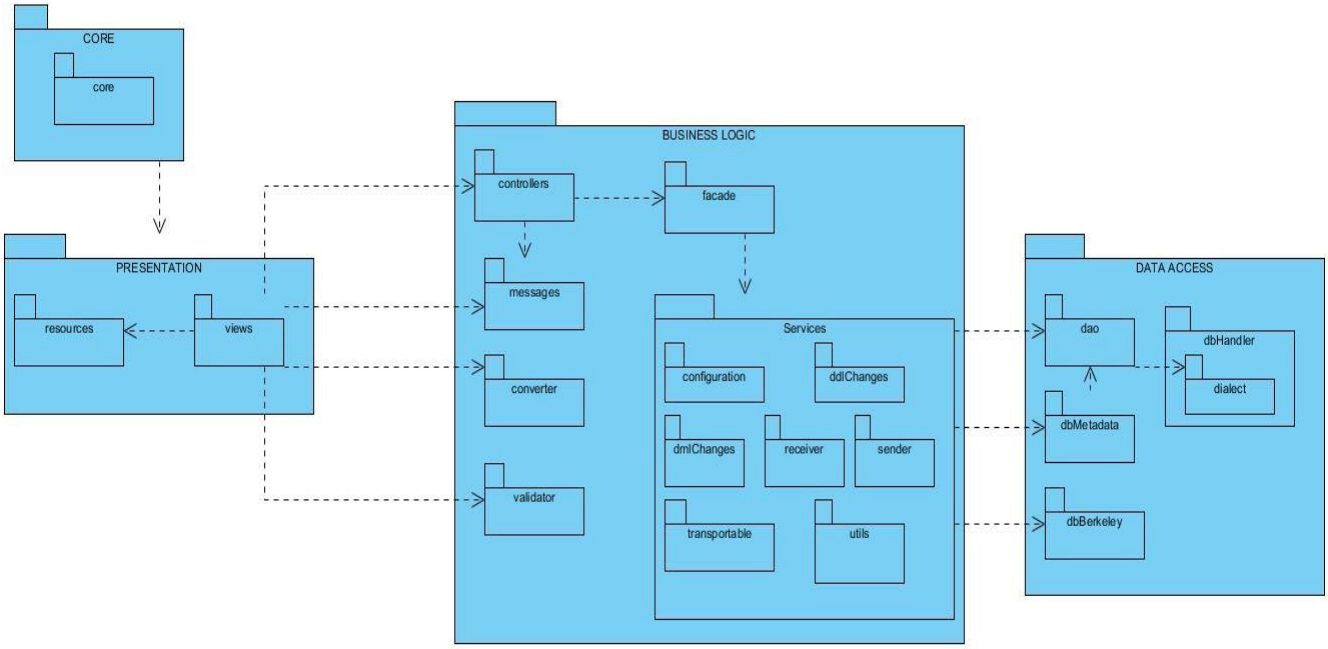


Figura 2 Arquitectura del Replicador de datos REKO

2.8 Patrones de diseño

Un patrón de diseño constituye una solución estándar para un problema común de programación en el desarrollo del software 41).

En el desarrollo de la solución se usaron los patrones de asignación de responsabilidades, conocidos como patrones GRASP (Patrones Generales de Software para Asignar Responsabilidades, en inglés *General Responsibility Assignment Software Patterns*) estos

describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones 42).

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 **DBStructureChangeManager** cuenta con toda la información necesaria para iniciar el proceso de captura de cambios de estructura en la BD.

Controlador: el patrón controlador actúa 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 del método solicitado. Por ejemplo, se utiliza la clase controladora: **ReplicationConfigController**, encargada de desencadenar eventos según las peticiones provenientes de la interfaz, separando la capa de presentación de la lógica de negocio lo cual permite una mayor reutilización del código.

Polimorfismo: tiene responsabilidad de definir la variación de comportamiento basado en tipos, se asigna a aquellos para los cuales esta variación ocurre. Esto se logra mediante el uso de operaciones polimórficas. Este patrón se aplica en la clase interfaz **HibernateDialect**, la cual es implementada por **PostgresDialect**, cuya función principal es crear las consultas SQL correspondiente al tipo de gestor de base de dato.

Alta cohesión: como cada clase tiene un conjunto de funcionalidades relacionadas directamente con la entidad que representan, no realizan un trabajo enorme y por tanto pueden ser calificadas como de alta cohesión. Este patrón se aplica a las clases **PostgresDialect** y **DBStructureChangeManager**, pues manejan solo la información de las funcionalidades que realizan.

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. Este patrón se aplica en las clases **DBStructureChangeManager**, **DialectUtils**, **HibernateDialect** y **PostgresDialect**, ya que la implementación de la solución se basa en funcionalidades de **DBStructureChangeManager** que dependen secuencialmente del resto.

Además, se utilizaron los patrones **GOF** (Banda de los Cuatro, del inglés *Gang-Of-Four*) los cuales se clasifican en dependencia del propósito para los que hayan sido definidos: creación, estructurales y de comportamiento 43).

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 **ReplicationConfigFacade** perteneciente al componente facade que ofrece un punto de acceso para iniciar el proceso de replicación.

Otro de los patrones utilizados para la solución propuesta es el patrón **DAO** (Objeto de Acceso a Datos, en inglés *Data Access Object.*), se dividen más las responsabilidades en la aplicación de tal forma que tendremos clases que se encargarán de la lógica de negocio y otras clases de la responsabilidad de persistencia. Fuera de las clases DAO no debe haber ningún tipo de código que acceda al repositorio de datos. Los Objetos de Acceso a Datos pueden usarse en Java para aislar a una aplicación de la tecnología de persistencia Java subyacente, la cual es JDBC □44).

Ejemplo del patrón DAO

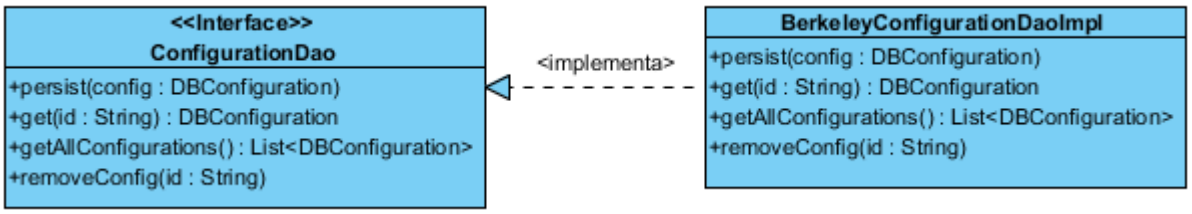


Figura 3 Ejemplo del patrón DAO

2.9 Modelo de diseño

El modelo de diseño está adaptado para modelar el entorno de implementación real, y sirve como una abstracción del código fuente □45).

2.9.1 Diagramas de paquetes

Un diagrama de paquetes en el Lenguaje Unificado de Modelado representa las dependencias entre los paquetes que componen un modelo. Es decir, muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre esas agrupaciones.

A continuación, se muestran los diagramas de paquetes de los RF4 y RF5 respectivamente, para consultar el resto ver Anexo 15.

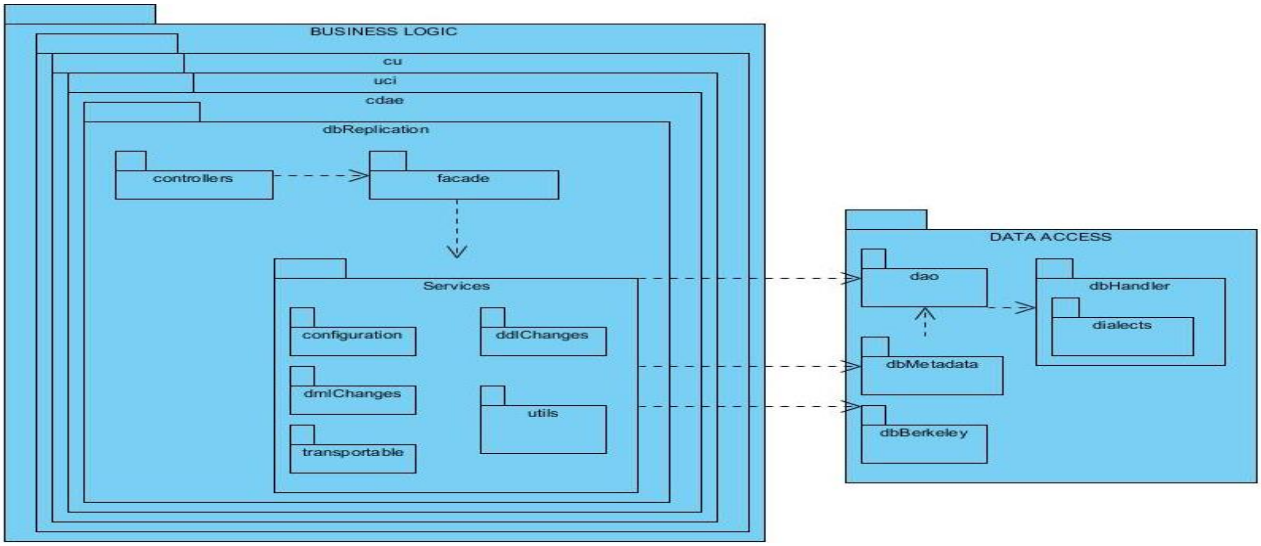


Figura 4 Diagrama de Paquetes RF4

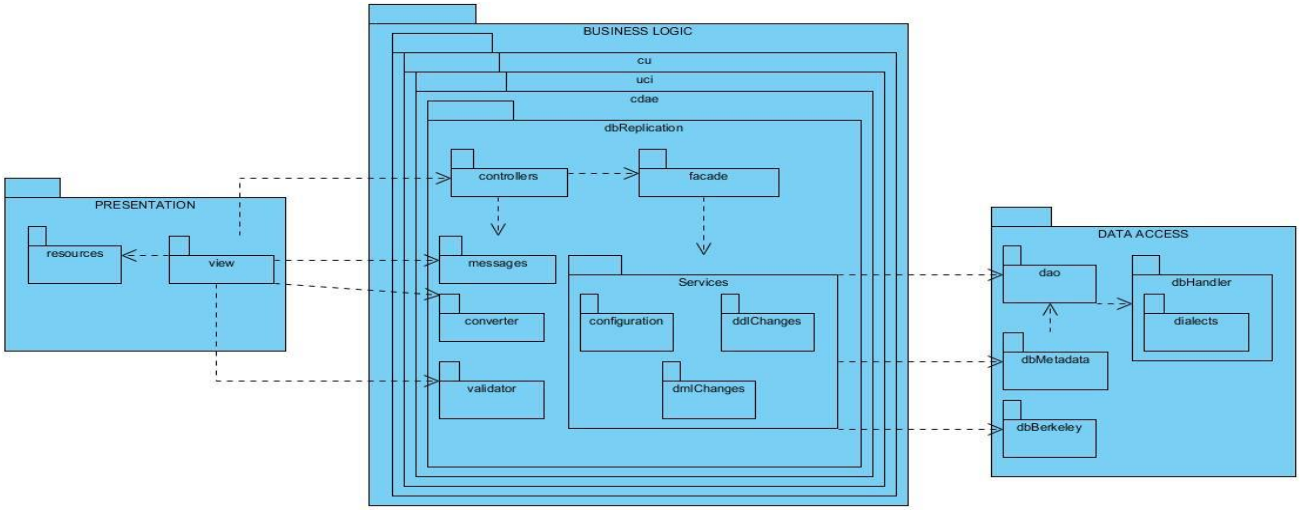


Figura 5 Diagrama de Paquetes RF5

2.9.2 Diagrama de clases

Durante el diseño, el Diagrama de Clase se elabora para tener en cuenta los detalles concretos de la implementación del sistema.

A continuación, se muestra los diagramas de clases de los RF4 y RF5 respectivamente, para consultar el resto ver Anexo 16.

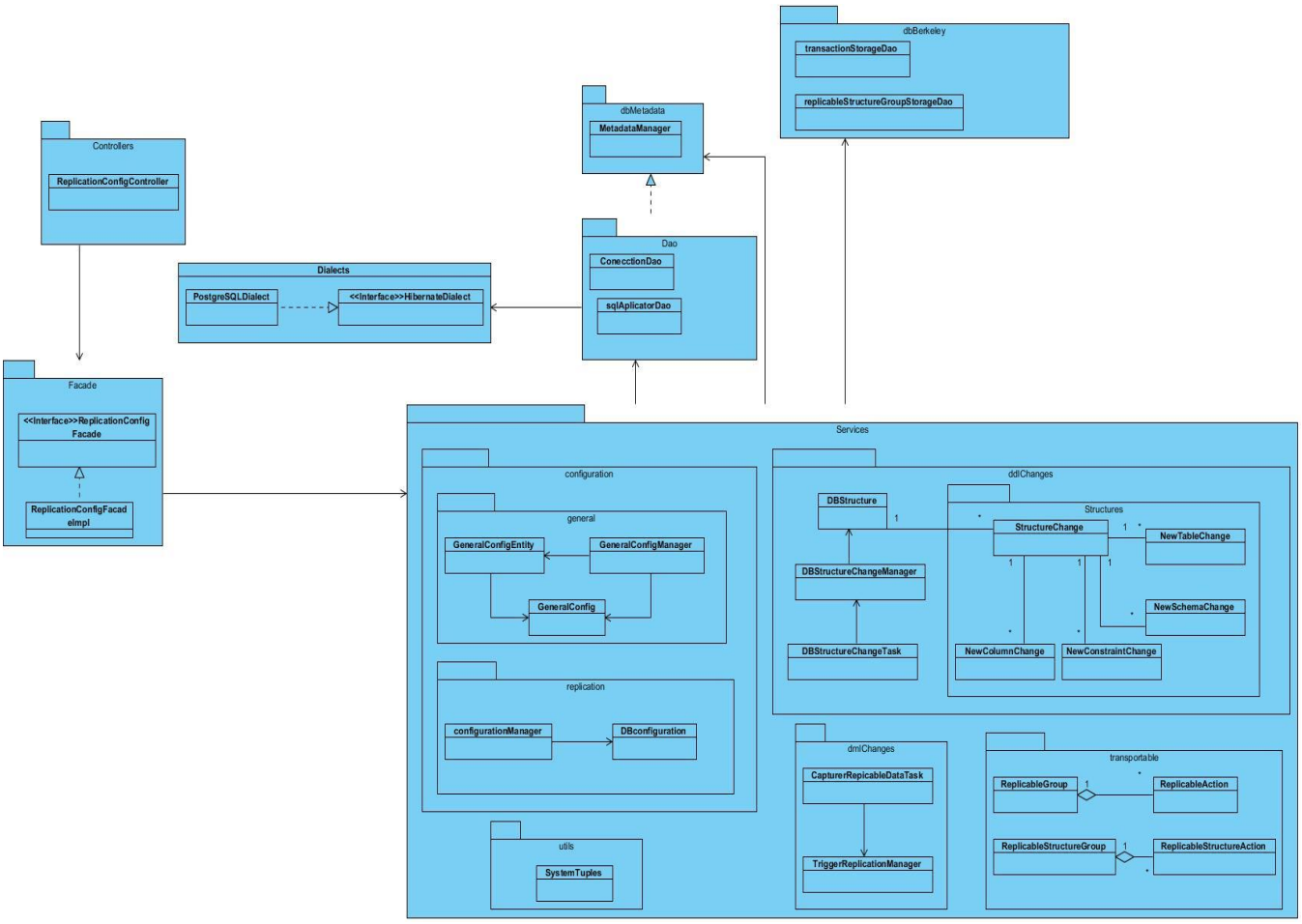


Figura 6 Diagrama de clases RF 4

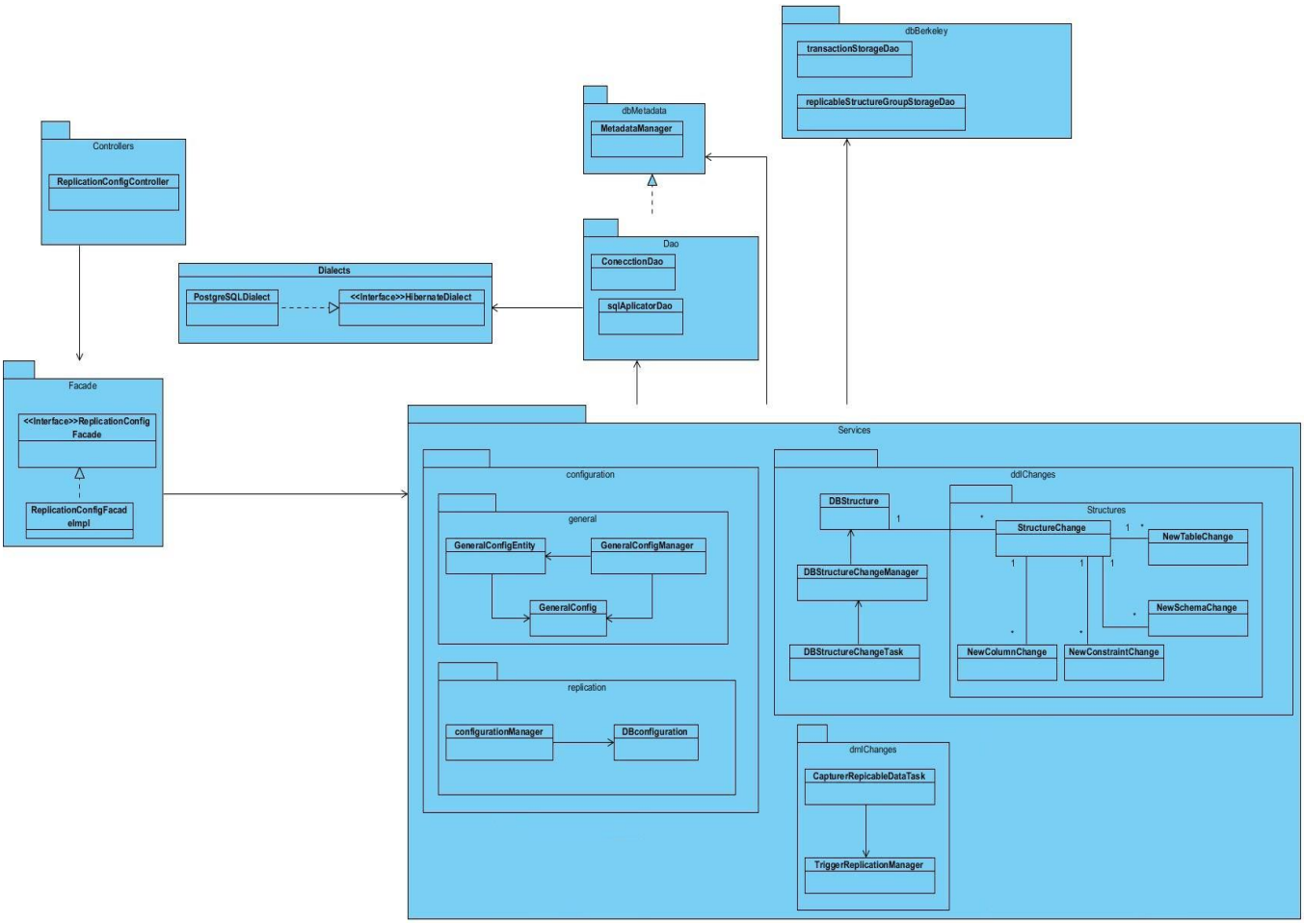


Figura 7 Diagrama de clases RF 5

2.9.3 Descripción de las clases

A continuación, se muestra la descripción de las clases DBStructureChangeManager que es la encargada de administrar la captura de los cambios de estructura y ConfigurationManager que es la encargada de administrar las configuraciones.

Para un mejor entendimiento de las descripciones de las clases ver Anexo del 17 al 19.

Tabla 5 Descripcion de la clase DBStructureChangeManager

Descripción de la clase DBStructureChangeManager	
Nombre: DBStructureChangetManager	
Tipo de clase: Controladora.	
Atributos	**Tipos**
ConnectionDao	private
sqlMetadaManager	private
configurationManager	private
Responsabilidades	
Nombre:	afterPropertiesSet() : void
Descripción:	Reajusta el proceso de captura de cambios de estructura cuando cambia la configuración de réplica
Nombre:	startStructureCaptureChanges() : void
Descripción:	Inicia el proceso de captura de cambios de estructura
Nombre:	createTablesOfGeneralControl(HibernateDialect dialect, Connection con) : void
Descripción:	Crea todas tablas de control general de cambios de estructura en la BD
Nombre:	createStructureControlTable(HibernateDialect dialect, Connection con): void
Descripción:	Crea la tabla de control de cambios de estructura en la BD
Nombre:	createTableCaptureSchema(HibernateDialect dialect, Connection con) : void
Descripción:	Crea la tabla de captura de estructura para los esquemas en la BD
Nombre:	createTableCaptureTable(HibernateDialect dialect, Connection con) : void
Descripción:	Crea la tabla de captura de estructura para las tablas en la BD
Nombre:	createTableCaptureColumn(HibernateDialect dialect, Connection con): void
Descripción:	Crea la tabla de captura de estructura para las columnas en la BD
Nombre:	createTableCaptureConstraint(HibernateDialect dialect, Connection con) : void
Descripción:	Crea la tabla de captura de estructura para las restricciones en la BD

Nombre:	stopStructureCaptureChanges(HibernateDialect dialect, Connection con) : void
Descripción:	Detiene el proceso de captura de cambios de estructura
Nombre:	dropStructureCaptureTriggers(HibernateDialect dialect, Connection con) : void
Descripción:	Elimina los triggers de captura de cambios de estructura de la BD
Nombre:	startStructureCapture(HibernateDialect dialect, Connection con) : void
Descripción:	Realiza la captura inicial de la estructura de la BD
Nombre:	createStructureCaptureFunction(HibernateDialect dialect, Connection con) : void
Descripción:	Crea la función de captura inicial de estructura en la BD
Nombre:	executeStructureCaptureFunction(HibernateDialect dialect, Connection con): void
Descripción:	Ejecuta la función de captura inicial de estructura en la BD
Nombre:	dropStructureCaptureFunction(HibernateDialect dialect, Connection con): void
Descripción:	Elimina la función de captura inicial de estructura de la BD

Tabla 6 Descripcion de la clase ConfigurationManager

Descripción de la clase ConfigurationManager	
Nombre: ConfigurationManager	
Tipo de clase: Controladora.	
Atributos	**Tipos**
connectionDao	private
ConfigurationDao	private
replicationManager	private
triggerReplicationManager	private
dbStructureChangeManager	private
sqlApplicatorDao	private
replicableStructureGroupStorageDao	private
transactionStorageDao	private
transactionProcessor	private
Responsabilidades	
Nombre:	persistConfiguration (DBConfiguration configuration) : void
Descripción:	Almacena la configuración de réplica establecida en la base de datos
Nombre:	removeConfig (String id) : void
Descripción:	Dado el identificador de una configuración de réplica pasado por parámetro la elimina si existe
Nombre:	isAnyNodeConfiguredInAnyConfig(): boolean
Descripción:	Devuelve true si existen algún nodo con configuración de réplica
Nombre:	startReplicConvergenceProcess(): boolean
Descripción:	Contiene el mecanismo de ejecución simultánea para los procesos de réplica
Nombre	updateAllReplicConfig(DBConfiguration configuration): voip
Descripción	Actualiza las configuraciones de réplica

2.10 Diagrama de despliegue

Establece una correspondencia entre la arquitectura de software y la arquitectura de hardware del sistema. El modelo de despliegue se utiliza para capturar los elementos de configuración

del procesamiento y las conexiones entre esos elementos, además para visualizar los nodos procesadores, la distribución de los procesos y los componentes 46).

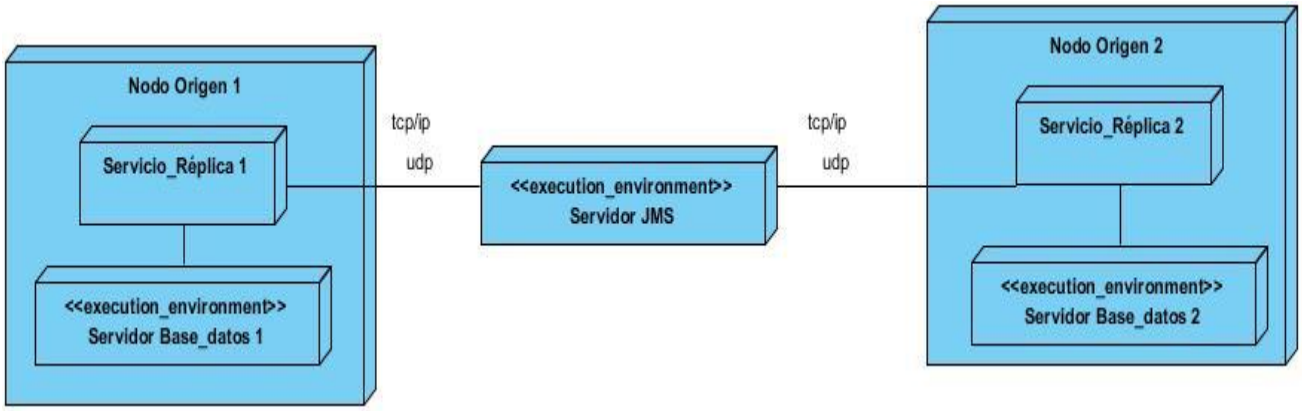


Figura 8 Diagrama de despliegue

2.11 Consideraciones parciales del capítulo

- La representación del modelo de dominio contribuyó a identificar las entidades del negocio y las relaciones entre ellas.
- El levantamiento de los requisitos funcionales de la aplicación permitió dar respuesta a las necesidades del problema. Por otra parte, los requisitos no funcionales definieron las restricciones del mismo.
- La definición de la propuesta de solución del problema, detallándola con la ayuda de los artefactos propuestos por la metodología AUP-UCI escenario 4, posibilitó estructurar todo el proceso de desarrollo.
- La selección del patrón arquitectónico, los patrones de diseño, así como los diagramas de paquetes y diagrama de clases, facilitó garantizar el correcto funcionamiento y organización de la solución.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS AL SISTEMA

En el presente capítulo se ejemplifica la implementación de la solución, se realizan los diagramas de componentes, así como la aplicación de las pruebas al sistema y se analizan las no conformidades encontradas en cada una de las iteraciones realizadas a la solución.

3.1 Diagramas de componentes

En los diagramas de componentes se muestran los elementos de diseño de un sistema de software. Un diagrama de componentes permite visualizar con facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. Se puede usar un diagrama de componentes para describir un diseño que se implemente en cualquier lenguaje o estilo. Solo es necesario identificar los elementos del diseño que interactúan con otros elementos del diseño a través de un conjunto restringido de entradas y salidas □47).

A continuación, se muestra un diagrama de componente perteneciente a la HU 7, para consultar el resto ver Anexo 20 y 21.

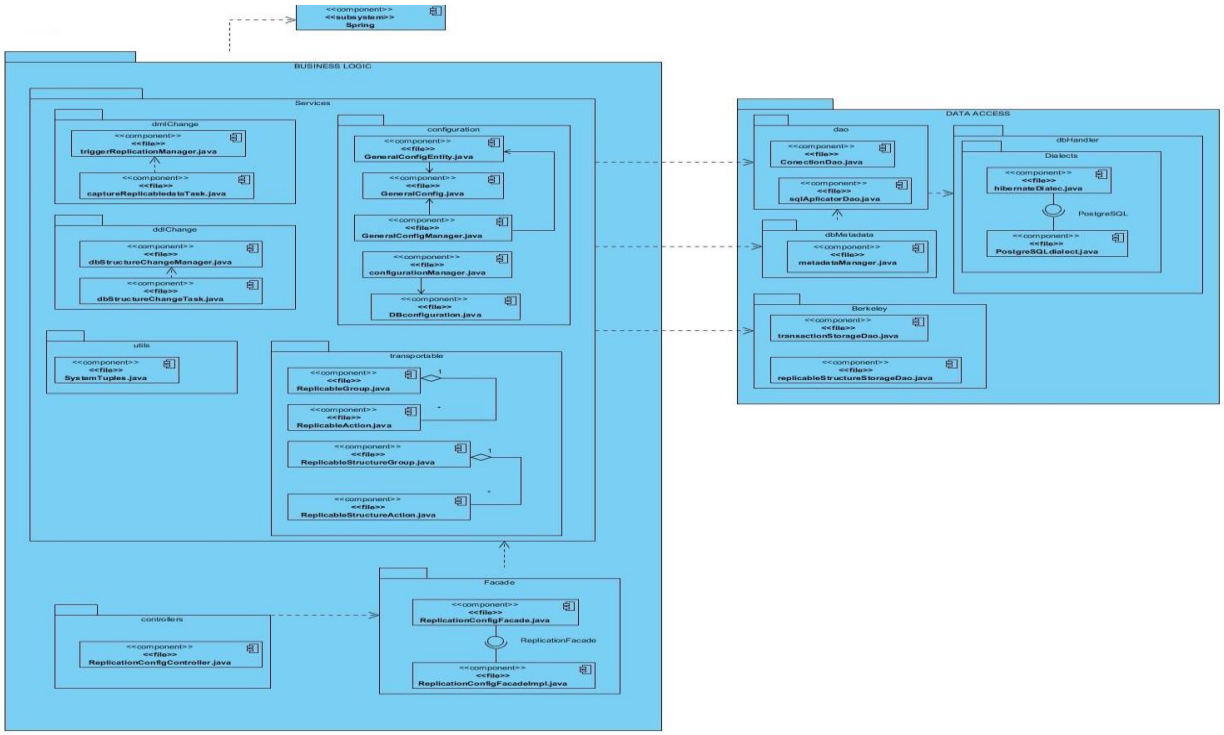


Figura 9 Diagrama de componente HU 7

3.2 Estándar de codificación

Las técnicas de codificación incorporan muchos aspectos del desarrollo del software. Aunque generalmente no afectan a la funcionalidad de la aplicación, sí contribuyen a una mejor compresión del código fuente. En esta fase se tienen en cuenta todos los tipos de código fuente, incluidos los lenguajes de programación, de secuencias de comandos, de marcado o de consulta 48).

A continuación, se muestra una parte del estándar de codificación empleado en la solución, para consultar el resto ver Anexo 22:

Variables y métodos: Empiezan con minúsculas y si estos identificadores están compuestos por varias palabras las siguientes empezarán con mayúscula.

Variables

```
tableConfigMap
```

```
schemaConfigMap
```

Método

- Los nombres de método deben iniciar con un verbo, ejemplo:

```
public void updateAllReplicConfig (...) {}
```

- Los obtenedores de campos privados en las clases tienen el prefijo "**get**", ejemplo:

```
publis SchemaConfig getSchemaConfig (..) {}
```

- Los obtenedores con el resultado de booleano tienen el prefijo "**is**", ejemplo:

```
public boolean isReplicateOnSchemaCreate () {}
```

Clases, enumeradores e Interfaces:

- Todas las palabras que componen a dichos identificadores empezarán con mayúscula, ejemplos:

```
public class DBStructureChangeManager {}
```

```
public class DBConfiguration {}
```

3.3 Evaluación del Diseño Aplicando Métricas de Software

Las métricas de software son medidas cuantitativas que permiten a los desarrolladores tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar los remedios y mejorar el proceso del software 49).

Para evaluar la calidad del diseño del Mecanismo de convergencia entre los procesos de réplica en el Replicador de Datos REKO fueron seleccionadas las métricas Tamaño Operacional de Clases (TOC) y Relaciones entre Clases (RC).

- **Responsabilidad**: consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- **Complejidad de implementación**: consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- **Reutilización**: consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento**: consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- **Complejidad del mantenimiento**: consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirectamente, pero fuertemente en los costes y la planificación del proyecto.
- **Cantidad de pruebas**: consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, módulo, clase, conjunto de clases, entre otras) diseñado 49).

3.3.1 Tamaño Operacional de la clase (TOC)

Esta métrica se refiere a la cantidad de métodos que tiene una clase 50).
En la siguiente tabla describe los atributos que se evalúan al aplicar la métrica TOC.

Para un mejor entendimiento de la aplicación de la métrica TOC ver Anexo 23.

Tabla 7 Tamaño operacional de clase (TOC)

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

Fuente: □50)

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

Tabla 8 Criterios y categorías de evaluación (TOC)

Atributos	**Categorías**	**Criterios**
Responsabilidad	Baja	TOC <= Promedio[29]
	Media	Promedio < TOC <= 2*Promedio
	Alta	TOC > 2*Promedio
Complejidad de implementación	Baja	TOC <= Promedio
	Media	Promedio < TOC <= 2*Promedio
	Alta	TOC > 2*Promedio
Reutilización	Baja	TOC > 2*Promedio
	Media	Promedio < TOC <= 2*Promedio
	Alta	TOC <= Promedio

Fuente: □50)

Resultados obtenidos en la aplicación de la métrica TOC

Luego de aplicar la métrica TOC, se puede comprobar que el modelo de diseño propuesto tiene una calidad aceptable, debido a que el comportamiento de los atributos de calidad es positivo.

La figura 10 muestra que un 68% de las clases posee una **responsabilidad** baja y sólo un 9% con una responsabilidad alta, por lo que, en caso de fallos, ningún componente es demasiado crítico como para dejar fuera de servicio el sistema.

La figura 10 describe resultados similares, pero evaluando la **complejidad de implementación**, demostrando que el sistema no es difícil de implementar.

[29] **Promedio de la cantidad de métodos por clase.**

La figura 10 muestra que un 68% de las clases tiene un alto grado de **reutilización** y sólo un 9% una reutilización baja; por lo que las funcionalidades pueden ser aprovechadas al máximo.

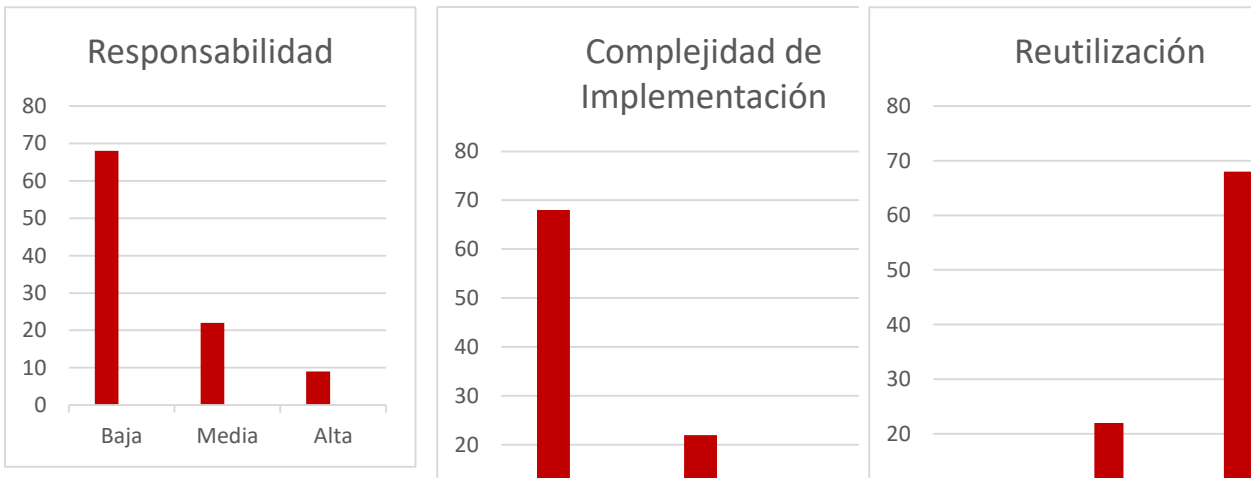


Figura 10 Evaluación mediante TOC

3.3.2 Relaciones entre clases (RC)

Esta métrica se refiere a la cantidad de relaciones de uso que tiene una clase con otras. □50)
En la siguiente tabla describe los atributos que se evalúan al aplicar la métrica RC.

Tabla 9 Relaciones entre clase (RC)

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

Fuente: □50)

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

Tabla 10 Criterios y categorías de evaluación (RC)

Atributos	Categorías	Criterios
Acoplamiento	Ninguno	RC = 0
	Baja	RC = 1
	Media	RC = 2
	Alta	RC > 2
Complejidad de mantenimiento	Baja	RC <= Promedio
	Media	Promedio < RC <= 2*Promedio
	Alta	RC > 2*Promedio
Reutilización	Baja	RC > 2*Promedio
	Media	Promedio < RC <= 2*Promedio
	Alta	RC <= Promedio
Cantidad de pruebas	Baja	RC <= Promedio
	Media	Promedio < RC <= 2*Promedio
	Alta	RC > 2*Promedio

Fuente: □50)

Resultados obtenidos en la aplicación de la métrica RC

Luego de aplicar la métrica RC, se puede comprobar que el modelo de diseño propuesto tiene una calidad aceptable, debido a que el comportamiento de los atributos de calidad también es positivo.

Ver Anexo (24) para una descripción más detallada.

La figura 11 muestra que un 41% de las clases posee un nivel de acoplamiento bajo, un 24% no tiene, el 22% acoplamiento medio y un 13% un nivel de acoplamiento alto; demostrando así, que el grado de dependencia entre las clases del modelo de diseño planteado es bajo.

La figura 11 especifica que el 64% de las clases tiene una baja complejidad de mantenimiento, por lo que no se requiere de gran esfuerzo para realizarle mejoras o correcciones al sistema en general.

La figura 11 detalla que un 64% de las clases posee una alta reutilización, dejando sólo un 35% de clases cuya reutilización es media o baja; por lo que se demuestra que el modelo de diseño propuesto, permite un elevado aprovechamiento de sus funcionalidades.

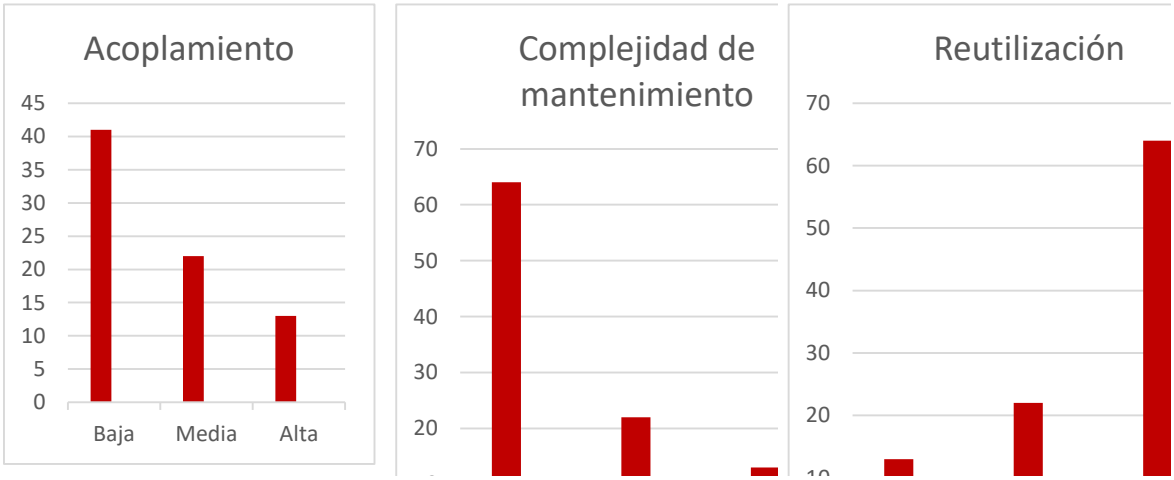


Figura 11 Evaluación mediante RC

3.3.3 Matriz de inferencia de indicadores de calidad

La matriz inferencia de indicadores de calidad, también llamada matriz de cubrimiento, es una representación estructurada de los atributos de calidad y métricas utilizadas para evaluar la calidad del diseño propuesto. Dicha matriz permite conocer si los resultados obtenidos de las relaciones atributo/métrica es positivo o no, llevando estos resultados a una escalabilidad numérica donde, si los resultados son positivos se le asigna el valor de 1, si son negativos toma valor 0 y si no existe relación es considerada como nula y es representada con un guion simple (-). Luego se puede obtener un resultado general para cada atributo promediando todas sus relaciones no nulas □50).

Tabla 11 Matriz de Interferencia de Indicadores de Calidad

Atributos\Métricas	TOC	RC	Promedio
Responsabilidad	1	-	1
Complejidad de implementación	1	-	1
Reutilización	1	1	1
Acoplamiento	-	1	1
Complejidad de mantenimiento	-	1	1
Cantidad de Pruebas	-	1	1

Resultados obtenidos después de aplicar las métricas de software

Una vez aplicadas las métricas TOC y RC, se puede concluir que el modelo de diseño propuesto tiene una calidad aceptable, debido a que el comportamiento de los atributos de calidad es positivo:

- El grado de **responsabilidad** del sistema es bajo, por lo que ningún proceso es demasiado crítico como para dejar fuera de servicio al sistema en caso de fallo.
- La **complejidad de implementación** del sistema es baja.
- El sistema posee un alto grado de **reutilización**, por lo que sus funcionalidades pueden ser aprovechadas al máximo.
- Las clases del sistema poseen un bajo nivel de **acoplamiento**, reduciendo las dependencias entre clases y permitiendo realizar cambios sin afectar en gran medida al resto de sistema.
- La **complejidad de mantenimiento** es baja, por lo que no se requiere de gran esfuerzo para realizarle mejoras o correcciones al sistema en general.
- No se necesita gran esfuerzo para realizar **pruebas** al sistema en general.

3.4 Pruebas del software

Luego de concluido un proyecto de desarrollo de software es necesario verificar la calidad del mismo antes de realizar su entrega. En el flujo de trabajo de pruebas es donde se examina el resultado de la implementación. Estas no son más que un conjunto de métodos y técnicas que garantizan el buen desempeño de una aplicación, para la validación del producto.

Según Pressman: Las pruebas de software constituyen una fase del proceso de desarrollo de un software centrada en la calidad, fiabilidad y robustez del mismo; dentro del contexto o escenario previsto para ser utilizado. Las mismas permiten detectar la presencia de errores que generen salidas o comportamientos inapropiados durante su ejecución 49).

3.4.1 Tipos de pruebas

Hay diferentes tipos de pruebas de software. Las que buscan probar una funcionalidad del software, las que buscan una característica no funcional, y las que buscan probar la estructura del software.

- **Pruebas Funcionales**: Este tipo de pruebas se basan en las funcionalidades de un sistema, que se describen en la especificación de los requisitos, es decir, lo que hace el sistema.
- **Pruebas no Funcionales**: Este tipo de pruebas tienen en cuenta el comportamiento externo del software, que incluyen las pruebas de: Rendimiento, Usabilidad, Fiabilidad o Portabilidad. Por tanto, se centran en los requisitos no funcionales.
- **Pruebas estructurales:** Las que se centran en los detalles de arquitectura o lógica 49).

Para el desarrollo de las pruebas se tienen en cuenta un conjunto de niveles de pruebas a seguir.

3.4.2 Pruebas Unitarias

Las pruebas unitarias se centran en un esfuerzo de verificación de la unidad más pequeña del diseño de software: el componente y el módulo del software. “Las pruebas de unidad se centran en la lógica del procesamiento interno y en las estructuras de datos dentro de los límites de un componente” 49).

Método de Caja Blanca

Se realiza un análisis riguroso de los detalles procedimentales, comprobando las rutas lógicas del programa. El objetivo principal de este método es probar que todos los caminos del código están correctos 49).

Para desarrollar la prueba de caja blanca se utilizó la técnica del camino básico.

Técnica Camino básico: La técnica del camino básico se basa en derivar casos de prueba a partir de un conjunto dado de caminos independientes .Para obtener el conjunto de caminos independientes se construirá el Grafo de Flujo asociado y se calculará su Complejidad Ciclomática 49).

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de

procesamiento o una nueva condición. El camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente 49).

Hay tres formas fundamentales de calcular la complejidad:

1-El número de regiones del grafo de flujo coincide con la complejidad ciclomática.

2-La complejidad ciclomática, V(G), también se define como:

V(G) = A - N + 2 donde: A es el número de aristas del grafo y N es el número de nodos.

3-V(G)= Nodos predicados, estos se definen como aquellos donde emergen dos o más aristas.

V(G) = P + 1 donde: P es el número de nodos predicado contenidos en el grafo G.

A continuación se muestra un ejemplo realizado al método **startReplicConvergenceProcess** que se encarga de establecer una convergencia entre los procesos de réplica a través de la ejecución simultánea de los mismos.

Tabla 12 Definición de los nodos de flujo en el método starReplicConvergeProcess()

startReplicConvergenceProcess	
0	`public synchronized boolean startReplicConvergenceProcess() throws SQLException, DataAccessException`

1	`boolean flag = true;`
	`//Comprobando si sea ha capturado`
	`int cant = connectionDao.getCountsReplicsDataStructureSQL();`
	`//Obteniendo cantidad maxima de acciones por grupos.`
	`int maxCant = Integer.parseInt(GeneralConfig.getCANT_ACTIONS_IN_GROUP());`

2	`if (cant != 0)`

3	`triggerReplicationManager.setDisabledExportBtn(false);`
	`applicationEventPublisher.publishEvent(new ExportEvent(this));`
	
	`logger.info("Existen elementos para replicar");`
	`List<SystemTuples> tuples = connectionDao.getNewSystemTuples(maxCant);//obtener todas las tuplas de mi tabla de control`
	`List<Integer> tuplesToDelete = new LinkedList<>();`
	`List<SystemTuples> tuplesPacket =connectionDao.getNewSystemTuples(maxCant);`
	`tuplesPacket.clear();`
	`int count = 0;`
	`int count1 = 0;`
	`try`

4	`for (SystemTuples tuples1 : tuples) {`

5	`if (tuples1.get_change_data() != null){`

6	`tuplesPacket.add(tuples.indexOf(tuples1),tuples1);`
	`tuplesToDelete.add(tuples1.get_id());`
	`int pos = tuples.indexOf(tuples1);`
	`if (pos == tuples.size()-1){`
	`triggerReplicationManager.startNewWayDataCapture1(tuplesPacket, cant);`
	`tuplesPacket.clear();`
	`}else {`
	`if (tuples.get(pos+1).get_change_data() == null) {//si no es del mismo tipo`
	`triggerReplicationManager.startNewWayDataCapture1(tuplesPacket,cant);`
	`tuplesPacket.clear();`

7	`else{tuplesPacket.add(tuples.indexOf(tuples1),tuples1);`
	`tuplesToDelete.add(tuples1.get_id());`
	`int pos = tuples.indexOf(tuples1);`
	`if (pos == tuples.size()-1){`
	`dbStructureChangeManager.sendStructuresChanges(tuplesPacket);`
	`tuplesPacket.clear();`
	`}else {`
	`if (tuples.get(pos+1).get_change_data() == null) {//si no es del mismo tipo`
	`dbStructureChangeManager.sendStructuresChanges(tuplesPacket);`
	`tuplesPacket.clear();`

8	`);`
9	`for (int tupleId : tuplesToDelete`
10	`connectionDao.dropElementsFromSystemId(tupleId);`
11	`return flag;`
12	`}//sale del método`

Se procede a la construcción del grafo correspondiente al código fuente mostrado anteriormente, cuyo procedimiento se ejecutó manual.

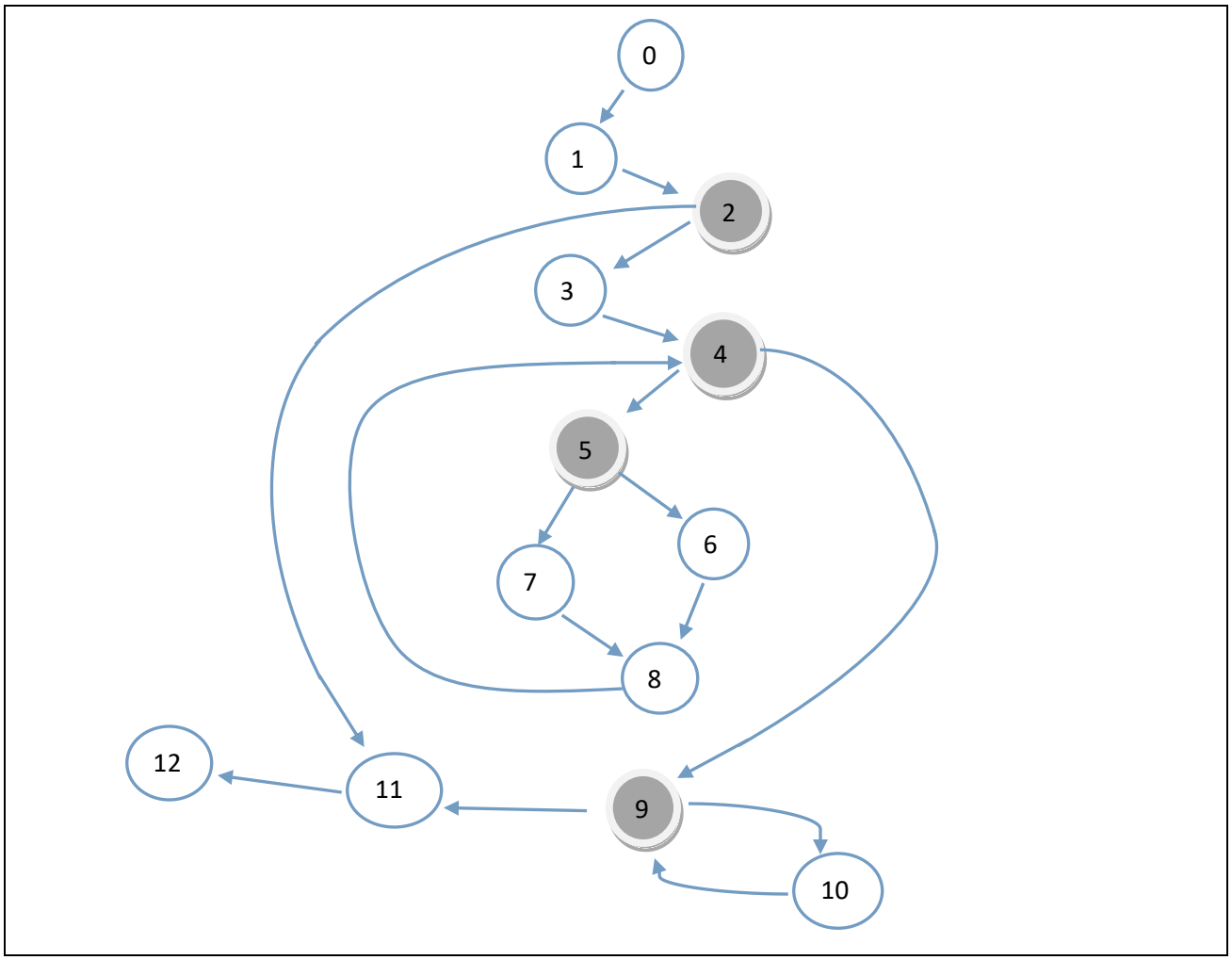


Figura 12 Flujo de control lógico correspondiente al método starReplicConvergeProcess()

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

Tabla 13 Complejidad ciclomática correspondiente al método starReplicConvergeProcess()

V(G) = R	**V(G) = A - N + 2**	**V(G) = P + 1**
R = 5 (regiones del grafo)	A = 16 (aristas) N = 13 (nodos)	P = 4 (nodos predicados)
	V(G) = 16 - 13 + 2	V(G) = 4 + 1
V(G) = 5	**V(G) =5**	**V(G) = 5**

Los cálculos realizados mediante las tres fórmulas anteriores dan el mismo resultado V(G) = 5, lo que revela que existen nueve posibles caminos por donde el flujo puede circular y determina el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.

R1: 0 1 2 11 12

R2: 0 1 2 3 4 5 6 8 4 9 10 9 11 12

R3: 0 1 2 3 4 5 7 8 4 9 10 9 11 12

R4: 0 1 2 3 4 5 6 8 4 5 6 8 4 9 10 9 11 12

R5: 0 1 2 3 4 5 6 8 4 5 7 8 4 9 10 9 11 12

Tabla 14 Casos de prueba por camino básico

Camino	Entrada	Resultado
1	No existen configuraciones de réplica	La tabla de control está vacía puesto que no se han capturado cambios.
2	Existe al menos una configuración de réplica datos	Se crea el grupo replicable, se adiciona al grupo la acción replicable que corresponde a la configuración de réplica obtenida de la tabla de control, se marca para eliminar de la tabla de control, se persiste y se envía el grupo, luego se actualiza la tabla de control eliminando las configuraciones previamente marcadas.
3	Existe al menos una configuración de réplica estructura	Se crea el grupo de estructura replicable, se adiciona al grupo la acción de estructura replicable que corresponde a la configuración de réplica obtenida de la tabla de control, se marca para eliminar de la tabla de control, se persiste y se envía el grupo, luego se actualiza la tabla de control eliminando las configuraciones previamente marcadas.
4	Existen más de una configuración de réplica del mismo tipo	Para las configuraciones de tipo DML, se crea el grupo replicable, se adicionan al grupo las acciones replicables que corresponden a las configuraciones de réplica obtenidas de la tabla de control, se marcan para eliminarse de la tabla de control, se persiste y se

		envía el grupo, luego se actualiza la tabla de control eliminando las configuraciones previamente marcadas. Para las configuraciones de tipo DDL, se crea el grupo de estructura replicable, se adicionan al grupo las acciones de estructura replicables que corresponden a las configuraciones de réplica obtenidas de la tabla de control, se marcan para eliminarse de la tabla de control, se persiste y se envía el grupo, luego se actualiza la tabla de control eliminando las configuraciones previamente marcadas.
5	Existen configuraciones de réplica de diferentes tipos	Se crea el grupo replicable para configuraciones de tipo DML y el grupo de estructura replicable para configuraciones de tipo DDL. Se van adicionando a cada grupo las acciones replicables para el tipo DML y las acciones de estructura replicables para el tipo DDL en cada caso, se marcan para eliminarse de la tabla de control, se persiste y se envía el grupo que se creó, luego se actualiza la tabla de control eliminando las configuraciones previamente marcadas.

Las pruebas de caja blanca aplicadas al código fuente de la solución, utilizando el método de camino básico, demuestran que los Requisitos Funcionales están correctamente implementados y que el sistema responde como se espera ante determinadas acciones.

3.4.3 Pruebas de Sistema

Estas buscan probar al sistema como un todo, las cuales permiten probar el correcto funcionamiento de los requisitos funcionales lo que incluye la entrada de datos, el procesamiento de estos y la obtención de los resultados.

Para la creación de casos de prueba de sistema se usa el método de caja negra mediante la técnica partición de equivalencia, procedimiento que permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata los errores 49).

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

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica 49).

A continuación, se muestra una síntesis de los resultados obtenidos al aplicar las pruebas de Caja Negra (ver Tabla 15), este método se aplicó a todos los requisitos funcionales, el cual mostró resultados satisfactorios. Los restantes casos de prueba se encuentran en los anexos 25 y 26.

Tabla 15 Caso de Prueba HU7

Caso de Prueba	
Código: SC_7	**Historia de usuario**: 7
Nombre: Establecer orden en el proceso de réplica	
Descripción: prueba para la funcionalidad establecer orden en el proceso de réplica	
Condiciones de Ejecución: • El usuario debe estar autenticado. • Establecer la configuración de BD para PostgreSQL en el Módulo de Configuración General. • Tener al menos un Nodo configurado.	
Respuesta del Sistema/Flujo Central • Al capturarse cambios de datos o estructuras el sistema deberá permitir almacenarlos en la tabla de control del esquema reko de forma ordenada por el tiempo de captura, es decir si se realiza una operación de inserción sobre una tabla que tiene configuración de réplica de datos, el sistema deberá almacenar el tipo de operación, sobre que tabla se realizó, los datos y el momento que se capturó el cambio. Sucede de igual manera cuando ocurre un cambio de estructura sobre un esquema u objeto dentro de un esquema que tiene configuración de estructura. Se capturan los datos asociados al cambio estructural y	

también el momento en que se efectuó dicho cambio. • Se estable un orden en el momento que se va a replicar gracias a que todos los cambios se encuentra almacenados en una sola tabla ordenados ascendentemente por el momento en el que fueron capturados.
Resultado Esperado: el sistema deberá establecer el orden de ejecución de los procesos de réplica.
Evaluación de la Prueba: bien.

3.4.4 Resultados de las pruebas de Caja Blanca y Caja Negra

Fue necesario realizar tres iteraciones de pruebas para verificar el correcto funcionamiento de la solución. En la primera iteración fueron encontradas un total de cuatro no conformidades (NC), asociadas a problemas funcionales y errores de ortografía:

Las NC que se detectaron fueron:

- No se crea la tabla de control en el esquema reko.
- No se almacena el objeto configuración correctamente.
- No se actualiza la configuración DDL de los esquemas.
- Existían errores ortográficos en la interfaz de configuración de réplica.

Una vez concluida la primera iteración fueron resueltas las cuatro NC detectadas, para el caso de la primera NC, se concatenó el nombre del esquema reko con el de la tabla de control, de esta forma la tabla de control se creó dentro del esquema reko. Para darle solución a la segunda NC fue necesario incluir antes de almacenar el objeto configuración el mapa que contiene todas las configuraciones de réplica sobre los esquemas y para solucionar la tercera NC fue necesario actualizar la configuración mediante la funcionalidad del Metadata que actualiza la configuración realizando una lectura en tiempo real de la BD.

En la segunda iteración se detectó una NC:

- No se almacenan los cambios de tipo DML en la tabla de control.

Esta NC surgió a causa de que se almacenaban los cambios DML en la antigua tabla de control y no en la nueva. Se identificó a partir de que se empezaron a capturar cambios DDL en la tabla de control tras la resolución de la NC de la iteración anterior. Para solucionar esta NC fue

necesario cambiar en los triggers DML el nombre de la tabla de control donde se almacenan los cambios DML, para que cuando se capturaran los cambios se almacenaran en la tabla de control propuesta como parte de la solución.

En la tercera iteración de pruebas no se detectaron nuevas NC. La figura 13 muestra la relación entre las NC detectadas y resueltas por cada iteración de pruebas:

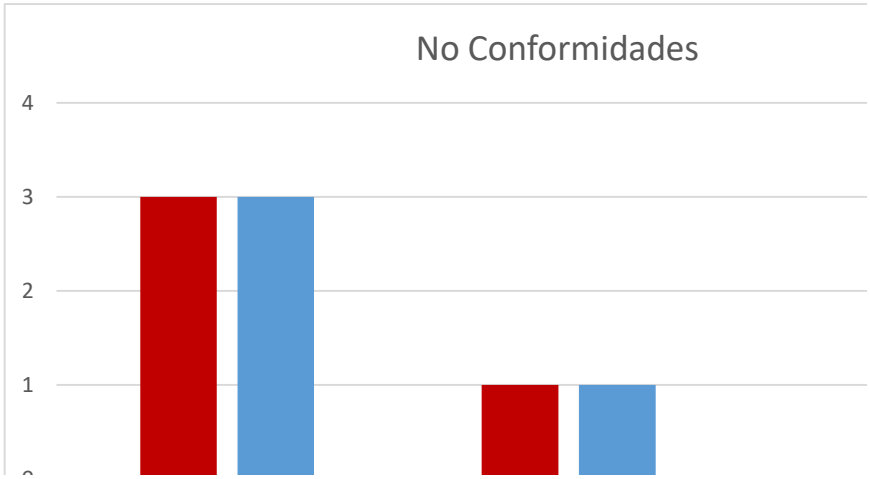


Figura 13 No Conformidades

3.4.5 Análisis de los casos críticos

Una vez alcanzado un estado de implementación libre de errores, cubriendo todos los tipos de combinaciones posibles para cada tipo de acción DML y DDL se procedió a analizar el comportamiento del sistema ante varios casos críticos. En la investigación constituyen casos críticos aquellas situaciones o escenarios en los que el sistema puede arrojar diferentes resultados. A continuación, se presentan y se describen dos de ellos.

Caso 1: Cuando ocurre una acción de tipo DDL y otra de tipo DML que depende de esta.

A continuación, se describe un ejemplo: Se utilizó un script en el cual se adicionó el SQL para crear una tabla y para insertar datos en la tabla anterior como se puede apreciar en la figura 14, en este orden. En la figura 15 se puede apreciar la ejecución de una acción DDL específicamente del tipo creación de tabla y en la figura 16 se evidencia la inserción de los datos en la tabla previamente creada., Al ejecutarse las sentencias satisfactoriamente, el sistema captura y registra en la tabla de control los cambios en el mismo orden.

```
CREATE TABLE public.rol
(
  id serial not null,
  nombre character varying(255),
  descripcion character varying(255)
);

INSERT INTO public.rol ("id", "nombre", "descripcion") VALUES ('1','Admin', 'Administrador del sistema');
```

Figura 14 Script SQL

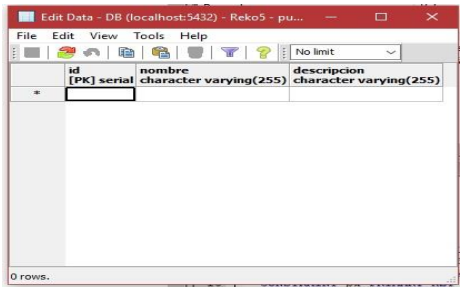


Figura 15 Creación de la tabla

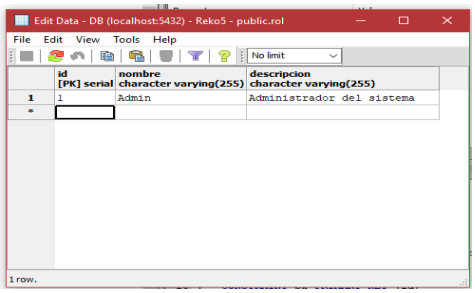


Figura 16 Inserción de los datos

Caso 2: Cuando ocurre una acción de tipo DDL y otra del mismo tipo que depende de la anterior.

A continuación, se describe un ejemplo: La figura 17 muestra los cambios que han sido capturados y ordenados por el tiempo de ocurrencia en la tabla de control, se puede apreciar que primero hay un cambio de tipo creación de tabla y luego otro que añade una nueva columna a la tabla previamente creada, ambos cambios se realizan a través de un script que se ejecutó en la BD como muestra la figura 18. Se evidencia que las acciones se capturan y se registran en la tabla de control en el mismo orden en que fueron realizadas.

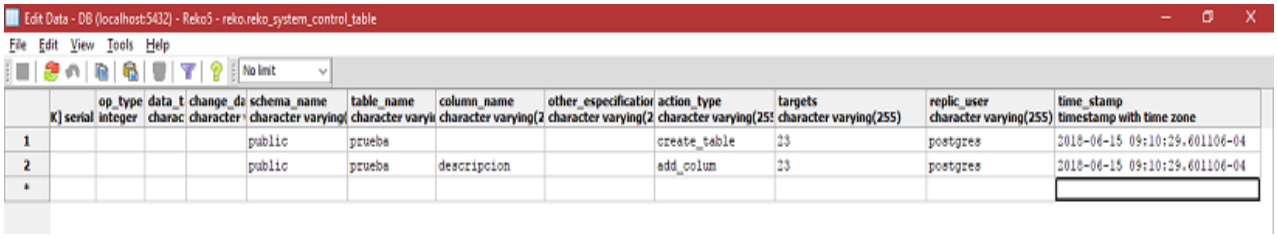


Figura 17 Tabla de control

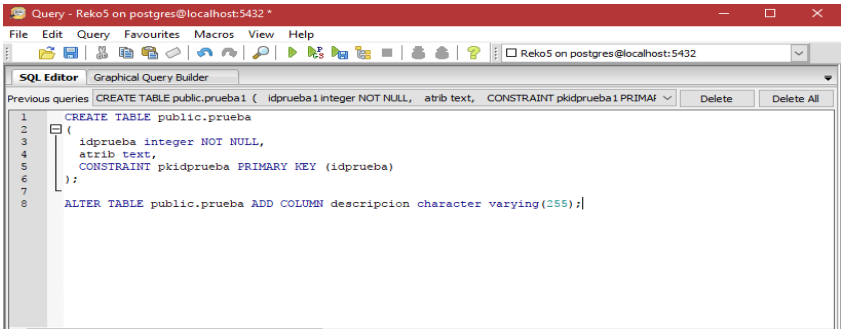


Figura 18 Script

Para este caso fue necesario realizar un estudio sobre los disparadores, su estructura y funcionamiento para determinar el orden cuando las acciones son dependientes entre sí, con el objetivo de verificar que las acciones que son dependientes entre ellas siempre se van a ejecutar de forma secuencial en el orden en el que se realizó cada acción. Como resultado de este análisis se corroboró que los triggers DDL dentro de su funcionamiento a partir de la versión 9.5 del SGBD PostgreSQL trabajan asociados a eventos, es decir, cuando se produce un cambio se registra un evento de tipo DDL lo que provoca la ejecución de los triggers de eventos sobre comandos DDL, por tanto el orden para la ejecución de estas acciones siempre va a ser el mismo en que fueron realizadas.

3.4.6 Pruebas de Aceptación

Las pruebas de aceptación son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto. El propósito es confirmar que la solución está terminada, que cumple con las necesidades de la organización y que es aceptado por los usuarios finales.

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 49).

Luego de la culminación de las pruebas unitarias y de sistema se procedió a realizar las pruebas de aceptación; la misma se llevaron a cabo por los clientes Gloria Raquel Leyva y Nayibi Martín, fueron realizadas bajo un ambiente controlado para determinar si la solución cumple con lo estipulado por el cliente y lo establecido en las historias de usuarios. Terminadas estas pruebas y corregidas las no conformidades el cliente avaló la solución quedando entonces el Certificado de Aceptación del Producto.

3.5 Consideraciones parciales del capítulo

- Al concluir el desarrollo de este capítulo se obtuvo una solución que cumple satisfactoriamente con los requerimientos definidos.
- Se realizaron las pruebas necesarias para el control de la calidad del producto que permitieron la corrección de las no conformidades detectadas.

CONCLUSIONES GENERALES

Con el desarrollo de la presente investigación se arriba a las siguientes conclusiones:

- El empleo de la metodología, tecnologías, lenguajes y herramientas definidas para la investigación, permitió obtener un producto acorde los estándares de desarrollo del Replicador de Datos REKO en su nueva versión.
- El proceso de implementación permitió desarrollar la solución a partir de los resultados del análisis y diseño, contribuyendo así a los requisitos establecidos por el cliente.
- La realización de pruebas a la solución y la aplicación de métricas de software, permitieron detectar y corregir a tiempo con el menor esfuerzo los errores existentes, obteniéndose un producto funcional y con la calidad requerida.

RECOMENDACIONES

- Se recomienda extender la solución propuesta para otros gestores.
- Se recomienda realizar un análisis de los triggers, su estructura y funcionamiento para los gestores que lo implementan, para verificar el orden de ejecución de sentencias de tipo DDL.

REFERENCIAS BIBLIOGRÁFICAS

1. DATE, Chris J. Introducción a los sistemas de bases de datos. Pearson Educación, 2001, 9684444192 – ISBN www.academia.edu/download/37358813/Fundamentos_de_Bases_de_Datos.pdf
2. MONTILLA DELGADO, Alejandro. Sistemas distribuidos. Replicación de datos. 2016. http://openaccess.uoc.edu/webapps/o2/bitstream/10609/53222/1/amontilla_TFG_0616.pdf
3. SIERRA, M. ¿Qué es una Base de Datos y cuáles son sus principales tipos? [Página web: Sitio oficial]. España: Aprender a programar, [Consultado el: noviembre de 2017]. Disponible en: http://www.aprenderaprogramar.com/index.php?option=com_attachments&task=download&id=500.
4. ROSA MARÍA MATO GARCÍA. Diseño de bases de datos Editorial Pueblo y 2009. 51 p. ISBN 978-959-13-1273-0.
5. CARRANZA ATHÓ, F. Bases de datos distribuidas. . Perú: Perú: Escuela Académico Profesional de Informática, 2006.
6. JEREZ, Gloria Raquel Leyva; PEÑA, Nayibi Martí. COMPONENTE PARA EL ENVÍO Y RECEPCIÓN DE DATOS DEL PROCESO DE RÉPLICA DE ESTRUCTURA PARA EL REPLICADOR REKO.
7. Soto MCDE. Base de Datos. Replicación de Datos en SQL Server. Disponible en: www.desarrollopro.com/tec/Unidad%20III%20BDD%20B.ppt.
8. Urbano R. Database Advances Replication 2010. Disponible en: http://docs.oracle.com/cd/E11882_01/server.112/e10706.pdf.
9. DHARMA ROJAS. Propuesta Metodológica para el Desarollo y la Elaboración de Estadísticas ambientales para los países de América Latina y el Caribe. United Nations Publications, 2009. 36 p p. ISBN 9213227787, 9789213227787.
10. JOSE ANTONIO OCAMPO. Registros Administrativos, Calidad de los Datos y Credibilidad Pública: Presentación y Debate de los Temas Sustantivos de la Segunda Reunión de la Conferencia Estadística de las Américas de la CEPAL. CEPAL, 2004. 18 p p. ISBN 9789213222836.

11. Real Academia Española (2000). Convergencia. En diccionario de la lengua Española. Madrid: España.

12. JEREZ, Gloria Raquel Leyva, et al. CONFIGURACIÓN Y MONITORIZACIÓN DE LA RÉPLICA DE ESTRUCTURA PARA EL REPLICADOR DE DATOS REKO.

13. SOFTWARE, H. DBMoto Cloud Edition generalidades [Consultado el: 16 de noviembre 2017]. Disponible en: http://www.hitsw.com/localized/spanish/products_services/dbmoto/dbmoto_cloud/DBMoto_Cloud_Edition.html.

14. YERANDY MANSO GUERRA, ENRIQUE JOSÉ ALTUNA, , ADRIÁN GARCÍA SÁNCHEZ, YOANDY PÉREZ CÁCERES. Experiencias en el uso del symmetric para la replicación de datos en la plataforma educativa ZERA. nº [Consultado el: 16/11/2017]. Disponible en: http://scielo.sld.cu/pdf/rcci/v9n4/rcci14415.pdf.

15. ORACLE. Data Replication and Integration [Página web:Sitio oficial]. Oracle,. [Consultado el: noviembre de 2017]. Disponible en: http://www.oracle.com/technetwork/database/information-management/streams-fov-11g-134280.pdf.

16. PostgreSQL: Announcing The Release Of pglogical 2.0 [0.52] ..pglogical_create_subscriber which can convert physical standby (or base backup) into pglogical subscriber Performance...https://www.postgresql.org/about/news/1744/

17. SERVER, M. S. Making Schema Changes on Publication Databases [Página web.]. [Consultado el: noviembre de 2017]. Disponible en: http://msdn.microsoft.com/es-es.

18. RÍOS, M. N. D. L. Manual de usuario Replicador de datos REKO. 2015. vol. 4.0, 97 p.

19. ING. EDISVEL MELO ESTEVEZ , I. A. G. S., ING. YANISLEYDIS RODRÍGUEZ TAMAYO. CONFIGURACIÓN Y MONITORIZACIÓN DE LA RÉPLICA DE ESTRUCTURA PARA EL REPLICADOR DE DATOS REKO Cuba : Universidad de las Ciencias Informáticas.: [Consultado el: 08/12 de 2017]. Disponible en: https://vemecuba.eventos.uci.cu/sites/default/files/public/ponencia/ponencia/node/Uciencia665.pdf.

20. SÁNCHEZ, T. R. Metodología de desarrollo para la Actividad productiva de la UCI. 1.2 ed. Universidad de las Ciencias Informáticas, [Consultado el: 15/11/2017].

21. FLORES, E. Metodologías Agiles Proceso Unificado Ágil (AUP) [Consultado el: 16/11 de 2017]. Disponible en: http://ingenieriadesoftware.mex.tl/63758_AUP.html.

22. RIVERA, F. L. O. Introducción a la Programación en Java. Un enfoque Práctico. 1ra ed. septiembre de 2007. ISBN 978-958-98314-8-9.

23. Mastermagazine. Disponible en: http://www.mastermagazine.info/termino/4404.php.

24. GALLARDO, D. Iniciándose en la plataforma Eclipse [Consultado el: 14/11 de 2017]. Disponible en: https://www.ibm.com/developerworks/ssa/library/os-ecov/.

25. Martínez R. Sobre PostgreSQL 2013. Disponible en: http://www.postgresql.org.es/sobre_postgresql#intro.

26. Marc Gibert Ginestà OPM. Bases de Datos en Postgres. Disponible en: http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02152.pdf.

27. Andalucía Jd. Spring. Disponible en: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/93.

28. EGAS CLAVIJO, Patricio Gustavo. Primefaces crud generador para Netbeans. 2015. Tesis de Licenciatura. Quito: UCE.

29. Squires IVMCIERC. Sistema para el manejo de órdenes de servicio para los especialistas del centro de ingeniería clínica y electromedicina: Universidad de las Ciencias Informáticas; 2013. Disponible en: http://www.informatica2013.sld.cu/index.php/informaticasalud/2013/paper/download/167/32

30. Yolanda ILPE. Modelado de Sistemas con UML. Disponible en: http://www.monografias.com/trabajos94/modelado-sistemas-uml/modelado-sistemas-uml.shtml.

31. Activemq. Active MQ 2011. Disponible en: http://activemq.apache.org/index.html.

32. Foundation A. Apache Tomcat. Disponible en: http://apachefoundation.wikispaces.com/Apache+Tomcat.

33. GUERRA, C. R. Configuración JMS en ORACLE WEBLOGIC v11 [Página web: Sitio oficial.]. Lima: Oracle®. [Consultado el: enero de 2018]. Disponible en: http://frameworksjava2008.blogspot.com/2012/05/configuracion-jms-en-oracle-weblogic.html.

34. GARCÍA-HOLGADO, A.; GARCÍA-PEÑALVO, F. J. Modelo de dominio. 2018.

35. ARIAS CHAVES, Michael. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. InterSedes: Revista de las Sedes Regionales, 2005, vol. 6, no 10.

36. SERNA-MONTOYA, Édgar. Estado actual de la investigación en requisitos no funcionales. Ingeniería y Universidad, 2012, vol. 16, no 1, p. 225-246.

37. JOSKOWICZ, José. Reglas y prácticas en eXtreme Programming. Universidad de Vigo, 2008, vol. 22.

38. REYNOSO, Carlos. Introducción a la Arquitectura de Software. Universidad de Buenos Aires, 2004, vol. 33.

39. Ingeniería de Software Basada en Componentes. [Consultado el: 21 de febrero del 2018]. Disponible en: https://sites.google.com/site/lawebdelsoftware/ingenieria-de-software-1/unidad-vi.

40. GONZALEZ, A. Sistemas de llamada y retorno 2 de Junio de 2016, [Consultado el: 10/03 de 2018]. Disponible en: https://prezi.com/wyylmlq8jpyo/sistemas-de-llamada-y-retorno/.

41. GAMMA, E. Patrones de diseño: elementos de software orientado a objetos reutilizables, ed. P. Educacion, 2006, vol. 1, nº [Consultado el: 22 de febrero de 2018]. Disponible en: http://dspace.ucbscz.edu.bo/dspace/handle/123456789/565. ISSN 84-7829-059-1.

42. LARMAN, C. UML y Patrones. Introducción al análisis y diseño orientado a objetos. 2005. 507 p p.

43. GUERRERO, C. A.; SUÁREZ, J. M., et al. Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. Información tecnológica, 2013, vol. 24, nº 3, p. 103-114. ISSN 0718-0764.

44. CECILIO ÁLVAREZ. Patrones de Diseño (Active Record vs DAO) [Página Web: Sitio Oficial]. [Consultado el: 22 de febrero de 2018]. Disponible en: http://www.genbetadev.com/java-j2ee/patrones-de-diseno-active-record-vs-dao.

45. JACOBSON, I.; BOOCH, G., et al. El proceso unificado de desarrollo de software. Addison Wesley Reading, 2000. vol. 7, ISBN 84-7829-036-2.

46. LARMAN, C. UML y Patrones. Introducción al análisis y diseño orientado a objetos. 2005. 507 p p.

47. RUMBAUGH, James, JACOBSON, Ivar y BOOCH, Grady,Addison Wesley. 2000. “El lenguaje unificado de modelado. Manual de referencia” . Capítulos 4 y 13 . 2000. págs. Páginas 42-48122-125, 131-143, 156, 162-170, 191- 197, 211, 299-303, 367-373, 399-402, 427, 434-435, 446-4.

48. CALLEJA, M. A. Carmen. *Estándares de codificación* [en línea]. S.l.: s.n. [Consultado 2 junio 2018]. Disponible desde: http://www.cisiad.uned.es/carmen/estilo-codificacion.pdf.

49. PRESSMAN, R. S. Ingenieria del Software, Un Enfoque Práctico. Editado por: Ed. España:: Editorial McGraw-Hil, 2005. ISBN 8448132149.

50. GÓMEZ BARYOLO, Oiner. *Solución Informática de Autorización en Entornos Multientidad y Multisistema*. La Habana: Universidad de las Ciencias Informáticas, 2010.

ANEXOS

Anexo 1: Descripción del **RnF1** Fiabilidad: Tolerancia a fallos y Recuperabilidad.

Tabla 16 Descripcion del RnF 1

Atributo de Calidad	**Fiabilidad.**
Sub-atributos/Sub-característica	Tolerancia a fallos y Recuperabilidad.
Objetivo	Capacidad del producto para operar según lo previsto en presencia de fallos de hardware o software. Capacidad del producto para recuperar los datos directamente afectados y reestablecer el estado deseado del sistema en caso de interrupción o fallos.
Origen	Proveedor de requisitos.
Artefacto	El sistema y el código fuente.
Entorno	El sistema desplegado y existen fallos en la red y eléctricos.
Estímulo **<<1>>. <<a>> <Recuperación ante un fallo de conexión al servidor de mensajería >**	**Respuesta: Flujo de eventos** **(Escenarios)**
Fallos de red.	1. El sistema se encuentra capturando y enviando acciones de réplica de estructura hacia el destino. 2. El sistema intenta enviar los datos, pero detecta que está desconectado y persiste en la base de datos local todas las acciones a replicar hacia el nodo destino. 3. El sistema se conecta al servidor de mensajería y envía las acciones cuando se

	reestablece la comunicación con el servidor de mensajería. 4. El sistema concluye el proceso antes afectado satisfactoriamente.
<<1>>. <<b>> <Recuperación ante un fallo del fluido eléctrico >	**Respuesta: Flujo de eventos** **(Escenarios)**
Fallos eléctricos.	1. El sistema se encuentra capturando y enviando acciones de réplica de estructura hacia el destino. 2. El sistema falla por ausencia del fluido eléctrico. 3. El sistema se reestablece antes el fallo eléctrico iniciando de forma automática con el sistema operativo y cargando las últimas acciones no ejecutadas de su base de datos local. 4. El sistema concluye el proceso antes afectado satisfactoriamente.
Medida de respuesta	
Navegar en el sistema en presencia de fallos eléctricos y de recursos de red.	

Anexo 2: Descripción del **RnF2** Mantenibilidad: Analizabilidad.

Tabla 17 Descripcion del RnF 2

Atributo de Calidad	**Mantenibilidad.**
Sub-atributos/Sub-característica	Analizabilidad.
Objetivo	Facilidad con la que se puede evaluar el impacto de un determinando cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.
Origen	Arquitecto de software.
Artefacto	El código fuente.
Entorno	El ambiente de desarrollo del sistema.
Estímulo **<<1>>. <<a>>< Impacto de un determinado cambio sobre el resto del sistema>**	**Respuesta: Flujo de eventos** **(Escenarios)**
Se evalúa el impacto de un cambio en el sistema.	1. El arquitecto de software identifica los paquetes y clases implicados en el cambio, así como las funciones que se reutilizarán o se crearán para introducir el cambio. 2. Se evalúa el impacto partiendo de los resultados que arrojó el análisis anterior con respecto a la arquitectura del sistema.
<<1>>. <<b>>< Diagnosticar las deficiencias o causas de fallos en el sistema>	**Respuesta: Flujo de eventos (Escenarios)**
Se diagnostica las deficiencias o causas de fallos en el sistema.	1. El arquitecto de software identifica las posibles deficiencias o causas de fallos que se pueden originar en el sistema. 2. Se diagnostican las deficiencias o causas de

	fallos partiendo de los resultados que arrojó el análisis anterior.
<<1>>. <<c>>< Identificar las partes a modificar>	**Respuesta: Flujo de eventos (Escenarios)**
Se identifica las partes a modificar en el sistema.	1. El arquitecto de software identifica los paquetes y clases implicados en el cambio, así como las funciones que se reutilizarán o se crearán para introducir el cambio.
Medida de respuesta	
Analizar el cambio en el sistema.	

Anexo 3: Descripción del **RnF2** Mantenibilidad: Modificabilidad.

Tabla 18 Descripcion del RnF 2

Atributo de Calidad	**Mantenibilidad.**
Sub-atributos/Sub-característica	Modificabilidad.
Objetivo	Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.
Origen	Arquitecto de software.
Artefacto	El código fuente.
Entorno	El ambiente de desarrollo del sistema.
Estímulo	**Respuesta: Flujo de eventos**
<<1>>. <<a>>< Capacidad de modificación del sistema>	**(Escenarios)**

La arquitectura del sistema está diseñada para brindar facilidades a la hora de introducir modificaciones en el sistema. Esto permite que se puedan introducir cambios o modificaciones, que no afecten el correcto desempeño del resto de la solución.	NA
Medida de respuesta	
Introducir una modificación al sistema.	

Anexo 4: Descripción del **RnF2** Mantenibilidad: Capacidad para ser probado.

Tabla 19 Descripcion del RnF 2

Atributo de Calidad	**Mantenibilidad.**
Sub-atributos/Sub-característica	Capacidad para ser probado.
Objetivo	Facilidad con la que se pueden establecer criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.
Origen	Arquitecto de software y el analista.
Artefacto	Diseños de casos de pruebas.
Entorno	El sistema desplegado.
Estímulo **<<1>>. <<a>>< Facilidad con la que se pueden establecer criterios de prueba para el sistema>**	**Respuesta: Flujo de eventos** **(Escenarios)**
Desde la concepción inicial del sistema se definen y delimitan las funciones según los requisitos funcionales y no funcionales a implementar especificando los argumentos o variables de entrada y salida del sistema,	NA

elementos que favorecen el proceso de identificación y elaboración de los distintos escenarios de prueba.	
Medida de respuesta	
Escenario de prueba del sistema.	

Anexo 5: Descripción del **RnF3** Portabilidad: Adaptabilidad.

Tabla 20 Descripcion del RnF 3

Atributo de Calidad	**Portabilidad.**
Sub-atributos/Sub-característica	Adaptabilidad.
Objetivo	Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.
Origen	Arquitecto de software.
Artefacto	El sistema.
Entorno	El sistema desplegado.
Estímulo **<<1>>. <<a>>< Capacidad del sistema de adaptarse de forma efectiva a diferentes entornos>**	**Respuesta: Flujo de eventos** **(Escenarios)**
El sistema está diseñado con tecnologías que permiten que este se adapte a cualquier sistema operativo. El *script* de inicio de la aplicación permite personalizar los recursos de RAM de uso de la aplicación. El fichero de configuración de las propiedades del sistema permite configurar los parámetros	NA

del sistema según las prestaciones que posea la estación de trabajo donde se encuentre instalado.	
Medida de respuesta	
El ambiente de despliegue del sistema.	

Anexo 6: Descripción del **RnF4** Adecuación funcional: Integridad funcional.

Tabla 21 Descripcion del RnF 4

Atributo de Calidad	**Adecuación funcional.**
Sub-atributos/Sub-característica	Integridad funcional.
Objetivo	Grado en el que el conjunto de funciones cubre todas las tareas y objetivos del usuario especificados.
Origen	Proveedor de requisitos.
Artefacto	El sistema.
Entorno	El sistema desplegado.
Estímulo	**Respuesta: Flujo de eventos**
<<1>>. <<a>>< Grado en el que el sistema cubre todas las tareas y objetivos especificados>	**(Escenarios)**
El desarrollo del sistema esta guiado por las necesidades expresadas por parte de los proveedores de requisitos, dándole total cumplimiento a sus especificaciones declaradas e implícitas.	NA
Medida de respuesta	
Analizar las funcionalidades del sistema.	

Anexo 7: Descripción del **RnF4** Adecuación funcional: Corrección funcional.

Tabla 22 Descripcion del RnF 4

Atributo de Calidad	**Adecuación funcional.**
Sub-atributos/Sub-característica	Corrección funcional.
Objetivo	Grado en que un producto o sistema proporciona los resultados correctos con el grado necesario de precisión.
Origen	Proveedor de requisitos.
Artefacto	El sistema.
Entorno	El sistema desplegado.
Estímulo **<<1>>. <<a>>< Grado en el que el sistema proporciona los resultados correctos con precisión>**	**Respuesta: Flujo de eventos** **(Escenarios)**
El sistema realiza el proceso de réplica de estructura con la exactitud necesaria en cada uno de los nodos de réplica, en caso de que existan errores la solución es capaz de gestionar los conflictos para mantener la integridad de las base de datos implicadas en el proceso de réplica.	NA
Medida de respuesta	
Navegar en el sistema.	

Anexo 8: Descripción del **RnF5** Seguridad: No repudio.

Tabla 23 Descripcion del RnF 5

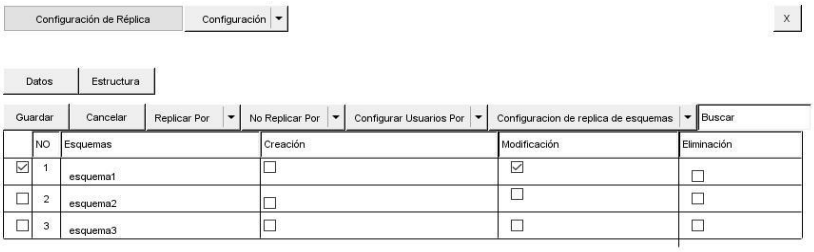
Atributo de Calidad	**Seguridad.**
Sub-atributos/Sub-característica	No repudio.
Objetivo	Grado en que las acciones o eventos pueden ser probados a haber tenido lugar, por lo que los eventos o acciones no pueden ser repudiados más tarde.
Origen	Arquitecto de software.
Artefacto	El sistema.
Entorno	El sistema desplegado.
Estímulo **<<1>>. <<a>>< Sistema de eventos o trazas.>**	**Respuesta: Flujo de eventos** **(Escenarios)**
El sistema una vez iniciado crea un conjunto de trazas a nivel de ficheros en la raíz de la solución en una carpeta nombrada log donde se almacenan tres archivos: Audit.log: contiene todas las trazas relacionadas con las respuestas de la aplicación al cliente. Error.log: contiene los errores ocurridos en el sistema. System.log: contiene todas las trazas relacionadas con las acciones ejecutadas por el sistema. Este sistema de trazas garantiza el no repudio sobre las acciones realizadas en el sistema.	NA
Medida de respuesta	

Navegar en el sistema.

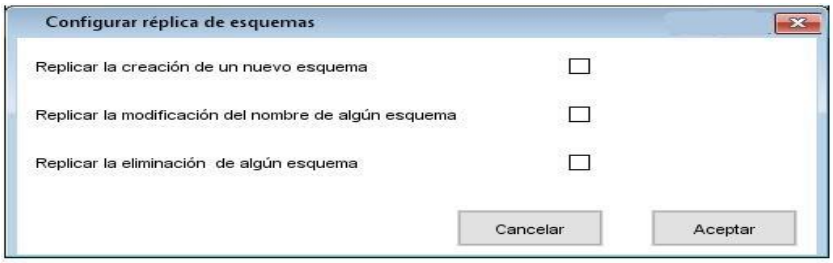
Anexo 9: Descripcion de la HU Adicionar configuración de réplica de estructura

Tabla 24 Descripción de HU del Adicionar configuración de réplica de estructura

Número: HU 1	**Nombre del requisito:** Adicionar configuración de réplica de estructura
Programador: Jessilié Paz Lara	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 15 días
Riesgo en Desarrollo: • Fallos eléctricos. • Planificación incorrecta.	**Tiempo Real:** 120 horas
Descripción: El sistema debe permitir al usuario adicionar una nueva configuración de réplica de estructura, debe mostrar una vista que le permita definir los siguientes parámetros: al listar los esquemas almacenados en la base de datos, debe brindar las opciones de “Replicar por”, “No Replicar por”, “Configurar Usuarios por”, las cuales se aplicarán a todos los objetos dentro de los esquemas seleccionados. La opción “Replicar por” debe permitir al usuario seleccionar el tipo de acción DDL deseada (“Creación”, “Modificación” y “Eliminación”). Al seleccionar esta acción se deberá registrar dicha acción para el esquema o los esquemas seleccionados y visualmente deberá aparecer un cuadro de chequeo activo indicando dicho proceso. La opción “No replicar por” deberá desactivar la acción para el esquema o los esquemas seleccionados y visualmente deberá aparecer un cuadro de chequeo no activo indicando dicho proceso. La opción “Configurar Usuarios por” permite adicionar los filtros de usuarios que se van a utilizar. El sistema también brinda la opción de realizar configuraciones para los esquemas existentes en la base de datos, es decir, “Replicar la creación de un nuevo esquema’, ‘Replicar la modificación del nombre de algún esquema” y ‘Replicar la eliminación de algún esquema” aplicándole el filtro de usuario a cada uno respectivamente. Al seleccionar el botón "Guardar" el sistema muestra un panel que permite elegir a que Nodo o Etiqueta se desea aplicar la configuración de réplica y crea las estructuras de control en la BD en caso de no existir. La opción “Cancelar” debe permitir al usuario regresar al estado anterior.	
Observaciones: NA	
Prototipo de interfaz:	



Interfaz para la configuración de réplica de estructura para cada objeto del esquema



Interfaz para la configuración de réplica de los esquemas

Anexo 10: Descripción de HU Editar configuración de réplica de estructura

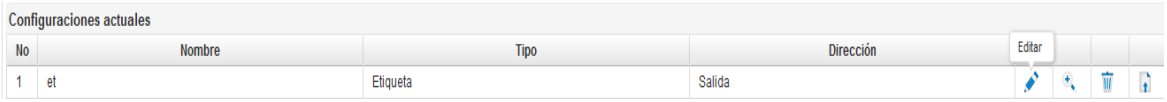
Tabla 25 Descripción de HU Editar configuración de réplica de estructura

Número: HU 5	**Nombre del requisito:** Editar configuración de réplica de estructura
Programador: Jessilié Paz Lara	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20 días
Riesgo en Desarrollo: Fallos eléctricos.	**Tiempo Real:** 160 horas
Descripción: Si el usuario desea editar una configuración de réplica que se tiene registrada. Para ello el sistema brinda la opción “Editar”, al usuario seleccionar esta opción, el sistema debe cargar automáticamente la configuración que se tiene almacenada para el Nodo o Etiqueta seleccionado. Visualmente se deberá mostrar un formulario con dos pestañas, uno para Datos y otra para Estructuras. Para el caso de Datos, el sistema debe permitir al usuario modificar los parámetros establecidos como el tipo de acción mediante las opciones “Replicar por” o “No replicar por” y los	

filtros de la configuración que se pudieron haber establecidos (Filtros de usuarios, Filtro sql, y Columnas por referencia). El sistema brinda las opciones “Guardar” y “Cancelar”. Mediante la opción “Guardar” el sistema debe actualizar los cambios modificados por el usuario en la configuración de réplica seleccionada. La opción “Cancelar” debe permitir al usuario regresar al estado anterior. Para el caso de Estructuras, el sistema debe permitir al usuario modificar los parámetros establecidos como el tipo de acción mediante las opciones “Replicar por” o “No replicar por” y los filtros de la configuración que se pudieron haber establecidos (Filtros de usuarios). El sistema brinda las opciones “Guardar” y “Cancelar”. Mediante la opción “Guardar” el sistema debe actualizar los cambios modificados por el usuario en la configuración de réplica seleccionada. La opción “Cancelar” debe permitir al usuario regresar al estado anterior.

Observaciones: NA

Prototipo de interfaz:



Interfaz donde se listan las configuraciones actuales y te permite editar

Interfaz que te permite modificar las configuraciones de réplica de datos

Interfaz que te permite modificar las configuraciones de réplica de estructura

Anexo 11: Descripción de HU Eliminar configuración de réplica de estructura

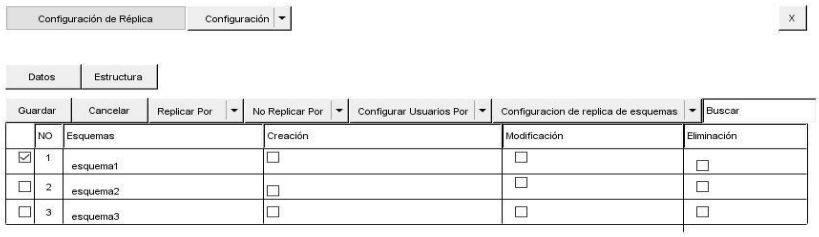
Tabla 26 Descripción de HU Eliminar configuración de réplica de estructura

Número: HU 6	**Nombre del requisito:** Eliminar configuración de réplica de estructura
Programador: Jessilié Paz Lara	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 15 días
Riesgo en Desarrollo: Fallos eléctricos.	**Tiempo Real:** 120 horas
Descripción: El sistema necesita eliminar una configuración de réplica previamente registrada. Para ello el sistema brinda la opción “Eliminar”, el sistema debe eliminar la configuración que el usuario seleccione. Se muestra una interfaz con las configuraciones almacenadas donde se podrá seleccionar la que se desee eliminar.	
Observaciones: NA	
Prototipo de interfaz: 	

Anexo 12: Descripción de HU Listar esquemas

Tabla 27 Descripción de HU Listar esquemas

Número: HU 2	**Nombre del requisito:** Listar esquemas
Programador: Jessilié Paz Lara	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 10 días
Riesgo en Desarrollo: Fallos eléctricos.	**Tiempo Real:** 80 horas
Descripción: El sistema muestra el listado de los esquemas de la Base de datos.	
Observaciones: NA	
Prototipo de interfaz:	

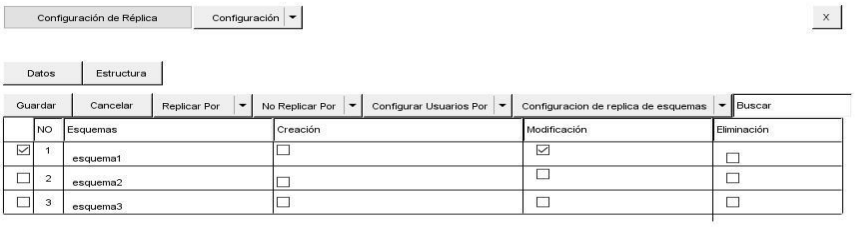


Interfaz que te muestra el listado de los esquemas

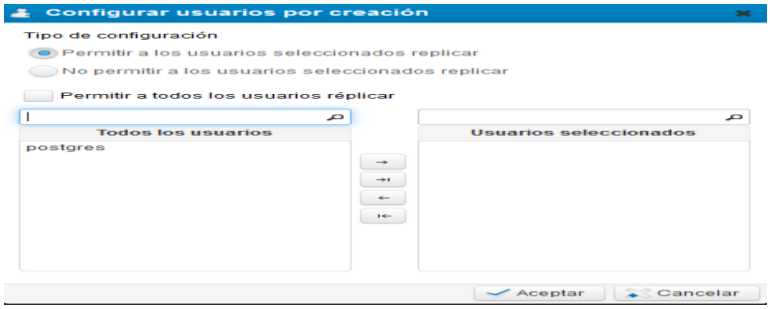
Anexo 13: Descripción de HU Adicionar filtros de usuarios

Tabla 28 Descripción de HU Adicionar filtros de usuarios

Número: HU 3	**Nombre del requisito:** Adicionar filtros de usuarios
Programador: Jessilié Paz Lara	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 15 días
Riesgo en Desarrollo: Fallos eléctricos.	**Tiempo Real:** 120 horas
Descripción: El sistema debe permitir aplicar filtros de usuarios sobre las acciones permitidas en la configuración de réplica de estructura en la opción **"Configurar Usuarios Por"**. Visualmente se muestra cuando uno o varios esquemas seleccionados tienen activo algunas de las acciones permitidas.	
Observaciones: Solo se le pueden aplicar filtros de usuarios de cualquier tipo de acción permitida a los esquemas que repliquen estas acciones.	
Prototipo de interfaz:	



Interfaz que muestra la opción **"Configurar Usuarios Por"**.



Interfaz que permite adicionar los filtros de usuarios

Anexo 14: Descripción de HU Buscar esquema

Tabla 29 Descripción de HU Buscar esquema

Número: HU 4	**Nombre del requisito:** Buscar esquema
Programador: Jessilié Paz Lara	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 5 días
Riesgo en Desarrollo: Fallos eléctricos.	**Tiempo Real:** 40 horas
Descripción: El sistema debe permitirle al usuario la búsqueda de un esquema por el nombre.	
Observaciones: NA	
Prototipo de interfaz:	

Interfaz para realizar una búsqueda por el nombre de un esquema

Anexo15: Diagrama de paquete RF1, RF1.1, RF1.2, RF1.3, RF2, RF3

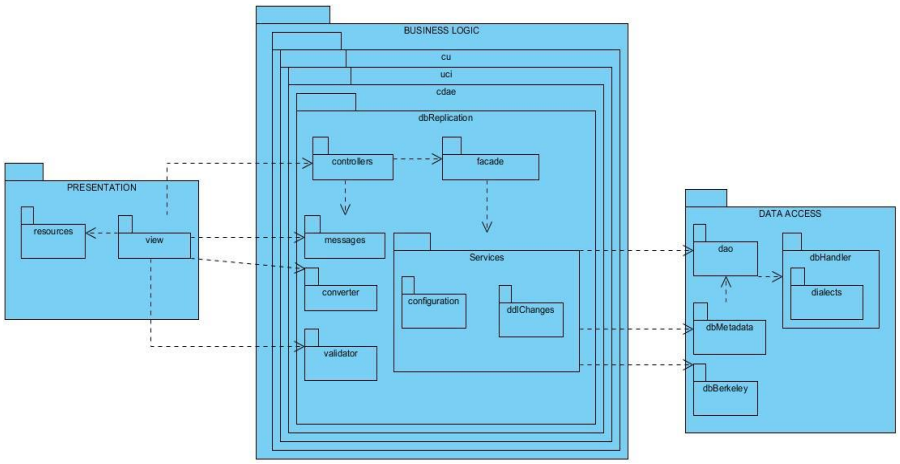


Figura 19 Diagrama de Paquete del RF 1, RF 1.1,RF 1.2, RF 1.3, RF 2, RF3

Anexo16: Diagrama de clase RF1, RF1.1, RF1.2, RF1.3, RF2, RF3

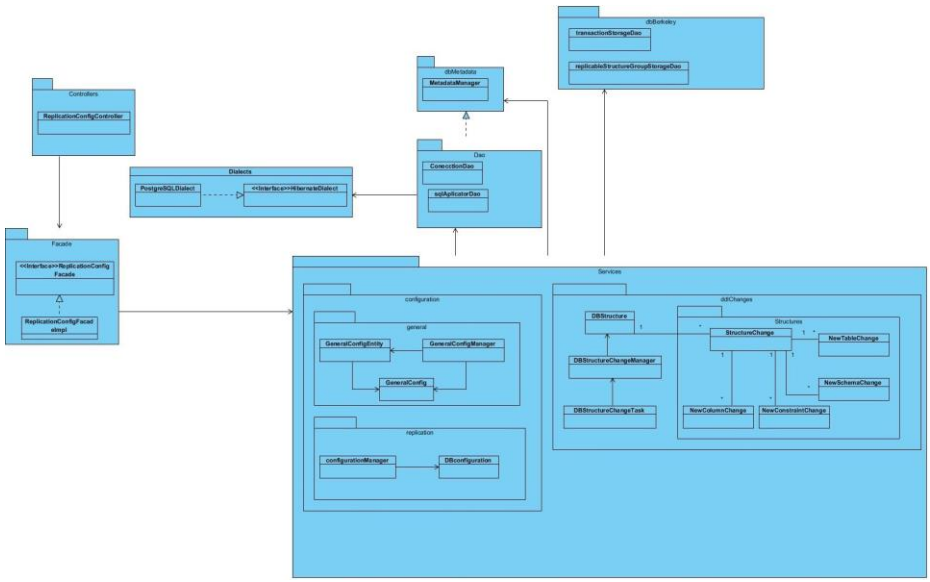


Figura 20 Diagrama de clase del RF 1, RF 1.1, RF 1.2, RF 1.3, RF 2, RF3

Anexo 17: Descripción de la clase DBConfiguration

Representa la configuración de un nodo.

Tabla 30 Descripción de la clase DBConfiguration

Descripción de la clase DBConfiguration	
Nombre: DBConfiguration	
Tipo de clase: Controladora.	
Atributos	**Tipos**
Id	private
tableConfigMap	private
schemaConfigMap	private
replicateOnSchemaCreate	private
replicateOnSchemaAlter	private
replicateOnSchemaDelete	private
users	private
configuration	private
Responsabilidades	
Nombre:	addTableConfig (TableConfig tableConfig):void
Descripción:	Adiciona la configuración de réplica de datos al mapa de configuración de tablas.
Nombre:	getTableConfig (String table):TableConfig
Descripción:	Dado el nombre de una tabla, retorna la configuración de réplica que tiene si existe en el mapa de configuraciones de réplica de datos.
Nombre:	getTableConfigMap ():Map<String, TableConfig>
Descripción:	Retorna el mapa de configuraciones de réplica de datos.
Nombre:	getTableMapConfigWithSchema ():Map<String, List<TableConfig>>
Descripción:	Retorna un mapa con una lista de configuraciones de réplica de datos que tiene.
Nombre:	addSchemaConfig (SchemaConfig schemaConfig):void
Descripción:	Adiciona la configuración para la réplica de estructura.
Nombre:	getSchemaConfig (String schemaName): SchemaConfig
Descripción:	Dado el nombre de un esquema, retorna la configuración de réplica que tiene si existe en el mapa de configuraciones de réplica de estructura.

Nombre:	getSchemasWithConfig ():List<String>
Descripción:	Retorna los nombres de todos los esquemas con configuración de réplica.
Nombre:	getUserFilterByActionType (int actionType):UserFilter
Descripción:	Retorna el filtro de usuario por cada tipo de acción.
Nombre:	getAllSchemaConfigs ():List<SchemaConfig>
Descripción:	Retorna un listado de configuraciones de esquemas.
Nombre:	isReplicateOnSchemaCreate ():boolean
Descripción:	Retorna verdadero para el tipo de accion Create.
Nombre:	isReplicateOnSchemaAlter ():boolean
Descripción:	Retorna verdadero para el tipo de accion Alter.
Nombre:	isReplicateOnSchemaDrop ():boolean
Descripción:	Retorna verdadero para el tipo de accion Drop.

Anexo 18: Descripción de la clase ReplicationConfigController

Clase controladora de las vistas de configuraciones de réplica.

Tabla 31 Descripción de la clase ReplicationConfigController

Descripción de la clase ReplicationConfigController	
Nombre: ReplicationConfigController	
Tipo de clase: Controladora.	
Atributos	**Tipos**
List<String> schemasList	private
List<String> schemasSelectConf	private
String schemaSelected	private
Map<String, SchemaConfig> schemaConfigMap	private
Boolean schemaFlag	private
SchemaConfig schemaConfigSelected	private
Responsabilidades	
Nombre:	getSchemasList():List<String>
Descripción:	Obtiene los esquemas de la BD y los adiciona al maps de configuraciones de esquemas.
Nombre:	getSchemasSelectConf():List<String>
Descripción:	Obtiene los esquemas seleccionados por el usuario en la interfáz de configuración de estructura.

Nombre:	saveConfiguration(): void
Descripción:	Guarda la configuración de réplica.
Nombre:	editReplicConfiguration(String idApplyTo): void
Descripción:	Editar configuración de réplica dado el identificador del nodo o etiqueta.
Nombre:	deleteConfiguration(): void
Descripción:	Elimina la configuración de réplica.
Nombre:	onSelectSchemaCreateAction(String Schema): void
Descripción:	Adiciona la acción Create al esquema seleccionado.
Nombre:	onUnSelectSchemaCreateAction(String Schema):void
Descripción:	Elimina la acción Create al esquema seleccionado.
Nombre:	onSelectSchemaAlterAction(String Schema):void
Descripción:	Adiciona la acción Alter al esquema seleccionado.
Nombre:	onUnSelectSchemaAlterAction(String Schema):void
Descripción:	Elimina la acción Alter al esquema seleccionado.
Nombre:	onSelectSchemaDropAction(String Schema):void
Descripción:	Adiciona la acción Drop al esquema seleccionado.
Nombre:	onUnSelectSchemaDropAction(String Schema):void
Descripción:	Elimina la acción Drop al esquema seleccionado.
Nombre:	onSelectMultipleSchemaCreateAction(): void
Descripción:	Adiciona la acción Create a los esquemas seleccionados.
Nombre:	onUnSelectMultipleSchemaCreateAction()void
Descripción:	Elimina la acción Create a los esquemas seleccionados.
Nombre:	onSelectMultipleSchemaAlterAction():void
Descripción:	Adiciona la acción Alter a los esquemas seleccionados.
Nombre:	onUnSelectMultipleSchemaAlterAction():void
Descripción:	Elimina la acción Alter a los esquemas seleccionados.
Nombre:	onSelectMultipleSchemaDropAction():void
Descripción:	Adiciona la acción Drop a los esquemas seleccionados.
Nombre:	onUnSelectMultipleSchemaDropAction():void
Descripción:	Elimina la acción Drop a los esquemas seleccionados.
Nombre:	onSelectAllSchemasAction(): void
Descripción:	Adiciona las acciones Create, Alter y Drop a los esquemas seleccionados.

Nombre:	onUnSelectAllSchemasAction(): void
Descripción:	Elimina las acciones Create, Alter y Drop a los esquemas seleccionados.
Nombre:	isCreateSchemasAction(String Schema) : Boolean
Descripción:	Verifica si una tabla almacenada en el mapa de configuraciones de estructura, posee tipo de acción Create.
Nombre:	isAlterSchemasAction(String Schema) : Boolean
Descripción:	Verifica si una tabla almacenada en el mapa de configuraciones de estructura, posee tipo de acción Alter
Nombre:	isDropSchemasAction(String Schema) : Boolean
Descripción:	Verifica si una tabla almacenada en el map de configuraciones de estructura, posee tipo de acción Drop.
Nombre:	cleanUsers(): void
Descripción:	Resetea los valores almacenados de la configuración de usuarios.
Nombre:	loadSchemasConfigUser(int actionType): void
Descripción:	Carga los valores almacenados de la configuración de usuarios de los esquemas y en el visual levanta la ventana que corresponde a la acción.
Nombre:	replicWin(int actionType): void
Descripción:	Carga los valores almacenados de la configuración de usuarios en la configuración general y en el visual levanta la ventana que corresponde a la acción.
Nombre:	isCreationSetOn(): boolean
Descripción:	Verifica si es premitido la accion Create en la configuración general.
Nombre:	isAlterSetOn(): boolean
Descripción:	Verifica si es premitido la accion Alter en la configuración general.
Nombre:	isDropSetOn(): boolean
Descripción:	Verifica si es premitido la accion Drop en la configuración general.
Nombre:	showDetails(String id): void
Descripción:	Muestra detalles de la conficuración.

Nombre:	searchScheme(String schema): void
Descripción:	Busca un esquema y lo muestra en el visual.

Anexo 19: Descripción de la clase PostgresDialect

Clase que representa el dialecto de Postgres.

Tabla 32 Descripción de la clase PostgresDialect

Descripción de la clase PostgresDialect	
Nombre: PostgresDialect	
Tipo de clase: Modelo.	
Atributos	**Tipos**
-	-
Responsabilidades	
Nombre:	dropStructureCaptureTriggers(): String
Descripción:	Devuelve la cadena SQL para eliminar los triggers de captura de cambios de estructura de la BD, según el dialecto de PostgreSQL
Nombre:	createStructureCaptureFunction(): String
Descripción:	Devuelve la cadena SQL para crear la función de captura de estructura inicial en la BD, según el dialecto de PostgreSQL
Nombre:	executeStructureCaptureFunction(): String
Descripción:	Devuelve la cadena SQL para ejecutar la función de captura de estructura inicial en la BD, según el dialecto de PostgreSQL
Nombre:	dropStructureCaptureFunction(): String
Descripción:	Devuelve la cadena SQL para eliminar la función de captura de estructura inicial de la BD, según el dialecto de PostgreSQL
Nombre:	createStructureCaptureTriggerCreateAlter(): String
Descripción:	Devuelve la cadena SQL para crear el trigger de captura de cambios de estructura de tipo CREATE / ALTER en la BD, según el dialecto de PostgreSQL

Nombre:	createStructureCaptureTriggerDrop(): String
Descripción:	Devuelve la cadena SQL para crear el trigger de captura de cambios de estructura de tipo DROP en la BD, según el dialecto de PostgreSQL
Nombre:	createNewStructureControlTableSQL(): String
Descripción:	Devuelve la cadena SQL para crear la tabla de control de cambios de réplica en la BD, según el dialecto de PostgreSQL
Nombre:	createCaptureSchemasTableSQL(): String
Descripción:	Devuelve la cadena SQL para crear la tabla de captura de estructura para los esquemas en la BD, según el dialecto de PostgreSQL
Nombre:	createCaptureTablesTableSQL(): String
Descripción:	Devuelve la cadena SQL para crear la tabla de captura de estructura para las tablas en la BD, según el dialecto de PostgreSQL
Nombre:	createCaptureColumnsTableSQL(): String
Descripción:	Devuelve la cadena SQL para crear la tabla de captura de estructura para las columnas en la BD, según el dialecto de PostgreSQL
Nombre:	createCaptureConstraintsTableSQL(): String
Descripción:	Devuelve la cadena SQL para crear la tabla de captura de estructura para las restricciones en la BD, según el dialecto de PostgreSQL
Nombre:	dropStructureCaptureTriggers(): String
Descripción:	Devuelve la cadena SQL para eliminar los triggers de captura de cambios de estructura de la BD, según el dialecto de PostgreSQL
Nombre:	createStructureCaptureFunction(): String
Descripción:	Devuelve la cadena SQL para crear la función de captura de estructura inicial en la BD, según el dialecto de PostgreSQL
Nombre:	executeStructureCaptureFunction(): String
Descripción:	Devuelve la cadena SQL para ejecutar la función de captura de estructura

	inicial en la BD, según el dialecto de PostgreSQL

Anexo 20: Diagrama de componente HU1, HU2, HU3, HU4, HU5, HU6

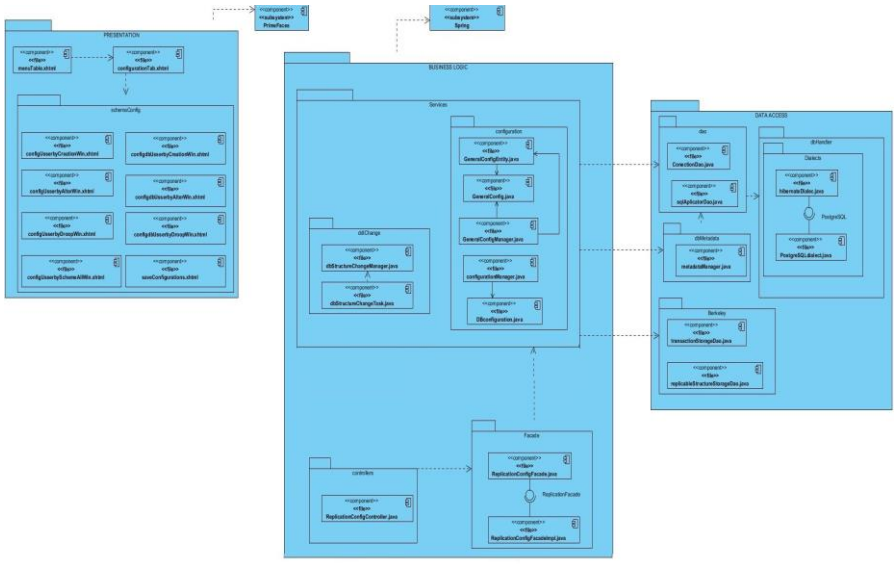


Figura 21 Diagrama de componente HU 1,HU2, HU3, HU4, HU5, HU6

Anexo 21: Diagrama de componente HU8

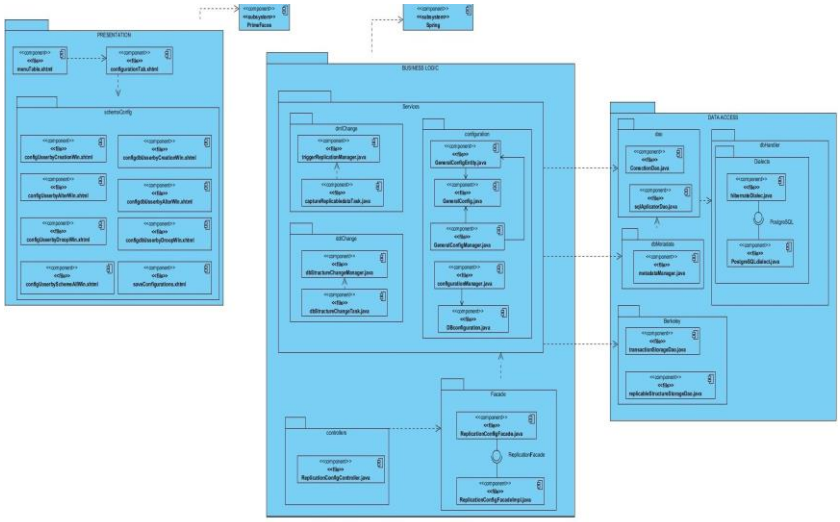


Figura 22 Diagrama de componente HU8

Anexo 22: Estándares de codificación java

Constantes: las constantes o campos finales son escritos en letras mayúsculas y separando las palabras por un guión bajo "_", ejemplo:

- `String schema_name = result.getString("SCHEMA_NAME");`

Clases:

Clases **Controladoras**: estas clases deben terminar con la palabra "**Manager** ", ejemplos:

- `public class DBStructureChangeManager {}`
- `public class MetadataManager {}`

Clases **Exceptions:** estas clases deben terminar con la palabra "**Exception**", ejemplo:

- `public class DataAccessException extends Exception{}`

Clases **DAO**: las clases que se utilicen como Objeto de Acceso a Datos deben terminar con la palabra "**Dao**", ejemplos:

- `public interface DBStructureDao {}`

Normas de Comentarios

Comentarios de Bloque: los comentarios de bloque se usan para proporcionar descripciones de ficheros, métodos, estructuras de datos y algoritmos. Deben ser utilizados al comienzo de cada archivo y antes de cada método, ejemplo:

```
/*---- Elimina de la tabla de control de estructuras
*  la tupla especificada -----*/
```

Comentario de una sola línea: este tipo de comentario puede aparecer al mismo nivel del código al que se desea comentar. Un comentario de una sola línea debe ser precedido por una línea en blanco. ejemplo:

```
// obtener el nombre del schema
String schemaName = schema_tableName[0];
// obtener el nombre de la tabla
String tableName = schema_tableName[1];
```

Comentario detrás del código: los comentarios cortos pueden aparecer en la misma línea que el código que describen, pero deben estar suficientemente separadas de las declaraciones. Si más de un comentario corto aparece en un fragmento de código, todos ellos deben tener la misma sangría, ejemplo:

```
public String columnDefinition(…) {
      return columQuery.toString(); /* retorna la columna */
}
```

Nota: No usar los comentarios "*//*" para escribir varias líneas de texto consecutivamente. **Comentarios de las clases**: al inicio de cada clase se debe describir con un comentario de bloque el propósito de la clase e instrucciones de uso. También se pueden incluir recordatorios o avisos acerca de las mejoras necesarias o convenientes. Es necesario especificar el autor y la versión de la clase, utilizando anotaciones de Java, ejemplo:

```
/**
 * Actualiza las configuraciones de replica existentes en el objeto
 * configuracion que se recive por parametro, se recorre el map de
configuraciones
 * de la tablas y sverifica que cada tablas que esta en la
configuracion del
 * objeto BDConfiuration exista en la BD real de la aplicacion, de nos
ser asi
 * se elimina esta configuracion asi como los trigger asociados a esta
y los rastros
 *de configuracion que fueron capturados en la tabla de control y que
esperan
 *para ser replicados
 * @author Jessilié Paz Lara
 * @param configuration
 */
public class updateAllReplicConfig {}
```

Anexo 23: Instrumentos de evaluación de la métrica TOC

	Clases	Métodos	TOC		
			Responsabilidad	Complejidad	Reutilización
3	GeneralConfigManager	1	Baja	Baja	Alta
4	ConfigurationManager	4	Baja	Baja	Alta
5	Dbconfiguration	12	Baja	Baja	Alta
6	DBStructureChangeManager	24	Media	Media	Media
7	ReplicationConfigController	30	Media	Media	Media
8	PostgresSQLDialect	14	Baja	Baja	Alta
9	MetadataManager	43	Alta	Alta	Baja
10	ReplicationConfigFacadeImpl	17	Baja	Baja	Alta
11	HibernateDialect	150	Alta	Alta	Baja
12	ConectionDao	34	Media	Media	Media
13	GeneralConfigEntity	2	Baja	Baja	Alta
14	DBStructure	2	Baja	Baja	Alta
15	DBStructureChangeTask	2	Baja	Baja	Alta
16	CaptureReplicableDataTask	1	Baja	Baja	Alta
17	TriggersReplicationManager	17	Baja	Baja	Alta
18	SystemTuples	28	Media	Media	Media
19	ReplicableGroup	3	Baja	Baja	Alta
20	ReplicableAction	14	Baja	Baja	Alta
21	ReplicableStructureGroup	3	Baja	Baja	Alta
22	ReplicableStructureAction	23	Media	Media	Media
23	transactionStorageDao	1	Baja	Baja	Alta
24	replicableStructureGroupStora	1	Baja	Baja	Alta
25					
26					
27					
28					
29	Promedio de métodos por clases	19.3636			

Indicadores de Calidad	Evaluación %		
	Baja	Media	Alta
Responsabilidad	68.1818182	22.7273	9.09091
Complejidad de implementaci	68.1818182	22.7273	9.09091
Reutilización	9.09090909	22.7273	68.1818

Figura 23 Aplicación de la métrica TOC

Anexo 24: Instrumentos de evaluación de la métrica RC

	Clases	Relaciones	Acoplamiento	Complejidad de mantenimiento	Reutilización	Cantidad de Pruebas
3	GeneralConfigManag	2	Media	Media	Media	Media
4	ConfigurationManag	4	Alta	Alta	Baja	Alta
5	Dbconfiguration	0	Ninguno	Baja	Alta	Baja
6	DBStructureChangeM	4	Alta	Alta	Baja	Alta
7	ReplicationConfigCo	4	Alta	Alta	Baja	Alta
8	PostgresSQLDialect	1	Baja	Baja	Alta	Baja
9	MetadataManager	0	Ninguno	Baja	Alta	Baja
10	ReplicationConfigFac	1	Baja	Baja	Alta	Baja
11	HibernateDialect	1	Baja	Baja	Alta	Baja
12	ConectionDao	0	Ninguno	Baja	Alta	Baja
13	GeneralConfigEntity	1	Baja	Baja	Alta	Baja
14	DBStructure	0	Ninguno	Baja	Alta	Baja
15	DBStructureChangeTa	1	Baja	Baja	Alta	Baja
16	CaptureReplicableDa	1	Baja	Baja	Alta	Baja
17	TriggersReplicationM	1	Baja	Baja	Alta	Baja
18	SystemTuples	0	Ninguno	Baja	Alta	Baja
19	ReplicableGroup	2	Media	Media	Media	Media
20	ReplicableAction	2	Media	Media	Media	Media
21	ReplicableStructureG	2	Media	Media	Media	Media
22	ReplicableStructureA	2	Media	Media	Media	Media
23	transactionStorageDa	1	Baja	Baja	Alta	Baja
24	replicableStructureG	1	Baja	Baja	Alta	Baja
25						
26						
27						
28						
29	Promedio	1.4090909				

Indicadores de Calidad	Baja	Media	Alta
Acoplamiento	40.909091	22.727273	13.636364
Complejidad de mantenimient	63.636364	22.727273	13.636364
Reutilización	13.636364	22.727273	63.636364
Cantidad de Pruebas	63.636364	22.727273	13.636364

Figura 24 Aplicación de la métrica RC

Anexo 25: Caso de prueba Adicionar configuración de réplica de estructura

Caso de Prueba	
Código: SC_1	**Historia de usuario**: 1
Nombre: Adicionar configuración de réplica de estructura	
Descripción: prueba para la funcionalidad adicionar configuración de réplica de estructura	
Condiciones de Ejecución: • El usuario debe estar autenticado. • Establecer la configuración de BD para PostgreSQL en el Módulo de Configuración General. • Tener al menos un Nodo configurado.	
Respuesta del Sistema/Flujo Central • En la interfaz de Configuraciones actuales el sistema brinda la opción de adicionar “Nueva” configuración, luego seleccionando el tab de “Estructura” el sistema muestra un panel con los esquemas existentes en la BD. En esta interfaz, el sistema ofrece las opciones “Replicar por” los tipos de acciones permitidas (Creación, Modificación, Eliminación y Todo), “No replicar por” (Creación, Modificación, Eliminación y Todo), “Configurar usuarios por” (Creación, Modificación, Eliminación y Todo) y “Configurar la réplica de esquemas”, que de seleccionar está opción el sistema muestra otro panel con las opciones de “Replicar la creación de un nuevo esquema”, “Replicar la modificación del nombre de algún esquema”, “Replicar la eliminación de algún esquema” y por cada una de estas brinda la posibilidad aplicar filtros de usuarios. • Al seleccionar el botón "Guardar" el sistema muestra un panel que permite elegir a que Nodo o Etiqueta se desea aplicar la configuración réplica. Crea las estructuras de control en la BD en caso de no existir, inicia la captura de cambios de estructura si existen configuraciones de réplica de este tipo.	
Resultado Esperado: si no existen configuraciones de réplica almacenadas, el sistema realiza la captura de la estructura inicial almacenándola en las tablas de control, de lo contrario el sistema actualiza la configuración de réplica que ya tiene e inicia la captura de cambios de estructura.	
Evaluación de la Prueba: bien	

Anexo 26: Caso de prueba Actualizar automáticamente configuración de réplica a partir de cambios de estructura

Caso de Prueba	
Código: SC_8	**Historia de usuario**: 8
Nombre: Actualizar automáticamente configuración de réplica a partir de cambios de estructura	
Descripción: prueba para la funcionalidad establecer orden en el proceso de réplica	
Condiciones de Ejecución: • El usuario debe estar autenticado. • Establecer la configuración de BD para PostgreSQL en el Módulo de Configuración General. • Tener al menos un Nodo configurado.	
Respuesta del Sistema/Flujo Central • El sistema deberá ser capaz de actualizar las configuraciones de réplica para cada cambio de estructura. Si es CREATE, se deberá actualizar en el caso de la vista de Datos las tablas que se listan al seleccionar un esquema y para el caso de la vista de Estructura los esquemas que se listan, de esta forma se le podrán adicionar configuración de réplica a las nuevas tablas o esquemas. • Si es DROP, el sistema deberá verificar si las tablas con configuración existen en la BD, si no el sistema eliminará las acciones DML con configuración de réplica sobre la tabla eliminada que están pendientes por ser replicadas. • Si es ALTER, el sistema deberá verificar si hay datos capturados en la tabla de control, por cada cambio el tipo de acción DDL a replicar, en caso de que sea algun tipo de acción alter, el sistema deberá eliminar las acciones DML sobre la tabla, los cambios DML que estan pendientes por replicar y deberá actualizar la cofiguracion de réplica para la tabla.	
Resultado Esperado: el sistema deberá actualizar las configuraciones de réplica.	
Evaluación de la Prueba: bien	