UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 5

Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE)

Solución para la visualización de configuraciones de réplica del Replicador de datos REKO

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Alejandro Ortega Baez

Tutor: Ing. Frank Rosales Muñoz

La Habana, Julio de 2015

"Año 57 del Triunfo de la Revolución Cubana"

"Our deepest fear is not that we are inadequate.
Our deepest fear is that we are powerful beyond measure.
It is our light, not our darkness, that most frightens us.
Your playing small does not serve the world.
There is nothing enlightened about shrinking
so that other people won't feel insecure around you.
We are all meant to shine as children do.
It's not just in some of us; it is in everyone.
And as we let our own lights shine,
we unconsciously give other people permission to do the same.
As we are liberated from our own fear,
our presence automatically liberates others"

Marianne Williamson

Declaración de autoría

Declaro ser el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____	_____
Alejandro Ortega Baez	Frank Rosales Muñoz
Autor	**Tutor**

Datos de Contacto

Tutor:

Ing. Frank Rosales Muñoz.

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Correo: frankr@uci.cu

Dedicatoria

A mi padre y madre, los mejores padres del mundo

Agradecimientos

A papá y a mamá por haberme dado la vida, criado y educado; por haberme enseñado con el ejemplo a siempre esforzarme y nunca rendirme.

A Anita, el ser tan extraordinario que tengo como hermana, siempre energética y llena de alegría, que es capaz de subir a la cima más alta sin importar los obstáculos que tenga por delante.

A tío Pablo que, por todo lo que ha hecho, se merece la mitad del título de ingeniero.

A abuelo Piter y abuela Betty, que se merecen la otra mitad, que siempre estuvieron al tanto de todo y siempre se preocuparon porque todo saliera bien.

A Susej por ser la mejor jefa de año del mundo, una persona increíble y ejemplo como ser humano.

A Zoraida, por ser un excelente ser humano y persona, que sólo tiene cariño y alegría que dar a sus alumnos.

Esas dos profes, que mejores que esas en el mundo no hay, que siempre sacan lo mejor de uno y nunca se rinden con nadie, siempre dan deseos de ser mejor y de esforzarse más.

Por supuesto a la gente del apartamento, al bobo con gorra, al calvito, a sonzana la jefa de apartamento, al hijo de Crane, a Ernesto, al rottweiler guardián que tenemos.

A amistades que sin cuya ayuda no hubiera podido terminar el trabajo a Susana, a Rosmery y a Armando que me brindaron su ayuda desinteresada cuando más la necesitaba y me impulsaron a salir adelante.

A Pedrito, por su orientación y apoyo desde que escogí esta carrera hasta hoy.

A Rafael que siempre estuvo ayudando y aconsejando durante todo el trayecto, y siempre sirvió de guía y de voz de experiencia.

A Yusmary y a Frank.

A Rigo que siempre muestra su preocupación y apoyo por sus estudiantes.

A Osmani, Orestes, Noel, Luis, por todas las sugerencias y consejos.

A Katia también por todos los consejos y sobre todo por ser estricta con el documento, lo necesitaba.

A Claudia.

A todas las personas que de una forma u otra me ayudaron y apoyaron durante todo el trayecto, que son tantos que no cabrían en las 80 páginas del documento. Sin su apoyo no hubiera sido posible lograrlo. Gracias por tener fe en mí, incluso en aquellos momentos cuando ni yo mismo sabía cómo saldría adelante.

Gracias a todos

Resumen

El software REKO, es un replicador de bases de datos distribuidas, desarrollado en la Universidad de las Ciencias Informáticas que pretende cubrir las principales necesidades de replicación. Es un sistema multiplataforma que soporta a los principales gestores de bases de datos y se utiliza como solución de réplica en el Sistema Gestión Penitenciario Venezolano, en el Sistema Gestión de Hospitales Alas-His, entre otros.

Actualmente no cuenta con un área de trabajo capaz de visualizar las configuraciones de réplica en escenarios en los cuales la misma alcanza un tamaño y complejidad extremadamente grandes. Para suplir esta necesidad se implementó una solución informática que se complementa al REKO. Dicho sistema permite tres funcionalidades fundamentales: importar la configuración general de un nodo REKO, listar las configuraciones de réplica actuales presentes en el mismo y eliminar la configuración deseada. Resultando en una solución práctica a un problema real existente, ahorrando al país de gasto de divisas en productos informáticos desarrollados por terceros.

Palabras clave: replicador, bases de datos distribuidas, configuración de réplica.

Índice

Índice de figuras

Índice de tablas

Introducción

En la actualidad, debido al desarrollo alcanzado en las organizaciones, la gestión y la administración de la información, se hace muy compleja y es imprescindible un enfoque distribuido, orientado a la red de redes. Dado los grandes volúmenes de información y las diferentes sucursales de una organización, los almacenes de datos superan fronteras geográficas.

Precisamente este hecho, es el que fomenta la creación de Sistemas de Bases de Datos Distribuidas (SBDD). Los SBDD ganan en disponibilidad de la información pero se complejiza la sincronización de los datos, y es para eso que se apoyan en sistemas administradores de bases de datos distribuidas, también conocidos como replicadores de bases de datos. La replicación se utiliza para descentralizar datos en un ambiente distribuido, con el objetivo de obtener mejor rendimiento y confiablidad, mediante la reducción de dependencia de un sistema de base de datos centralizado.

En la Universidad de las Ciencias Informáticas (UCI), específicamente en el Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE), se ha desarrollado una solución de réplica en ambientes distribuidos, llamada REKO. Esta herramienta multiplataforma pretende 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 (1), la protección y la recuperación de los datos, así como la satisfacción de otras necesidades relacionadas con la distribución de datos entre los gestores más populares. Actualmente está siendo usado en el Sistema de Gestión Penitenciario Venezolano, en el Sistema de Gestión de Hospitales Alas-His, entre otros.

Atendiendo a las reglas de su configuración, los replicadores determinan qué, cómo, cuándo y hacia dónde realizar la réplica. Si no se establece ninguna configuración, el sistema interpreta que no se desea realizar ninguna réplica. El Replicador de datos REKO es capaz de realizar réplica de datos y de estructura, cada una con una configuración independiente de la otra, aunque se refieran a la misma tabla de la base de datos a replicar; ambas partes conforman la configuración de réplica.

Además que cada configuración de réplica puede ser lógicamente dividida por su tipo, éstas a su vez son también lógicamente separables por cada acción replicable que pueda afectar a la tabla, dígase inserción, actualización o eliminación.

Cada acción replicable puede ser configurada independientemente de las demás. A cada una de éstas últimas subdivisiones se les pueden definir una serie de filtros, para determinar específicamente en qué situación se quiere realizar la réplica y cuándo no. Los filtros son los siguientes:

- **Filtros SQL:** permiten crear reglas atendiendo a condiciones o valores en las tablas, que activan o inhiben la réplica. Pueden ser tan complejos como una sentencia SQL válida.
- **Filtros de usuarios:** precisan cuáles usuarios tienen permitido replicar. También pueden enunciar los que tienen explícitamente prohibido hacerlo.
- **Columnas de referencia:** permiten especificar cuáles columnas de la tabla son las que van a replicar, para situaciones en las que no se requiera replicar la tabla completa. Este filtro incluye además un filtro SQL interno que permite crear una regla atendiendo a los valores que las columnas de la tabla almacenan. Como en el caso anterior, este filtro SQL soporta cualquier regla que se defina.

También es posible incorporar filtros de archivos de referencia y transformaciones de datos. Como se puede apreciar, la configuración de la réplica en el Replicador de datos REKO es altamente versátil y permite adaptarse prácticamente a cualquier necesidad que el usuario pueda tener. A medida que la configuración se hace más específica, va incorporando a su cuerpo más filtros y la configuración crece en tamaño y complejidad.

El Replicador de datos REKO es alojado en un contenedor de servlet[1], desde donde se genera la interfaz visual en forma de página web dinámica que permite la gestión y administración del mismo. Al contar con una interfaz visual, determinar el estado de las

[1] Servlet es un programa en Java que se ejecuta dentro de un servidor web y sirve construir páginas web dinámicas. Se utilizan principalmente para ampliar las capacidades de un servidor y permitir recibir y responder dinámicamente a peticiones de clientes web. Todo servlet debe implementar la interfaz Servlet, la cual determina los métodos de su ciclo de vida.

configuraciones de réplica actuales se realiza de forma sencilla e intuitiva, por mera observación.

Esta gestión y administración del Replicador de datos REKO se realiza mediante peticiones HTTPS[2] POST[3]. Para visualizar el listado de las configuraciones de réplica presentes se genera una petición POST con todos los datos sobre las mismas. En un escenario real donde es empleado el Replicador de datos REKO como solución de réplica, por ejemplo el caso del Sistema de Gestión de Hospitales Alas-His, la cantidad de tablas a replicar ascienden al orden de los miles; aumentando consecuentemente la complejidad y el tamaño de la configuración de la réplica, y por ende el tamaño de las peticiones POST.

En estas situaciones, la petición POST generada es extremadamente grande para ser procesada correctamente por el contenedor de *servlet* que aloja al Replicador de datos REKO. Por tanto la visualización de las configuraciones de réplica actuales se torna lenta y dificultando el trabajo con las mismas. La situación se agrava más aún cuando se selecciona ver los detalles de las configuraciones, ya que se deben cargar los filtros asociados a cada tabla que la configuración afecte, esto impacta negativamente porque no se puede interactuar adecuadamente las configuraciones de réplica bajo estas circunstancias, afectando el desempeño de los administradores de réplica en el uso del Replicador de datos REKO.

Partiendo de lo planteado anteriormente, surge entonces el siguiente **problema a resolver:** ¿Cómo garantizar un adecuado proceso de visualización de configuraciones de réplica del Replicador de datos REKO sobre cualquier escenario de replicación?

Basado en el problema a resolver se define como **objeto de estudio:** Proceso de configuración de réplica en sistemas de base de datos distribuidas. El **campo de acción** es: Proceso de visualización de configuraciones de réplica del Replicador de datos REKO.

[2] Protocolo Seguro de Transferencia de Hipertexto, del inglés *Hypertext Transfer Protocol Secure.* No es más que la versión segura del Protocolo de Transferencia de Hipertexto (del inglés *Hypertext Transfer Protocol,* HTTP).
[3] Es uno de los métodos que define HTTP, es comúnmente usado para acceder a recursos dinámicos.

Para dar solución al problema planteado se define como **objetivo general:** Desarrollar una solución informática que permita garantizar un adecuado proceso de visualización de configuraciones de réplica del Replicador de datos REKO sobre cualquier escenario de replicación.

Teniendo en cuenta el objetivo general, se plantea como **idea a defender:** La implementación de una solución informática, garantizará un adecuado proceso de visualización de configuraciones de réplica del Replicador de datos REKO sobre cualquier escenario de replicación.

Tareas de investigación:

- Confección del marco teórico-metodológico de la investigación.
- Análisis del estado del arte acerca de diferentes replicadores existentes y la forma en que aquellos realizan las configuraciones de réplica.
- Definición de los requisitos funcionales y no funcionales de la aplicación.
- Definición de la arquitectura que soporte la implementación de las funcionalidades.
- Selección de herramientas para la solución del problema.
- Generación de los artefactos para la disciplina de análisis y diseño.
- Implementación de las funcionalidades que den cumplimiento a los requisitos identificados.
- Realización de pruebas a las funcionalidades para detectar posibles errores.

A continuación se muestran una breve descripción de los métodos de investigación y empíricos utilizados para el desarrollo de la aplicación.

Métodos de Investigación:

- **Histórico - Lógico:** Permite conocer el desarrollo evolutivo de los sistemas de generación de configuración de réplica, de las distintas herramientas y tecnologías utilizadas para configurar la réplica de datos.
- **Analítico - Sintético:** Permite estudiar por separado las distintas herramientas que se tienen que emplear y luego integrarlas todas en la solución final.

Métodos empíricos:

- **Observación científica** se empleó con el objetivo de observar el funcionamiento de algunos replicadores de bases de datos existentes, para obtener un registro de las características comunes en estos y que pueden formar parte de la solución.

Para el desarrollo del presente trabajo se definieron tres capítulos, organizados de la siguiente forma:

Capítulo 1. Fundamentación teórica:

Son descritos los principales conceptos comprendidos en la presente investigación. Incluye la confección del marco teórico metodológico, análisis y estudio de sistemas homólogos existentes. Se realiza además la selección de la metodología y las herramientas a utilizar en el desarrollo de la solución.

Capítulo 2. Propuesta de solución:

Incluye la descripción, diseño y análisis de la solución que se propone para darle respuesta a la problemática planteada. La descripción de los requisitos funcionales y no funcionales. La definición de las historias de usuarios. La arquitectura propuesta para la solución al problema y diagramas de clases que describan la misma.

Capítulo 3. Implementación y pruebas:

Descripción del flujo de implementación. Se exponen los patrones de diseños aplicados en la construcción de la solución, así como el estándar de código empleado. Se exponen las pruebas que le serán realizadas al sistema para demostrar la robustez del mismo.

Capítulo 1. Fundamentación teórica

En el presente capítulo se abordarán los aspectos y conceptos esenciales relacionados con la generación de la configuración de réplica, los cuales servirán de apoyo para lograr un mayor entendimiento del problema a resolver. Se realizará un estudio del estado del arte de los replicadores de datos existentes, profundizando en los mecanismos de configuración de réplica que emplean los mismos. Serán caracterizadas las técnicas, métodos y herramientas que se utilizan en el ambiente de desarrollo del Replicador de datos REKO y que harán posible la implementación del módulo.

1.1 Marco Conceptual

En este epígrafe se definen una serie de conceptos relacionados con la presente investigación, con el objetivo de lograr una mejor comprensión del trabajo.

1.1.1 Base de Datos

El autor de la tesis asume el siguiente concepto pronunciado por Christopher J. Date[4] por considerarlo válido y ser consecuente con el objetivo del trabajo.

Un sistema de Base de Datos (BD) es básicamente un sistema computarizado para guardar registros; es decir, es un sistema computarizado cuya finalidad general es almacenar información y permitir a los usuarios recuperar y actualizar esa información con base en peticiones (2).

El autor concuerda con que cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más columnas y filas. Donde las columnas guardan una parte de la información sobre cada elemento que queramos guardar en la tabla, y cada fila de la tabla conforma un registro o una tupla.

[4] Autor, conferencista, investigador y consultor independiente, especializado en la tecnología de bases de datos relacionales.

1.1.2 Base de Datos Distribuida

Atendiendo a las definiciones enunciadas por C.J. Date y Abraham Silberschatz[5]:
Una Base de Datos Distribuida (BDD) es un conjunto de bases de datos parcialmente independientes que (idealmente) comparten un esquema común y coordinan el procesamiento de transacciones que acceden a datos remotos. Los procesadores se comunican entre sí a través de una red de comunicación que gestiona el encaminamiento y las estrategias de conexión (3).

El soporte completo para las bases de datos distribuidas implica que una sola aplicación debe ser capaz de operar de manera transparente sobre los datos que están dispersos en una variedad de bases de datos diferentes, administradas por una variedad de distintos Sistemas Gestores de Bases de Datos (SGBD del inglés *Database Management System* (DBMS)), ejecutadas en diversas máquinas diferentes, manejadas por varios sistemas operativos diferentes y conectadas a una variedad de redes de comunicación distintas; donde el término “de manera transparente significa” que la aplicación opera desde un punto de vista lógico como si todos los datos fueran manejados por un solo DBMS y ejecutados en una sola máquina (2).

El autor, concuerda con las definiciones anteriores en que las principales características de las bases de datos distribuidas son:

- Un conjunto de múltiples bases de datos lógicamente relacionadas las cuales se encuentran típicamente distribuidas en diferentes localizaciones geográficas comunicadas entre sí por una red de comunicación.
- Operan de manera transparente para el usuario como si todos los datos se encontraran localizados en un solo lugar.
- Soportan arquitecturas heterogéneas, por lo que un sistema de BDD puede incluir en su diseño tanto hardware como software diferente en cada nodo.

[5] Dr. En Ciencias de la Computación, autor e investigador, actualmente profesor en la Universidad de Yale, miembro de la ACM y de la IEEE, ha obtenido disímiles premios académicos por su trayectoria lo que le ha otorgado reconocimiento internacional.

1.1.3 Réplica de Datos

A continuación se citan algunos conceptos emitidos por varios autores:

La replicación es un conjunto de tecnologías destinadas a la copia y distribución de datos y objetos de base de datos desde una base de datos a otra, para luego sincronizar ambas bases de datos y mantener su coherencia. La replicación permite distribuir datos entre diferentes ubicaciones y entre usuarios remotos o móviles mediante redes locales y de área extensa, conexiones de acceso telefónico, conexiones inalámbricas e Internet (4).

La replicación es un mecanismo utilizado para propagar y diseminar datos en un ambiente distribuido, con el objetivo de tener mejor rendimiento y confiabilidad, mediante la reducción de dependencia de un sistema de base de datos centralizado (5).

Replicación de Base de Datos, es el término que usamos para describir la tecnología para mantener una copia de un conjunto de datos en un sistema remoto (6).

En el presente estudio se utilizará el concepto definido por la Lic. Vivian Romero Buchilla (5) por considerarlo actualizado, correctamente formulado y acorde a la temática de la investigación.

1.2 Estudio de Homólogos

En la actualidad existen diversos sistemas replicadores de bases de datos. En la presente investigación se consultó un grupo de estos replicadores en búsqueda de una solución para la visualización de configuraciones de réplica para el Replicador de datos Reko.

1.2.1 Slony-I

Slony es un sistema replicador “maestro-múltiples esclavos” para PostgreSQL que soporta la replicación en cascada y es tolerante a fallos. Slony es un sistema diseñado para usarlo en centros de datos y sitios de respaldo, donde el modo de operación normal es que todos los nodos se encuentren disponibles (7).

Slony-I no cuenta con una interfaz visual y funciona con comandos definidos en un lenguaje propio, para el que el equipo de desarrollo de Slony-I libera manuales de usuario en cada

nueva versión del software. Por tanto visualizar el estado de las configuraciones de réplica resulta engorroso, más aún cuando se trata de la administración de numerosos nodos.
Entre las principales características del software se encuentran (8):

- Permite la replicación entre diferentes versiones de PostgreSQL.
- Basado en *triggers* (gatillos, disparadores de eventos en las bases de datos).
- Permite la replicación parcial de bases de datos.
- Permite replicar datos entre diferente hardware y sistemas operativos.
- Permite diferentes servidores de bases de datos para diferentes tablas, en el origen (en el master).

Entre sus principales limitaciones se encuentran (9):

- Sólo permite un nodo maestro, siendo todos los otros esclavos.
- Sólo soporta servidores de base de datos PostgreSQL.
- No propaga automáticamente los cambios de esquemas.
- No permite replicar objetos de gran tamaño.

1.2.2 SymmetricDS

Es un software que permite la sincronización asincrónica de datos y soporta sincronización bidireccional. Utiliza tecnologías web y de base de datos para replicar la información almacenada en las tablas entre bases de datos relacionales. SymmetricDS fue diseñado para el trabajo a través de conexiones de bajo ancho de banda y soportar períodos de interrupción de la conexión entre los nodos (10).
SymmetricDS está escrito en Java, es un software de código abierto bajo licencia LGPL[6]. Cuenta con dos ediciones principales, la profesional y la comunitaria. Ambas ediciones cuentan con el mismo motor de replicación y la consola de comandos. La gran diferencia entre las ediciones es que la profesional tiene licencia comercial privativa y cuenta con

[6] Licencia Pública General Reducida de GNU (por sus siglas en inglés “GNU no es Unix”) del inglés *GNU Lesser General Public License*, pretende garantizar la libertad de compartir y modificar el software cubierto por ella, asegurando que el software es libre para todos sus usuarios. Así como permitir usar el software amparado bajo esta licencia para ser usado por otros productos de licencia privativa, sin tener que publicar la totalidad del código fuente.

muchas más utilidades que la edición comunitaria, esta última a su vez tiene una licencia GNU GPL[7] gratuita.

SymmetricDS en su versión profesional ofrece una interfaz de usuario en forma de página web que simplifica la visualización, configuración, monitorización y la detección de errores. Dicha interfaz visual es muy útil pero sólo está incorporada en la versión profesional. En la edición comunitaria para visualizar, configurar, monitorizar o detectar errores en las configuraciones de réplica, se utiliza la consola de comandos escritos en un lenguaje propio del SymmetricDS.

SymmetricDS garantiza la captura de los cambios de los datos en las tablas de una base de datos mediante el uso de *triggers*. El soporte para varias bases de datos esta dado mediante el uso de dialectos, existen implementaciones para: PostgreSQL, MySQL Server, Oracle, MS SQL Server, DB2, Firebird, HSQLDB, H2, y Apache Derby (10).

1.2.3 DBMoto

DBMoto es un software replicador de bases de datos a tiempo real, privativo y que soporta los principales gestores de bases de datos. Es una aplicación basada en la nube para su funcionamiento (11).
Cuenta con una interfaz de usuario para visualización y gestión de las configuraciones de réplica del sistema. Gracias a la interfaz de usuario el estado de las configuraciones de réplica puede ser fácilmente apreciado por el cliente. Esta forma de visualizar la información es mucho más intuitiva para el usuario que una consola de comandos.
A continuación se exponen algunas de las características más relevantes de DBMoto:

- Soporta la réplica de datos entre bases de datos heterogéneas.
- Permite monitorear a tiempo real la réplica de datos.
- Emplea registros log para grabar la actividad de réplica, el desempeño y cualquier error posible para ser analizado.

[7] Licencia pública general de GNU, garantiza a los usuarios finales la libertad de usar, estudiar, compartir (copiar) y modificar el software amparado bajo esta licencia. Si un programa utiliza fragmentos de código GPL (y el programa es distribuido) el código fuente en su totalidad debe estar disponible, bajo la misma licencia.

- Presenta funciones de comparación de resultados para verificar la actualización de datos posterior a una replicación.
- Usa tecnología en la nube para replicar datos de diferentes orígenes.
- Soporta datos de cualquier tamaño.
- Soporta trasformaciones de los datos.
- Soporta integración de datos con soluciones de almacenes de datos (12).

1.2.4 Daffodil Replicator[8]

Daffodil Replicator está escrito en Java y amparado bajo la licencia de código abierto de GNU GPL. Asegura alta disponibilidad. El replicador de datos Daffodil es un sistema potente que permite la integración de datos, migración de los mismos y protección a tiempo real. Permite la réplica bidireccional y la sincronización entre bases de datos homogéneas y heterogéneas. Permite replicar datos a través de Internet usando servicios web, similares a RMI. Además soporta la detección y la resolución de conflictos (13).

No fue posible encontrar referencias sobre como la herramienta Daffodil Replicator realiza el proceso de visualización de configuraciones de réplica.

Entre los gestores de bases de datos soportados se encuentran (13):
Microsoft SQL Server, Oracle, Daffodil DB, DB2, Derby, MySQL, PostgreSQL y Firebird.

1.2.5 Oracle streams[9]

Es un replicador de bases de datos que basa su funcionamiento en mensajes. Los cambios producidos en la base de datos por los Lenguajes de Definición de Datos (DDL[10]) y por los Lenguajes de Manipulación de Datos (DML[11]) son capturados y empaquetados en Registros

[8] En inglés, replicador.
[9] La traducción literal sería corriente, torrente, chorro, fluir. En este caso hace referencia al flujo de mensajes que utiliza para realizar la réplica, característica principal del replicador.
[10] Del inglés *Data Definition Language*. Lenguaje utilizado por sistemas gestores de bases de datos (como el Oracle) que permite a los usuarios definir la base de datos y especificar los tipos de datos, estructuras y las restricciones en los datos.
[11] Del inglés *Data Manipulation Language*. Comandos que pueden ser utilizados para manipular datos de las bases de datos existentes.

de Cambios Lógicos (LCR[12]). Estos registros de cambios son los que son enviados posteriormente utilizando mensajes hacia sus respectivos destinos, atendiendo al sistema de reglas definido por el usuario, que es el que configura la réplica en sí (14).

Las configuraciones de réplica se determinan y almacenan en una serie de reglas y no cuenta con una interfaz visual intuitiva que permita determinar el estado de las configuraciones de réplica actuales. En vez de eso hay que interpretar las reglas establecidas para saber cómo se encuentra configurado el replicador.

Oracle *streams* puede ser usado para la replicación de datos entre bases de datos Oracle (homogéneas) y bases de datos no Oracle (heterogéneas). Además permite su uso para guardar datos en almacenes de datos. Soporta la notificación de eventos y está habilitado para garantizar la protección de los datos. Dado a su funcionamiento basado en mensajes, Oracle *streams* está perfectamente habilitado para construir, operar y administrar almacenes de datos distribuidos. Soporta encriptación de los datos para garantizar su integridad (14).

1.2.6 Replicador de datos REKO

Es un software de réplica de datos entre Bases de Datos. Pretende cubrir las principales necesidades relacionadas con la distribución de datos entre los gestores más populares como la protección, recuperación, sincronización de datos, transferencia de datos entre diversas localizaciones o la centralización de la información en una única localización (15). El Replicador de datos REKO constituye un software maduro y una solución robusta ante un alto conjunto de escenarios de replicación para los cuales ha sido probado (15).

El Replicador de datos REKO ha sido implementado en la plataforma Java Enterprise Edition(JEE) que provee una arquitectura robusta para el desarrollo de aplicaciones empresariales en el lenguaje Java utilizando un modelo multi-capas e incluye una serie de tecnologías, herramientas de desarrollo y especificaciones e implementaciones de referencia de los servicios que brinda la plataforma (15).

[12] Del inglés *Logical Change Records.*

Principales Funcionalidades:

- Soporte para réplica de datos en ambientes con conexión y sin conexión.
- Soporte para réplica de ficheros externos a la Base de Datos.
- Soporte para la Sincronización de datos en ambientes con conexión y sin conexión.
- Selección de los datos de réplica ajustada por filtros.
- Soporte para réplica de datos entre gestores diferentes (PostgreSQL – Oracle - MySQL – MS SQL Server).
- Soporte para resolución de conflictos automática y manual.
- Monitoreo en tiempo real de los datos de réplica.
- Réplica de datos de gran tamaño con capacidad de resumir el envío y el recibo de los mismos en caso de desconexión.
- Soporte para programación del momento de captura y envío de los datos de réplica.
- Soporte réplica sin conexión automática.
- Soporte para la réplica de estructura (PostgreSQL).

Beneficios:

- Mejorar la distribución de los datos.
- Monitorear el flujo de cambios en la información que se almacena en el sistema distribuido.
- Garantizar la seguridad de los datos que se replican.

1.2.7 Resultados del estudio de homólogos

A continuación se muestran tres tablas que sintetizan el estudio realizado a los sistemas replicadores de datos actuales.

TABLA 1 RESULTADO DE ESTUDIO DE HOMÓLOGOS

Características/ Herramienta	Slony-I	SymmetricDS	DB Moto	Daffodil	Oracle streams	REKO
Medio por el que se visualizan las configuraciones de réplica						
Comandos	X	X(comunitaria)		-	X	
Interfaz gráfica		X(privativa)	X	-		X
Licencia						
LGPL / Licencia Postgres	X	X		X		X
Privativa			X		X	
Dirección de la replicación						
Maestro – esclavo(s)	X					
Multi - maestro		X	X	X	X	X

FUENTE: ELABORACIÓN PROPIA

TABLA 2 GESTORES SOPORTADOS

Características/ Herramienta	Slony-I	SymmetricDS	DB Moto	Daffodil	Oracle streams	REKO
Gestores Soportados						
PostgreSQL	X	X	X	X		X
Oracle		X	X	X	X	X
MySQL		X	X	X		X
MS SQL Server			X	X	X	X
IBM DB2		X	X	X		
Otros Gestores		X	X	X	X	

FUENTE: ELABORACIÓN PROPIA

TABLA 3 OTRAS CARACTERÍSTICAS

Características/ Herramienta	Slony-I	SymmetricDS	DB Moto	Daffodil	Oracle streams	REKO
Otras características						
Monitoreo cercano a tiempo real	**X**	**X**				
Monitoreo a tiempo real			**X**	**X**		**X**
Replica ficheros externos a la BD		**X**				**X**
Tolerante a fallos	**X**	**X**	**X**	**X**		**X**
Replica en ambientes de conexión y sin conexión						**X**
Resolución de conflictos		**X**	**X**	**X**	**X**	**X**

FUENTE ELABORACIÓN PROPIA

Concluido el estudio de los sistemas replicadores de bases de datos existentes, se procedió a realizar una comparación entre los mismos como se ilustra en las tablas anteriores.

Las soluciones anteriormente estudiadas no satisfacen las necesidades de la investigación, casi todas las herramientas estudiadas no cuentan con una interfaz de usuario capaz de visualizar de forma sencilla el estado de las configuraciones de réplica actuales. En caso de contar con una interfaz similar se trata de una herramienta privativa o la edición privativa de una de ellas. En vez de eso, utilizan comandos propios escritos en consola y almacenados en ficheros que dificultan el trabajo de la visualización de las configuraciones de réplica.

Por los elementos analizados anteriormente el autor de la presente investigación decide desarrollar una aplicación propia que apoye el trabajo del Replicador de datos REKO, permitiendo reutilizar componentes ya desarrollados del software REKO. Además una solución de este tipo aportaría independencia tecnológica al país, liberándolo del pago por software y licencias privativas a terceros.

1.3 Metodología de desarrollo: *AUP [13] variación UCI*

Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas (…). Estas prácticas se centran en el desarrollo de productos y servicios de calidad (16).

Descripción de las fases:

De las 4 fases que propone AUP (*Inicio, Elaboración, Construcción, Transición*) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de *Inicio*, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos *Ejecución* y se agrega la fase de *Cierre* (16).

Descripción de las disciplinas:

AUP propone 7 disciplinas (*Modelo*, *Implementación*, *Prueba*, *Despliegue*, *Gestión de configuración*, *Gestión de proyecto* y *Entorno*), se decide para el ciclo de vida de los proyectos de la UCI tener 8 disciplinas, pero a un nivel más atómico que el definido en AUP (16).

Las disciplinas que propone AUP variación UCI son: *Modelado de negocio* (la cuál es opcional), *Requisitos*, *Análisis y diseño*, *Implementación*, *Pruebas internas*, *Pruebas de liberación*, *Pruebas de aceptación* y *Despliegue* (también opcional).

Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación y la disciplina Despliegue se considera opcional.

[13] Proceso Unificado Ágil por sus siglas en inglés.

Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto) (16).

Todas las disciplinas antes definidas (desde Modelado de negocio hasta Despliegue) se desarrollan en la Fase de Ejecución, de ahí que en la misma se realicen iteraciones y se obtengan resultados incrementales (16).

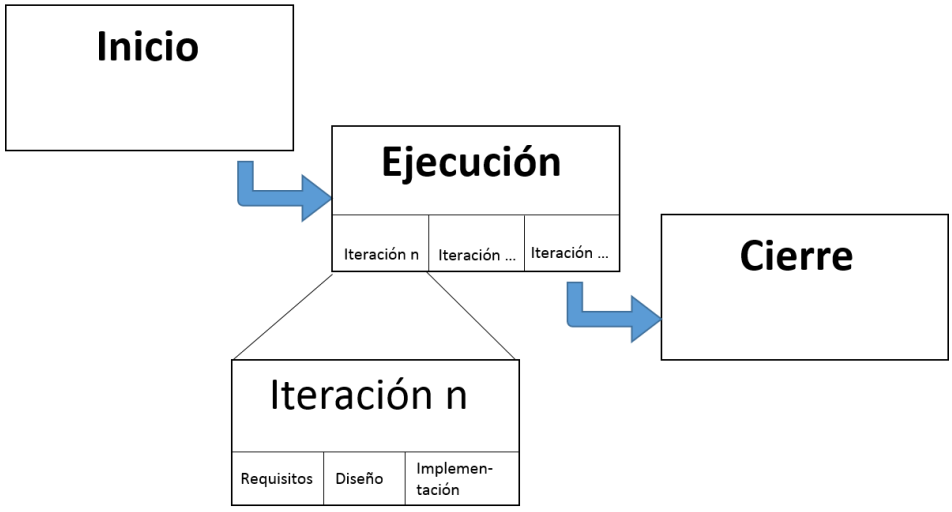


FIGURA 1 FASES E ITERACIONES DE AUP VARIACIÓN UCI **(16)**

1.4 Herramientas y lenguajes para el desarrollo

A continuación se especifican las herramientas que por sus características fueron seleccionadas para su uso en la elaboración de la solución.

1.4.1 Lenguaje de modelado: UML

El Lenguaje de Modelado Unificado (UML por sus siglas en inglés), es el lenguaje de modelado estándar para el desarrollo de software y sistemas. Un modelo, es una abstracción de la realidad. Cuando se modela un sistema, se abstraen los detalles irrelevantes que pueden potencialmente crear confusión. Un modelo es una simplificación del sistema real, por lo que permite el diseño y da paso a que un sistema sea entendido,

evaluado y criticado de forma mucho más rápida que si se tuviera que evaluar el sistema en sí. Entre las principales ventajas del UML se encuentran: Es un lenguaje formal, conciso, comprensivo, escalable, construido sobre la experiencia acumulada y además es estándar por excelencia (17).

1.4.2 Herramienta de modelado: Visual Paradigm

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso y generar documentación. La herramienta también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML (18).

1.4.3 Lenguaje de programación: Java

Java es un lenguaje de programación de propósito general orientado a objetos desarrollado por Sun Microsystems. También se puede decir que Java es una tecnología que no sólo se reduce al lenguaje, sino que además provee de una máquina virtual, que permite ejecutar código compilado del lenguaje, sea cual sea la plataforma que exista como base; tanto hardware, como software (el sistema operativo que soporte ese hardware) (19).
Java es un lenguaje simple. Orientado al objeto, distribuido, interpretado, sólido, seguro, de arquitectura neutral, portable, de alto desempeño, multi-hilos y dinámico (20) .

El diseño de Java aportó una respuesta eficaz a esas necesidades (21):

- Lenguaje de sintaxis sencilla, orientada a objetos e interpretada, que permite optimizar el tiempo y el ciclo de desarrollo (compilación y ejecución).
- Las aplicaciones son portables sin modificación en numerosas plataformas físicas y sistemas operativos.
- Las aplicaciones son resistentes porque el motor de ejecución de Java se encarga de la gestión de memoria (Java *Runtime Enviroment*), y es más fácil escribir programas sin error en comparación a C++, debido a un mecanismo de gestión de errores más evolucionado y estricto.

- Las aplicaciones y en particular las aplicaciones gráficas son eficaces debido a la puesta en marcha de varios procesos ligeros (*Thread*[14] y *multithreading*[15]).

1.4.4 Entorno de desarrollo integrado[16]*: Spring Tool Suite (STS)*

Versión: 3.6.1
Plataforma: Eclipse Luna 4.4
El STS es un ambiente de desarrollo basado en Eclipse que está diseñado para el desarrollo de aplicaciones Spring. Provee un entorno listo para usar útil para implementar, depurar, correr, y desplegar aplicaciones Spring, incluyendo la integración para Pivotal tc Server, Pivotal Cloud Foundry, Git, Maven, AspectJ, y viene sobre las últimas versiones de Eclipse (22).

1.4.5 Framework: Spring 2.0

El Framework Spring es de plataforma Java y provee soporte comprensivo de infraestructura para el desarrollo de aplicaciones Java. Spring se encarga de la infraestructura para que el usuario se pueda enfocar en la aplicación. Es una solución ligera y potente para desarrollar aplicaciones. Spring es modular, permitiendo usar solamente aquellas partes que necesites, sin que tengas que cargar el resto. Soporta la administración de transacciones declarativas, soporta el acceso remoto a través de RMI[17] o servicios web, y varias opciones para preservar los datos. Ofrece un marco de trabajo Modelo-Vista-Controlador completamente equipado, además de habilitar e integrar transparentemente la programación orientada a aspectos al software (23).

1.4.6 Protocolo simple de acceso a datos[18]

Protocolo para el intercambio de mensajes, que usa el Lenguaje de Empaquetado Extensible (del inglés *eXtensible Markup Language*, XML) como lenguaje de codificación (…). SOAP permite la comunicación entre máquinas en un entorno distribuido y

[14] Del inglés, hilo, en informática se refiere a hilo de procesos.
[15] Del inglés, multi hilo, en informática se refiere a la ejecución de múltiples hilos de procesos.
[16] En inglés Integrated development environment (IDE).
[17] Llamada a procedimiento remoto, del inglés Remote Method Invocation.
[18] También conocido como SOAP (del inglés Simple Object Access Protocol).

descentralizado, cuya ventaja fundamental es su independencia de la plataforma y lenguaje de programación. (…) Su uso proporciona las ventajas de un estándar aceptado a nivel mundial, de forma que la interacción de servicios y aplicaciones externas se puede llevar a cabo fácilmente. Debido a que SOAP ha sido ampliamente aceptado, hay librerías disponibles para una gran variedad de lenguajes de programación (24).

SOAP es un paradigma de mensajería de una dirección, sin estado, que puede ser utilizado para formar patrones de interacción más complejos (ejemplo: petición respuesta, petición respuesta múltiple, etc.) al combinar varios intercambios unidireccionales. (…) Ofrece un marco de trabajo de mensajería que permite transmitir datos de forma extensible (25).

Algunas de las Ventajas de SOAP son (26):

- **No está asociado con ningún lenguaje**.
- **No se encuentra fuertemente asociado a ningún protocolo de transporte**: La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.
- **No está atado a ninguna infraestructura de objetos distribuidos.**
- **Aprovecha los estándares existentes en la industria**: Los principales contribuyentes a la especificación SOAP evitaron intencionadamente reinventar las cosas. Optaron por extender los estándares existentes para que coincidieran con sus necesidades. Por ejemplo, SOAP aprovecha XML para la codificación de los mensajes, en lugar de utilizar su propio sistema. Como ya se ha mencionado SOAP no define un medio de trasporte de los mensajes; los mensajes de SOAP se pueden asociar a los protocolos de transporte existentes como HTTP y SMTP.
- **Permite la interoperabilidad entre múltiples entornos**: SOAP se desarrolló sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dichos estándares pueden comunicarse mediante mensajes SOAP con aplicaciones que se ejecuten en otras plataformas.

1.4.7 Empaquetado de datos mediante: XML

XML es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones. Las tecnologías XML son un conjunto de módulos que ofrecen servicios útiles a las demandas más frecuentes por parte de los usuarios. XML sirve para estructurar, almacenar e intercambiar información (27).

Los objetivos de diseño para XML son (28):

1. XML debe ser utilizable directamente sobre internet.
2. XML debe soportar una amplia variedad de aplicaciones.
3. XML debe ser compatible con SGML.
4. Debe ser fácil escribir programas que procesen documentos XML.
5. El número de características opcionales en XML debe ser mantenido en un mínimo, idealmente cero.
6. Los documentos XML deben ser legibles por un humano y razonablemente claros.
7. El diseño de XML debe ser preparado rápidamente
8. El diseño de XML debe ser formal y conciso.
9. Los documentos XML deben ser fáciles de crear.
10. La brevedad en la marcación es de mínima importancia
11. Esta especificación, junto con los estándares asociados, provee toda la información necesaria para entender XML Versión 1.0 y construir programas informáticos que lo procesen.

Esta versión de la especificación de XML puede ser distribuida libremente, mientras todo el texto y las anotaciones legales permanezcan intactas.

1.4.8 Java Architecture for XML Binding

JAXB[19] de Oracle es una herramienta de enlace de datos (*data binding*) que proporciona una API y un variado grupo de herramientas para automatizar el mapeo entre documentos XML y objetos Java, facilitando la tarea de los desarrolladores de incorporar datos XML y funciones para su procesamiento en aplicaciones Java. Partiendo de esquemas XML, JAXB genera clases Java que incorporan métodos para realizar el desempaquetado de documentos XML convirtiéndolos en un árbol de objetos Java, así como para realizar el empaquetado de dicho tipo de árboles de vuelta a un documento XML (29).

Los componentes principales de JAXB son (30):

- Personalización del enlace de datos entre XML y Java.
- Un compilador que realiza el enlace entre un esquema XML origen y un conjunto de clases Java.
- Una herramienta de ejecución de enlace que proporciona las interfaces para acceder a las operaciones de empaquetado, desempaquetado y validación que permiten manipular documentos XML y objetos Java.
- JAXB mapea la mayor parte de los tipos primitivos de XML Schema a tipos primitivos y nativos de Java.

1.4.9 Publicación y descripción del servicio mediante: WSDL[20]

WSDL fue desarrollado por Microsoft e IBM para describir servicios Web independientemente del método de transporte o método de codificación final (31).

WSDL es un lenguaje de descripción de servicios Web que permite definir la funcionalidad abstracta y la forma de acceder a un servicio, ya que los protocolos de comunicación y los formatos de los mensajes se están estandarizando en la comunidad web, se va haciendo cada vez más importante la descripción de las comunicaciones de forma estructurada. WDSL satisface esa necesidad al definir una gramática XML para describir servicios de red como colecciones de agentes de comunicación capaces de intercambiar mensajes (32) .

[19] Del inglés Java Architecture for XML Binding.
[20] Por sus siglas en inglés Web Service Description Language.

1.4.10 SoapUI 5.1.3

SoapUI es una herramienta de gran alcance diseñada para ayudar en la prueba y el desarrollo de aplicaciones. Permite efectuar pruebas de la web, con docenas de características incluyendo un interfaz de usuario, permite la utilización de métodos de captura y repetición, siendo una herramienta de gran ayuda en la realización de pruebas de carga de gran alcance, informes detallados, gráficos, etc (33).

SoapUI permite (33):

- Modificar capturas y poder volverlas a ejecutar en cualquier momento sin necesidad de volver ejecutarlas.
- Grabar las capturas como scripts y poderlas compartir dentro del entorno de trabajo.
- Capturar la estadística del funcionamiento mientras se ejecuta una prueba.
- Prueba de la regresión en áreas enteras de los sitios web complejos.

1.4.11 Herramienta de diseño de prototipos de interfaz: Balsamiq Mockups

Versión: 2.1.13

Es una herramienta muy útil para diseñar y preparar prototipos de aplicación de forma sencilla y rápida. Las funciones de importación y exportación garantizan una integración transparente con todas las versiones de maquetas. Además puedes cargar y guardar archivos de varias maquetas, utilizar los métodos abreviados de teclado y mucho más. Esta herramienta permite exportar el trabajo final a PNG o PDF. Cuenta con una variada gama de componentes e íconos para adaptarse a cualquier necesidad (34).

1.4.12 Herramienta generadora de diagramas de flujo: Visustin

Versión: 7

Visustin para desarrolladores de software. Visustin es un programa automatizado para la creación de diagramas de flujo para desarrolladores de software y escritores de documentos. Visustin usa la ingeniería inversa en su código fuente para crear diagramas de flujo o diagramas de actividad UML (35).

1.5 Sistemas gestores de bases de datos que soporta el Replicador de datos REKO

A continuación una breve reseña sobre los principales gestores de bases de datos a los cuales el Replicador de datos REKO respalda:

PostgresSQL es un sistema administrador de bases de datos objeto-relacional (ORDBMS[21]) basado en Postgres, desarrollado en la Universidad de California en el Departamento de Ciencias Informáticas de Berkeley. PostgresSQL es un descendiente de código abierto del original elaborado en la Universidad de Berkeley, soporta una gran parte del estándar SQL y ofrece muchas prestaciones modernas como: consultas complejas, llaves foráneas, *triggers*, vistas actualizables, integridad transaccional y control de concurrencia para diferentes versiones. Y además debido a su licencia libre, PostgresSQL puede ser usado, modificado y distribuido por cualquiera libre de cargos judiciales para cualquier propósito sea privado, comercial o académico (36).

Microsoft SQL Server es un sistema de manejo de bases de datos del modelo relacional, desarrollado por la empresa Microsoft. SQL Server solo está disponible para sistemas operativos Windows de Microsoft.

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle *Corporation*[22] desde abril de 2009 se desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Oracle Database es un sistema de gestión de base de datos objeto-relacional (u ORDBMS por el acrónimo en inglés de *Object-Relational Data Base Management System*), desarrollado por Oracle *Corporation*. Se considera a Oracle *Database*[23] como uno de los sistemas de bases de datos más completos, destacando: soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma.

[21] Por sus siglas en inglés *Object-Relational Database Management System*.
[22] En inglés corporación.
[23] En inglés base de datos.

1.6 Conclusiones parciales del capítulo

La elaboración del marco teórico brindó un acercamiento a los principales conceptos, técnicas y métodos a utilizar durante el resto de la investigación, lo que permitió profundizar los aspectos teóricos, el análisis de tendencias y soluciones existentes, propiciando una base de conocimientos útil para el desarrollo de la investigación. Además, la indagación sobre las metodologías, técnicas y herramientas consolidaron las propuestas tecnológicas necesarias para la realización del sistema, lográndose una fundamentación técnica que justifica como solución escogida.

Capítulo 2. Propuesta de solución

Sobre las bases de las ideas expuestas que se utilizarán en el desarrollo de la aplicación informática, se realiza el levantamiento de los requisitos. En este capítulo se define la propuesta del sistema, los requisitos funcionales y no funcionales, así como el diseño de la arquitectura del sistema propuesto. El proceso de desarrollo a seguir estará guiado por las pautas de la metodología AUP (variación UCI). Explica cada artefacto generado por la metodología empleada. Especifica cada una de las historias de usuarios del sistema, definidos según los requisitos identificados y se define la arquitectura a utilizar. Los elementos anteriores serán la base para la implementación la solución para la configuración de réplica del Replicador de datos REKO.

2.1 Propuesta de solución

Considerando la situación actual del proceso de visualización de configuraciones de réplica del Replicador de datos REKO, se propone la implementación de un sistema informático que evite los problemas asociados a peticiones POST excesivamente grandes durante dicho proceso. Teniendo en cuenta lo anterior, se sugiere una aplicación de escritorio para la visualización de configuraciones de réplica. En aras de gestionar las configuraciones actuales en la aplicación, se planea crear un canal de comunicación con el Replicador de datos REKO. A través de dicho canal de comunicación deberá transmitirse toda la información necesaria para la correcta gestión de las configuraciones de réplica. Para ello se propone la creación de dos servicios web independientes, uno encargado de exportar la configuración general de un nodo del Replicador de datos REKO y otro encargado de exportar las configuraciones de réplica almacenadas en el mismo. La aplicación deberá consumir dichos servicios y reflejar el estado actual del negocio en una interfaz visual con la que el usuario pueda interactuar. Las modificaciones que el usuario realice mediante la interfaz visual deben ser persistidas en el negocio correctamente.

2.2 Modelo de dominio

La finalidad del análisis orientado a objetos es crear una descripción del dominio desde la perspectiva de la clasificación de objetos. Una descomposición del dominio conlleva una identificación de los conceptos, atributos y asociaciones que se consideran significativas. El resultado se puede expresar en un **modelo del dominio,** que se ilustra mediante un conjunto de diagramas que muestran los objetos o conceptos del dominio (37).
En la Figura 2 se muestra el modelo de dominio propuesto para el sistema.

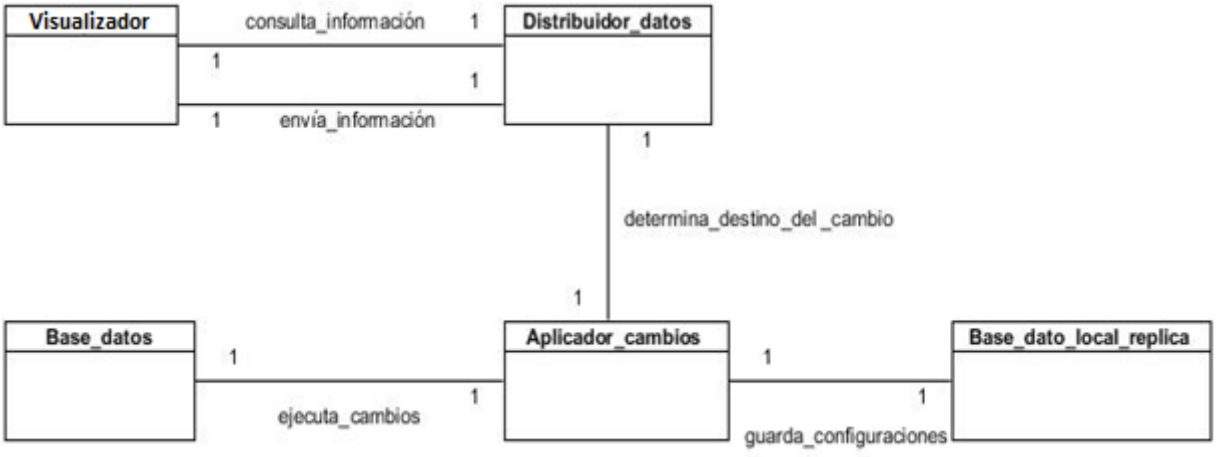


FIGURA 2 MODELO DE DOMINIO (ELABORACIÓN PROPIA)

Distribuidor_datos: concepto que determinará el destino de cada configuración de cambio.
Aplicador_cambios: concepto que ejecutará sobre la Base_datos los cambios que sean replicados hacia ella.
Base_datos_local_réplica: concepto que guardará las configuraciones propias de la réplica, así como acciones sobre la Base_datos.
Base_datos: concepto que representa la Base de Datos que se está replicando. Los cambios ejecutados sobre ella serán enviados, así como serán aplicados otros cambios provenientes de otros nodos de réplica.

Visualizador: elemento que consume los servicios publicados por el distribuidor. Encargado de visualizar a petición del usuario, las configuraciones de réplica.

2.3 Requisitos funcionales

Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema de sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema (38).

Los requisitos funcionales representan el comportamiento que tendrá el sistema, describen las funcionalidades que debe cumplir o que se espera que este provea. Deben ser lo más completo, claro y conciso posible, además se pueden definir a partir de reglas del negocio o de la propia interacción de los usuarios. Específicamente los requerimientos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer (38).

A continuación se muestran los requisitos funcionales propuestos para el sistema:

TABLA 4 REQUISITOS FUNCIONALES

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF1	Exportar configuraciones de réplica desde un nodo del Replicador de datos REKO	Permite exportar las configuraciones de réplica presentes en un nodo del Replicador de datos REKO.	Alta	Alta
RF2	Exportar configuración general desde un nodo del Replicador de datos REKO	Permite exportar la configuración general de un nodo del Replicador de datos REKO.	Alta	Alta
RF3	Listar configuraciones de réplica actuales	Permite listar las configuraciones de réplica actuales importadas previamente desde un	Media	Alta

		nodo del Replicador de datos REKO.		
RF4	Ver detalles de configuración de réplica	Permite ver los detalles de una configuración de réplica seleccionada por el usuario.	Baja	Alta
RF5	Importar configuraciones de réplica	Permite importar las configuraciones de réplica que exporta un nodo del Replicador de datos REKO.	Alta	Alta
RF6	Importar configuración general	Permite importar la configuración general que exporta un nodo del Replicador de datos REKO.	Alta	Alta
RF7	Eliminar configuración de réplica	Permite eliminar una configuración de réplica seleccionada por el usuario.	Alta	Alta
RF8	Actualizar	Permite actualizar el listado de configuraciones de réplica, acorde a modificaciones que puedan sufrir las mismas, desde un nodo del Replicador de datos REKO o desde la propia solución.	Alta	Alta

FUENTE ELABORACIÓN PROPIA

2.4 Requisitos no funcionales

Los requerimientos no funcionales, como su nombre sugiere, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema (38).

A continuación se muestra un resumen de los requisitos no funcionales del sistema. Para más detalles sobre los requisitos no funcionales y su uso en la implementación de la solución véase la sección de Anexos.

TABLA 5 REQUISITOS NO FUNCIONALES

Nº	Nombre	Descripción
RnF1	Fiabilidad	El sistema debe ser capaz de satisfacer las necesidades de fiabilidad en condiciones normales.
RnF2	Usabilidad	El sistema debe ser de fácil manejo para los usuarios que tengan niveles básicos de trabajo con ordenadores.
RnF3	Portabilidad	El sistema debe permitir ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.
RnF4	Adecuación funcional	El sistema debe cubrir todos los objetivos del usuario especificados.
RnF5	Restricciones de diseño	Debe estar instalada la máquina virtual de Java JDK 1.7.

FUENTE ELABORACIÓN PROPIA

2.5 Historias de usuarios

Las historias de usuario son la técnica para especificar los requisitos del software. Se realiza una por cada funcionalidad del sistema, se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos de iteraciones del producto. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. Las historias de usuarios deben ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a las tres semanas, debe ser

dividida en dos o más historias. A continuación se describen las historias de usuarios de mayor prioridad para la propuesta de solución.

Definiciones y acrónimos empleados en el siguiente epígrafe:

- HU Historia de usuario
- NA No aplica

TABLA 6 HU1

Historia de Usuario	
Número: 1	**Nombre:** Exportar configuraciones de réplica desde un nodo del Replicador de datos REKO
Usuario:	**Iteración asignada:** 1
Prioridad en el Negocio: Alta	**Puntos estimados:** 2
Riesgo en desarrollo: Alto	**Puntos reales:** 2

Descripción:

El Replicador de datos REKO, desde el momento de ser iniciado, debe establecer de forma automática un canal de comunicación que permita exportar las configuraciones de réplica presentes en el mismo. De cada configuración de réplica es necesario exportar los siguientes datos:

- Identificador de la réplica.
- Si existen transformaciones en la réplica de datos.
- Si realiza o no réplica por creación de estructura.
- Si realiza o no réplica por modificación de estructura.
- Si realiza o no réplica por eliminación de estructura.
- **Configuraciones de réplica de datos**, si existen.
- **Configuraciones de réplica de estructura**, si existen.
- **Filtros de usuarios**, si existen.

- Por cada configuración de réplica de datos es necesario exportar:
  - El nombre de la tabla a la que hace referencia.
  - El identificador del destino (hacia dónde va a replicar).
  - Si realiza o no réplica por creación de datos.
  - Si realiza o no réplica por modificación de datos.

  - SI realiza o no réplica por eliminación de datos.
  - Los **filtros de usuario** por cada acción de réplica, si existen.
  - Los **filtros SQL** por cada acción de réplica, si existen.
  - Los **filtros de archivos de referencia** por cada acción de réplica, si existen.
- Por cada filtro de usuario es necesario exportar:
  - Los usuarios que son afectados por este filtro.
  - Si tienen permitido o no replicar.
  - Si afecta a todos los usuarios o solamente a una parte de ellos.
- Por cada filtro SQL es necesario exportar:
  - La sentencia SQL que produce el filtro.
  - El atributo que se va tener en cuenta.
  - El determinador de la condición.
  - El operador de la condición.
  - El valor del primer elemento de la condición.
  - El valor del segundo elemento de la condición.
  - Si es una condición compuesta.
  - Si es una condición de tipo SQL.
  - Los **hijos** de dicha condición, si existen.
- Por cada filtro de archivo de referencia es necesario exportar:
  - Los **filtros de columnas de referencia**, si existen.
- Por cada filtro de columna de referencia es necesario exportar:
  - Un **filtro SQL**, si está definido.
  - La expresión regular que se aplica al valor de la columna para extraer la ruta del archivo.
  - El nombre de la columna.
  - La ruta para replicar en el anfitrión.
- Por cada configuración de réplica de estructura es necesario exportar:
  - El nombre del mismo.
  - El identificador del objetivo.
  - Si realiza réplica por creación de estructura.

- Si realiza réplica por modificación de estructura. - Si realiza réplica por eliminación de estructura. - Los **filtros de usuarios** asociados a cada acción replicable, si existen.
Observaciones: Las configuraciones de réplica de datos y las configuraciones de réplica de estructura, pueden tener definido un filtro diferente e independiente por cada acción de réplica a configurar según las tablas involucradas en el proceso de configuración. Si las configuraciones de réplica son modificadas por usuarios del el Replicador de datos REKO, el canal de comunicación que las publica debe reflejar dichos cambios.
Prototipo de Interfaz: NA

FUENTE ELABORACIÓN PROPIA

TABLA 7 HU2

Historia de Usuario	
Número: 2	**Nombre:** Importar configuraciones de réplica
Usuario:	**Iteración asignada:** 1
Prioridad en el Negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Alto	**Puntos reales:** 1
Descripción: La propuesta de solución, desde el momento de ser iniciada, debe importar automáticamente las configuraciones de réplica que exporta el nodo del Replicador de datos REKO. Para ello debe conectarse al canal de comunicación establecido por el nodo del Replicador de datos REKO e importar las configuraciones de réplica del mismo, debe ser capaz de importar todos los elementos definidos en la descripción de la HU1.	
Observaciones: Previamente debe encontrarse desplegado y en estado de ejecución un nodo del Replicador de datos REKO.	
Prototipo de Interfaz: NA	

FUENTE ELABORACIÓN PROPIA

TABLA 8 HU3

Historia de Usuario	
Número: 3	**Nombre:** Exportar configuración general desde un nodo del Replicador de datos REKO
Usuario:	**Iteración asignada:** 2
Prioridad en el Negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Alto	**Puntos reales:** 1
Descripción: El Replicador de datos REKO, desde el momento de ser iniciado, debe establecer de forma automática un canal de comunicación que permita exportar la configuración general presente en el mismo. De la configuración general es necesario exportar los siguientes datos: - El identificador del nodo del Replicador de datos REKO. - El nombre del nodo del Replicador de datos REKO. - La descripción del nodo del Replicador de datos REKO.	
Observaciones: Si la configuración general es modificada por usuarios en el Replicador de datos REKO, el canal de comunicación que exporta la misma debe reflejar estos cambios.	
Prototipo de Interfaz: NA	

FUENTE ELABORACIÓN PROPIA

TABLA 9 HU4

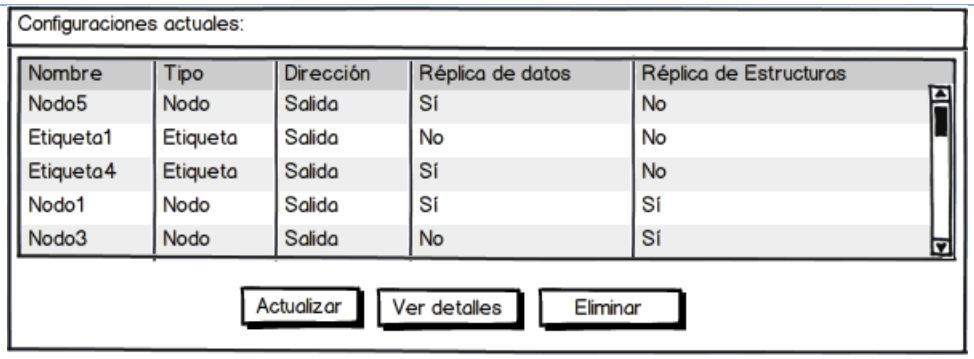
Historia de Usuario	
Número: 4	**Nombre:** Importar configuración general
Usuario:	**Iteración asignada:** 2
Prioridad en el Negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Alto	**Puntos reales:** 1
Descripción: La propuesta de solución, desde el momento de ser iniciada, debe importar automáticamente la configuración general que exporta el nodo del Replicador de datos REKO. Para ello debe conectarse al canal de comunicación establecido por el nodo del	

Replicador de datos REKO e importar la configuración general del mismo, del cual serán empleados los elementos definidos en la descripción de la HU3.
Observaciones: Debe encontrarse previamente desplegado y en estado de ejecución un nodo del Replicador de datos REKO.
Prototipo de Interfaz: NA

FUENTE ELABORACIÓN PROPIA

TABLA 10 HU5

Historia de Usuario	
Número: 5	**Nombre:** Listar configuraciones de réplica actuales
Usuario:	**Iteración asignada:** 2
Prioridad en el Negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Medio	**Puntos reales:** 1
Descripción: Permite visualizar un listado de las configuraciones de réplica, previamente exportadas por el Replicador de datos REKO e importadas por la propuesta de solución. El sistema debe mostrar en la sección “Configuraciones Actuales” el listado de las configuraciones actuales de réplica, mostrando de estas la siguientes características: - Nombre de la configuración. - Tipo de configuración. - Dirección de la réplica. - Si realiza réplica de datos. - Si realiza réplica de estructura. Debe permitir seleccionar una configuración de réplica, ya sea para ver más detalles de su configuración o para eliminarla; mediante los botones “Ver detalles” y “Eliminar” respectivamente. Además debe brindar la opción de actualizar el listado de configuraciones mediante el botón “Actualizar”.	
Observaciones: NA	
Prototipo de interfaz:	



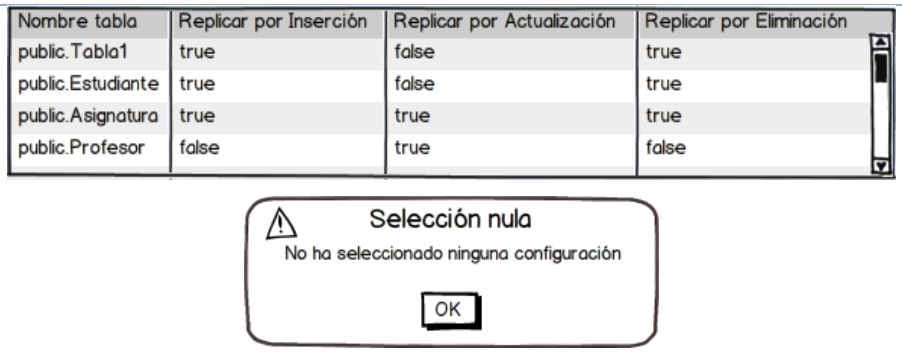
FUENTE ELABORACIÓN PROPIA

TABLA 11 HU6

Historia de Usuario	
Número: 6	**Nombre:** Ver detalles de configuración de réplica
Usuario:	**Iteración asignada:** 3
Prioridad en el Negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Bajo	**Puntos reales:** 1

Descripción:
Debe permitir seleccionar una configuración de réplica dándole clic a la misma y luego de oprimir el botón "Ver detalles", se debe mostrar específicamente de la configuración seleccionada los siguientes atributos:
- Los nombres de las tablas que afecta dicha configuración.
- Por cada tabla, si realiza o no réplica por:
  o inserción.
  o actualización.
  o eliminación.	

Observaciones: Si no ha sido seleccionada ninguna configuración y se presiona el botón "Ver detalles", se debe mostrar un mensaje de alerta con el siguiente mensaje: "No ha seleccionado ninguna configuración".	
Prototipo de interfaz:	

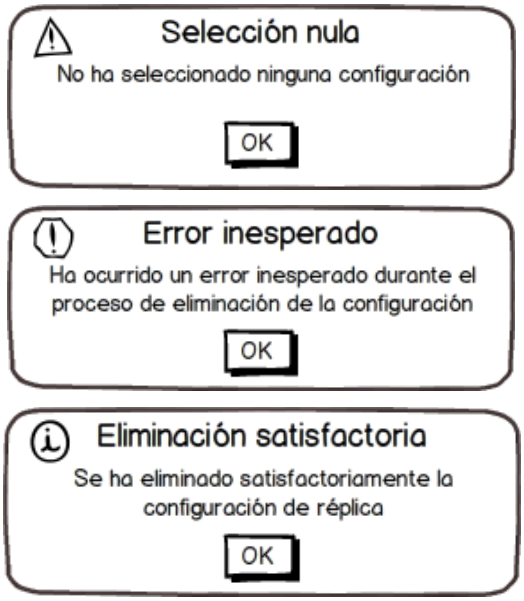


FUENTE ELABORACIÓN PROPIA

TABLA 12 HU7

Historia de Usuario	
Número: 7	**Nombre:** Eliminar configuración de réplica
Usuario:	**Iteración asignada:** 3
Prioridad en el Negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Alta	**Puntos reales:** 1
Descripción: El sistema debe permitir eliminar las configuraciones de réplica listadas en la propuesta de solución. Además debe permitir en la sección “Configuraciones Actuales” seleccionar una configuración a eliminar; una vez seleccionada una configuración y oprimido el botón “Eliminar” el sistema debe eliminar automáticamente la configuración asociada en la aplicación y enviar los cambios realizados a través de un canal de comunicación con el Replicador de datos REKO. Mientras esto sucede la aplicación debe ser capaz de actualizar la sección “Configuraciones Actuales” con las configuraciones actuales almacenadas en el nodo del Replicador de datos REKO.	
Observaciones: Si no ha sido seleccionada ninguna configuración, se debe mostrar un mensaje de alerta: “No ha seleccionado ninguna configuración”. Si ocurriera algún error durante el proceso de eliminación de la réplica se debe mostrar un mensaje de error: “Ha ocurrido un error durante el proceso de eliminación de la configuración”. Luego de haber realizado una eliminación satisfactoria, se deberá actualizar la lista de configuraciones actuales mostradas en la sección “Configuraciones actuales” y se debe mostrar un mensaje de información: “Se ha eliminado satisfactoriamente la configuración de réplica”.	

Prototipo de interfaz:



FUENTE ELABORACIÓN PROPIA

TABLA 13 HU8

Historia de Usuario	
Número: 8	**Nombre:** Actualizar
Usuario:	**Iteración asignada:** 3
Prioridad en el Negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Alta	**Puntos reales:** 1
Descripción: Debe permitir actualizar el listado de configuraciones de réplica mediante el botón "Actualizar", el mismo debe proporcionar el mecanismo que actualiza la sección "Configuraciones actuales" mostrando cualquier modificación que hayan sufrido las configuraciones de réplica. Para ello la propuesta de solución debe volver a importar los datos de las configuraciones de réplica y la configuración general del nodo del Replicador de datos REKO, finalmente lista las configuraciones de réplica actuales, con los datos actualizados.	
Observaciones: Como resultado deben encontrarse importadas y listadas las configuraciones de réplica actuales del nodo del Replicador de datos REKO. Igualmente debe encontrarse importada	

la configuración general del nodo del Replicador de datos REKO.
Prototipo de interfaz: NA

FUENTE ELABORACIÓN PROPIA

2.6 Arquitectura

El diseño arquitectónico representa la estructura de los datos y los componentes del programa que se requieren para construir un sistema basado en computadora, constituye el estilo arquitectónico que tendrá el sistema, la estructura y las propiedades de los componentes que ese sistema comprende, y las interrelaciones que tienen lugar entre todos los componentes arquitectónicos del sistema. Además, se describen las propiedades de los componentes y sus relaciones (interacciones) (39).

La arquitectura que se propone como solución es una arquitectura por capas. A continuación se muestra un diagrama con los principales elementos que la integran:

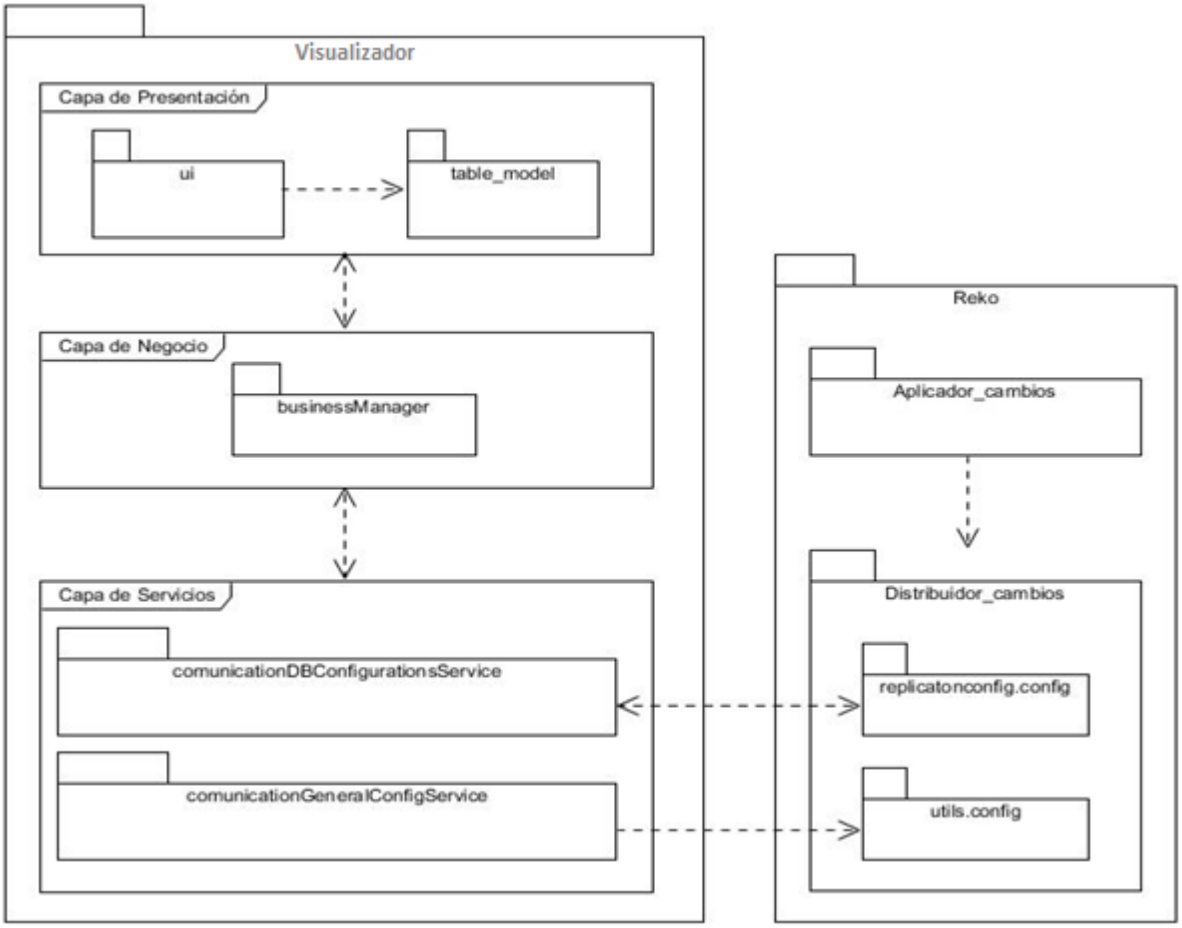


FIGURA 3 ARQUITECTURA PROPUESTA (ELABORACIÓN PROPIA).

Nota: El Replicador de datos REKO tiene una arquitectura basada en componentes, no una arquitectura por capas como la que se propone para la solución. El diagrama anterior es una abstracción donde solamente se representa la parte del Replicador de datos REKO que interactúa con la propuesta de solución.

La aplicación estará dividida en tres capas descritas a continuación:

Capa de Presentación: Esta capa es responsable de mostrar información al usuario e interpretar sus acciones. Los componentes de esta capa implementan la funcionalidad requerida para que los usuarios interactúen con la aplicación. Genera la interfaz visual con la información sobre las configuraciones actuales.

Capa de Negocio: Esta capa es responsable de representar conceptos de negocio, información sobre la situación de los procesos de negocio e implementación de las reglas

del dominio. También debe contener los estados que reflejan la situación de los procesos del negocio.

Capa de servicios: Es la capa encargada de consumir el servicio publicado por el Replicador de datos REKO. Es la que debe implementar código que gestione la semántica de comunicaciones con dicho servicio e incluso tareas adicionales como mapeos entre diferentes formatos de datos.

2.7 Diagrama de clases

Un diagrama de clases es una presentación gráfica de la vista estática, que muestra una colección de elementos declarativos (estáticos) del modelo, como clases, tipos, y sus contenidos y relaciones. Un diagrama de clases puede mostrar una vista de un paquete y contener símbolos de paquetes anidados. Un diagrama de clases contiene ciertos elementos materializados de comportamiento, como operaciones, pero cuya dinámica está representada en otros diagramas, como diagramas de estados o diagramas de colaboración (40).

A continuación se muestran los principales diagramas de clases presentes en la propuesta de solución:

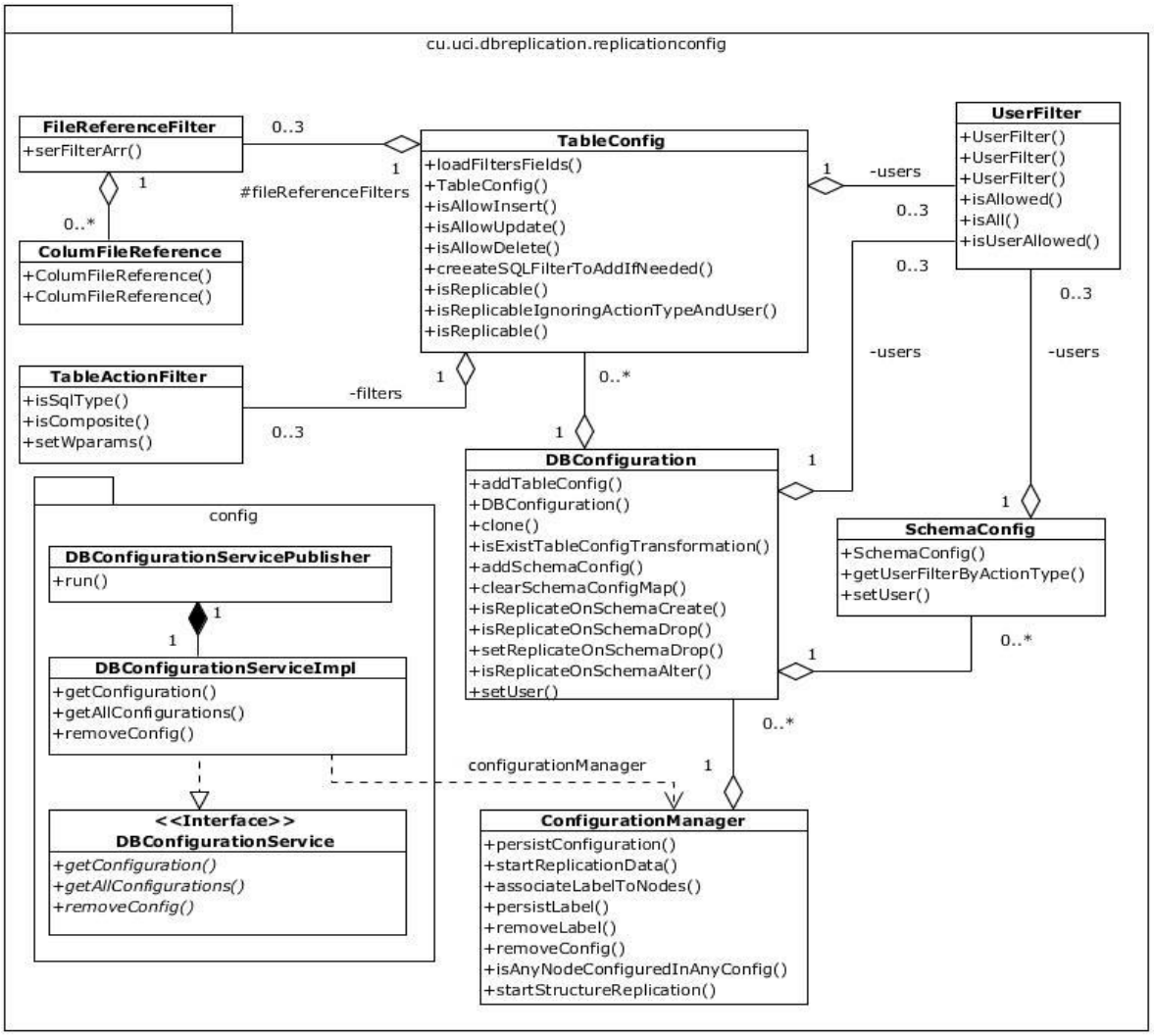


FIGURA 4 DIAGRAMA DE CLASES HU EXPORTAR CONFIGURACIÓN DE RÉPLICA (ELABORACIÓN PROPIA).

ConfigurationManager: clase responsable de administrar y gestionar las configuraciones de réplica de un nodo del Replicador de datos REKO.

DBConfiguration: es la clase que define la configuración de réplica. Contiene conjuntos de TableConfig y SchemaConfig, ambas clases encargadas de determinar la réplica de datos y de estructura respectivamente.

TableConfig: clase que contiene todos los datos y operaciones vinculados a la réplica de datos en una configuración de réplica.

SchemaConfig: clase que contiene todos los datos y operaciones vinculados a la réplica de estructuras en una configuración de réplica.

Paquete config: contiene las clases encargadas de exportar las configuraciones de réplica que serán consumidas por la solución.

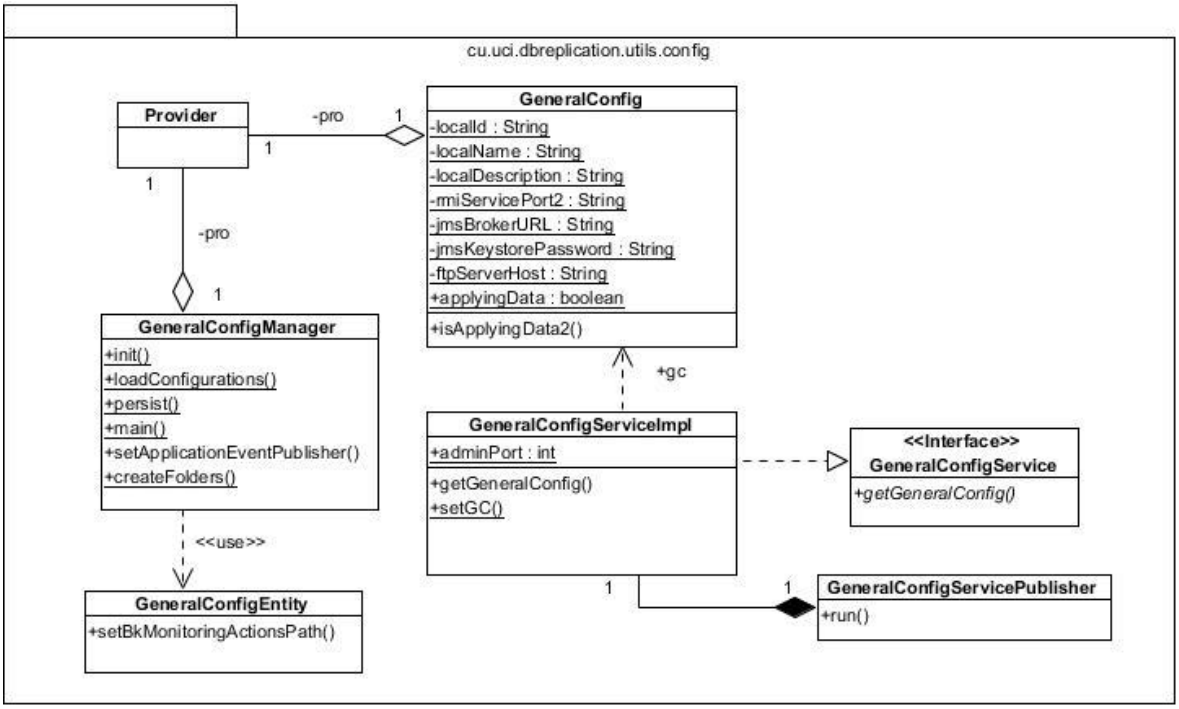


FIGURA 5 DIAGRAMA DE CLASES HU EXPORTAR CONFIGURACIÓN GENERAL (ELABORACIÓN PROPIA)

GeneralConfigManager: clase responsable de administrar y gestionar la configuración general de un nodo del Replicador de datos REKO.

GeneralConfig: clase que contiene todos los datos y operaciones vinculados a la configuración general de un nodo del Replicador de datos REKO.

GeneralConfigService: clase que propone una serie de métodos para interactuar con la configuración general de un nodo del Replicador de datos REKO y hacerla accesible a la propuesta de solución.

GeneralConfigServiceImpl: clase que implementa los servicios de la interfaz GeneralConfigService.

GeneralConfigServicePublisher: publica el servicio web que exporta la configuración general de un nodo del Replicador de datos REKO.

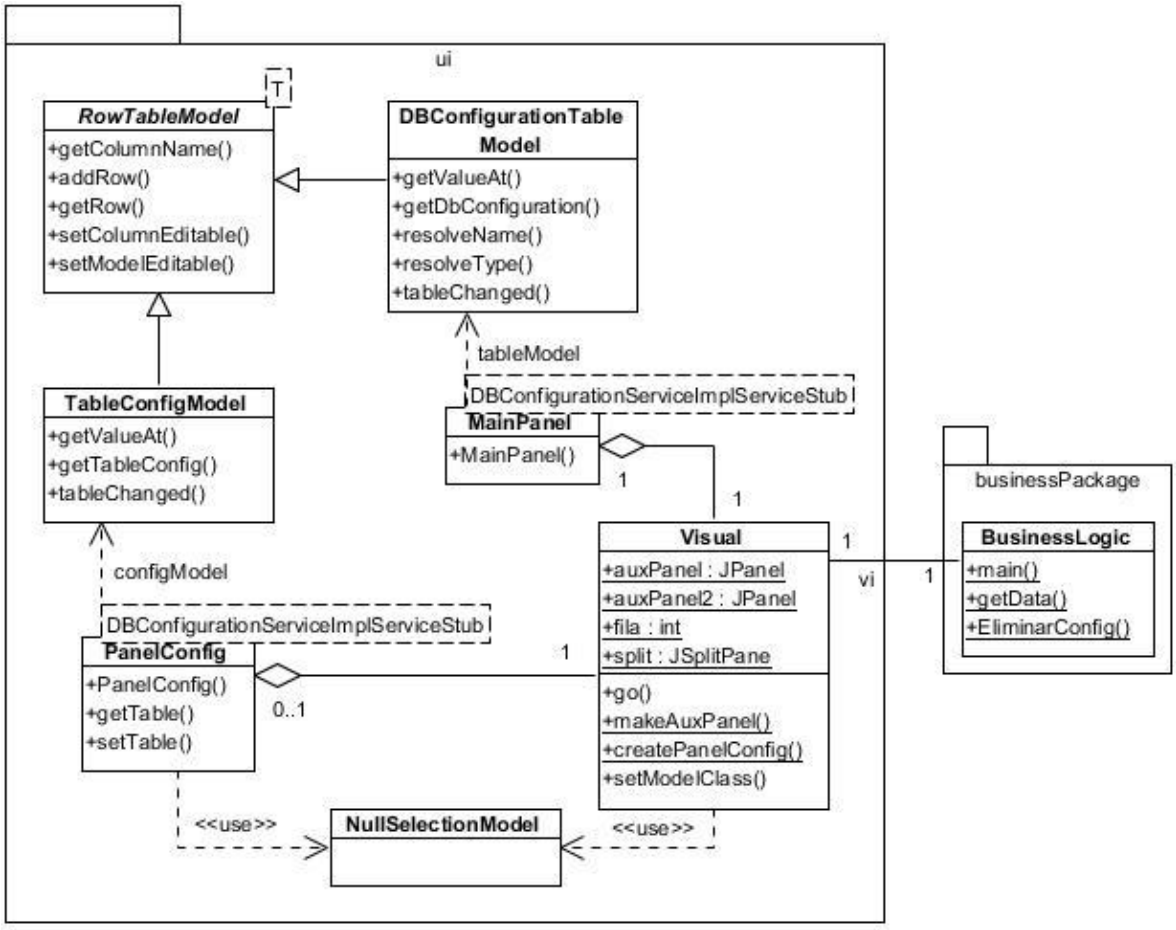


FIGURA 6 DIAGRAMA DE CLASES SISTEMA CLIENTE (ELABORACIÓN PROPIA)

BusinessLogic: clase que engloba la lógica del negocio, esta clase es la que se comunica con las librerías “ComunicationDBConfiguration” y “ComunicationGeneralConfig” que son las que actúan como la capa de servicios consumiendo los servicios “DBConfigurationWebSerice” y “GeneralConfigWebService” respectivamente. Es la encargada de solicitar consumir las operaciones publicadas en los servicios. Responsable de almacenar el estado del negocio. Encargada de hacer disponible a la interfaz de usuario los datos necesarios para su correcta creación y visualización.

Visual: clase principal dentro del paquete de interfaz de usuario. Es su responsabilidad crear una adecuada interfaz de usuario, que satisfaga todas sus necesidades funcionales, así como estar atento a las acciones que el usuario realice en la misma y actuar en consecuencia. Debe reflejar y hacer visible al usuario el estado del negocio.

2.8 Conclusiones parciales del capítulo

El estudio realizado en este capítulo ha facilitado un entendimiento de la dinámica y estructura de la aplicación, contribuyendo a una mejor comprensión del problema que el sistema a desarrollar debe resolver, se realizó una descripción de las características del mismo, se definieron los requerimientos funcionales y los no funcionales, sirviendo de base para el comienzo del desarrollo del sistema. Y a través del modelo del dominio quedaron plasmados los conceptos significativos. Se logró la modelación del sistema con la realización de los diagramas de clases y la descripción de ellas, permitiendo dar paso a la realización posterior de la implementación y a la realización de las pruebas.

Capítulo 3. Implementación y pruebas

Partiendo del resultado del análisis y diseño, en el presente capítulo se definen los patrones de diseño y el estándar de código utilizados. Se generan los artefactos referentes a la fase de implementación y prueba del software. Como principales elementos se definen los diagramas de despliegue y de componentes. Además muestra los casos de pruebas aplicados a la solución desarrollada para validar su correcto funcionamiento.

3.1 Patrones de diseño

El diseño de la solución fue elaborado siguiendo patrones que pretenden evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente, formalizar un vocabulario común entre diseñadores y estandarizar el modo en que se realiza el diseño.

Patrones Generales de Software para Asignación de Responsabilidades (del inglés General Responsibility Assignment Software Patterns, GRASP) son una serie de patrones que describen los principios fundamentales de la asignación de responsabilidades a objetos y son considerados una serie de buenas prácticas en el diseño de software.

Se encontrarán patrones repetidos de clases y objetos de comunicación, en muchos sistemas orientados a objetos. Estos patrones resuelven problemas específicos del diseño y vuelven al diseño orientado a objetos más flexible, elegante y extremadamente reutilizable. Ayudan a los diseñadores a reutilizar diseños exitosos basando nuevos diseños en experiencia previa (39).

A continuación se citan los patrones utilizados en la fase de implementación de la solución.

TABLA 14 PATRONES EMPLEADOS

Nombre del patrón	Descripción
Experto	Asignan una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Plantea que se debe asignar la responsabilidad de realizar determinada operación, a la clase que tiene la información necesaria para cumplir con dicha responsabilidad. El experto es usado más que cualquier otro patrón en la asignación de responsabilidades, es un principio usado continuamente en el diseño orientado a objetos
Bajo Acoplamiento	Asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible. Este patrón se tiene en cuenta por la importancia de realizar un diseño de clases independiente que puedan soportar los cambios de una manera fácil y permitan la reutilización. El uso de los patrones Experto y Creador favorecen al bajo acoplamiento entre las clases del sistema.
Alta Cohesión	Asignan a las clases responsabilidades para que trabajen sobre una misma área de la aplicación y que no tengan mucha complejidad. Este patrón se evidencia en las clases del sistema, las cuales tienen sólo los atributos y métodos necesarios para que estas cumplan con su función. Se recomienda tener un mayor grado de cohesión con un menor grado de acoplamiento.
Controlador	Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control. Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento.
Creador	Este patrón se usa en la creación de una instancia de una clase B en una clase A que la crea o la utiliza, guía por lo tanto la asignación de responsabilidades desde el punto de vista de la creación de objetos, lo que es muy frecuente en todo sistema orientado a objetos.

FUENTE ELABORACIÓN PROPIA

A continuación se presentan algunos ejemplos de utilización de patrones GRASP en el código fuente generado en la propuesta de solución:

En la clase *BusinessLogic* se evidencia la utilización de los patrones experto y creador, ya que la clase tiene toda la información necesaria para crear la interfaz visual, por lo que asume la responsabilidad de crear la misma. Estos patrones al ser bien utilizados en conjunto, producen un diseño de bajo acoplamiento, mayor claridad del código y facilitan la reutilización del mismo. También se evidencia el empleo del patrón controlador ya que en la clase se implementan métodos como el *getData*() o el *deleteConfig*() que utiliza la interfaz visual, separando la lógica del negocio de la capa de presentación.
MyRowTableModel es una de las clases donde se evidencia el empleo del patrón alta cohesión ya que sólo posee métodos y atributos relacionados con su función.

3.2 Estándar de código

El estándar de código utilizado fue el *CamelCase*.
La primera letra del identificador está en minúscula y la primera letra de las siguientes palabras concatenadas en mayúscula, por ejemplo *tableConfig*
A continuación una descripción de la **convención de nombres** utilizada para la solución:
Clases:
El nombre de la clase debe estar en singular, salvo que la clase represente multiplicidad de cosas. La letra inicial del nombre de la clase debe ser mayúscula al igual que la primera letra de las siguientes palabras concatenadas. Los nombres de las clases deben ser sustantivos descriptivos, ejemplo: *DBConfigurationService*, *Visual*, *MyRowTableModel*.
Métodos:
Los nombres de los métodos deben ser verbos, dado que describen una acción, ejemplo: *getData(), makeAuxPanel().* Los métodos dentro de las clases siempre deben declarar su visibilidad tales como *privadas, protegidas, públicas, etc.*
Variables:
Evitar variables que sean de un solo carácter, Los nombres comunes para las variables temporales son i, j, k, m. Los nombres de variables sólo pueden contener caracteres alfanuméricos. Además los nombres de variables deben ser camelCase, ejemplo: *dbConfigurations, generalConfig*.

3.3 Diagrama de despliegue

La vista de despliegue describe la configuración de los nodos de un sistema en ejecución y la organización de los componentes y objetos en él, incluyendo posibles migraciones de contenido entre nodos.

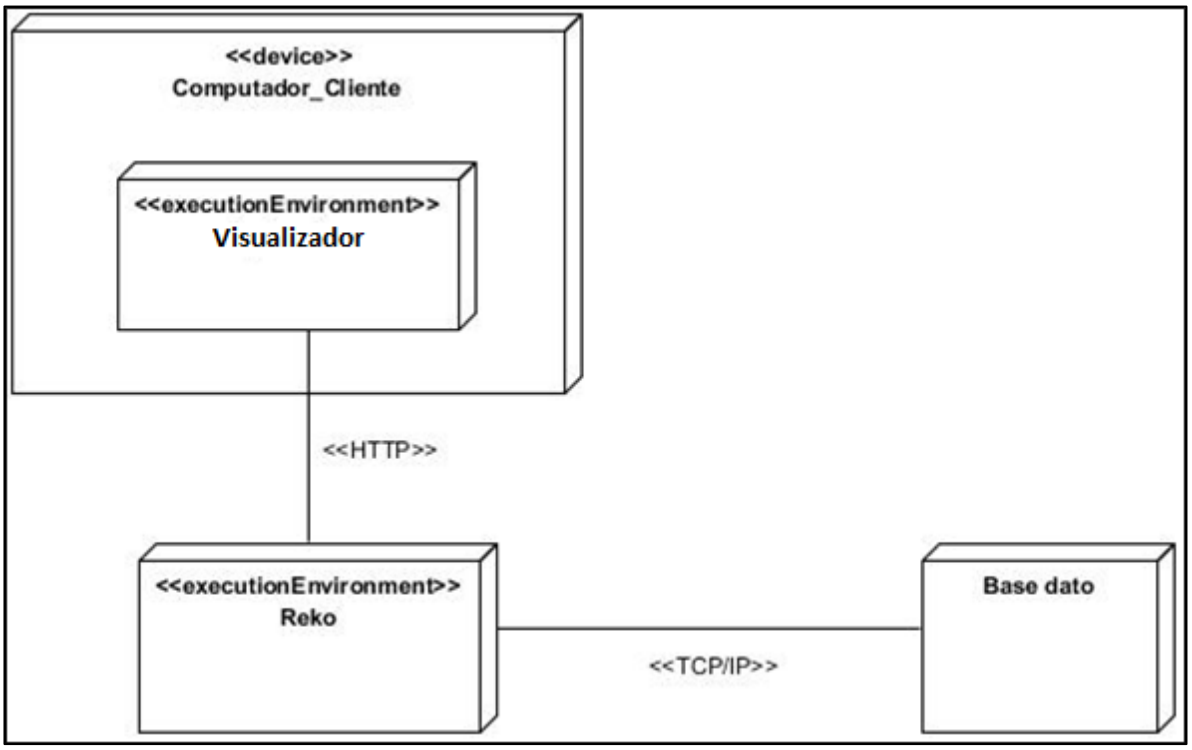


FIGURA 7 DIAGRAMA DE DESPLIEGUE (ELABORACIÓN PROPIA)

3.4 Pruebas de software

Las pruebas del software son la actividad más común de control de la calidad realizada en los proyectos para asegurar el correcto funcionamiento del software. Tienen como objetivos la verificación de la correcta implementación de los requisitos explícitamente establecidos, la adecuada integración de los componentes que conforman el sistema y la ejecución de casos de prueba que permitan detectar el mayor número de no conformidades y corregirlas antes de la entrega del software al cliente.

Vale mencionar que las pruebas minimizan la probabilidad de que aparezcan problemas que no son visibles en el software, pero si no se encuentra ninguna anomalía, no es una garantía su corrección. Se trata de diseñar pruebas que tengan alta probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y de tiempo.

3.5 Prueba de caja negra

También denominada prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. (…) La prueba de caja negra intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación (39).

La **partición equivalente** es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. Los diseños de caso de prueba para este método se basan en la evaluación de las clases de equivalencia para una condición de entrada (39).

3.5.1 Casos de prueba de aceptación

Los casos de prueba de aceptación son utilizadas para evaluar si al final de la iteración se obtuvo la funcionalidad deseada por el cliente. Su objetivo es comprobar, desde la perspectiva del usuario final, el cumplimiento de las especificaciones definidas para el producto.

A continuación se muestran los resultados obtenidos al aplicar las pruebas de aceptación con la utilización de la técnica partición equivalente.

TABLA 15 CASO DE PRUEBA LISTAR CONFIGURACIONES DE RÉPLICA

Caso de prueba	
Código: HU5_P1	Historia de usuario: 5
Nombre: Listar configuraciones de réplica.	
Descripción: Prueba para la funcionalidad listar configuraciones de réplica. Vale destacar que para que ocurra un correcto listado de las configuraciones de réplica actuales, previamente debe haber una correcta exportación e importación de datos de configuraciones de réplica y configuración general. Por lo que si esta prueba da resultados satisfactorios significa que dichas funciones también operan correctamente. Por lo tanto también se utiliza para medir las funcionalidades: - Exportar configuraciones de réplica. - Exportar configuración general. - Importar configuración de réplica. - Importar configuración general.	
Condiciones de ejecución: • Debe encontrarse desplegado y en estado de ejecución un nodo del Replicador de datos REKO. • La solución debe estar correctamente configurada. Los puertos predeterminados para establecer la comunicación con el Replicador de datos REKO son el 9870 y el 7978, para importar las configuraciones de réplica y la configuración general del nodo respectivamente. • Los puertos predeterminados pueden ser cambiados en acuerdo con el administrador de red local para evitar solapamientos.	
Entrada/Pasos de ejecución: **1.** Se ejecuta el sistema.	
Resultado esperado: Se deben listar el estado actual de las configuraciones de réplica en la sección “Configuraciones actuales:”.	

Resultado obtenido: Se obtiene el listado de las configuraciones de réplica actuales.

Nombre	Tipo	Dirección	Réplica de Datos	Réplica de estructuras
Nodo5	Nodo	Salida	Sí	No
Etiqueta3	Etiqueta	Salida	Sí	Sí

Evaluación de la prueba: Satisfactoria

FUENTE ELABORACIÓN PROPIA

TABLA 16 CASO DE PRUEBA VER DETALLES DE CONFIGURACIÓN DE RÉPLICA

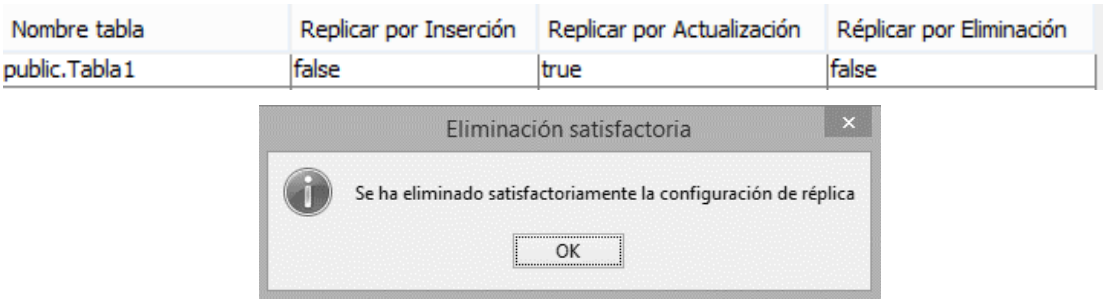
Caso de prueba	
Código: HU6_P1	Historia de usuario: 6
Nombre: Ver detalles de configuración de réplica.	
Descripción: Prueba para la funcionalidad ver detalles de la configuración de réplica.	
Condiciones de ejecución: Las configuraciones de réplica se encuentran listadas en el sistema en la sección “Configuraciones actuales:”.	
Entrada/Pasos de ejecución: **1.** El cliente da clic sobre una configuración determinada del listado, seleccionándola. **2.** Clic en el botón “Ver detalles”	
Resultado esperado: Se deben listar los detalles de la configuración seleccionada por el usuario.	
Resultado obtenido: Se muestra el listado de los detalles de la configuración seleccionada por el usuario.	

Nombre tabla	Replicar por Inserción	Replicar por Actualización	Réplicar por Eliminación
public.Tabla1	false	true	false
public.Estudiante	true	false	true

Evaluación de la prueba: Satisfactoria

FUENTE ELABORACIÓN PROPIA

TABLA 17 CASO DE PRUEBA ELIMINAR CONFIGURACIÓN DE RÉPLICA

Caso de prueba	
Código: HU7_P1	Historia de usuario: 7
Nombre: Eliminar configuración de réplica.	
Descripción: Prueba para la funcionalidad eliminar configuración de réplica.	
Condiciones de ejecución: Las configuraciones de réplica se encuentran listadas en el sistema, en la sección "Configuraciones actuales:".	
Entrada/Pasos de ejecución: **1.** El cliente da clic sobre una configuración determinada del listado, seleccionándola. **2.** Clic en el botón "Eliminar"	
Resultado esperado: El sistema debe eliminar automáticamente la configuración asociada y actualizar la sección "Configuraciones Actuales" con las configuraciones restantes. Se debe notificar al usuario que la eliminación ha sido realizada de forma satisfactoria, mediante un mensaje de información: "Se ha eliminado satisfactoriamente la configuración de réplica".	
Resultado obtenido: Se muestra el listado actualizado de configuraciones de réplica de la sección "Configuraciones actuales" y se muestra un mensaje al usuario, informándole del éxito de la eliminación. 	
Evaluación de la prueba: Satisfactoria	

FUENTE ELABORACIÓN PROPIA

TABLA 18 CASO DE PRUEBA ACTUALIZAR

Caso de prueba	
Código: HU8_P1	Historia de usuario: 8
Nombre: Actualizar	
Descripción: Prueba para la funcionalidad actualizar listado de configuraciones de réplica.	

Condiciones de ejecución: El sistema se encuentra en ejecución, las configuraciones de réplica han sido cargadas y listadas en la sección “Configuraciones actuales”
Entrada/Pasos de ejecución: **1.** El cliente da clic en el botón “Actualizar”.
Resultado esperado: El sistema debe automáticamente actualizar el listado de configuraciones actuales, reflejando cualquier cambio que éste pudiera haber sufrido.
Resultado esperado: Se muestra el listado actualizado de configuraciones de réplica de la sección “Configuraciones actuales”.
Evaluación de la prueba: Satisfactoria

FUENTE ELABORACIÓN PROPIA

3.5.2 Resultados de las pruebas de aceptación

A continuación se presentan los resultados arrojados durante las pruebas aplicadas:

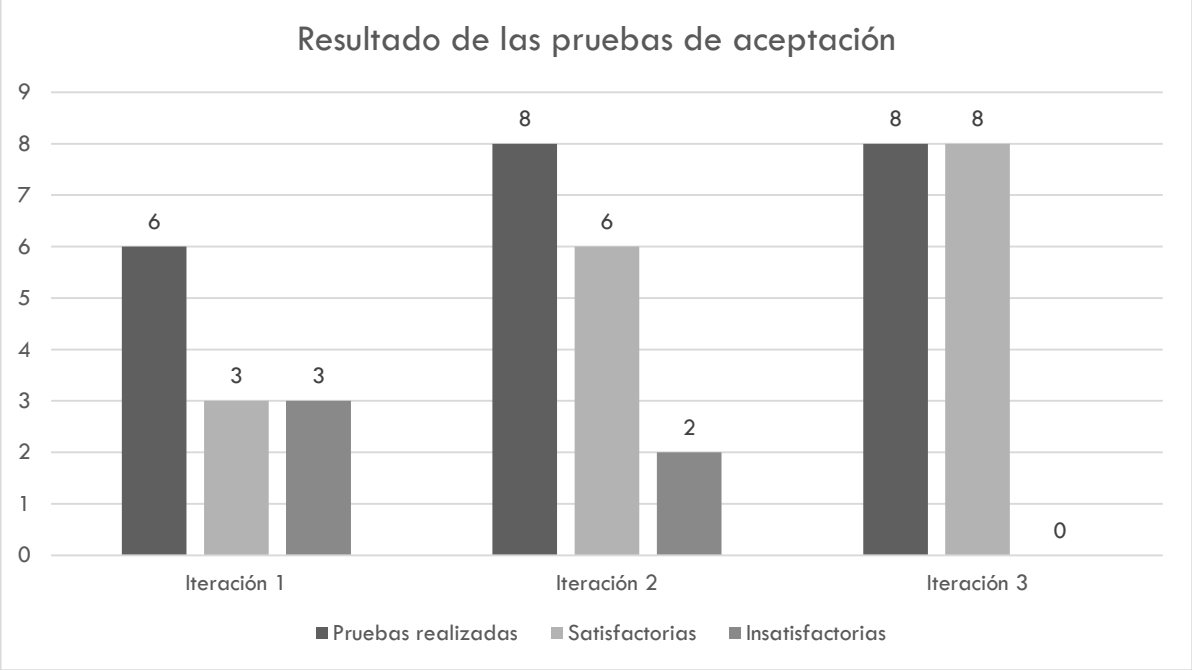


FIGURA 8 RESULTADOS DE LAS PRUEBAS DE ACEPTACIÓN (ELABORACIÓN PROPIA)

Se ejecutaron tres iteraciones de pruebas. En la primera iteración fueron realizadas un total de seis pruebas y como resultado fueron detectadas tres no conformidades. Una vez detectadas estas no conformidades, se llegó al consenso con el tutor de erradicarlas en un

intervalo de tres días. En una segunda iteración fueron realizadas un total de ocho pruebas, en las cuales se comprobó la corrección de las tres no conformidades detectadas en la primera iteración. Como resultado de la segunda iteración se descubrieron dos nuevas no conformidades y se definió un plazo de dos días para corregirlas. En una tercera iteración se realizaron un total de ocho pruebas, comprobando la adecuada corrección de las dos no conformidades detectadas en la segunda iteración. En esta tercera iteración no se encontraron nuevas fallas, corroborando el correcto funcionamiento de los requisitos implementados en la solución.

3.6 Pruebas de caja blanca

La prueba de caja blanca, denominada a veces *prueba de caja de cristal* es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; ejecuten todos los bucles en sus límites y con sus límites operacionales; y ejerciten las estructuras internas de datos para asegurar su validez (39).

Para probar los principales flujos de control de la aplicación se utilizó la técnica de camino básico.

3.6.1 Pruebas del camino básico

La prueba del camino básico es una técnica de prueba de caja blanca que le permite al diseñador de casos de prueba obtener una medida de complejidad lógica de un diseño procedimental y que use esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para ejercitar el conjunto básico, deben garantizar que se ejecuta cada instrucción del programa por lo menos una vez durante la prueba (39).

A continuación se muestra el caso de prueba para el método *actualizar*, el cual se localiza en la clase MainPanel. El objetivo del método es actualizar el listado de configuraciones actuales que se muestra en la interfaz visual de la solución. En la figura 9 se muestra el código fuente referente al método descrito anteriormente y en la figura 10 se aplica la técnica de camino básico, cuyo procedimiento se realizó automáticamente con la utilización de la herramienta Visustin referente al código fuente presentado. Para consultar las pruebas de camino básico realizadas a otros métodos clave de la solución véase la sección de Anexos.

```
161    public void actualizar() throws RemoteException, DataAccessExceptionException {
162        Visual.setMyList(null);
163        if(Visual.auxPanel2!=null)
164        {
165            Visual.setFila(-1);
166            Visual.marco.getContentPane().remove(Visual.auxPanel2);
167            Visual.auxPanel2.validate();
168            Visual.auxPanel2 = null;
169        }
170        Visual.marco.validate();
171        BusinessLogic.getData();
172        Visual.makeAuxPanel();
173        myModel = new MyRowTableModel(clases2, miListaColumnas);
174        table.setModel(myModel);
175        table.validate();
176        configurarTabla();
177        Visual.marco.validate();
178    }
```

FIGURA 9 CÓDIGO FUENTE CORRESPONDIENTE AL MÉTODO *ACTUALIZAR* DE LA CLASE MAINPANEL(ELABORACIÓN PROPIA)

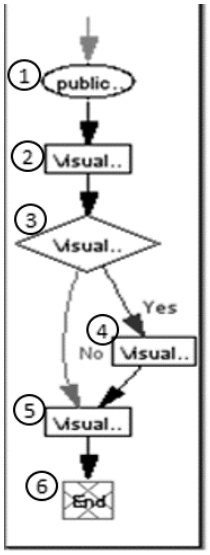


FIGURA 10 FLUJO AUTOMÁTICO CORRESPONDIENTE AL MÉTODO *ACTUALIZAR* (ELABORACIÓN PROPIA)

TABLA 19 COMPLEJIDAD CICLOMÁTICA MÉTODO *ACTUALIZAR*

V(G) = A – N + 2	V(G) = R	V(G) = P + 1
A = 6 (aristas) N = 6 (nodos)	R = 2 (regiones)	P = 1 (nodos predicados)
V(G) = 6 – 6 + 2 V(G) = 2	V(G) = 2	V(G) = 1 + 1 V(G) = 2

FUENTE ELABORACIÓN PROPIA

El cálculo efectuado mediante las tres fórmulas ha dado como resultado el mismo valor, por lo que se puede afirmar que la complejidad ciclomática del código es 2 lo cual entra en el límite para el número de caminos independientes que componen el conjunto básico y, consecuentemente, un valor límite superior para el número de pruebas que se deben diseñar y ejecutar con el propósito de garantizar que se cubran todas las sentencias de los procedimientos.

Conjunto de caminos independientes correspondientes al método *actualizar*:

Camino 1: 1, 2, 3, 4, 5, 6

Camino 2: 1, 2, 3, 5, 6

TABLA 20 CASOS DE PRUEBA QUE FUERZAN LA EJECUCIÓN DE CADA CAMINO DEL MÉTODO *ACTUALIZAR*

No.	Caso de prueba	Resultado esperado
1	Si no se ha seleccionado ninguna fila.	Se actualiza la interfaz visual con los datos actualizados.
2	Si se ha seleccionado alguna fila.	Se actualiza la interfaz visual con los datos actualizados.

FUENTE ELABORACIÓN PROPIA

3.6.2 Resultados de las pruebas de camino básico

Vale destacar que a medida que se fue desarrollando el software se fueron aplicando pruebas a nivel del programador, las cuales consisten en ir probando las funcionalidades implementadas con el propósito de verificar su correcto funcionamiento. Este procedimiento permitió obtener un menor número de errores al realizar las pruebas. Concluidas las pruebas de camino básico, se puede afirmar que se ejercitaron todos los caminos independientes (caminos básicos) de la estructura de control, con el fin de asegurar que todas las sentencias de la solución informática se ejecutan por lo menos una vez con resultados satisfactorios. Por lo que se demuestra que el sistema cumple con cada uno de los requisitos establecidos.

3.7 Conclusiones parciales del capítulo

Durante el desarrollo de este capítulo se definieron los diferentes patrones de diseño utilizados así como la descripción de los mismos. Se logró conocer la distribución física del sistema en términos de nodos a través del diagrama de despliegue. Se implementó la propuesta solución y se obtuvieron importantes resultados sobre el comportamiento del sistema aplicando las pruebas correspondientes, éstas permitieron comprobar los requisitos funcionales definidos para el sistema, a partir de los diseños de casos de pruebas de caja negra, evaluándose la calidad del mismo. Además, luego de la realización de las pruebas de caja blanca, se puede afirmar que se ejecutaron todas las sentencias de la solución informática por lo menos una vez, demostrando el correcto funcionamiento de las mismas.

Conclusiones generales:

El desarrollo de la presente investigación permitió elaborar un visualizador de configuraciones de réplicas para el Replicador de datos REKO, dando cumplimiento a los objetivos trazados, destacándose de manera general los siguientes aspectos:

- Mediante el estudio de la bibliografía especializada en el campo de la réplica de información, se logró sentar las bases teóricas que sustentaron la presente investigación.
- Se realizó un estudio de los sistemas replicadores de bases de datos existentes, arrojando como resultado las deficiencias y fortalezas de cada uno, lo cual demostró que estas no brindan solución al problema planteado, por lo que fue necesario desarrollar una solución propia.
- Mediante la implementación de la propuesta de solución se obtuvo una aplicación informática capaz de visualizar las configuraciones de réplica del Replicador de datos REKO.
- Las pruebas realizadas a la solución, corroboraron el correcto funcionamiento del software y el cumplimiento de todos los requerimientos definidos para el mismo. Por lo que se puede afirmar que el software desarrollado asegura una correcta visualización de las configuraciones de réplica.

Recomendaciones:

Agregar al sistema la posibilidad de editar una configuración de réplica existente, así como la posibilidad de crear una nueva.

Bibliografía

1. **Velázquez Vizcay, Adriel, y otros.** Compumat-2013. *Compumat.* [En línea] 25 de Mayo de 2013. [Citado el: 23 de Octubre de 2013.]

2. **Date, Christopher J.** *Introducción a los Sistemas de Bases de Datos.* [trad.] Sergio Luis María Ruiz Faudón. Séptima. México : Pearson Education, 2001. pág. 960.

3. **Silberschatz, Abraham.** *Fundamentos de Bases de Datos.* Madrid : McGRAW-HILL, 2002. 0-07-228363-7.

4. **Microsoft.** Microsoft Developer Network. *Replicación de SQL Server.* [En línea] Microsoft, 2015. [Citado el: 12 de Abril de 2015.] https://msdn.microsoft.com/es-es/library/ms151198.aspx.

5. **Buchilla, Lic. Vivian Romero.** Monografias.com. *Monografias.com.* [En línea] 2009. [Citado el: 4 de Abril de 2015.] http://www.monografias.com/trabajos82/replicaciondatos/replicaciondatos.shtml#ixzz3X9VC7IIo. s.n..

6. **Riggs, Simon y Krosing, Hannu.** *PostgresSQL 9 Administration Cookbook.* Birmingham : Packt Publishing Ltd., 2010. 978-1-849510-28-8.

7. **Slony Development Group.** Slony-I. [En línea] Command Prompt, Inc., 2010. [Citado el: 14 de Abril de 2015.] http://www.slony.info/.

8. **Christopher Browne.** *Slony-I 2.2.4 Documentation.* s.l. : The PostgreSQL Global Development Group, 2015. pág. 298.

9. **The PostgreSQL Global Development Group.** PgAdmin.org. [En línea] [Citado el: 14 de Abril de 2015.] http://dave.webdev.pgadmin.org/docs/1.6/slony/slonyintro.html.

10. **Long, Eric, y otros.** SymmetricDS User Guide. [En línea] 23 de Julio de 2012. http://www.symmetricds.org/doc/3.0/pdf/user-guide.pdf.

11. **Hit Software, Inc. BackOffice Associates, LLC.** www.hitsw.com/localized/spanish. [En línea] 2013. http://www.hitsw.com/localized/spanish/products_services/dbmoto/dbmoto_cloud/DBMoto_Cloud_Edition.html.

12. **HiT Software, Inc. BackOffice Associates, LLC.** http://www.hitsw.com. [En línea] 2013. http://www.hitsw.com/localized/spanish/products_services/dbmoto/dbmoto.html?__hstc=753710.b33da759d6a55fc4e7f4344143ce2ca9.1432278088132.1432278088132.1433826949173.2&__hssc=753710.1.1433826949173&__hsfp=1134820550.

13. **Daffodil Replicator Team.** Daffodil Software. [En línea] Daffodil Software Ltd., 2007. http://www.daffodilsw.com/.

14. **Oracle.** *Oracle streams an Oracle withe paper.* 2002. pág. 18.

15. **Mena Rodriguez, Luis Edgardo y Pérez Alfonso, Damián.** Módulo de Reko para la replicación entre bases de datos con. La Habana : s.n., 2012. Vol. 5, 4.

16. **Sánchez Rodríguez, Tamara.** *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n., 2014. pág. 13, Programa de Mejora.

17. **Hamilton, Kim y Miles, Russell.** *Learning UML 2.0.* s.l. : O'Reilly, 2006. pág. 286. 0-596-00982-8.

18. **Visual-Paradigm Development Group.** Visual-Paradigm. [En línea] Febrero de 2011. [Citado el: 26 de Marzo de 2015.] www.visual-paradigm.com.

19. **Oracle.** Oracle.com. *Oracle.com.* [En línea] [Citado el: 26 de Marzo de 2015.] http://www.oracle.com/es/technologies/java/features/index.html.

20. **Departamento de Informática Universidad de Velladolid.** Departamento de Informática. *http://www.infor.uva.es/.* [En línea] http://www.infor.uva.es/~jmrr/tgp/java/JAVA.html.

21. **Groussard, Thierry.** *Java7 Los fundamentos del lenguaje Java.* [ed.] Estelle Dechenaud y Juan Javier Piqueres. Barcelona : Ediciones Eni, 2012. 978-2-7460-7318-0.

22. **Pivotal Software.** Spring. *Spring Tool Suite.* [En línea] [Citado el: 13 de Abril de 2015.] https://spring.io/tools/sts.

23. **Johnson, Rod, y otros.** Spring.io. *Docs.spring.io.* [En línea] [Citado el: 14 de Abril de 2015.] http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/.

24. **Gómez-Arribas, Francisco J., Holgado, Susana y López de Vergara, Jorge E.** Departamento de Ingeniería de Sistemas Telemáticos - ETSIT - UPM. [En línea] 16 de Septiembre de 2005. http://neweb.dit.upm.es/~jlopez/publicaciones/ucami05.pdf. 84-9732-442-0.

25. **W3C Recommendation.** www.w3.org. [En línea] 1.2, 27 de Abril de 2007. [Citado el: 19 de Noviembre de 2014.] http://www.w3.org/TR/2007/REC-soap12-part0-20070427/#L1161.

26. **Ing. C, Benjamín González.** http://www.desarrolloweb.com. [En línea] 07 de Julio de 2004. [Citado el: 19 de Noviembre de 2014.] http://www.desarrolloweb.com/articulos/1557.php.

27. **w3c.** http://www.w3c.es. [En línea] [Citado el: 11 de Noviembre de 2014.] http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML.

28. **W3C.** www.sidar.org. [En línea] 1998. http://www.sidar.org/recur/desdi/traduc/es/xml/xml1/index.html.

29. **Oracle.** http://www.oracle.com/. [En línea] http://java.sun.com/xml/jaxb/.

30. **Sun Microsystems, Inc.** *JavaTM Architecture for XML Binding (JAXB) Specification.* Sun Microsystems, Inc. Santa Clara, California : s.n., 2002. pág. 188, Especificación.

31. **Vallecillo, Antonio.** www.lcc.uma.es. *Departamento Lenguajes y Ciencias de la Computación Universidad de Málaga.* [En línea] http://www.lcc.uma.es/~canal/sabc/curso0405/WebServices.pdf.

32. **Marsh, Jonathan, Rogers, Tony y Le Hégaret, Philippe.** http://www.w3.org. [En línea] 2007. http://www.w3.org/2002/ws/desc/.

33. **Sociedad Informática del Gobierno Vasco.** *SoapUI. Manual de usuario v2.2.doc.* EJIE, S.A Sociedad Informática del Gobierno Vasco. Vitoria-Gasteiz : s.n., 2011. Manual de Usuario.

34. **Balsamiq Studios.** Balsamiq. [En línea] 2008. https://balsamiq.com/.

35. **Aivosto.** www.aivosto.com/index.html. *Aivosto.* [En línea] 2014. http://www.aivosto.com/visustin-es.html.

36. **The PostgreSQL Global Development Group.** PostgreSQL 9.4.0 Documentation. *PostgreSQL.org.* [En línea] [Citado el: 14 de Abril de 2015.] http://www.postgresql.org/files/documentation/pdf/9.4/postgresql-9.4-A4.pdf.

37. **Larman, Craig.** *UML y Patrones.* Segunda. s.l. : Prentice Hall, 2002. 978-84-205-3438-1.

38. **Sommerville, Ian.** *Ingienería del software.* [ed.] Marta Caicoya Miguel Martín-Romo. [trad.] Antonio Botía Martínez, Francisco Mora Lizán, José Pascual Trigueros Jover María Isabel Alfonso Galipienso. Séptima. Madrid : Pearson Educación, S.A., 2005. 84-7829-074-5.

39. **Pressman, Roger S.** *Ingienería de software Un enfoque práctico.* Quinta. s.l. : Mc Graw Hill, 2000.

40. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** *El lenguaje unificado de modelado.* s.l. : Addison Wesley, 2006. 978-84-7829-076-5.

Anexos

Anexo 1 Descripción de atributo de calidad: fiabilidad, madurez.

Atributo de Calidad	Fiabilidad.
Sub-atributos/Sub-característica	Madurez.
Objetivo	Capacidad del producto para satisfacer las necesidades de fiabilidad en condiciones normales.
Origen	Proveedor de requisitos.
Artefacto	El sistema.
Entorno	El sistema desplegado y funcionando en periodos de tiempos determinados.
Estímulo **<<1>>. <<a>> <El sistema desplegado en los escenarios de réplica especificados por el cliente>**	**Respuesta: Flujo de eventos (Escenarios)**
El sistema funciona correctamente bajo periodos de tiempos determinados por el cliente, mientras no ocurra un evento extraordinario en el servidor donde se encuentra desplegada la solución. O en caso de reinicio o apagado del mismo.	NA
Medida de respuesta	
Desplegar el sistema.	

Anexo 2 Descripción de atributo de calidad: fiabilidad, disponibilidad.

Atributo de Calidad	Fiabilidad.
Sub-atributos/Sub-característica	Disponibilidad.
Objetivo	Capacidad del producto de estar operativo y accesible para su uso cuando se requiere.
Origen	Proveedor de requisitos.
Artefacto	El sistema.
Entorno	El sistema desplegado, funcionando la mayor cantidad de horas del día.

Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
<<1>>. <<a>> < El sistema desplegado, funcionando la mayor cantidad de horas del día.>	
El sistema funcionando normalmente la mayor cantidad de horas al día, mientras no ocurra un evento extraordinario en el servidor donde se encuentra desplegada la solución. O en caso de reinicio o apagado del mismo.	NA
Medida de respuesta	
Desplegar el sistema.	

Anexo 3 Descripción de atributo de calidad: usabilidad, protección contra errores humanos.

Atributo de Calidad	Usabilidad.
Sub-atributos/Sub-característica	Protección contra errores de usuarios.
Objetivo	Capacidad del producto para proteger a los usuarios de cometer errores.
Origen	Proveedor de requisitos.
Artefacto	Los mensajes de alerta para prevenir al usuario ante un error en la configuración y administración del producto.
Entorno	El sistema desplegado.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
<<1>>. <<a>>< Mensajes de alerta para prevenir al usuario ante un error en la configuración y administración del producto.>	
El usuario intenta eliminar una configuración sin antes haber seleccionado una configuración de réplica.	1. El usuario presiona el botón ¨Eliminar¨. 2. El sistema muestra un mensaje de alerta, informándole al usuario que no hay configuración de réplica seleccionada para realizar la eliminación.
Medida de respuesta	
Notificar al usuario.	

Anexo 4 Descripción de atributo de calidad: usabilidad, operabilidad.

Atributo de Calidad	Usabilidad.
Sub-atributos/Sub-característica	Operabilidad.
Objetivo	Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.
Origen	Proveedor de requisitos.
Artefacto	Interfaz de usuario.
Entorno	El sistema desplegado.
Estímulo **<<1>>. <<a>>< Capacidad del producto de ser operarlo y controlarlo con facilidad >**	**Respuesta: Flujo de eventos (Escenarios)**
Interactuar con el listado de configuraciones de réplica.	1. El sistema muestra el listado de configuraciones de réplica. 2. El usuario selecciona una configuración específica y selecciona la opción acorde a sus necesidades, seleccionando entre los botones “Ver detalles” y “Eliminar”
Medida de respuesta	
Ver detalles de la configuración o eliminar configuración, respectivamente.	

Anexo 5 Descripción de atributo de calidad: portabilidad, adaptabilidad.

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	Proveedor de requisitos.
Artefacto	El sistema.
Entorno	El sistema desplegado.
Estímulo **<<1>>. <<a>>< Capacidad del sistema de adaptarse de forma efectiva a diferentes**	**Respuesta: Flujo de eventos (Escenarios)**

entornos>	
El sistema está diseñado con tecnologías que permiten que este se adapte a cualquier sistema operativo.	NA
Medida de respuesta	
El ambiente de despliegue del sistema.	

Anexo 6 Descripción del atributo de calidad: adecuación funcional, integridad funcional.

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 **<<1>>. <<a>>< Grado en el que el sistema cubre todas las tareas y objetivos especificados>**	**Respuesta: Flujo de eventos (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 atributo de calidad: adecuación funcional, corrección funcional.

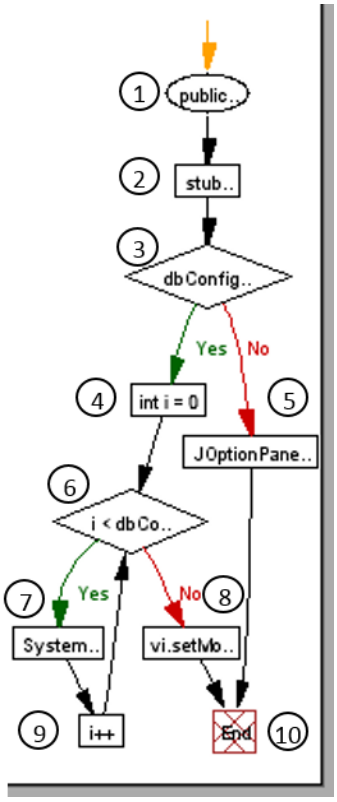
Atributo de Calidad	Adecuación funcional.
Sub-atributos/Sub-característica	Corrección funcional.
Objetivo	Grado en que un producto o sistema proporciona los resultados correctos con el grado necesario de precisión.

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 los procesos solicitados por el cliente con la exactitud necesaria en cada uno de ellos.	NA
Medida de respuesta	
Navegar en el sistema.	

Anexo 8 Código fuente correspondiente al método *getData* de la clase BusinessLogic

```
77    public static void getData() throws RemoteException, DataAccessExceptionException {
78        // TODO Auto-generated method stub
79        stub = new DBConfigurationServiceImplServiceStub();
80        allConfigurationsE = new GetAllConfigurationsE();
81        allConfigurations = new GetAllConfigurations();
82
83        allConfigurationsE.setGetAllConfigurations(allConfigurations);
84
85        allConfigurationsResponseE = stub.getAllConfigurations(allConfigurationsE);
86
87        if(allConfigurationsResponseE!=null)
88        {
89            dbConfigurations =
90            allConfigurationsResponseE.getGetAllConfigurationsResponse().get_return();
91
92            vi.setModelClass(dbConfigurations);
93        }
94    }
```

Anexo 9 Flujo automático correspondiente al método *getData*



Anexo 10 Complejidad ciclomática método *getData*

V(G) = A – N + 2	V(G) = R	V(G) = P + 1
A = 11 (aristas) N = 10 (nodos)	R = 3 (regiones)	P = 2 (nodos predicados)
V(G) = 11 – 10 + 2 V(G) = 3	V(G) = 3	V(G) = 2 + 1 V(G) = 3

Anexo 11 Conjunto de caminos independientes correspondientes al método *getData*

Conjunto de caminos independientes:

Camino 1: 1, 2, 3, 5, 10

Camino 2: 1, 2, 3, 4, 6, 8, 10

Camino 3: 1, 2, 3, 4, 6, 7, 9, 8, 10

Anexo 12 Conjunto de caminos independientes correspondientes al método *getData*

No.	Caso de prueba	Resultado esperado
1	No existen configuraciones de réplica en el Replicador de datos REKO para importar	Se muestra un mensaje advertencia, indicando que no hay configuraciones de réplica para importar
2	Hay una sola configuración	Se importa la configuración
3	Hay más de una configuración	Se importan todas las configuraciones

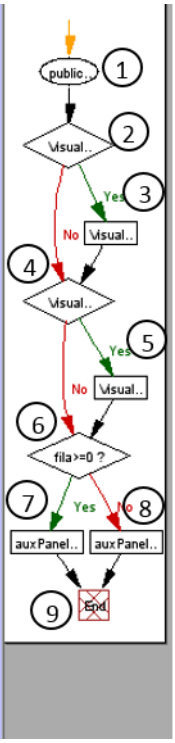
FUENTE ELABORACIÓN PROPIA

Anexo 13 Código y flujo automático correspondiente al método makeAuxPanel de la clase MainPanel

```
public static void makeAuxPanel() {
   if(Visual.auxPanel!=null)
   {
     Visual.marco.getContentPane().remove(Visual.auxPanel);
   }
   if(Visual.split!=null)
   {
     Visual.marco.getContentPane().remove(Visual.split);
     //Visual.split.remove(auxPanel2);
   }

   if(fila>=0)
   {
     auxPanel = null;
     auxPanel = new MainPanel(Visual.getMyList());
     auxPanel.validate();
      split = null;
     split = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, auxPanel, auxPanel2);
     split.setDividerLocation(505);
     marco.getContentPane().add(split);
     marco.validate();
   }
   else{
     auxPanel = null;
     auxPanel = new MainPanel(Visual.getMyList());
     auxPanel.validate();
      split = null;
     split = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, auxPanel, scrollAux);
     split.setDividerLocation(505);
     marco.getContentPane().add(split);
     marco.validate();
   }

}
```



Anexo 14 Complejidad ciclomática método *makeAuxPanel*

V(G) = A – N + 2	V(G) = R	V(G) = P + 1
A = 11 (aristas) N = 9 (nodos)	R = 4 (regiones)	P = 3 (nodos predicados)
V(G) = 11 – 9 + 2 V(G) = 4	V(G) = 4	V(G) = 3 + 1 V(G) = 4

Anexo 15 Conjunto de caminos independientes correspondientes al método *makeAuxPanel*

Conjunto de caminos independientes:

Camino 1: 1, 2, 4, 6, 8, 9

Camino 2: 1, 2, 4, 6, 7, 9

Camino 3: 1, 2, 3, 4, 5, 6, 7, 9

Camino 4: 1, 2, 3, 4, 6, 8, 9

Anexo 16 Conjunto de caminos independientes correspondientes al método *makeAuxPanel*

No.	Caso de prueba	Resultado esperado
1	No hay ningún componente de la clase Visual previamente creado, no se ha seleccionado ninguna configuración de réplica.	Se crea el panel principal a partir de los datos importados
2	Hay algún componente de la clase Visual previamente creado	Se vacían dichos componentes y se vuelven a llenar a partir de los datos importados
3	Se están visualizando los detalles de una configuración de réplica y es seleccionada otra	Se vacía el panel de configuraciones y se vuelve a crear el panel principal con los datos de la otra configuración de réplica
4	No se está visualizando ninguna configuración y es selecciona una	Se vacía el panel auxiliar y se crea el panel con los detalles de la configuración de réplica

FUENTE ELABORACIÓN PROPIA