Universidad de las Ciencias Informáticas

Facultad de Ciencias Tecnologías Computacionales

“Réplica y Sincronización de datos para Xilema Suria 3.0”

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

Autor:

Alejandro De León Abreu

Tutores:

Ing. Yenisel Tirado Gutiérrez

Ing. Guillermo Luzua Farias

La Habana, junio del 2017

"No existe una manera fácil. No importa cuán talentoso seas, tu talento te va a fallar si no lo desarrollas. Si no estudias, si no trabajas duro, si no te dedicas a ser mejor cada día."

-Will Smith

Declaración de Autoría

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

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

Alejandro de León Abreu

_____
Firma del Autor

Ing. Yenisel Tirado Gutiérrez

_____

Firma Tutor

Ing. Guillermo Luzua Farias

_____

Firma Tutor

Datos de contacto

Tutores:

Ing. Yenisel Tirado Gutiérrez.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Correo electrónico: ytgutierrez@uci.cu

Ing. Guillermo Luzua Farias.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Correo electrónico: gluzua@uci.cu

Resumen

La Universidad de las Ciencias Informáticas cuenta con el centro de Geoinformática y Señales Digitales, dedicado al desarrollo de aplicaciones que abarcan los sistemas de geoinformación, la gestión y transmisión de archivos audiovisuales y la Video Vigilancia. Actualmente el centro se encuentra trabajando en una solución de Video Vigilancia llamada Xilema Suria 3.0. Este sistema está concebido bajo una filosofía de desarrollo por módulos, los cuales brindan funcionalidades que satisfacen diferentes necesidades como la vigilancia de las principales zonas de interés para las empresas e instituciones. El sistema presenta afectaciones con la disponibilidad y almacenamiento de la información que se maneja. Lo que trae como consecuencia incoherencia en los datos almacenados. Es por ello que la presente investigación definió como objetivo el desarrollo de una herramienta para aumentar la disponibilidad de los datos almacenados en el sistema Xilema Suria 3.0. De esta forma se garantiza la persistencia de la información que maneja el sistema, así como su disponibilidad. Para el desarrollo de la herramienta fueron combinadas varias tecnologías tales como MySQL para la persistencia de los datos y AUP-UCI como metodología usada para guiar los procesos de implementación de la herramienta. También se exponen los resultados obtenidos producto al estudio realizado a varias soluciones de réplica y sincronización de bases de datos, lo cual proporcionó el conocimiento requerido para la posterior implementación del sistema en cuestión. Igualmente, para evaluar los resultados de la investigación se realizaron las pruebas de caja negra y caja blanca. Las cuales demostrando que el sistema cumple con los requisitos especificados y con la calidad requerida para ser integrado al sistema de video vigilancia Xilema Suria 3.0.

Palabras claves: Disponibilidad, Réplica, Sincronización.

Abstract

The Universidad de las Ciencias Informáticas has the Center of Geoinformatics and Digital Signals, dedicated to the development of applications covering geoinformation systems, the management and transmission of audiovisual files and Video Surveillance. The center is currently working on a Video Surveillance solution called Xilema Suria 3.0. This system is conceived under a modular development philosophy which provide functionalities that satisfy different needs such as monitoring the main areas of interest for companies and institutions. The system is affected by the availability and storage of the information that is handled. This results in inconsistency in the stored data. This is why the present research defined as objective the development of a tool to increase the availability of data stored in the Xilema Suria 3.0 system. This ensures the persistence of the information that the system handles, as well as its availability. For the development of the tool were combined several technologies such as MySQL for data persistence and AUP-UCI as a methodology used to guide the implementation processes of the tool. The results obtained are also presented to the study of several solutions of database replication and synchronization, which provided the required knowledge for the subsequent implementation of the system in question. Likewise, to evaluate the results of the investigation were carried out the tests of black box and white box. Which demonstrates that the system complies with the specified requirements and with the quality required to be integrated into the Xilema Suria 3.0 video surveillance system.

Keywords**:** Availability, Replication, Synchronization.

Índice General

Índice de tablas

Índice de figuras

Introducción

El crecimiento constante y acelerado de la información ha traído como consecuencia que con el paso del tiempo la forma tradicional de almacenarla y consultarla se convirtiera en una actividad demasiado costosa. Sin embargo, desde hace unas décadas, con el desarrollo de las Tecnologías de la Informática y las Comunicaciones (TIC), han surgido un conjunto de herramientas que permiten realizar este proceso de forma automática. Una de las tecnologías existentes son las Bases de Datos (BDs), las cuales permiten guardar gran cúmulo de información de manera organizada para que luego se pueda consultar y utilizar fácilmente, manteniendo una estricta seguridad. (Belloch Ortí, 2006)

Originalmente la información se almacenaba de forma centralizada, pero con el transcurso del tiempo, la necesidad de acceder a los datos aumentó y a su vez surgieron inconvenientes, a los cuales no era posible dar una solución eficiente desde un modelo centralizado. Estos problemas dieron paso a la creación de los Sistemas de Bases de Datos Distribuidas (SBDD). Estos responden a la relación que existe entre diferentes BDs las cuales se encuentran en ordenadores ubicados geográficamente distantes e interconectados por una red de comunicaciones. (Lic. D. Ramon, 2007)

El uso de los sistemas distribuidos ha propiciado que las BDs que lo conforman tengan la necesidad de mantener la actualización y sincronización de los datos entre ellas. Para lograr esto fue necesario desarrollar herramientas que permitieran copiar y distribuir los objetos de una Base de Datos (BD) hacia otra, manteniendo la coherencia y consistencia de la información. Este proceso se dio a conocer como réplica de datos.

Xilema-Suria 3.0 es un sistema de video vigilancia en desarrollo por el Centro de Geoinformática y Señales Digitales (GEYSED) de la Universidad de las Ciencias Informáticas (UCI). Las versiones 1.0 y 2.0 de este sistema están desplegadas en la universidad, donde se utilizan para el control y la monitorización de áreas de interés dentro del centro de altos estudios. Actualmente, en las versiones desplegadas se tiene que realizar las salvas de la información de las BDs de manera manual. Siendo el operador de monitorización una de las personas que realiza esta operación sobre los datos almacenados por el sistema. El mismo en ocasiones no cuenta con todos los conocimientos técnicos necesarios para poder realizar estas operaciones. Esto trae como consecuencia que existan archivos que posean un contenido que no se hayan salvado correctamente y que los datos estén corruptos o defectuosos debido a la mala manipulación de las BDs al realizar la salva de la información. A su vez, los medios de salva guarda y creación de los datos no son confiables; ya que al poner en un dispositivo de almacenamiento esta información para ser llevados al servidor se incurren en accesos no autorizados a los datos. Lo que crea como consecuencia que

informaciones que no pueden ser vistas por los usuarios que tienen iguales privilegios sobre zonas diferentes queden expuestas, acarreando fallas de accesibilidad y permisos sobre información priorizada. Igualmente, al contar con una BD centralizada se da el escenario donde por problemas de comunicación no se tiene acceso a los datos y el sistema pierde toda disponibilidad de procesamiento y almacenamiento de la información.

Lo descrito anteriormente introduce como **problema de investigación**: ¿Cómo aumentar la disponibilidad de los datos almacenados en la base de datos de Xilema Suria 3.0?

Se define como **objetivo general:** Desarrollar una herramienta para aumentar la disponibilidad de los datos almacenados en el sistema Xilema Suria 3.0.

Para darle cumplimiento al objetivo de la investigación se define como **objeto de estudio:** los procesos de réplica y sincronización de datos. Acotando el **campo de acción:** las herramientas de réplica y sincronización de información para bases de datos.

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

1. Realización de un estudio del estado del arte basado en los procesos de administración de las herramientas de sincronización y réplica de datos.
2. Selección de las tecnologías, estándares y metodologías necesarias a utilizar en el desarrollo de la herramienta de réplica y sincronización de datos para el sistema Xilema Suria 3.0.
3. Obtención de los requisitos funcionales y no funcionales de la propuesta de solución para representar sus características.
4. Implementación de la herramienta de sincronización y réplica de datos para el sistema Xilema Suria 3.0.
5. Selección de las técnicas y métodos para la validación de la solución.

Según los elementos anteriormente expuestos se plantean como **preguntas científicas:**

1. ¿Qué elementos se deben tener en cuenta para implementar una solución de sincronización y réplica de datos?
2. ¿Qué metodología de software, herramientas y tecnologías se utilizarán para guiar el proceso de desarrollo de la solución?
3. ¿Cómo estructurar el proceso de desarrollo de la herramienta de sincronización y réplica de datos?
4. ¿Qué actividades se le aplicarán a la herramienta de sincronización y réplica de datos para la comprobación y validación de la misma?

Para la realización de las tareas trazadas anteriormente, se utilizarán los siguientes **métodos de investigación:**

Métodos Teóricos:

Crean las condiciones para ir más allá de las características fenoménicas y superficiales de la realidad, permiten explicar los hechos y profundizar en las relaciones esenciales y cualidades fundamentales de los procesos, hechos y fenómenos. (Zayas, 1995)

Analítico - Sintético: Permite la descomposición de un todo complejo en sus partes y cualidades. La síntesis, por su parte, establece la unión entre las partes, previamente analizadas y posibilita descubrir relaciones y características generales entre los elementos de la realidad. Se utiliza para realizar un estudio de la bibliografía referente a los diferentes conceptos asociados a la investigación, permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio. (Hernández León, y otros, 2002)

Histórico - Lógico: El método histórico estudia la trayectoria real de los fenómenos y acontecimientos en el transcurso de su historia. El método lógico investiga las leyes generales del funcionamiento y desarrollo de los fenómenos (Hernández León, y otros, 2002)**.** Este método se emplea para analizar la evolución de los conceptos asociados a la réplica y sincronización de bases de datos, acotado en el espacio de tiempo comprendido desde 1995 a la actualidad.

Modelación: Se utiliza para la representación del modelado mediante diagramas. Esto es evidenciado en los artefactos que son generados durante el desarrollo del software, dando un mejor entendimiento de cómo implementar la solución.

Técnica de recopilación de información

Entrevista: Se utiliza con el objetivo de entrevistar a algunos profesionales del proyecto para un mayor dominio del tema a investigar. Además, se emplea con el objetivo de obtener algunas características y funcionalidades a tener en cuenta para el desarrollo de la solución de réplica y sincronización de datos para Xilema-Suria 3.0. Se aplica el tipo de entrevista no estructurada, ya que en ella se trabaja con preguntas abiertas, sin un orden preestablecido, adquiriendo las características de la conversación y permitiendo la espontaneidad.

Capítulo 1: Fundamentos teóricos sobre los sistemas de réplica de datos.

En este capítulo se abordan los conceptos y características fundamentales relacionados con la replicación de datos y los sistemas para la gestión de réplica de datos. Se realizará un estudio de las herramientas de réplica de bases de datos. Se analizarán las tecnologías usadas actualmente para el desarrollo de aplicaciones informáticas posibilitando dar solución al problema de investigación.

1.1 Conceptos asociados al dominio del problema

1.1.1 Base de datos

Varios autores (Pérez Valdes, 2007) y (Alburquerque Arias, 2007) han emitido determinadas valoraciones con respecto a las BDs:

- Serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular. (Pérez Valdes, 2007)
- Colección o depósito de datos, donde los mismos están lógicamente relacionados entre sí, tienen una definición, descripción común, y están estructurados de una forma particular. (Alburquerque Arias, 2007)

Los conceptos presentados anteriormente son asumidos para la presente investigación. Además, se puede agregar que una BD es la estructura donde se almacena la información que caracteriza determinado entorno. Esta información es parte del patrimonio de la entidad que la gestione y es por ello que elementos tales como seguridad, integridad y disponibilidad son priorizados a la hora de definir un esquema de BD. Los datos se almacenan en estructuras denominadas Tablas y sobre estas se pueden realizar diversas operaciones de inserción, actualización, eliminación y funciones, entre otras.

1.1.2 Tablas

Tablas en las bases de datos, se refiere al tipo de modelado de datos, donde se guardan los datos recogidos por un programa. Son utilizadas para organizar y presentar información. Las Tablas se componen de filas y columnas que definen celdas, las que acogen la información en función de las necesidades para las cuales son definidas. (Pérez Porto , et al., 2010)

Otro de los elementos que se pueden encontrar en las Tablas son los Disparadores. Estos se encargan de emitirse para actualizar todos los objetos que se encuentren a la escucha de los cambios que puedan suceder en las Tablas.

1.1.3 Disparadores o *Triggers*

Los Disparadores o *Triggers* son objetos que se asocian con Tablas y se almacenan en la base de datos. Su nombre se deriva por el comportamiento que presentan en su funcionamiento, ya que se ejecutan cuando sucede algún evento sobre las Tablas a las que se encuentra asociado. Los eventos que hacen que se ejecute un *trigger* son operaciones de inserción (*INSERT*), eliminación (*DELETE*) o actualización (*UPDATE*), de los datos de una Tabla. (Loney, et al., 2003)

1.1.4 Base de datos relacional

Una base de datos relacional es una base de datos que se trata como un conjunto de Tablas y se manipula de acuerdo con el modelo de datos relacional. Contiene un conjunto de objetos que se utilizan para almacenar y gestionar los datos, así como para acceder a los mismos. Las estructuras, vistas, índices, funciones, activadores y paquetes son ejemplos de estos objetos. (IBM, 2013)

1.2 Réplica de base de datos

Replicación es mucho más que simplemente mover datos de un lado a otro. Esta técnica demanda de una tecnología extrema. Por lo tanto, el éxito de los sistemas de replicación depende de la tecnología y directrices de un amplio rango de necesidades en la organización. (Heredia, y otros, 2013)

La replicación en el contexto de los SGBD, se entiende como el proceso de copiar y mantener objetos de una base de datos, entiéndase Tablas, Secuencias y/o Disparadores en múltiples bases de datos, de tal forma que los cambios realizados localmente, sean enviados a las bases de datos remotas y aplicados. (Oracle, 2009)

1.2.1 Beneficios de la réplica de datos según (MongoDB, 2011)

- **Disponibilidad:** Es la posibilidad de tener acceso a los recursos y los datos cuando son necesitados.
- **Fiabilidad:** Al haber múltiples copias de los datos disponibles en el sistema, se dispone de un mecanismo excelente de recuperación cuando existan fallos en nodos.
- **Rendimiento:** Se mejora para las transacciones de consulta cuando se introduce la replicación en un sistema que estuviera aquejado de sobrecarga de recursos centralizados.
- **Reducción de la carga:** Modo en que se utiliza la replicación para distribuir datos en ubicaciones remotas.

- **Copia de seguridad:** En condiciones normales, una base de datos replicada de forma correcta es válida como copia de seguridad. Además, se puede realizar copias de seguridad usando un servidor esclavo para así no interferir al servidor maestro.
- **Mejorar la escalabilidad:** Podríamos configurar nuestras aplicaciones para balancear las consultas de lectura (*SELECT*) entre los servidores replicados.

Las réplicas locales constituyen una ayuda especialmente útil cuando se desea trabajar en una computadora que en ocasiones no estará conectada a la red donde se encuentra el servidor en el que reside el recurso.

1.2.2 Modelos de distribución de datos

Existen dos modelos de distribución de datos esencialmente aplicados a la replicación de datos según (Vega, 2007):

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

En la presente investigación se propone el uso del modelo de distribución de los datos Asincrónico. El mismo permitirá que la información se encuentre disponible en los sitios remotos creados como estructura de respaldo de la información.

1.2.3 Entorno de replicación

A continuación, se sintetizan las características de la réplica según la forma de transmisión de los datos definidas por (Tardia, 2011):

- **Maestro-Esclavo (*master-slave*):** En este caso la replicación se realiza en un único sentido, desde un nodo maestro a uno o varios nodos esclavos.

- **Multi-Maestro (*multi-master*):** Se configuran varios nodos como maestros, que pueden replicar las bases de datos entre sí.

El entorno de réplica de los datos que se propone utilizar es el Multi-Maestro por la característica de replicar los datos entre todas las bases de datos que conforman el esquema de respaldo. Igualmente se acopla con la propuesta de utilización del modelo Asincrónico de distribución de los datos.

1.2.4 Captura y almacenamiento de los cambios a replicar

- **Basada en *triggers*:** Se crean una serie de *triggers* en la base de datos, que permiten capturar las operaciones de inserción, actualización y eliminación realizadas sobre las Tablas a replicar.
- **Basada en *logs*:** Se sostiene en la lectura de *logs* de cambios que proporcionan algunos gestores como Oracle Database.

1.3 Administración de las soluciones de réplica de datos

El proceso de administración en cualquier empresa, entidad o proceso resulta de gran importancia porque permite que el flujo de trabajo en cada uno de ellos se realice de forma correcta, sin errores y sin arrastrar consecuencias que puedan afectarlos. Cuando se desarrolla una solución de réplica de datos también se realizan tareas de administración que están dirigidas al proceso de sincronización entre las bases de datos donde se tienen en cuenta aspectos como:

- La forma en que será transmitida la información, si va a ser empujada por el nodo servidor o extraída por el nodo cliente.
- La cantidad de Tablas que son replicadas.
- El estado de los datos que contienen las Tablas replicadas.
- Las acciones que se realizan con los datos como insertar, modificar o eliminar.
- Verificar el registro de réplica de datos.

Además, permiten gestionar cada uno de los componentes de un escenario de sincronización una vez instalada y configurada inicialmente la réplica de datos. Estos escenarios, también conocidos como *Set* de réplica están compuestos por grupos de nodos, enlaces entre los grupos, la rutas que permitirán la transportación de los datos, la forma de envío y captura de los datos y tienen como objetivo garantizar la sincronización entre las bases de datos. Las tareas de administración en todas las soluciones de réplica de datos no se realizan de la misma forma porque dependen en gran medida del tipo de réplica de datos que

se desee implementar, de la herramienta de réplica de datos que se utilice, del entorno de réplica de datos y de las necesidades del negocio implicado. Pero sí están orientadas a:

- Validar datos periódicamente.
- Visualizar los hilos de empuje y tirado de los datos en el envío y captura de los datos.
- Verificar los lotes de datos que entran y salen de un nodo.
- Verificar las propiedades del sistema donde está montada la réplica de datos.
- Gestionar nuevos nodos en la réplica de datos.

1.4 Soluciones existentes

En este epígrafe se realizará el análisis de algunas herramientas de replicación existentes con el objetivo de conocer cómo se manifiestan las tareas de administración y monitorización buscando una tendencia en la forma de administrar y monitorear las soluciones de réplica de datos. A continuación, se describen:

PyReplica

Según (Reingart, 2012) es una herramienta desarrollada en el lenguaje Python, permite réplicas Maestro-Esclavo y Multi-Maestro limitado, funciona de forma asincrónica y especialmente para el gestor PostgreSql. Se caracteriza por ser multiplataforma y fácil de usar, permitiendo:

- La replicación condicional.
- La detección de conflictos.
- El monitoreo de las conexiones.
- Notificaciones vía correo electrónico.

Esta solución no soporta:

- Replicación de DDL [1] automática (pero puede usarse para propagar órdenes DDL a varios servidores).
- Replicación sincrónica.
- Resolución de conflictos, los cuales se pueden evitar con reglas o disparadores.

IBReplicator

[1] ***Data definition language (DLL):*** Lenguaje de definición de datos. Comandos SQL que permiten crear nuevas tablas, campos e índices como: *CREATE*, *DROP* y *ALTER*.

En (IBPhoenix, 2017) se define IBReplicator como un conjunto de herramientas para la implementación y el control de la replicación de bases de datos en todas las versiones de *Firebird*[2] e *InterBase* [3]. IBReplicator es un software que se integra con las BDs, lo que permite a los usuarios reproducir datos sin problemas y de forma transparente entre las BDs en ambos sitios locales y remotos.

Esta herramienta le permite seleccionar qué Tablas y campos van a ser replicados, no solo para ver y editar las configuraciones opcionales; sino que también genera los disparadores necesarios en la base de datos fuente.

La forma en que se realizará el proceso de réplica depende de cómo se desea hacer, por lo que se brinda cierta flexibilidad en la configuración de réplicas. Estas formas son:

- Múltiples BD origen y destino: IBReplicator le permite publicar los datos de muchas bases de datos, así como recibir datos de varias fuentes.
- Flexible en el tiempo: La replicación puede ocurrir ya sea sincrónica o asincrónica, es decir, a petición, a intervalos de tiempo, o en respuesta a eventos de base de datos.
- N-manera: Cada base de datos de destino también puede ser una base de datos fuente a su vez, con la réplica ser controlados por diferentes instancias del servidor de replicación, o incluso por el mismo.
- Resolución de conflictos: IBReplicator proporciona tres formas de manejar los casos en que replican los conflictos de datos con los datos existentes en la BD de destino:
    - **Basada en prioridades:** Las bases de datos pueden tener asignadas prioridades: la base de datos con la prioridad más alta tiene prioridad sobre el resto.
    - **Marca de tiempo:** Ya sea la primer o último cambio que tuvo efecto.
    - **Maestro-esclavo:** La base de datos de origen siempre tiene prioridad.

Reko

En la investigación (Ortiz Noriega, et al., 2010) se plantea que Reko es un sistema implementado en la UCI en el año 2009, la administración y configuración de la réplica de datos se realiza a través de una interfaz web, por lo que es administrable vía remota usando un navegador. Además, permite conocer el estado de los datos transmitidos y puede realizarse en tiempo real a través de la web, así como dar un seguimiento al funcionamiento interno del mecanismo. También permite detectar errores de conexión, mantiene los datos

[2] Sistema de administración de base de datos relacional de código abierto.
[3] Sistema de gestión de bases de datos relacionales.

de la réplica de datos en un estado estable en caso de desconexión. Sincroniza automáticamente los datos entre las bases de datos al restablecerse la conexión y la transferencia de datos de replicación puede realizarse haciendo uso de los protocolos TCP/IP[4], HTTP[5] o por ficheros de forma manual.

Realiza las operaciones de réplica y sincronización de información para las bases de datos creadas en PostgreSQL y Oracle Database.

Aunque el Replicador de Datos Reko brinda un conjunto de funcionalidades, presenta varias limitantes en el proceso de resolución de conflictos, las cuales se desglosan en:

- **Mecanismo de captura:** Durante la detección de conflictos sólo diferencia los conflictos de unicidad y de eliminación de datos. Los demás conflictos los clasifica como conflictos desconocidos.
- **Mecanismo de resolución automática:** Sólo permite realizar dos acciones automáticas sobre los conflictos que se detectan. Ignora los conflictos detectados y en presencia de un conflicto de unicidad permite convertir la acción de inserción que generó el conflicto, en una acción de actualización.

Slony-I

Según (Browne, 2016)**,** es un software de replicación Maestro-Esclavo, que tiene la capacidad de replicar grandes bases de datos. Funciona como maestro a múltiples esclavos y a su vez en cascada, el maestro puede alimentar a un nodo esclavo y este último a otros nodos esclavos de un nivel inferior. Es asincrónica utilizando disparadores para recoger actualizaciones de las Tablas, incluye todas las características necesarias para replicar base de datos grandes con un número razonable de réplicas. Slony-I es un sistema diseñado para su uso en centros de datos y sitios de respaldo, donde el modo normal de operación es que todos los nodos estén disponibles.

Solo soporta el servidor de BD PostgreSql, replica solo entre bases de datos con estructuras iguales y no tiene la capacidad de replicar objetos de gran tamaño. Para la resolución de conflictos establece que se deben resolver automáticamente según las políticas que están establecidas y en caso que no exista una política para un conflicto determinado se detiene el proceso de réplica hasta que manualmente se resuelva el problema. (Browne, 2016)

En (Slony Development Group , 2010) se plantea que Slony-I no es factible para una red inestable; no cuenta con ninguna funcionalidad dentro de ella que le permita detectar el fallo de un determinado nodo.

[4] ***Transmission control protocol (TCP), Internet protocol (IP):*** Protocolo de control de transmisión más el protocolo de internet. Denominación de la familia de protocolos de internet.

[5] ***Hypertext transfer protocol (HTTP):*** Protocolo de transferencia de hipertexto.

SymmetricDS

SymmetricDS es un programa desarrollado en el lenguaje Java, funciona con el controlador JDBC[6], que permite la replicación de datos de forma asincrónica para transmitir los cambios en el entorno de réplica de datos Multi-Maestro atendiendo al ambiente de replicación y sincronización unidireccional o bidireccional. Esta aplicación está diseñada para replicar gran cantidad de bases de datos, trabajar con conexiones de bajo ancho de banda y poder resistir a períodos de inoperatividad de la red (periodos inestables en la conexión). Además, soporta la sincronización entre diferentes plataformas de base de datos (MySQL, MariaDB, Derby, Greenplum, Informix, Interbase, SQLite, Sybase ASE, Redshift, Oracle, SQL Server, PostgreSql, DB2, Firebird, HSQLDB y H2) y puede ser utilizado tanto en el sistema operativo Linux como en Windows. (JumpMind, 2017)

La herramienta cuenta con una aplicación del lado del servidor y otra del lado del cliente, las cuales interactúan entre sí para realizar la réplica de datos. A cada instancia de la aplicación SymmetricDS se le denomina “Nodo” los cuales son inicializados mediante un fichero (*.properties*) y cada Nodo es configurado insertando datos de configuración en una serie de Tablas en la base de datos. Su funcionamiento se basa en el uso de Disparadores y su instalación se realiza en la base de datos para garantizar que los cambios de la información sean capturados. Además, para realizar el transporte de datos utiliza HTTP o HTTPS [7].

Atendiendo a la descripción anterior sus principales características se resumen en:

- **Compatibilidad:** Posee soporte para 16 plataformas de bases de datos importantes siendo la elección perfecta para entornos empresariales heterogéneos.
- **Escalar:** Optimizado para el rendimiento y la escalabilidad. La herramienta SymmetricDS puede replicar miles de bases de datos de forma asíncrona en tiempo casi real.
- **Configuración Flexible:** Configura las Tablas para que se sincronicen de forma unidireccional o bidireccional. Utiliza el filtro horizontal o vertical de subseries de Tablas y combina, filtra o cambia datos con transformaciones de gran alcance.

Como requisitos del sistema que deben existir para poder utilizar esta herramienta se encuentran:

[6] ***Java Database Connectivity (JDBC):*** Conectividad con la base de datos mediante java.
[7] ***Hypertext Transfer Protocol Secure (HTTPS):*** Protocolo seguro de transferencia de hipertexto.

- Requiere *Java Runtime Environment* (JRE por sus siglas en inglés) *Standard Edition* (SE por sus siglas en inglés) o *Java Development Kit* (JDK por sus siglas en inglés) *Standard Edition* (SE por sus siglas en inglés) versión 7.0 o superior.
- Utiliza cualquier base de datos que almacene los datos haciendo uso de disparadores y un controlador JDBC.
- Memoria del sistema: 64 (MB) disponibles.
- Almacenamiento: 256 (MB) Disponibles.

Arquitectura de SymmetricDS

Cada subsistema en el nodo es responsable de parte del movimiento de datos y es controlado a través de la configuración. Los datos fluyen a través del sistema en los siguientes pasos:

- **Capturar** en una Tabla en tiempo de ejecución en la base de datos de origen.
- **Encaminar** para la entrega a los nodos objetivo y agrupar en lotes.
- **Extraer y transformar** en las filas, columnas y valores necesarios para el lote saliente.
- Enviar el **lote saliente** a los nodos de destino.
- Recibir el **lote entrante** en el nodo objetivo.
- **Transformar** en las filas, columnas y valores necesarios para el lote entrante.
- **Cargar** datos y devolver un acuse de recibo al nodo de origen.

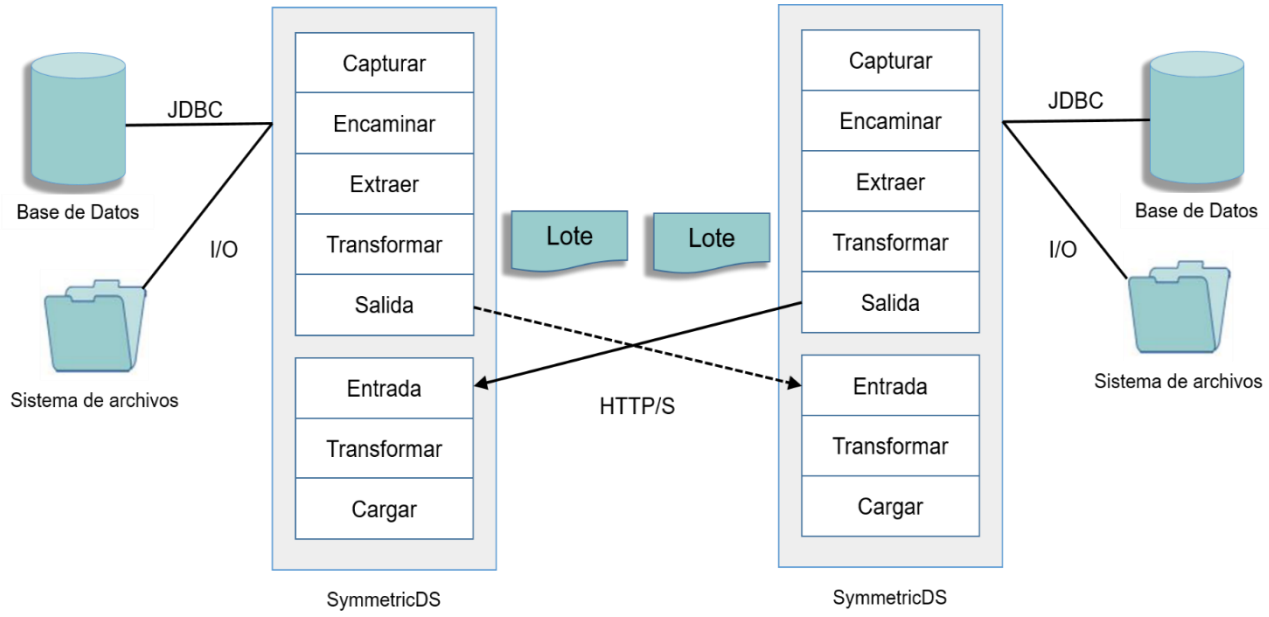


Fig 1: Arquitectura de SymmetricDS según (JumpMind, 2017).

Principales elementos de configuración

Configurar SymmetricDS es el proceso de configuración del escenario de sincronización. El mismo define elementos tales como:

Groups (Grupos)

En SymmetricDS, las reglas de configuración se aplican a grupos de nodos contra nodos individuales. Un grupo es una categorización de nodos con necesidades de sincronización similares.

Group Links: (Enlace entre grupos)

Los enlaces de grupo definen a un nivel alto cómo se mueven los datos en todo el escenario de sincronización. El enlace de grupo define qué grupos de nodos sincronizarán los datos con otros grupos de nodos y dentro de ese intercambio, el grupo de nodos que iniciará la conversación para ese intercambio.

Routers: (Rutas)

Los *routers* viajan en la parte superior de los enlaces de grupo. Mientras que un enlace de grupo especifica que los datos deben ser movidos desde nodos aglutinados en grupos. Definiendo el grupo de nodos origen desde donde salen los datos hacia el grupo de destino. Los enrutadores definen más específicamente qué

datos capturados de un nodo de origen deben enviarse a qué nodos específicos de un grupo de nodos de destino.

Chanels: (Canales)

Una vez que se definen los enlaces de grupo y los enrutadores, se debe completar la configuración para especificar qué datos (Tablas o sistemas de archivos) deben sincronizarse sobre esos enlaces y enrutadores. El siguiente paso para definir qué información específica de la base de datos se traslada para su réplica es definir agrupaciones lógicas para los datos; que son tomadas como canales. Como ejemplo de estas agrupaciones se puede definir un conjunto de Tablas que contienen datos de usuarios, estos pueden agruparse lógicamente en un canal de usuarios. Mientras que las Tablas con los datos de las cámaras, pueden agruparse lógicamente en un canal de cámaras. SymmetricDS crea automáticamente un canal predeterminado llamado *"default",* el que agrupa todas las Tablas; a menos que se creen y se especifiquen otros canales independientes.

1.4.1 Análisis de las soluciones existentes

Después de hacer un estudio de las principales características de cada una de estas herramientas se procede a representar en la Tabla 1 una comparación de las soluciones atendiendo a los parámetros que se identifican como fundamentales en el entorno que va a ser utilizada la propuesta de solución. Los mismos están en función de las necesidades del sistema Xilema Suria 3.0. Los parámetros a evaluar son: servidor de BD, comportamiento de transmisión de datos, plataforma soportada, forma de propagación de los datos, sincronización en tiempo real, resolución automática de conflictos y el trabajo durante periodos de inoperatividad de red.

Tabla 1: Comparativa de las herramientas

	Reko	IBReplicator	SymmetricDS	PyReplica	Slony-I
Servidor de BD soportados					
MariaDB			✓		
Comportamiento de transmisión de los datos					
Maestro-Esclavo			✓	✓	✓

Multi-Maestro	✓	✓	✓	✓	
Plataforma					
Windows	✓	✓	✓	✓	✓
Linux	✓	✓	✓	✓	✓
Forma de propagación de los datos					
Sincrónica		✓		✓	
Asincrónica	✓	✓	✓		✓
Sincronización en tiempo real					
Sí	✓	✓	✓	✓	
No					✓
Resolución automática de conflictos					
Sí	✓	✓	✓	✓	
No					✓
Periodo de desconexión de red					
Sí			✓		
No	✓	✓		✓	✓

Analizando las principales características de funcionamiento de las soluciones estudiadas, se arriba a la selección de SymmetricDS como herramienta a utilizar en la solución del problema de la presente investigación. Esta solución se acopla con las necesidades de sincronización y réplica de datos del sistema Xilema Suria 3.0, pero las tareas de configuración SymmetricDS deben realizarse a través de una terminal de comando. Lo que requiere de un estudio profundo de la herramienta SymmetricDS para su configuración y puesta en explotación. Esto trae como consecuencia que los administradores de la réplica de datos deban

consultar de forma manual los *logs* de SymmetricDS para ver el estado de los procesos de réplica y sincronización, convirtiéndose en una tarea compleja. El principal inconveniente de la herramienta radica en no poseer una interfaz de usuario para configurar el ambiente de respaldo de la información, elemento fundamental que deberá ser desarrollado para su integración final con el sistema. Xilema Suria 3.0.

Conclusiones del capítulo

En el presente capítulo fueron expuestos conceptos indispensables para la comprensión del proceso de salva y respaldo de datos. Además del estudio de las principales herramientas de réplica y sincronización de datos actuales. Dando por vencidas las dos primeras tareas de la investigación y arribando a las siguientes conclusiones:

- Las réplicas de BDs proporcionan importantes beneficios a los sistemas que lo implementan como la disponibilidad, fiabilidad y copia de seguridad de los datos. Elementos que son fundamentales para la integración con el sistema Xilema Suria 3.0.
- Se seleccionó de SymmetricDS como herramienta para el respaldo de datos al brindar soporte a las necesidades de réplica y sincronización automática de los datos de la BD del sistema Xilema Suria 3.0.
- Se detectó la necesidad del desarrollo de una interfaz de usuario para la administración de la herramienta SymmetricDS ya que la misma no posee una manera intuitiva de configuración. Tendrá los elementos necesarios para configurar la herramienta en función de la réplica y sincronización de los datos del sistema Xilema Suria 3.0.

Capítulo 2: Características de la herramienta de sincronización y réplica de los datos para el sistema Xilema Suria 3.0

En el presente capítulo se acotan las herramientas, los lenguajes y las tecnologías utilizadas para el desarrollo de la solución. Además, se realiza la captura de los requisitos mediante una técnica de recopilación de información y se detallan los requisitos no funcionales presentes en el sistema.

2.1 Metodología de desarrollo de software

La metodología de desarrollo de software es un enfoque estructurado para el desarrollo de software que incluye modelos de sistemas, notaciones, reglas, sugerencias de diseño y guías de procesos. En los últimos años el uso de estas ha incidido positivamente en la calidad del software. Las metodologías de desarrollo de software se dividen en dos vertientes: tradicionales o pesadas y metodologías ágiles. (Pressman, 2010)

Las tradicionales llevan una documentación exhaustiva de todo el proyecto. Se utilizan en grandes equipos de desarrollo y necesitan generar una gran cantidad de documentación para mantener controlado el proceso de desarrollo y gestionar la comunicación entre los departamentos en los que puede dividirse el equipo. Las metodologías ágiles son flexibles ante requisitos cambiantes y son utilizadas en equipos pequeños de desarrollo. (Pressman, 2010)

Según (Sánchez, 2015) la metodología de desarrollo a emplearse en los proyectos productivos de la UCI es una variación de la metodología AUP [8]en unión con el modelo CMMI-DEV [9]versión 1.3. En el desarrollo de la solución se utilizará esta metodología para generar todos los artefactos según la línea de desarrollo certificada en la universidad para el proceso productivo. Esto permitirá que los artefactos estén acordes con los generados en el proyecto Sistema de Video Vigilancia Xilema Suria 3.0. El escenario a aplicar según la metodología es el 2, el cual define que todos los proyectos que modelen el negocio con Modelo Conceptual, pueden modelar el sistema con Casos de Uso del Sistema.

2.2 Lenguaje de Modelado Unificado (UML)

Un lenguaje de modelado es un conjunto de símbolos y reglas que están estandarizados y se utilizan para modelar parte de un diseño de software orientado a objetos. Comúnmente son utilizados en combinación con una metodología de desarrollo de software para llegar de una especificación inicial a la implementación. (Larman, 2003)

[8] ***Agile unified process (AUP):*** Proceso unificado ágil.
[9] ***Capability maturity model integration for development*** **(CMMI-DEV):** Integración de modelos de madurez de capacidades para el desarrollo.

Para el desarrollo de la solución, se hará uso de UML en su versión 2.0 por ser el estándar más utilizado para especificar y documentar cualquier sistema de forma precisa. Además, es utilizado por los especialistas del proyecto Sistema de Video Vigilancia Xilema Suria 3.0.

2.3 Herramientas CASE[10]

Modelan la información de negocios cuando ésta se transfiere entre distintas entidades organizativas en el seno de una compañía. El objetivo primordial de las herramientas de esta categoría consiste en representar objetos de datos de negocios, sus relaciones, y ayuda a comprender mejor la forma en que fluyen estos objetos de datos entre distintas zonas de negocio en el seno de la compañía. Estas herramientas proporcionan una ayuda importante cuando se diseñan nuevas estrategias para los sistemas de información y cuando los métodos y sistemas no satisfacen las necesidades de la organización. (Asensio, 2016)

Visual Paradigm 8.0

Visual Paradigm para *UML Standard Edition* (VP-UML SE por sus siglas en inglés) es una plataforma de modelado diseñada para apoyar a los arquitectos de sistemas, desarrolladores y diseñadores UML para acelerar el proceso de análisis y diseño de aplicaciones empresariales complejas. (Visual Paradigm International Ltd., 2010)

Para el desarrollo de esta investigación se utilizará esta herramienta para generar los artefactos ingenieriles a través del ciclo de vida de desarrollo del sistema a desarrollar. También se utilizará por las siguientes características que presenta:

- Disponibilidad en múltiples plataformas, lo cual facilita que la elaboración de diagramas de modelado no dependa del sistema operativo que se tenga instalado en el equipo de trabajo.
- Uso de un lenguaje estándar que es común a todo el equipo de desarrollo facilitando la comunicación, permitiendo la realización de diagramas y modelos durante el proceso de desarrollo.
- Licencia gratuita y comercial, fácil de instalar y actualizar.

2.4 Servidor de Base de Datos MariaDB v10.2.1

Es uno de los servidores de bases de datos más populares del mundo. Está hecho por los desarrolladores originales de MySQL y garantizado para permanecer como código abierto. MariaDB convierte datos en información estructurada en una amplia gama de aplicaciones, que van desde banca a sitios web. Es un reemplazo mejorado para MySQL. MariaDB es rápido, escalable y robusto, con un rico ecosistema de

[10] ***Computer Aided Software Engineering (CASE):*** Ingeniería de Software Asistida por Computación.

motores de almacenamiento, complementos y muchas otras herramientas que lo hacen muy versátil para una amplia variedad de uso. (MariaDB, 2017)

MariaDB es el servidor de bases de datos utilizado por el proyecto Sistema de Video Vigilancia Xilema Suria 3.0 para el manejo de su información. Este servidor se integra con SymmetricDS que es la herramienta de réplica y sincronización de datos seleccionada para la propuesta de solución.

2.5 Entorno de desarrollo integrado (IDE)

Según (NetBeans.org, 2016) un IDE es una aplicación visual que sirve para la construcción de aplicaciones a partir de componentes. Entre sus principales componentes se encuentran un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

Netbeans IDE 8 es un entorno de desarrollo escrito en java para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones web o para dispositivos móviles. Da soporte a varios lenguajes de programación como Java, PHP, C/C++ y HTML. Además, puede instalarse en varios sistemas operativos. Es un producto libre y gratuito sin restricciones de uso. (NetBeans.org, 2016)

Características principales que permitieron emplearlo en la solución:

- Posee un editor de código robusto, con el habitual coloreado y sugerencias de código, acceso a clases pinchando en el código, control de versiones, ubicación de la clase actual, comprobaciones sintácticas, semánticas y plantillas de código.
- Simplifica la gestión de proyectos con el uso de diferentes vistas, asistentes de ayuda y estructurando la visualización de manera ordenada, lo que ayuda en el trabajo diario.
- Contiene herramientas para depurado de errores: el depurador que incluye el IDE es bastante útil para encontrar fallos en el código.

2.6 Lenguaje de programación utilizado

Java

Java se define en (Programación, 2016) como un lenguaje orientado a objetos, de una plataforma independiente, desarrollado por la compañía *Sun Microsystems*. Los pilares en los que se sustenta Java son cinco: la programación orientada a objetos, la posibilidad de ejecutar un mismo programa en diversos sistemas operativos, la inclusión por defecto de soporte para trabajo en red, la opción de ejecutar el código en sistemas remotos de manera segura y la facilidad de uso.

Características de Java según (Pérez Porto, et al., 2013):

- **Es orientado a objetos:** La programación orientada a objetos resulta muy conveniente para la mayoría de las aplicaciones. Entre las ventajas más evidentes que ofrece se encuentra un gran control sobre el código y una mejor organización, dado que basta con escribir una vez los métodos y las propiedades de un objeto, independientemente de la cantidad de veces que se utilicen.
- **Es flexible:** Java es un lenguaje especialmente preparado para la reutilización del código; permite a sus usuarios tomar un programa que hayan desarrollado tiempo atrás y actualizarlo con mucha facilidad, sea que necesiten agregar funciones o adaptarlo a un nuevo entorno.
- **Funciona en cualquier plataforma:** A diferencia de los programas que requieren de versiones específicas para cada sistema operativo (tales como Windows o Mac), las aplicaciones desarrolladas en Java funcionan en cualquier entorno, dado que no es el sistema quien las ejecuta, sino la máquina virtual (conocida como *Java Virtual Machine* o JVM).
- **Integración con SymmetricDS:** Los archivos de configuración de los nodos para SymmetricDS son del tipo (.*properties*), nativo del propio lenguaje, el cual presenta funciones predefinidas para el trabajo con este tipo de archivos.

2.7 Modelo de dominio

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan los conceptos que existen o los eventos que suceden en el entorno en el que trabaja el sistema. (Pressman, 2010)

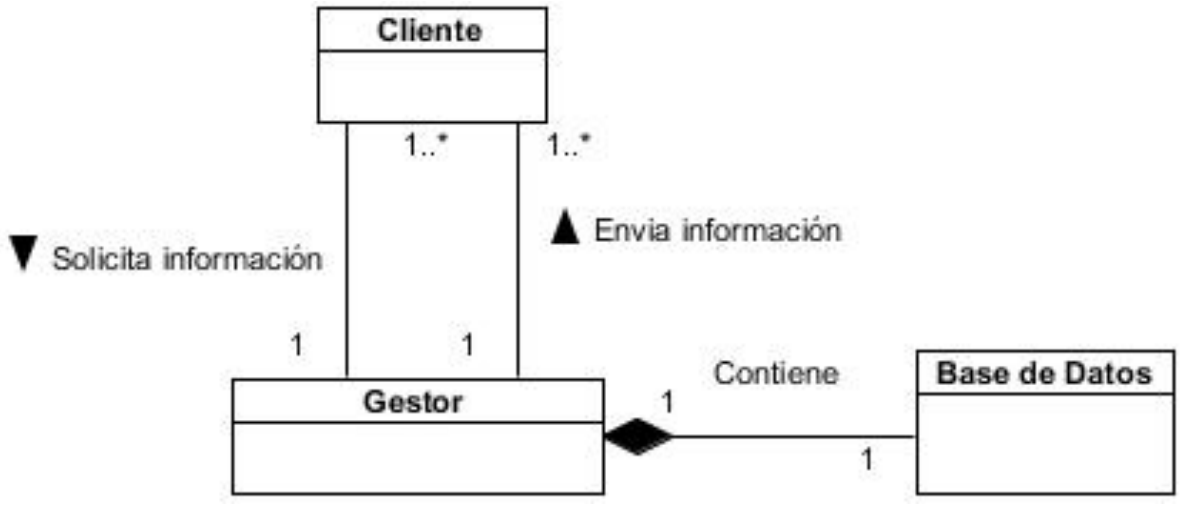


Fig 2: Modelo de Dominio.

2.7.1 Descripción de los conceptos que intervienen en el modelo de dominio

- **Cliente:** Representa la Interfaz General, de Administración, Visor, Visor APK[11] y Recuperador.
- **Servidor de administración:** Maneja las operaciones de análisis, grabación y almacenamiento.
- **Base de datos:** Contiene información necesaria para el funcionamiento del sistema.

2.7.2 Especificación de requisitos

La especificación de requisitos en el proceso de desarrollo del software es de vital importancia. Tener los requisitos bien claros y definidos permite comprender desde un inicio la línea a seguir en el desarrollo y así garantizar la eficiencia y calidad del software. (Pressman, 2010)

2.7.3 Requisitos funcionales y Casos de Uso

Para poder identificar qué debe hacer el sistema y entender su funcionamiento, es fundamental conocer los requisitos funcionales que el mismo debe cumplir. La técnica utilizada para la obtención de requisitos fue la entrevista realizada a los especialistas del proyecto Sistema de Video Vigilancia Xilema Suria 3.0.

A continuación, se muestran los requisitos funcionales (RF) agrupados por caso de uso (CU):

CU1: Gestionar nodo de réplica y sincronización

- **Adicionar nodo de réplica y sincronización:** El sistema debe permitir adicionar un nodo para la réplica y sincronización de datos.
- **Editar nodo de réplica y sincronización:** El sistema debe permitir modificar un nodo de réplica y sincronización de datos.
- **Eliminar nodo de réplica y sincronización:** El sistema debe permitir eliminar un nodo de réplica y sincronización de datos.
- **Describir nodo de réplica y sincronización:** El sistema debe permitir mostrar una descripción del nodo con sus principales atributos (nombre, id, tipo y estado).

CU2: Administrar servicio de réplica y sincronización

- **Iniciar servicio de réplica y sincronización:** El sistema debe permitir iniciar el servicio de réplica y sincronización de datos.
- **Detener servicio de réplica y sincronización:** El sistema debe permitir detener el servicio de réplica y sincronización de datos.

[11] ***Android application package (APK):*** Paquete de aplicación de Android.

- **Comprobar estado del servicio de réplica y sincronización:** El sistema debe permitir consultar el estado de réplica y sincronización de datos.

CU3: Establecer eventos del servicio de réplica y sincronización

- **Visualizar eventos del servicio de réplica y sincronización:** El sistema debe permitir visualizar información del servicio de réplica y sincronización de datos.
- **Limpiar registro de eventos del servicio de réplica y sincronización:** El sistema debe permitir borrar los eventos del servicio de réplica y sincronización de datos mostrados en pantalla.
- **Copiar registro de eventos del servicio de réplica y sincronización:** El sistema debe permitir copiar el registro de eventos del servicio de réplica y sincronización de datos.

CU4: Registrar nodo en el servidor

- **Registrar nodo:** El sistema debe permitir registrar un nodo en el servidor maestro.

2.7.4 Requisitos no funcionales.

Los requisitos no funcionales (RNF) detallan las propiedades o cualidades que el producto debe tener. Estos se encuentran separados en las siguientes categorías según (Pressman, 2010):

RNF1: Presentación: La interfaz debe resultar fácil de utilizar por parte del usuario e intuitiva.

RNF2: Software: El equipo servidor deberá tener instalado el software que se especifica a continuación.

- Sistema Operativo Microsoft Windows 10.
- Sistema Gestor de Base de Datos MariaDB v 10.2.
- Java SE Development Kit (JDK) versión 7.0 o superior.

RNF3: Hardware: El equipo servidor deberá cumplir con los requisitos mínimos que se especifican a continuación.

- Procesador: Core 2 Duo a 2.2 GHz o superior.
- Memoria RAM: 2 GB

2.7.5 Casos de uso del sistema

Los casos de uso son un conjunto de escenarios que identifican una línea de utilización para el sistema que va a ser desarrollado, facilitando una descripción de cómo se usará el sistema. Un diagrama de caso de uso muestra la relación entre los actores del sistema y los casos de uso. (Pressman, 2010)

Diagrama de casos de uso del sistema

El Diagrama de Casos de Uso refleja cómo los actores interactúan con los casos de uso.

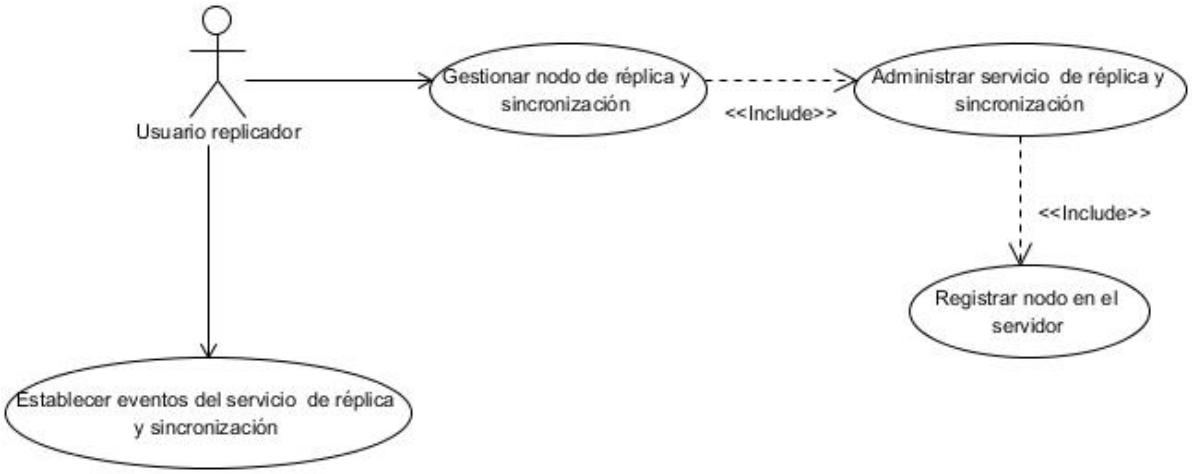


Fig 1: Diagrama de Casos de Uso.

Descripción de los actores del sistema

Los actores del sistema intercambian información con él, aunque no forman parte de este. Pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado. A continuación, se muestra en la Tabla 2 el actor y la justificación que tienen en el sistema.

Tabla 2: Descripción del actor del sistema

Actor	Descripción
Usuario replicador	Es la persona encargada de la administración y control del sistema; tiene acceso a todas las funcionalidades.

Descripción de los casos de uso del sistema

Una descripción amplia de los casos de uso del sistema, permite realizar el diseño e implementación de la solución. A continuación, se describe en la Tabla 3 el caso de uso principal del sistema: Gestionar nodo de réplica y sincronización.

Tabla 3: Descripción de casos de uso Gestionar nodo de réplica y sincronización.

Objetivo	Gestionar nodo de réplica y sincronización: Adicionar, editar, describir y eliminar un nodo de réplica y sincronización.
Actores	Usuario replicador (inicia): Gestionar nodo de réplica y sincronización.
Resumen	El caso de uso inicia cuando el actor desea adicionar, editar, describir o eliminar determinado nodo de réplica y sincronización en el sistema. Termina este caso de uso

	cuando se realiza alguna acción de gestión de nodos de réplica y sincronización.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	
Postcondiciones	Se adicionó, editó, describió o eliminó un nodo de réplica y sincronización en el sistema.
Referencia	**RF:** Adicionar nodo de réplica y sincronización. **RF:** Editar nodo de réplica y sincronización. **RF:** Eliminar nodo de réplica y sincronización. **RF:** Describir nodo de réplica y sincronización.

Flujo de eventos

Flujo básico Gestionar nodo de réplica y sincronización

	Actor	**Sistema**
1.	Inicia la interfaz principal de la aplicación.	
2.		Muestra una interfaz que permite realizar varias acciones: Adicionar nodo de réplica y sincronización. **Ver Paso 3:** Flujo Básico Gestionar nodo de réplica y sincronización. Editar nodo de réplica y sincronización. **Ver Sección 1:** Editar nodo de réplica y sincronización. Eliminar nodo de réplica y sincronización. **Ver Sección 2:** Eliminar nodo de réplica y sincronización.
3.	Presiona el botón Agregar.	
4.		Muestra la interfaz con los campos necesarios para adicionar un nodo de réplica y sincronización: Parámetros del nodo:

		Id externo: Campo numérico de 3 dígitos. (obligatorio) **Nombre:** Campo de texto. (obligatorio) **Tiempo de envío de datos:** Campo numérico de valores superiores a 1. (obligatorio) **Hacer maestro:** Casilla de selección. – **Observación:** Por defecto el sistema selecciona el nodo tipo “Nodo maestro”. Si el usuario deselecciona esta opción. **Ver Sección 3: Establecer tipo de nodo.** Parámetros para la conexión con la base de datos: **Tipo:** Lista de selección. **Nombre:** Campo de texto. (obligatorio) **Ubicación:** Campo de texto. (obligatorio) – **Observación:** Por defecto el sistema llena este campo con el nombre de la PC. **Usuario:** Campo de texto. (obligatorio) **Contraseña:** Campo de texto protegido. (obligatorio)
5.	Adiciona datos a los campos requeridos.	
6.	Presiona el botón Agregar.	
7.		Valida los datos de entrada.
8.		Crea el fichero de configuración del nodo. Adiciona las tablas necesarias del nodo creado en la BD. Carga el fichero (.*sql*) de configuración en la base de datos.
9.		Regresa a la interfaz principal. Visualiza la información del nodo creado. Termina el CU.
Flujo Alterno		

6a Presiona el botón Cancelar		
	Actor	**Sistema**
7a	Presiona el botón Cancelar	
8a		Cancela la acción. Muestra la interfaz principal del sistema. Termina el CU.
Flujo Alterno		
7a Existen campos vacíos.		
	Actor	**Sistema**
8a	El usuario no inserta todos los campos obligatorios.	
9a		Muestra el mensaje de error: “Por favor, llene todos los campos”.
Flujo Alterno		
7b Existen datos incorrectos.		
	Actor	**Sistema**
8b	El usuario inserta datos incorrectos en el campo Frecuencia.	
9b		Muestra el mensaje de error: “El tiempo de envío de datos no puede ser inferior a (2)”.
Sección 1 Editar nodo de réplica y sincronización		
Flujo básico Editar nodo de réplica y sincronización		
	Actor	**Sistema**
1.	Selecciona el nodo que desee editar.	
2.	Presiona el botón Editar.	
3.		El sistema carga la información de nodo y muestra la interfaz con los datos para editar: Parámetros del nodo: **Tiempo de envío de datos:** campo numérico de valores superiores a 1. (obligatorio)

		Hacer maestro: Casilla de selección. – **Observación:** Por defecto el sistema muestra la casilla seleccionada si el nodo es de tipo “Nodo maestro” y deseleccionada si es del tipo “Nodo”. Parámetros para la conexión con la base de datos: **Tipo:** Lista de selección. **Nombre:** Campo de texto. (obligatorio) **Ubicación:** Campo de texto. (obligatorio) **Usuario:** Campo de texto. (obligatorio) **Contraseña:** Campo de texto protegido. (obligatorio) – **Observación:** Por defecto el sistema muestra este campo vacío.
4.	Modifica los campos a editar	
5.	Presiona el botón Aplicar.	
4.		Valida los datos de entrada.
5.		Actualiza el fichero de configuración del nodo modificado. Adiciona las tablas necesarias del nodo creado en la BD.
6.		Muestra el mensaje: “Los cambios realizados tendrán efecto en el próximo inicio de SymmetricDS” Muestra la interfaz principal. Termina el CU.
Flujo Alterno		
5a Presiona el botón Cancelar		
	Actor	**Sistema**
6a	Presiona el botón Cancelar	
7a		Cancela la acción. Muestra la interfaz principal del sistema. Termina el CU.
Flujo Alterno		

4a Existen campos vacíos.		
	Actor	**Sistema**
5a	No inserta todos los campos obligatorios.	
6a		Muestra el mensaje de error: “Por favor, llene todos los campos”
Flujo Alterno		
4b Existen datos incorrectos.		
	Actor	**Sistema**
5b	Inserta datos incorrectos	
6b		Muestra el mensaje de error: “El tiempo de envío de datos no puede ser inferior a (2)”.
Sección 2 Eliminar nodo de réplica y sincronización		
Flujo básico Eliminar nodo de réplica y sincronización		
	Actor	**Sistema**
1.	Presiona el botón Eliminar.	
2.		Muestra mensaje de confirmación: “¿Está seguro que desea eliminar el nodo?”
3.	Acepta el mensaje de confirmación	
4.		Elimina las tablas de la BD del nodo seleccionado. Elimina el archivo de configuración del nodo.
5.		Muestra los eventos en la consola de eventos. Termina el CU
Flujo Alterno		
3a Presiona el botón No		
	Actor	**Sistema**
4a	Presiona el botón No	
5a		Cancela la acción. Muestra la interfaz principal del sistema. Termina el CU.

Relaciones	**CU Incluidos**	No existen.
	CU Extendidos	No existen.

Conclusiones del capítulo

En este capítulo se abordaron diferentes elementos que sustentan la descripción de la solución propuesta, dándole a su vez cumplimiento a varias de las tareas de la investigación planteadas. Arribándose finalmente, a las siguientes conclusiones:

- La selección de AUP-UCI como metodología sirvió como guía para documentar el proceso de desarrollo del sistema. Generando todos los artefactos de documentación acorde con los utilizados en el proyecto. Esto facilita la comprensión de la solución desarrollada para futuras versiones o el mantenimiento de la aplicación.
- El estudio realizado de las tecnologías y lenguajes a utilizar, permitió determinar la base tecnológica necesaria para desarrollar el sistema en cuestión. Definiendo Java como lenguaje de programación junto con Netbeans como IDE. Esta combinación facilita la integración de ambas tecnologías en función de editar los archivos de configuración de los nodos de réplica y sincronización de datos de SymmetricDS.
- La utilización de la técnica de obtención de información permitió obtener requisitos funcionales y no funcionales del sistema para su posterior implementación. Identificándose 11 requisitos funcionales, agrupados en 4 casos de usos. Igualmente se identificaron 3 requisitos no funcionales que aportaron características fundamentales en la descripción de los elementos de instalación y configuración de la solución.

Capítulo 3: Diseño e implementación de la solución propuesta

En el siguiente capítulo se describen los elementos relacionados con el diseño e implementación. Se definen los patrones de diseño y arquitectónico. Además, de los estándares de codificación y las pruebas realizadas al sistema.

Propuesta de Solución:

Con el objetivo de darle cumplimiento al objetivo general de la presente investigación se propone como solución al problema general, la utilización de la herramienta de sincronización de datos SymmetricDS en su versión 3.8.12 para la realización del proceso de réplica y sincronización de datos de la BD del sistema Xilema Suria 3.0. Además del desarrollo de una interfaz de administración con entorno de escritorio, partiendo de la inconveniente de que solo es posible la administración de la herramienta mediante un terminal o consola. Para el despliegue de la solución se tendrán en cuenta los siguientes aspectos:

- Todos los nodos creados estarán en un mismo grupo, permitiendo que la forma de transmisión de los datos sea la misma entre los distintos nodos presentes en el sistema.
- La comunicación entre nodos será bajo los parámetros de configuración estándares que propone SymmetricDS, proponiendo como protocolo de comunicación HTTP.
- Los nodos y bases de datos estarán en las PCs servidores del sistema Xilema Suria 3.0.
- El archivo ejecutable que muestra la interfaz a desarrollar estará en un directorio junto a SymmetricDS.

Parámetros de configuración para la sincronización

- **Group Links:** (Enlaces entre grupos) Valor por defecto *Push* [P]. Indica que los nodos del grupo de nodos de origen iniciarán la comunicación a través de un HTTP PUT[12] y enviarán datos a los nodos del grupo de nodos de destino.
- **Routers:** (Rutas) Valor por defecto [*default*]. Un enrutador que envía todos los datos capturados a todos los nodos que pertenecen al grupo de nodos objetivo definido en el enrutador.
- **Channels:** (Canales) Valor por defecto [*default*]. Todos los cambios que ocurren en una transacción serán agrupados juntos. Varias transacciones se procesarán por lotes y se mezclarán juntas hasta que no haya más datos que se envíen o se alcance el tamaño *max_batch_to_send* [100] o *max_batch_size* [1000].

[12] ***Request method for http (HTTP PUT):*** Método de petición de http. El método PUT actualiza las propiedades literales y las propiedades de recurso local, y suprime las propiedades de recurso local que no están incluidas en la petición. (IBM, 2016)

Parámetros de configuración inicial

- **auto.sync.configuration:** Valor por defecto [*true*]. Captura y envía los cambios en la configuración hacia los nodos clientes.
- **job.push.period.time.ms:** Valor definido por el usuario. Define el intervalo de tiempo necesario para ejecutar el trabajo de envío de datos.
- **initial.load.create.first:** Valor por defecto [*true*]. Permite crear las tablas e índices en la base de datos objetivo, antes de iniciar la carga inicial de datos.

3.1 Arquitectura de software

La arquitectura de software de un programa de computo es la estructura o las estructuras del sistema, que incluyen los componentes del software, las propiedades visibles externamente de sus componentes y las relaciones entre ellos. (Pressman, 2010)

Según (Pressman, 2010), la arquitectura no es el software operativo, sino una representación que permite que un ingeniero de software:

- Analice la efectividad del diseño para cumplir con los requisitos establecidos.
- Considere opciones arquitectónicas en una etapa en la que aún resulta relativamente fácil hacer cambios al diseño.
- Reduzca los riesgos asociados a la construcción del software.

3.1.1 Patrón arquitectónico N-Capas

El patrón arquitectónico en capas pertenece a la familia del estilo en llamada y retorno, donde cada capa proporciona servicios a la capa superior y se sirve de las prestaciones que le brinda la inferior. Al dividir un sistema en N-capas, cada capa puede tratarse de forma independiente, sin tener que conocer los detalles de las demás. La división de un sistema en capas facilita el diseño modular, la comunicación entre capas está basada en una abstracción que proporciona un bajo acoplamiento entre ellas y con alta cohesión internamente, lo cual posibilita variar de una forma sencilla diferentes implementaciones o combinaciones de capas. (Barroso, 2010)

Arquitectura 2 capas:

Para el desarrollo la herramienta de réplica se definió una arquitectura de dos capas como una variante del patrón N-capas. Las mismas se denominan capa de presentación y capa de lógica de negocio. La capa de presentación es la que interactúa con el usuario, esta capa es la encargada de comunicar y capturar la

información (Pressman, 2005). La capa de lógica de negocio es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso (Pressman, 2005).

3.1.2 Diagrama de clases del diseño

El diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema. Este muestra sus clases, atributos y las relaciones entre ellos. Son utilizados durante el proceso de diseño. A continuación, en la figura 3 se representa el diagrama de clases del diseño del sistema.

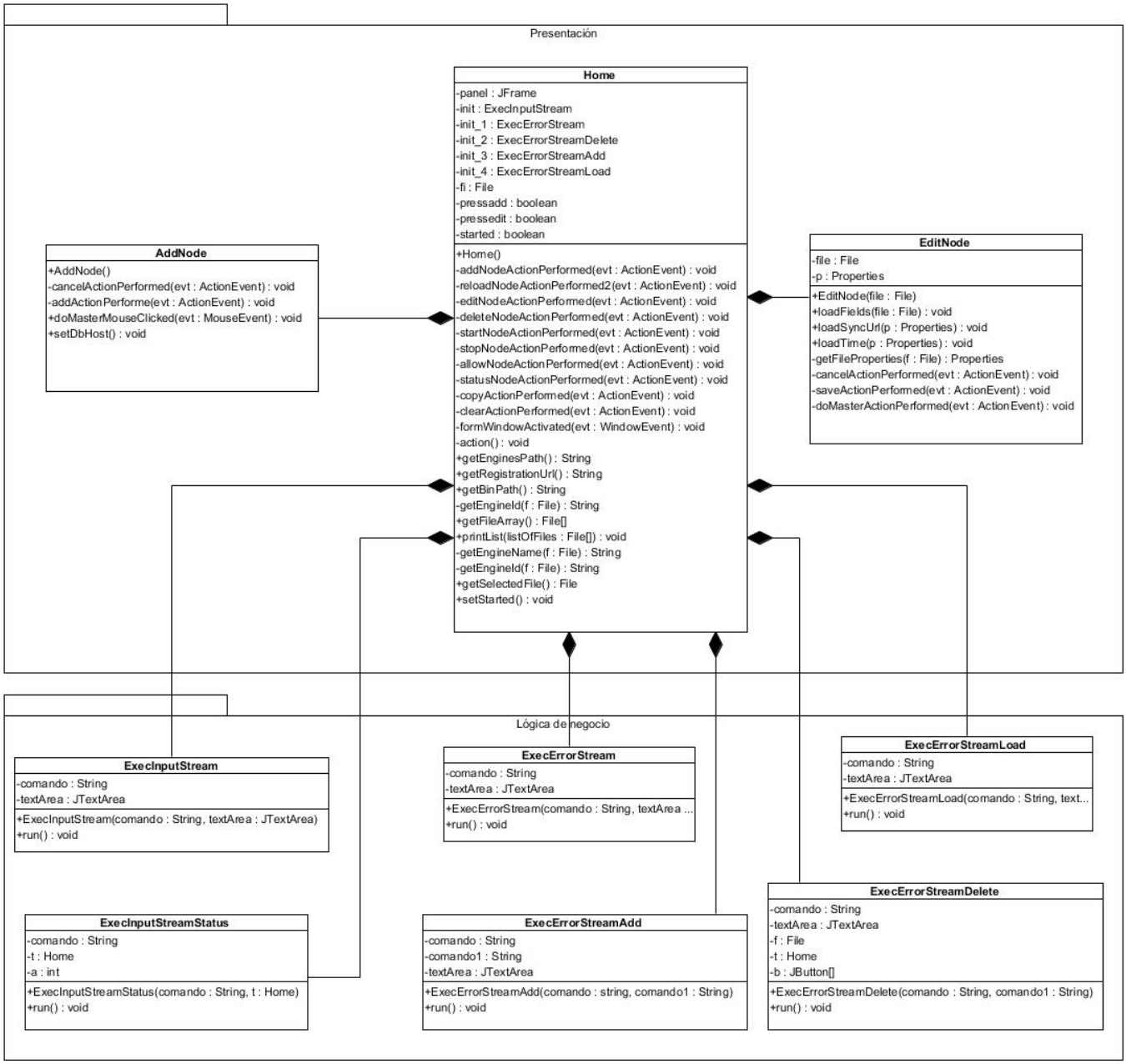


Fig 3: Diagrama de clases del diseño.

3.2 Patrones

Un patrón detalla un problema que ocurre una y otra vez en el ambiente, y luego describe el núcleo de la solución a ese problema, de tal manera que se puede usar esa solución un millón de veces más, sin repetir lo mismo dos veces. (Kruchten, 2000)

Los patrones son utilizados en el desarrollo del software para establecer una manera de organizar y estructurar el desarrollo. Estos son una guía para realizar algunas acciones dentro del proceso de desarrollo, especificando un conjunto de subsistemas predefinidos y las funcionalidades de cada uno. Además facilitan la comprensión de la situación en que se encuentre algún problema que se quiera solucionar. (Kruchten, 2000)

3.2.1 Patrones de diseño.

Un patrón de diseño constituye una solución estándar para un problema común de programación en el desarrollo del software. Además, es una técnica muy eficaz para flexibilizar el código haciéndolo satisfacer ciertos criterios, así como permite una manera más práctica de describir ciertos aspectos de la organización de un programa. (Gamma, 2013)

Los patrones GRASP[13] describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (Larman, 2004)

A continuación, se definen según (Larman, 2004) los patrones GRASP que se usaron en el desarrollo de la aplicación.

- **Experto:** Las responsabilidades deben ser asignadas a las clases que poseen la información para realizar dicha responsabilidad. El sistema hace uso de este patrón y se evidencia cuando se desea obtener toda la información de los nodos creados, ya que la única clase con la responsabilidad de conocer esta información es *Home.java.*
- **Creador:** Se utiliza para asignarle a una clase la responsabilidad de crear una instancia de otra. Dentro del sistema este patrón se evidencia en la acción crear objetos de los nodos. Ejemplo de ello es, la clase *Home.java*.
- **Controlador:** Es el encargado de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Se evidencia el uso de este patrón en la herramienta, ya que para cada acción que se genere en la misma, existe una clase controladora llamada *Home.java*, con la responsabilidad de manejar todos los eventos del sistema, es decir, obtener y devolver una respuesta.
- **Alta Cohesión:** Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Significa que las clases del sistema tienen asignadas solo las responsabilidades que les corresponde y mantienen una estrecha relación con el

[13] ***General Responsibility Assignment Software Patterns (GRASP):*** Patrones generales de software para la asignación de responsabilidades.

resto de las clases, es decir, una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas. Ejemplo de ello en la aplicación es la clase *Home.java.*

- **Bajo Acoplamiento:** Determina el nivel de dependencia de una clase con respecto a otras. Una clase con bajo acoplamiento no depende de muchas otras. Ejemplo de ello en la aplicación es la clase *AddNode.java.*

Los patrones GoF[14] describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad. Permiten crear grupos de objetos para ayudarnos a realizar tareas complejas. (Tello, 2014)

- **Singleton:** Asegura que una clase posea una sola instancia y proporciona un punto de acceso global a ella. Este patrón se ve evidenciado en la clase *Home.java*, debido a que siempre se hace una sola instancia de esta clase.

3.3 Modelo de implementación

En este modelo se describe cómo se implementan en términos de componentes, ficheros de código fuente o ejecutables, los elementos del modelo, tales como las clases, entre otros. Este modelo describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje de programación utilizado, y cómo dependen los componentes unos de otros. Se realizan con el objetivo de poseer una vista de forma general del sistema a partir de las dependencias e integraciones de los componentes y módulos. (Pressman, 2010)

A continuación, se muestra el diagrama de componentes de código ejecutable de la herramienta de réplica y sincronización:

[14] ***Gang of Four (GOF):*** Pandilla de los Cuatro.

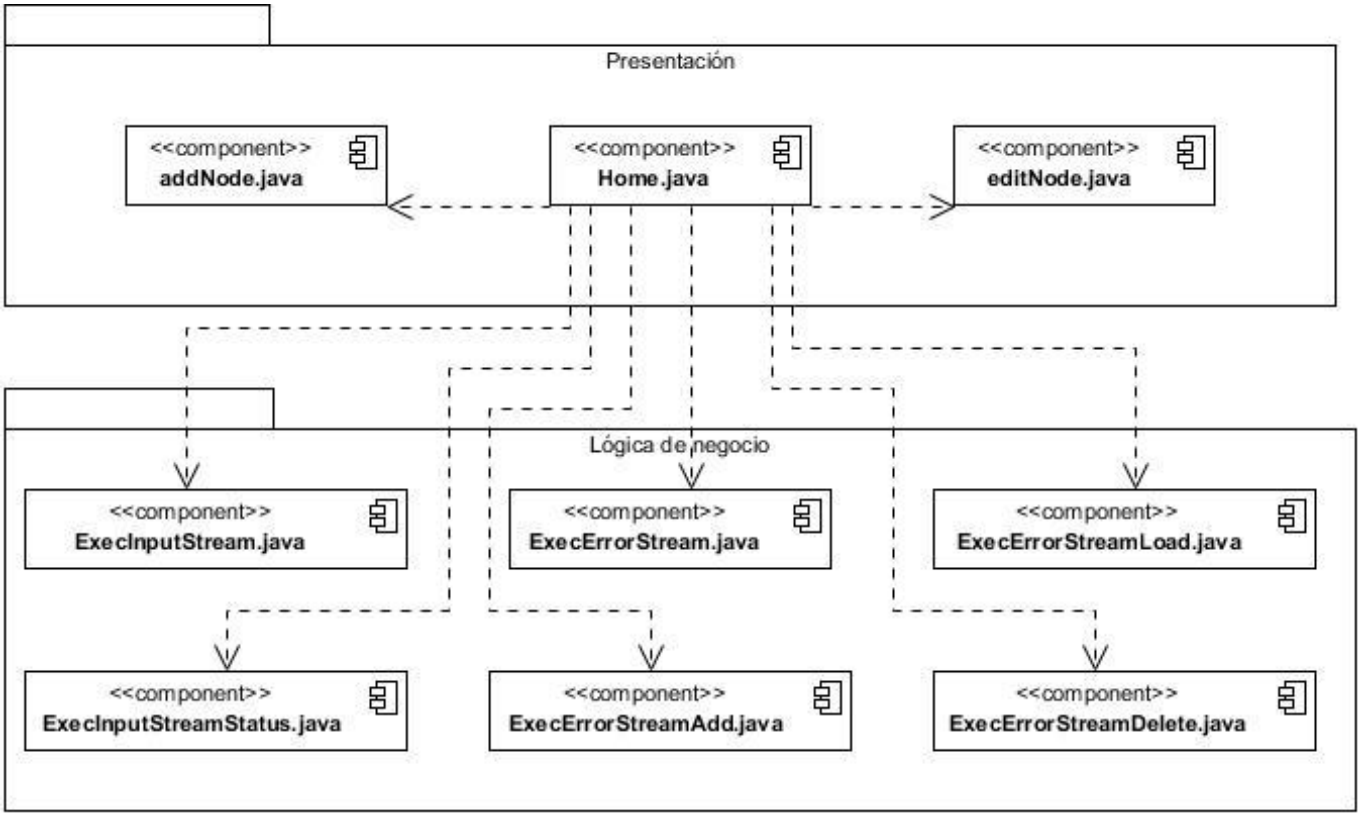


Fig 4: Diagrama de componentes.

3.4 Diagrama de despliegue

La vista de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los vínculos de comunicación entre ellos y las instancias de los componentes. El diagrama está compuesto por nodos, dispositivos y conectores. El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento y las conexiones entre estos elementos en el sistema. (Pressman, 2010)

A continuación, se presenta el modelo de despliegue para el sistema.

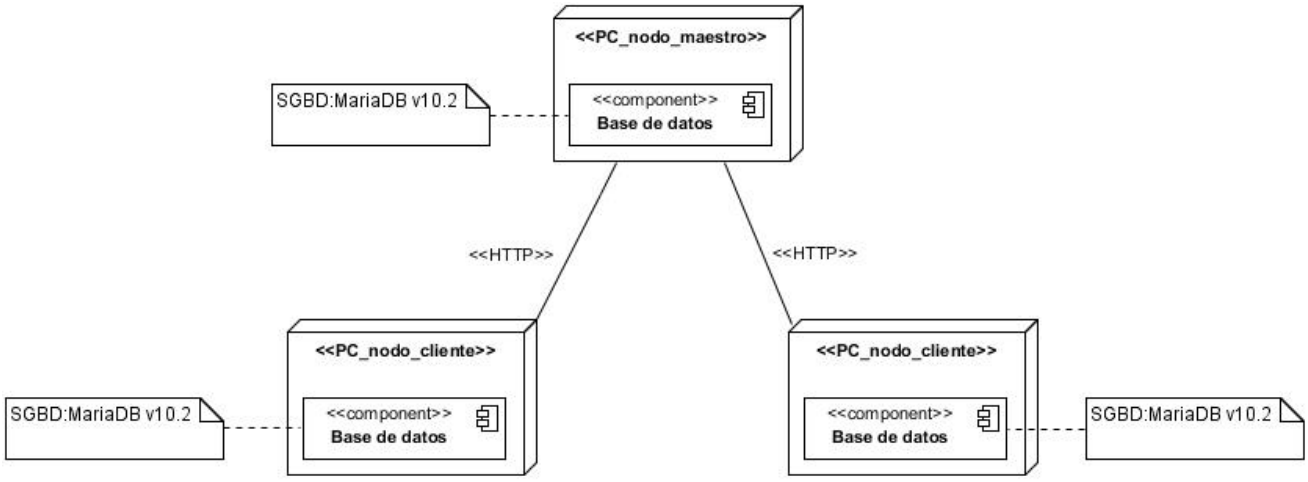


Fig 5: Diagrama de despliegue.

PC_node_maestro: Representa la computadora donde estará configurado el nodo maestro para el proceso de réplica. Este nodo será único en todo el sistema.

PC_nodo_cliente: Representa la computadora donde estará configurado el nodo cliente para el proceso de réplica. Podrá haber más de un nodo cliente en el sistema.

Base de datos: El componente representa cada base de datos asociada a un nodo de réplica.

Nota: La comunicación entre nodos se realiza mediante el protocolo de comunicación HTTP, parámetro tomado de la configuración por defecto de SymmetricDS.

3.4.1 Flujo de información

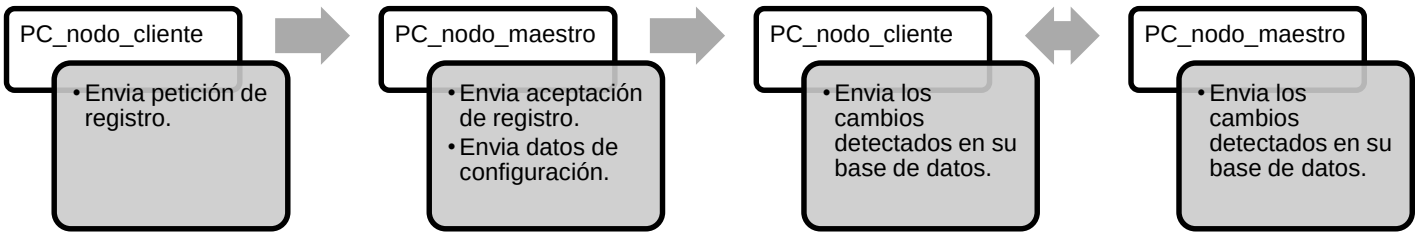


Fig 6: Flujo de información.

3.5 Entorno de réplica y sincronización

El proceso de réplica y sincronización será realizado a la base de datos del sistema Xilema Suria 3.0. La misma está descrita en (Jiménez, 2016) donde especifican todos los elementos que la componen y su relación entre ellos.

Las Tablas “*rule*” y “*autonomyrule*” no entrarán dentro del proceso de réplica y sincronización debido que las mismas almacenan reglas que solo pueden ser aplicadas a un Servidor de Administración de Xilema Suria 3.0 en específico. Trayendo como consecuencia que si estas reglas fueran distribuidas por todas las BDs se afectaría la configuración de cada Servidor de Administración debido a que se aplican las configuraciones sin distinguir el servidor al que pertenecen. La herramienta SymmetricDS utiliza las instancias de tiempos en que ocurren los cambios en la BDs para el proceso de réplica y sincronización, así como para la resolución de conflictos. Debido a esta característica, las computadoras en que se despliegen los nodos de respaldo deberán estar conectadas a un servidor de nombre de dominio, de donde se obtendrá la hora.

3.6 Estándar de codificación

Los estándares de codificación no están enfocados a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. (Pressman, 2010)

Si bien los programadores deben implementar un estándar de forma prudente, debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Además, si se aplica de forma continua un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas y, posteriormente, se efectúan revisiones del código de rutinas, existen posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener. (Pressman, 2010)

Estilos de codificación empleados:

Durante la implementación del software fue necesario utilizar algunos estilos de codificación en busca de un estándar que aportara una mayor organización al código.

Establecer un modo común para comentar el código, utilizar separadores, línea, espacios en blanco y márgenes, de forma tal que sea comprensible el código, siguiendo las pautas que a continuación se muestran:

- Comentar al inicio de la clase o función especificando el objetivo de la misma.
- Dejar una línea en blanco antes y después de la declaración de una clase o de una estructura y de la implementación de una función.
- Evitar comentar cada línea de código. Cuando el comentario se aplica a un grupo de instrucciones debe estar seguido de una línea en blanco. En caso de que se necesite comentar una sola instrucción se suprime la línea en blanco o se escribe a continuación de la instrucción.

```
// Devuelve el valos del parametro engine.id dado en archivo
private String getEngineId(File f){...17 lines }

// Devuelve el archivo correspondienta al elemento seleccionado en Node List
private File getSelectedFile() {...4 lines }
```

Fig 7: Ejemplo de comentarios de código

Para la definición de las clases y métodos en el código de la aplicación es utilizado el estándar *CamelCase*. Este es un estilo de escritura que se aplica a frases o palabras compuestas.

- ***UpperCamelCase*:** Cuando la primera letra de cada una de las palabras es mayúscula. Este estándar se utilizó para los nombres de las clases.

```
import javax.swing.JOptionPane;

/**
 *
 * @author Alejandro
 */
public class EditNode extends javax.swing.JFrame {
private File file;
private Properties p;
```

Fig 8: Ejemplo de *UpperCamelCase*.

- ***lowerCamelCase:*** Igual que la anterior con la excepción de que la primera letra es minúscula. Este estándar se utilizó para los nombres de los métodos.

```
private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {...10 lines }

public static String getEnginesPath() {...3 lines }

public static String getBinPath() {...3 lines }

public static File[] getFileArray() {...8 lines }

public void printList(File[] listOfFiles) {...12 lines }

private String getEngineName(File f){...17 lines }
```

Fig 9: Ejemplo de *lowerCamelCase*.

3.7 Interfaces de la solución propuesta.

La interfaz de una aplicación permite el flujo de información entre el usuario y el sistema. A continuación, se muestran algunas interfaces del sistema.

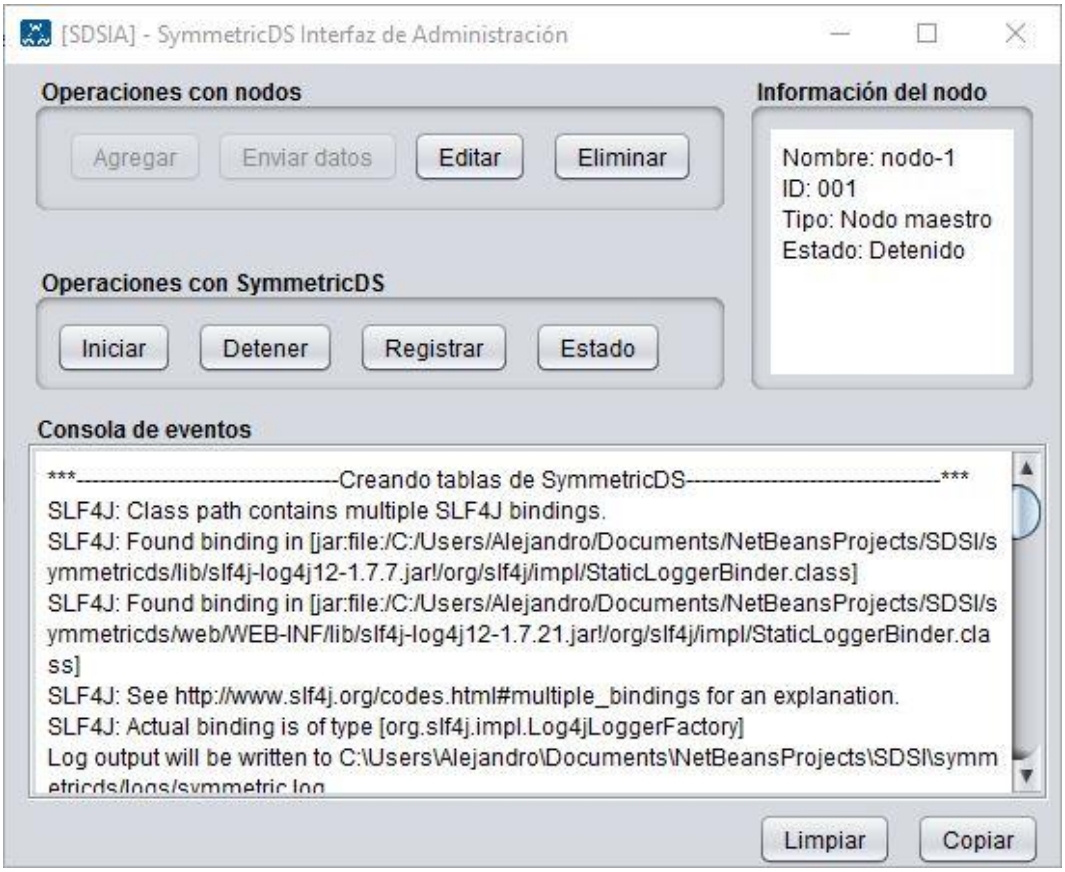


Fig 10: Interfaz principal.

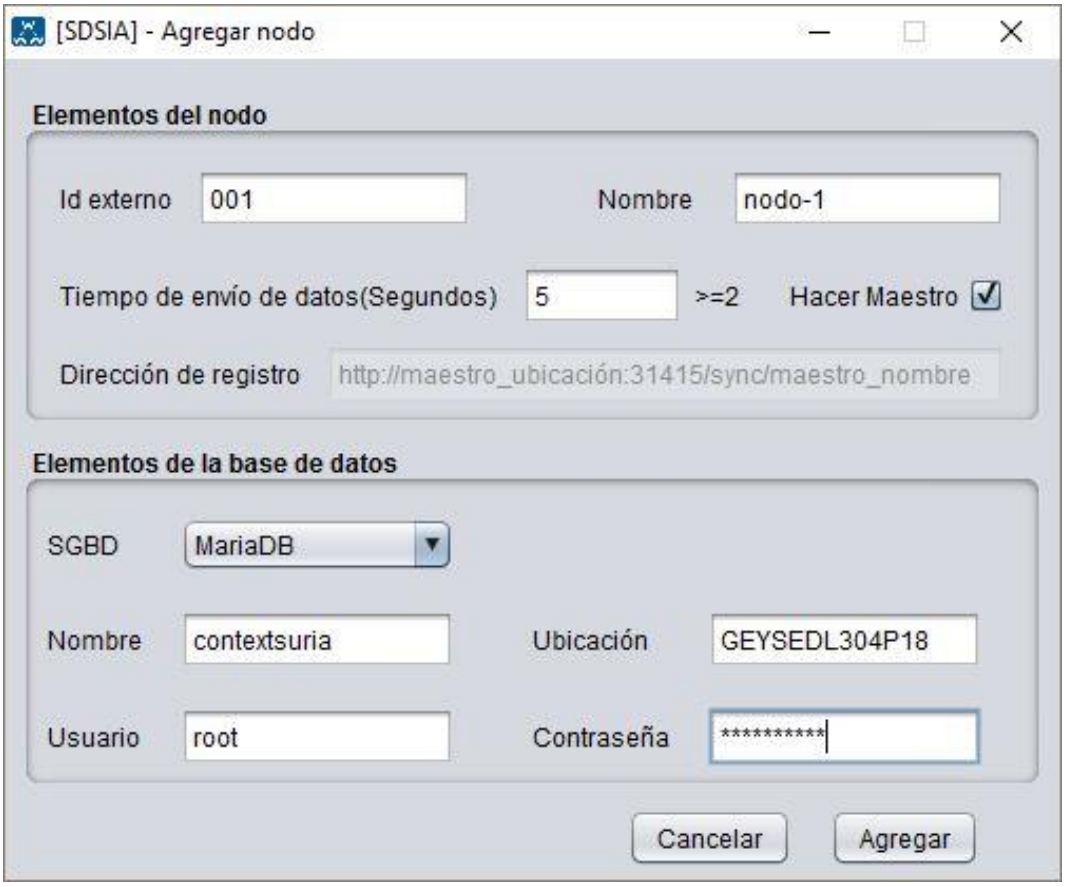


Fig 11: Interfaz correspondiente a la creación de nodo.

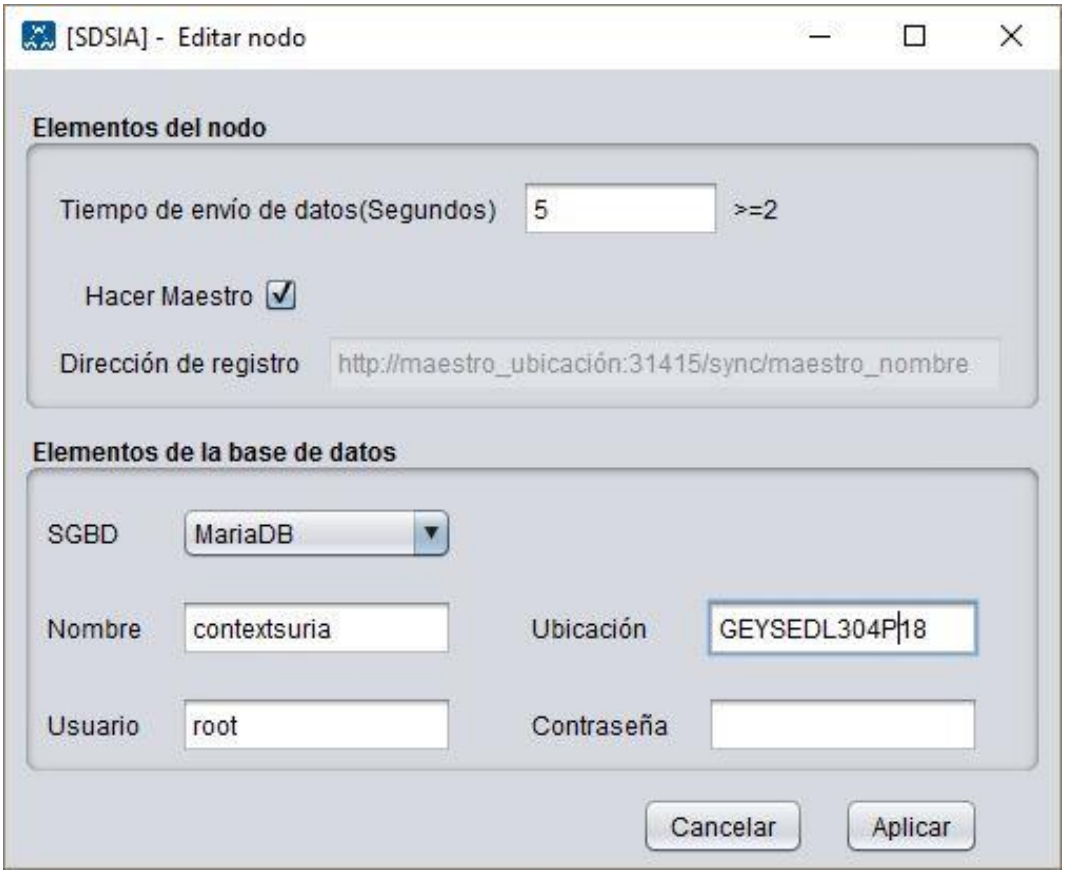


Fig 12: Interfaz correspondiente a la edición de nodo.

3.8 Pruebas del sistema

En la presente sección se expondrá todo lo referente a la fase de pruebas realizadas al sistema desarrollado, así como los resultados obtenidos. Dentro de esta fase se desarrollan distintos tipos de pruebas en función de los objetivos de las mismas. Estas fueron las pruebas de Caja Negra y Caja Blanca.

3.8.1 Pruebas realizadas.

- **Prueba de Caja Blanca**: La prueba de caja blanca del software se basa en el examen interno de los detalles de un procedimiento. Las rutas lógicas a través del software y las colaboraciones entre componentes se ponen a prueba al revisar conjuntos específicos de condiciones y/o bucles. (Pressman, 2010)
- **Prueba de Caja Negra:** Se realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar. Las pruebas de caja negra intentan

encontrar errores en las categorías siguientes: funciones incorrectas o faltantes, errores de interfaz, errores en las estructuras de datos o en el acceso a bases de datos externas, errores de comportamiento o rendimiento y errores de inicialización y terminación. (Pressman, 2010)

Cada prueba tiene técnicas asociadas, las cuales se aplicaron Camino Básico para Caja Blanca y Partición Equivalente para Caja Negra. (Pressman, 2010)

- **Camino básico o trayectoria básica:** Permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño de procedimiento y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para revisar el conjunto básico tienen garantía para ejecutar todo enunciado en el programa, al menos una vez durante la prueba.
- **Partición de equivalencia:** Es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos de los que pueden derivarse casos de prueba. Un caso de prueba ideal descubre de primera mano una serie de errores (por ejemplo, procesamiento incorrecto de todos los datos del tipo cadena) que de otro modo podrían requerir la ejecución de muchos casos de prueba antes de observar el error general.

3.8.2 Resultados finales de las pruebas realizadas a la plataforma.

A continuación, se presenta el caso de prueba correspondiente al caso de uso “Administrar nodo de réplica y sincronización”. El entorno de trabajo en el que fueron realizadas las pruebas cumple con las características de los requisitos no funcionales de hardware especificados.

- **Prueba de Caja Negra**

Aplicando el método de partición de equivalencia se realizaron las pruebas de caja negra, arrojando los siguientes resultados:

Condiciones de ejecución: El Usuario Replicador debe seleccionar previamente la opción “Agregar”.

Tabla 4: Diseño de prueba de caja negra

	Sección “Adicionar nodo de réplica y sincronización”						
Escenario	**Descripción**	**Nombre**	**Tiempo de envío de datos (Segundos)**	**Dirección de registro**	**Hacer maestro**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Adicionar nodo correctamente	El usuario Replicador inserta correctamente los datos necesarios para adicionar un nodo al sistema.	V Nodo1	V 2	V http://maestro_ubicación:31415/sync/maestro_nombre	V Seleccionar/No Seleccionar	El sistema valida la entrada de datos. En la interfaz principal muestra la visualización de la ejecución de los datos insertados.	Interfaz principal/ Clic en el botón Agregar/ Introducir datos necesarios/ Clic botón Aceptar.
EC 1.2 Adicionar nodo incorrectamente.	El usuario Replicador inserta incorrectamente los datos necesarios para adicionar un nodo al sistema.	V	I	V	V	Muestra el mensaje de error: *La frecuencia no puede tener valores inferiores a 2.*	Interfaz principal/ Clic en el botón Agregar/ Introducir datos necesarios/ Clic botón Aceptar
EC 1.3 Adicionar nodo incorrectam	Se inserta un campo incorrecto.	Nodo1	1/0	http://maestro_ubicación:31415/sync/maestro_nombre	Seleccionar/No Seleccionar	Redirige hacia el formulario de registro y	1. Se selecciona la opción “Registrarse”

ente. Campos vacíos						muestra un mensaje error.	en la página inicial de la plataforma. 2. Se llenan los campos del formulario de registro. 3. Se selecciona la opción “Registrarse”. 4. Se muestra mensaje de error.

A partir del diseño y ejecución de los casos de prueba creados, se detectaron errores tales como: opciones que no funcionan, errores de idioma, validación de los datos de entradas, y funcionalidades incorrectas, para un total de 11 no conformidades durante todas las iteraciones efectuadas.

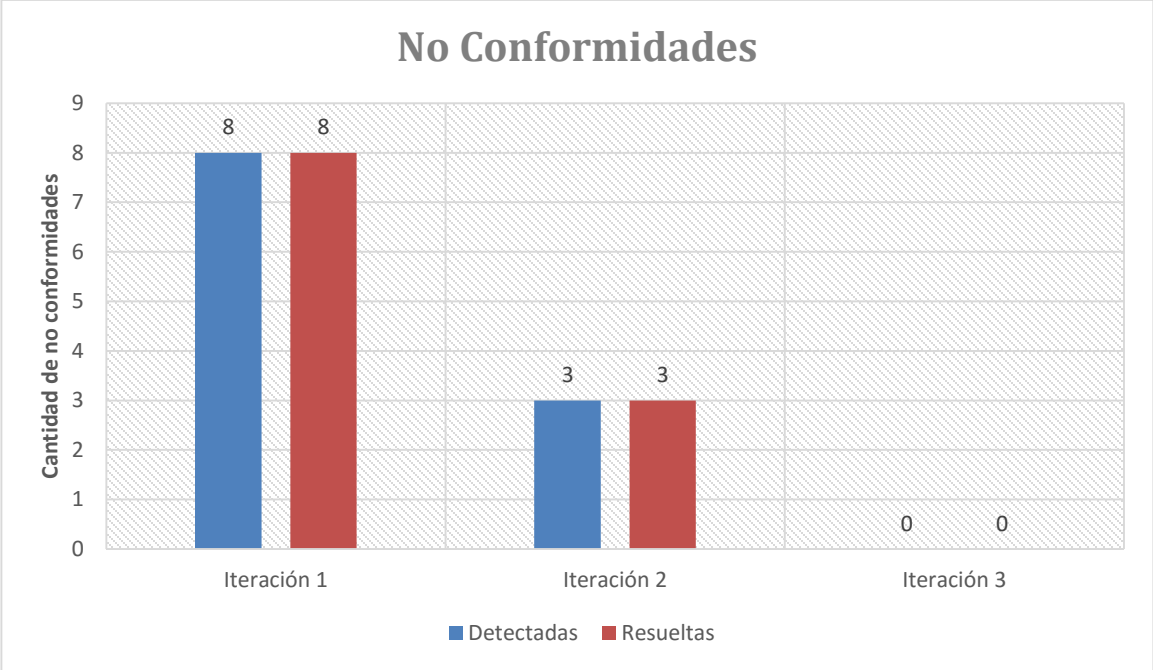


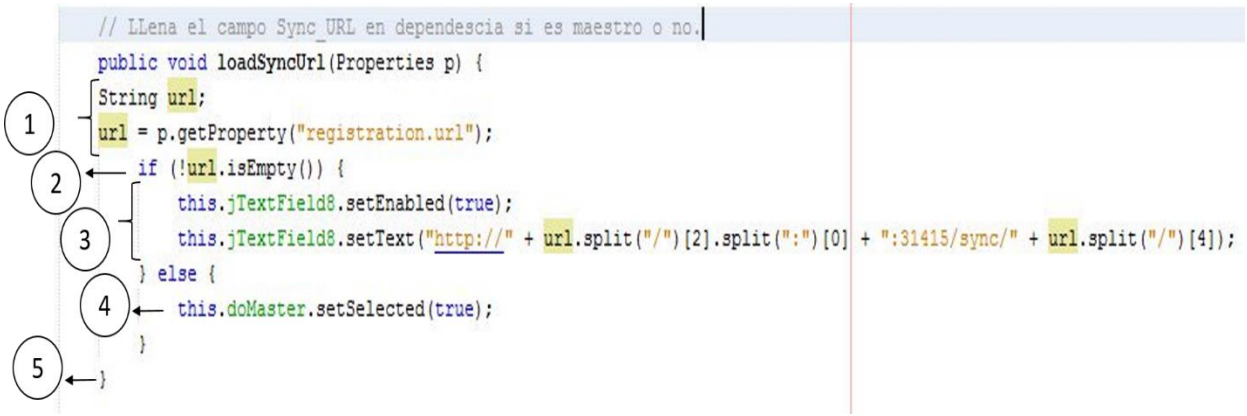
Fig 13: Gráfica de no conformidades de la pruebas funcionales

La gráfica anterior representa los resultados alcanzados en las pruebas funcionales, representando la cantidad de no conformidades detectadas en cada iteración. Al realizar un análisis de los resultados de la primera iteración, fueron detectadas 8 no conformidades asociadas principalmente a errores de validación de datos, opciones que no funcionan y funcionalidades incorrectas. Después de corregir los errores de la primera iteración se prosigue con una segunda en la cual se evaluaron nuevamente las funcionalidades, siendo detectadas solo 3 no conformidades relacionadas a problemas de interfaz. Después de darle solución a estas se prosigue con una última iteración obteniendo resultados satisfactorios sin no conformidades detectadas.

- **Pruebas de Caja Blanca**

Para la aplicación de las pruebas de Caja Blanca se utilizó la técnica del Camino Básico ya que permite obtener una medida de la complejidad lógica del diseño procedimental sobre un conjunto básico de caminos de ejecución.

A continuación, se muestra en la Fig 14 el código, que pertenece a la funcionalidad de llenar el campo “Dirección de registro”.




```
// LLena el campo Sync_URL en dependescia si es maestro o no.
public void loadSyncUrl(Properties p) {
String url;
url = p.getProperty("registration.url");
if (!url.isEmpty()) {
    this.jTextField8.setEnabled(true);
    this.jTextField8.setText("http://" + url.split("/")[2].split(":")[0] + ":31415/sync/" + url.split("/")[4]);
} else {
    this.doMaster.setSelected(true);
}
}
```

Fig 14: Código para la realización de la prueba de caja blanca.

En la Fig 15 se muestra el grafo obtenido.

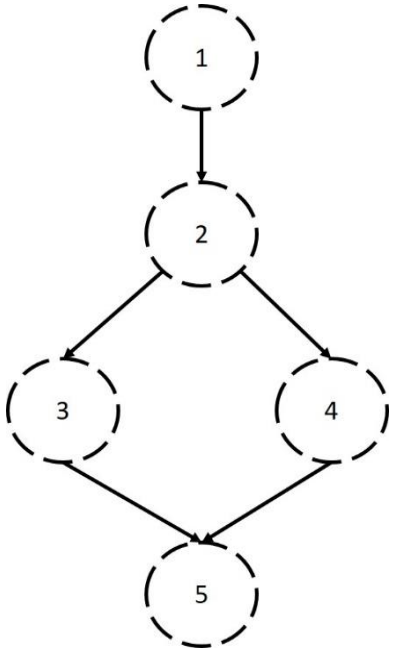


Fig 15: Grafo obtenido del código seleccionado.

El cálculo de la complejidad ciclomática proporcionó un límite inferior de la cantidad de pruebas que se deben aplicar a la porción de código, estableciendo una para cada camino independiente que posea el grafo. La fórmula para el cálculo de la complejidad ciclomática es: V (G) = A – N + 2, donde [V (G)] es la complejidad ciclomática, [A] representa a la cantidad de aristas y [N] a la de nodos con los que cuenta el grafo de flujo asociado al código. Luego de aplicar la fórmula para los valores del grafo queda de la siguiente manera:

V (G) = A – N + 2

V (G) = 5– 5+ 2

V (G) = 2

El valor obtenido para la complejidad ciclomática es 2, lo que deja claro que existen 2 posibles caminos a recorrer para el flujo; de igual forma es la cantidad mínima de pruebas que se le realizó al código. La Tabla 5 plasma los caminos básicos del flujo.

Tabla 5: Caminos básicos obtenidos de la tabla.

No.	Caminos
1	1-2-3-5

2	1-2-4-5

Dando continuidad al método y contando con el grafo y los caminos identificados previamente se pasa a la creación de casos de pruebas para cada uno de estos caminos. La Tabla 6 muestra el caso de prueba para el camino 1-2-3-5.

Tabla 6: Caso de prueba para el camino básico No.1 Prueba de Caja Blanca.

Caso de prueba para el camino básico No.1	
Camino	1-2-3-5
Descripción de los parámetros de entrada	Se le pasa por parámetro una variable *p* que contiene la información del nodo de réplica y sincronización necesaria para cargar el campo: Dirección de registro.
Resultado esperado	1-2-3-5

Conclusiones del capítulo

En este capítulo se ha generado toda la información necesaria para el desarrollo de la solución de réplica y sincronización de datos, dando respuesta a las tareas de la investigación 4 y 5. Se expusieron los resultados de las principales pruebas realizadas a la herramienta desarrollada. Por lo que se puede concluir que:

- La utilización de los patrones arquitectónicos y de diseño permitió aplicar una guía para la construcción de la herramienta de réplica y sincronización.
- La realización de los artefactos ingenieriles permitió documentar todo el proceso de construcción del software.
- La realización del flujo de trabajo de implementación permitió poner en práctica los estándares de codificación definidos. Esto ayuda a entender el código escrito, y permite realizar el mantenimiento o ampliación de la solución por otros desarrolladores.
- La detección y solución de las no conformidades permitió aumentar la calidad final de la solución para asegurar el correcto funcionamiento de la misma.

- La aplicación de distintas pruebas permitió comprobar que los resultados obtenidos en el sistema se corresponden con las necesidades del cliente, quedando implementados todos los requisitos pactados.

Conclusiones generales

Con el presente trabajo de diploma se logró dar cumplimiento a las tareas de investigación propuestas, obteniendo como resultado una herramienta que permite la réplica y sincronización de datos del Sistema Xilema-Suria 3.0. La investigación permitió arribar a las siguientes conclusiones:

- El estudio de las diversas soluciones existentes en Cuba y el mundo, permitió identificar como herramienta a utilizar a SymmetricDS para desarrollar el proceso de réplica y sincronización de datos para el Sistema Xilema Suria 3.0.
- La definición de los requisitos funcionales de la herramienta en conjunto con el diseño de clases del diseño posibilitó la identificación de las funcionalidades para cumplir con los objetivos planteados.
- La utilización del patrón arquitectónico N-Capas permitió estructurar la implementación en dos capas: Presentación y Lógica de Negocio, admitiendo el desarrollo de la herramienta en varios niveles, lo que permite modificar el código de una capa sin afectar al resto del sistema.
- Las pruebas realizadas permitieron comprobar el correcto funcionamiento de la herramienta de réplica y sincronización de datos del Sistema Xilema Suria 3.0.

Recomendaciones

Luego de haber analizado los resultados del presente trabajo de diploma, se arriba a plantear las siguientes recomendaciones:

- Implementar el intercambio de datos realizado por SymmetricDS sobre el protocolo seguro HTTPS.
- Implementar el intercambio de datos realizado por SymmetricDS entre los nodos hermanos de manera directa sin pasar por el nodo maestro.

Bibliografía

Pérez Porto , Juliány y Gardey, Ana . 2010. Definición de. [En línea] 09 de Junio de 2010. http://definicion.de/tabla/.

Alburquerque Arias, Amado. 2007. *Sistema Integrado de Gestión Estadística (SIGE).* Universidad de las CIencias Informáticas. Habana : s.n., 2007.

Asensio, Rafael Menéndez-Barzanallana. 2016. Ingeniería del software. [En línea] 9 de Septiembre de 2016. [Citado el: 19 de Noviembre de 2016.] http://www.um.es/docencia/barzana/IAGP/Enlaces/CASE_principales.html.

Barroso, Miguel Angel, Cesar de La Torre Llorente,, Unai Sorrilla Castro. 2010. *Guía de Arquitectura N-Capas orientada al Dominio con .NET.* 2010.

Belloch Ortí, Consuelo. 2006. *Las tecnologías de la información y comunicación (TIC).* Valencia : s.n., 2006.

Browne, Christopher. 2016. Slony-I: enterprise-level replication system. [En línea] 30 de Mayo de 2016. http://www.slony.info/adminguide/2.2/doc/adminguide/slony.pdf.

EcuRed. 2017. EcuRred: Conocimientos con todos y para todos. [En línea] 2017. [Citado el: 20 de Ablil de 2017.] https://www.ecured.cu/Bases_de_datos.

Gamma. 2013. *Desing Patterns.* 2013.

Garlan, Shaw. 1996. *Software Arquitecture. Prentice Hall.* 1996.

Heredia, Caldeón y Jeferson, Christian. 2013. *Replicación en aplicaciones distribuidas y su aplicación al sistema de administración académica de la UTN.* 2013.

Hernández León, Rolando Alfredo y Coello González, Sayda. 2002. *El Paradigma Cuantitativo de la Investigación Científica.* La Habana : Editoral Universitaria, 2002.

IBM. 2013. IBM Knowledge Center. [En línea] 2013. [Citado el: 20 de Ablil de 2017.] https://www.ibm.com/support/knowledgecenter/es.

—. 2016. IBM Knowledge Center. [En línea] 2016. [Citado el: 20 de Mayo de 2017.] https://www.ibm.com/support/knowledgecenter/es/SS8CCV_7.6.0/com.ibm.mif.doc/gp_intfrmwk/oslc/c_oslc_put_method.html.

IBPhoenix. 2017. IBPhoenix: your premier source of firebird supprt. [En línea] 21 de Mayo de 2017. https://www.ibphoenix.com/products/software/ibreplicator.

Jiménez, Ramón Antonio Morales. 2016. *GEYSED_Video Vigilancia SURIA 3.0_Arquitectura Vista de Datos.* 2016.

JumpMind. 2017. SymmetricDS. [En línea] 10 de Enero de 2017. https://www.symmetricds.org/.

Kruchten. 2000. *The Rational Unified Process: An Introduction.* 2000.

Larman, Craig. 2004. *UML y Patrones.* 2004.

—. 2003. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* México : Prentice Hall, 2003.

Larman, Craig, Prentice Hall. 2003. *UML y patrones.* 2003.

Lic. D. Ramon, Hugo. 2007. *Conceptos de Bases de Datos Distribuidas.* 2007.

Loney, Kevin y Koch, George. 2003. *ORACLE 9I: MANUAL DE REFERENCIA.* s.l. : S.A. MCGRAW-HILL / INTERAMERICANA DE ESPAÑA, 2003. ISBN:9788448139377.

MariaDB. 2017. MariaDB Fundation. [En línea] MariaDB Fundation, 2017. [Citado el: 20 de Febrero de 2017.] https://mariadb.com/kb/en/mariadb/what-is-mariadb-102/.

Módulo de Reko para la replicación entre bases de datos con. **Mena Rodriguez, Luis Edgardo y Pérez Alfonso, Damian. 2011.** Habana : Serie Científica, 2011.

MongoDB. 2011. Replication Introduction. [En línea] 2011. [Citado el: 13 de Febrero de 2017.] http://docs.mongodb.org/manual/core/replication-introduction/.

NetBeans.org. 2016. NetBean.org. [En línea] 2016. [Citado el: 8 de Mayo de 2017.] https://netbeans.org/index_es.html.

Oracle. Oracle | Integrated Cloud Applications and Platform Services:. [En línea] [Citado el: 27 de Octubre de 2016.] http://https//www.oracle.com/lad/products/mysql/overview/index.html.

Ortiz Noriega, Dresky, y otros. 2010. *Módulo de Reko para la resolución de conflictos.* UCI. Habana : s.n., 2010. Tesis.

Pérez Porto, Julián y Gardey, Ana . 2013. Definición de. [En línea] 2013. [Citado el: 8 de Mayo de 2017.] http://www.definicion.de/java/.

Pérez Valdes, E. 2007. *Fundamentos de bases de datos.* 2007.

Pressman, Roger S. 2010. Ingenieria de Software: Un enfoque práctico : s.n., 2010.

—. 2005. Ingeniería de Software: Un enfoque diferente : s.n., 2005.

—. 2010. *Ingenieria de Software: Un enfoque práctico.* 2010. Vol. VI.

—. 2010. Software engineering: a practitioner's approach. 7th ed. 2010.

Programación, Lenguajes de. 2016. Lenguajes de Programación. [En línea] 2016. [Citado el: 15 de Marzo de 2017.] http://www.lenguajes-de-programacion.com/programacion-java.shtml.

Reingart, Mariano. 2012. *PyReplica Sistema de replicación simple para PostgreSQL programado en Phyton.* 2012.

Sánchez, Tamara Rodríguez. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* Habana : s.n., 2015.

Slony Development Group . 2010. Slony-I. *Slony-I.* [En línea] Command Prompt, Inc., 2010. [Citado el: 25 de Noviembre de 2016.] http://www.slony.info/.

Tardia, M.A.M. 2011. *Replicación.* 2011.

Tello, Jeús Cáceres. 2014. *Design patterns: example of application in the Generative Learning Object.* s.l. : Dpto.Ciencias de la Computación (Universidad de Alcalá), 2014.

UCI. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* Habana : s.n., 2015.

Vega, Norge Fajardo. 2007. *Sistema de réplica para bases de datos distribuidas en PostgreSQL.* Habana : s.n., 2007.

Visual Paradigm International Ltd. 2010. Visual Paradigm for UML Standard 8.0. *Soft 112.* [En línea] 16 de Agosto de 2010. [Citado el: 19 de Noviembre de 2016.] http://visual-paradigm-for-uml-standard.soft112.com/.

WebFinance Inc. 2015. BusinessDictionary. [En línea] 2015. [Citado el: 12 de Febrero de 2017.] http://www.businessdictionary.com/definition/programming.

Zayas, Carlos Alvarez de. 1995. *Metodología de la Investigación Científica.* Santiago de Cuba : s.n., 1995.