Universidad de las Ciencias Informáticas

Facultad 6

Interfaz web para la administración y monitorización de la herramienta de réplica de datos SymmetricDS

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

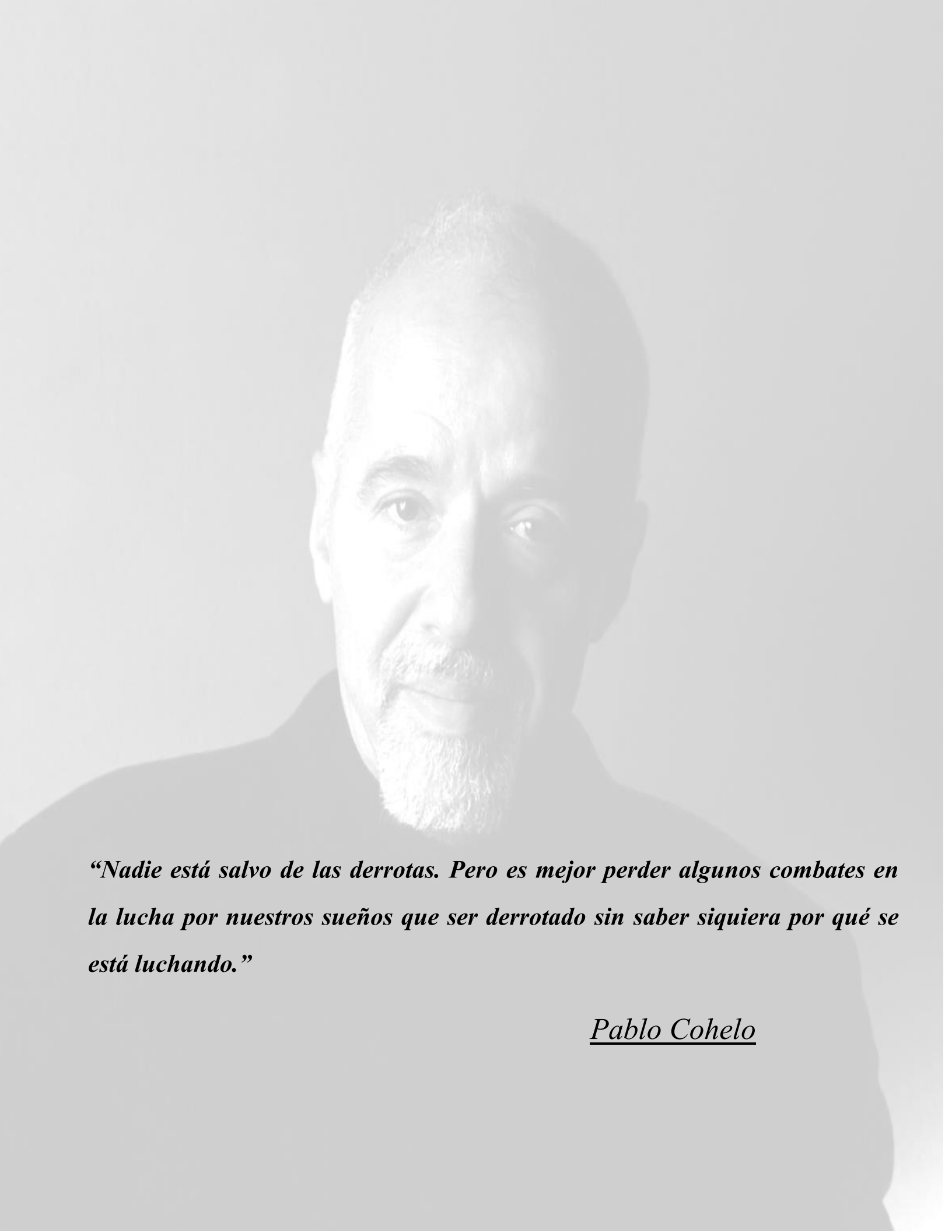
Autores: Arianne Ferrer Carcasés

Alfredo Sabina Diez

Tutores: Ing. Adrián Misael Peña Montero

Ing. Marcos Michel Martínez Pérez

La Habana, junio 2014



“Nadie está salvo de las derrotas. Pero es mejor perder algunos combates en la lucha por nuestros sueños que ser derrotado sin saber siquiera por qué se está luchando.”

Pablo Cohelo

Declaramos ser autores del presente trabajo de Diploma 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 _____.

Arianne Ferrer Carcasés

_____

Firma del Autor

Alfredo Sabina Diez

_____

Firma del Autor

Ing. Adrián M. Peña Montero

_____

Firma Tutor

Ing. Marcos M. Martínez Pérez

_____

Firma Tutor

Tutores:

Ing. Adrián Misael Peña Montero.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Correo electrónico: ampena@uci.cu

Ing. Marcos Michel Martínez Pérez.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Correo electrónico: mmartinezp@uci.cu

Arianne

Esta Tesis constituye el fruto de tantos mese de esfuerzo y sacrificio y la quiero dedicar en primer lugar a mis padres Xiomara y Andrés por guiarme todo este tiempo y corresponder a mis sueños de graduarme. A mi hermanito José, que siempre se las arreglaba para transformar un instante de estrés en alegría.

A mis abuelos Andrés y Nilda y a los ya no presentes en este mundo José y Reina.

A mis amigas Lisbita, Michi, Dannelis (La Flaca), Mely por aguantar mis pesadeces y compartir estos 5 años llenos de tristezas y muchas cositas lindas.

A mis tías Berta, Rebe, mi tío Israel y mi prima Claire por convertirme en una integrante más de su familia

A mi tío Ariel por abrirme sus brazos desde el primer momento que llegue a esta escuela.

A mi novio Frank por soportar mis malacrianzas.

A esta Revolución Cubana que me dio la oportunidad de estudiar en una institución como esta y graduarme satisfactoriamente.

Alfredo

Esta tesis va dedicada a mi abuela Josefa que en este momento por desgracia de la vida no pueden estar conmigo.

A lo que más quiero en este mundo mi abuela Ninfa, una mujer increíble y me brindó su amor, cariño y muy buenos consejos.

A mis amigos Ana, Héctor, Yoel, Oscar, porque siempre me dieron mucha fuerza para seguir adelante en los momentos más difíciles de mi carrera.

A ese pilar de madre que tengo, por siempre darme su amor y cariño.

A mi hermano Yudiel por siempre cuidar de mí y saber aconsejarme en momentos difíciles de la vida.

A la Revolución Cubana por la oportunidad que me ha dado siempre de estudiar y de llegar a estos resultados que hoy tengo y disfruto tanto.

Arianne

En la vida no se debe recorrer lo desconocido sin una luz que te guie por el camino correcto, para que sufras menos y sepas afrontar las dificultades y pesares del día a día.

Quiero agradecer primero a nuestro Comandante en Jefe Fidel Castro Ruz y a esta hermosa Revolución por permitirme vivir el sueño de muchos jóvenes en el mundo, ser un graduado universitario.

Agradezco a mi papito por todas las horas que ha dedicado a verme crecer, por ser un ejemplo a seguir como padre y como profesional, por saber entender mis emociones y permitirme ver que la vida es más que juegos y fiestas. Gracias papito lindo.

A mi mama porque me brindó la oportunidad de vivir este lindo sueño dándome la Vida, porque es un ejemplo de mujer trabajadora, por ser madre, hermana, amiga en estos años tan difíciles para mí. Te quiero Mamita.

A mi hermanito José por ser lo que me inspira cada mañana, por ayudarme a convertirme en un buen ejemplo para ti. Te quiero Flaco.

A mi abuela Nilda porque siempre se preocupa por mi salud, mi bienestar, las cosas de la escuela, y aunque no me pudo arreglar el pelo esta vez la quiero igual.

A mi tía Rebe, Bertha, mi prima Claire, mi tío Israel por abrirme las puertas de su casa y de su corazón en el momento que más lo necesitaba. Por ayudarme a seguir con este sueño hecho realidad Gracias.

A mis amigas las perras como les digo cariñosamente: Nely aunque estés muy lejos quiero que sepas que te quiero mucho y gracias por estar ahí desde los primeros meses de mi vida. Lisbita te agradezco por ser como mi segunda mamá desde que nos conocemos hace ya casi 18 años te quiero mami. Michi tu eres para mí como un ejemplo de supervivencia y te agradezco todos estos años de vida que pasamos juntas. Mely desde primer año te colaste en mi corazoncito como si vivieras en él hace mucho más tiempo, quiero

agradecerte por estos cinco años de carrera. A la loki de Claudita por ser una excelente compañera de cuarto y amiga sobre todo, gracias por tus consejos y apoyo.

A mi dúo de tesis Alfredo por los días sufridos, por las noches sin dormir, por las veces que dejó de comer para que esto funcionara. Te agradezco por este día por este momento de felicidad de los dos.

A mis tutores Marcos y Adrián por guiarnos en este trabajo y soportar nuestras malacrianzas. Y no se me puede olvidar una persona que me ayudó mucho casi en el final del desarrollo de esta tesis el profe Yovanoti como le dicen sus amigos.

A mi novio Franky por las noches sin dormir, por llorar junto conmigo en los momentos de desesperación, por su amor, comprensión, por ser más que amigo y estar siempre a mi lado y soportar mis ratos de bobita.

Alfredo

Agradezco a los tutores, Marcos Michel y Adrián Misael por el apoyo y la ayuda brindada durante el desarrollo de la Tesis.

A mi compañera de tesis Arianne (La titi) por su apoyo en todos los momentos difíciles que pasamos en el transcurso de este año.

Le agradezco a la UCI y a todos los profesores que me dieron clases durante estos 5 años.

A mi grande e inmensa familia pues ellos son la razón de mi vida, y todos los días le doy gracias a dios por tenerlos a mi lado.

A todos mis amigos los amigos que hice durante todo este tiempo en especial Ana, Yoel, Héctor, Luis Miguel, Oscar, Juan Miguel, Claudia, Mavis, Liosdanys, Laritza, Rizo, Batista, gracias a todos pues fueron los que mayormente estuvieron en las buenas y en las malas, apoyándome en todo y siendo incondicionalmente mis verdaderos amigos.

Le agradezco a todos los que han estado en mi grupo docente, agradezco su apoyo cuando de verdad lo necesité, me llevo un pedacito de ustedes en mi corazón.

A mi madre y mi abuelita Ninfa, estos dos pilares de mujer que he tenido durante todo mi vida, gracias por su apoyo incondicional.

A mi hermano Yudiel y mi tía Teresita por siempre estar presente cuando los he necesitado.

A Liliana y Sandra gracias por el apoyo que me brindaron desde el momento que la vida me puso en su camino.

Las Tecnologías de la Información y las Comunicaciones (TIC en lo adelante) son un factor de vital importancia en la transformación de la sociedad, especialmente para áreas del conocimiento como la Gestión de Datos. Actualmente los usuarios que utilizan la aplicación *SymmetricDS* para gestionar soluciones de réplica de datos en el Centro de Tecnologías de Gestión de Datos de la Universidad de las Ciencias Informáticas, sólo pueden configurar la herramienta a través de la terminal de comandos utilizando un *Shell Script* implementado para la solución de réplica de datos del Sistema de Información de Gobierno. A pesar de que existe una herramienta profesional que administra la aplicación *SymmetricDS,* ésta posee licencia privativa. Por tanto surge la necesidad de crear una alternativa que opere de la misma manera pero sin pagar altos costos económicos. Es por ello que la presente investigación se enmarca en la implementación de una interfaz web que permita administrar y monitorear la aplicación *SymmetricDS* para facilitar la gestión de las soluciones de réplica de datos desarrolladas en el departamento PostgreSQL del Centro de Tecnologías de Gestión de Datos. La interfaz web obtenida permite a los administradores de base de datos controlar y mantener en correcto funcionamiento de cualquier solución de réplica de datos que desarrollen. Para lograr este resultado fueron aplicadas las pruebas de aceptación, pruebas de sistema y pruebas de regresión, y en cada una de ellas fueron resueltas todas las no conformidades encontradas.

Palabras Claves: Administración, Monitorización, Interfaz web, Soluciones de réplica de datos, SymmetricDS.

The Information Technology and Communications (ICT hereafter) are a vital factor in the transformation of society, especially for areas of knowledge such as Data Management. Currently users, who use the application to manage SymmetricDS data replication solutions in the Technology Center Data Management at the University of Information Sciences, can only configure the tool through the command terminal using a Shell Script implemented to the solution data replication Information System Government. Although there is a professional tool that manages SymmetricDS application, it has proprietary license. Thus arises the need to create an alternative that operates in the same manner but without paying high economic costs. That is why this research is part of the implementation of a web interface to manage and monitor the implementation SymmetricDS to facilitate the management of data replication solutions developed in the department Center PostgreSQL Data Management Technologies. The web interface allows database administrators to monitor and maintain the proper functioning of any data replication solution development. To achieve this result the acceptance tests were applied, system testing and regression testing, each and all not-conformities were resolved.

Keywords: Administration, Monitoring, web interface, replica Solutions, SymmetricDS.

Introducción

En Cuba el proceso de informatización en todas las esferas de la sociedad ha propiciado el desarrollo de software siendo muchas las instituciones del país que se han vinculado a este proceso, como la Universidad de las Ciencias Informáticas (UCI en lo adelante) que tiene como misión dirigir la formación de sus ingenieros con conocimientos, habilidades y valores sólidos, sustentados en una concepción científica y dialéctico-materialista del mundo, que estén comprometidos con su Patria y que actúen como profesionales responsables, honestos, honrados, creativos, modestos, solidarios y con ética revolucionaria en el campo de la informática, poseedores además de una cultura general integral.

En la institución antes mencionada, se han creado un grupo de centros productivos destinados al desarrollo de aplicaciones informáticas, como es el Centro de Tecnologías y Gestión de Datos (DATEC en lo adelante), especializado en el desarrollo de tecnologías de base de datos, donde además se implementan soluciones que benefician a muchas instituciones del país como por ejemplo el Sistema de Información de Gobierno (SIGOB en lo adelante) para la Oficina Nacional de Estadísticas e Información (ONEI en lo adelante), con el objetivo de garantizar a las instituciones gubernamentales del país todos los datos estadísticos referente a cada esfera de la sociedad, ya sea educación, cultura, deporte, salud, industria y servicios.

Este sistema posee una sede en cada provincia del país incluyendo al municipio especial Isla de la Juventud, con una sede central en la Capital. Atendiendo a la necesidad de establecer una vía de comunicación permitiera la sincronización de la base de datos de la sede central con cada una de las bases de datos de las sedes provinciales y viceversa y con el fin de poder gestionar en tiempo real la información se desarrolla en el centro DATEC una solución de réplica de datos para el SIGOB.

Para el desarrollo de la solución de réplica de datos es necesario implementar tres tipos de réplica de datos: maestro-esclavos, maestros-esclavo y bidireccional. Precisamente para dar cumplimiento a este requisito se realizó un estudio detallado de las herramientas de réplica de datos existentes, escogiendo a *SymmetricDS* como la más adecuada por brindar soporte a las necesidades del sistema que se desea construir y ser liberada bajo la licencia de software libre.

En un principio se implementó una aplicación *Shell Script* orientada a las terminales *GNU/Linux* que permite la configuración de *SymmetricDS* para gestionar la solución de réplica de datos de SIGOB y brinda la posibilidad de adicionar nuevas tablas a la réplica. Una limitante del *Shell Script* es que está desarrollado específicamente para el entorno de réplica de SIGOB por tanto no pueden ser ejecutadas otras soluciones de réplica de datos sino se implementa una instancia de dicha solución.

Durante las pruebas piloto que le fueron realizadas a la aplicación *Shell Script* se identificó que las tareas de configuración sobre el *SymmetricDS* no permiten obtener reportes para verificar el estado de la réplica, el mal funcionamiento de los nodos de los nodos o errores de sincronización que pueden surgir. Estas deficiencias traen como consecuencia que los administradores de la réplica de datos deban consultar de forma manual los logs de *SymmetricDS*, convirtiéndose en una tarea compleja y agotadora. De forma general se puede decir que la aplicación Shell Script no brinda la posibilidad de tener un mayor control sobre la solución de réplica de datos, resultando imprescindible la realización de tareas de administración y monitorización sobre la herramienta *SymmetricDS*.

La búsqueda de una solución a estas deficiencias permiten definir el siguiente **problema científico:** ¿Cómo administrar y monitorear la herramienta SymmetricDS para realizar una gestión adecuada de las soluciones de réplica de datos que la utilizan, desarrolladas en el departamento *PostgreSQL* del centro DATEC?

Tomando como **objeto de estudio:** la administración y monitorización en las herramientas de réplica de datos, identificando el **campo de acción**: la administración y monitorización en *SymmetricDS*.

Para dar solución al problema planteado anteriormente se define el siguiente **objetivo general**: Desarrollar una interfaz web para la administración y monitorización de la herramienta *SymmetricDS* que posibilite la gestión adecuada de las soluciones de réplica de datos que la utilizan desarrolladas en el departamento *PostgreSQL* del centro DATEC.

El objetivo general definido anteriormente se desglosa en los siguientes **objetivos específicos:**

- Realizar un estudio del estado del arte basado en los procesos de administración y monitorización de las herramientas de réplica de datos.

- Desarrollar las funcionalidades de la aplicación que permitirá administrar y monitorear la herramienta *SymmetricDS.*
- Validar el funcionamiento del sistema desarrollado.

Para dar cumplimiento a los objetivos específicos se plantean las siguientes **tareas de la investigación**:

- Realización de un estudio del estado del arte basado en los procesos de administración y monitorización de las herramientas de réplica de datos
- Selección de las tecnologías, herramientas, estándares y metodologías necesarias a utilizar en el desarrollo de la aplicación que permitirá administrar y monitorear la herramienta *SymmetricDS*.
- Realización del diseño de las funcionalidades de la aplicación que permitirá administrar y monitorear la herramienta *SymmetricDS*.
- Implementación de las funcionalidades de la aplicación que permitirá administrar y monitorear la herramienta *SymmetricDS.*
- Selección de las técnicas y métodos para la validación de la solución.
- Validación del funcionamiento de la aplicación que permitirá administrar y monitorear la herramienta *SymmetricDS.*

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

- ¿Qué elementos se deben tener en cuenta para implementar una solución de réplica de datos?
- ¿Cómo efectúan las tareas de administración y monitorización las herramientas de réplica de datos para mantener controladas las soluciones de réplica de datos?
- ¿Existe alguna herramienta que permita la administración y monitorización de *SymmetricDS*?

Durante el transcurso de la investigación se utilizan los siguientes **métodos científicos**:

Analítico – sintético: Se emplea para lograr extraer los elementos esenciales de las bibliografías consultadas con el objetivo de caracterizar los procesos de replicación y las soluciones de réplica de

datos. Además de permitir realizar un análisis de soluciones semejantes a la que se desea implementar para obtener información que beneficie el proceso investigativo.

El trabajo de diploma está estructurado de la siguiente forma:

Capítulo 1: Fundamentación teórico-metodológico sobre los sistemas de réplica de datos: Incluye los resultados del estudio del estado del arte de la investigación, donde se realiza una breve reseña histórica del surgimiento y evolución de los sistemas de réplica de datos. Además se sintetizan los conceptos principales relacionados con la réplica de datos, fundamentando las metodologías, tecnologías y herramientas que se utilizan para el desarrollo de la solución.

Capítulo 2: Diseño de la solución propuesta: Se describe cómo se va a implementar la solución que permitirá la administración y monitorización de la herramienta *SymmetricDS*, incluyendo cada uno de los módulos que la componen. Se muestra el modelo de dominio para explicar la interacción de los conceptos asociados al negocio y se presenta el diseño de la solución que incluye las historias de usuario, la lista de reserva del producto, el plan de iteraciones, las tarjetas CRC y el diagrama de clases, además la arquitectura conformada por los patrones de arquitectura y patrones de diseño.

Capítulo 3: Implementación y validación de la solución propuesta: En este capítulo se realiza una descripción de la implementación de la interfaz según las tareas de ingeniería y los estándares de codificación. Describiendo el diagrama de despliegue y las interfaces principales de la aplicación También se investigan las diferentes estrategias de pruebas definidas por la metodología XP, que permitan comprobar el correcto funcionamiento de la solución, teniendo en cuenta las tipos de pruebas y las técnicas para ejecutarlas. Además las funcionalidades implementadas serán sometidas a un proceso de prueba para comprobar los resultados y corregir los errores encontrados.

Capítulo 1: Fundamentación teórico-metodológico 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 realiza un estudio de las aplicaciones existentes para la administración de la herramienta *SymmetricDS*. Se analizan las metodologías, herramientas y tecnologías usadas actualmente para el desarrollo de aplicaciones informáticas posibilitando dar solución a los problemas de administración y monitorización de SymmetricDS.

1.1. Conceptos asociados a la réplica de datos

En la actualidad son varias las empresas, instituciones, organizaciones que pese a su acelerado crecimiento por todo el mundo sienten la necesidad de establecer sedes o sucursales distribuidas en diferentes lugares geográficamente. Esta situación trae como consecuencia un gran flujo de información que debe ser analizada en cada una de las sedes por tanto el uso de una base de datos centralizada ocasionaría problemas de rendimiento por aumentar los tiempos de respuesta de la base de datos, además disminuirían los niveles de disponibilidad de la información. Como alternativa a estos problemas surge la replicación de datos.

1.1.1. Réplica de datos

La replicación [1] es un conjunto de tecnologías destinadas a la copia y distribución de datos desde una base de datos hacia otra o múltiples bases de datos, para luego sincronizarlas y mantener su coherencia. Además permite distribuir datos entre diferentes ubicaciones y entre usuarios remotos o móviles mediante redes locales y de área extensa, conexiones de acceso telefónico, conexiones inalámbricas e Internet.

Entre las distintas ventajas que ofrece este proceso se encuentran [2]:

- Alta disponibilidad: Se puede incrementar la disponibilidad de una base de datos mediante la replicación en un sistema distribuido. Si una de las máquinas del sistema falla, las otras podrán satisfacer las necesidades del cliente.

- Balance de carga: La replicación se puede utilizar para hacer un balance de carga. Ésta es una técnica usada para compartir el trabajo a realizar entre varias computadoras.

- Soporte para aplicaciones de alto consumo: Se puede satisfacer las necesidades de ciertos clientes que requieren un alto consumo en consultas, que sería muy costo en rendimiento, o hasta imposible, en una base de datos sin replicación.

- Confiabilidad: Debido a que existen varias copias de los datos disponibles en el sistema, se cuenta con un mecanismo confiable de recuperación de datos ante fallos en algún nodo.

Sin embargo, estas ventajas tienen también un coste asociado. Cuando las peticiones atendidas impliquen una actualización en el estado de la aplicación, dicha actualización debe realizarse en todas las réplica de datos, y esto debe hacerse de una manera ordenada para que todas ellas mantengan un estado consistente. Esto implica que las operaciones de actualización tendrán un tiempo de servicio mayor que en el caso no replicado, pues habrá que proceder a la propagación de las actualizaciones sobre todas las réplica de datos y para ello será necesario emplear algún mecanismo de difusión.

1.1.2. Soluciones de réplica de datos

Dada la diversidad de contextos donde se aplican mecanismos de replicación, se puede disponer de una gama de posibilidades en vez de utilizar una única forma de replicar datos, y esto se conoce como soluciones de réplica [2]. El objetivo fundamental de las soluciones de réplica está orientado a mantener sincronizadas varias fuentes de datos optimizando tanto recursos de red como de *hardware* y tareas administrativas a la réplica.

Clasificación de la réplica de datos

Las soluciones de réplicas son implementadas atendiendo a las diferentes clasificaciones que reciben las réplicas de datos. A continuación se sintetizan las características de la réplica según la forma de transmisión de los datos [3] y [4]:

Entorno de replicación:

- 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.

Forma de transmitir los cambios en el entorno de replicación:

- Síncrona: Los cambios ejecutados en un nodo maestro son aplicados instantáneamente en el nodo origen y los nodos destino dentro de una misma transacción. En caso de no poder ejecutarse la acción en cualquiera de los nodos, la transacción completa es retrotraída en todos los nodos. Requiere una alta disponibilidad de recursos de red.

- Asíncrona: Los cambios ejecutados en una tabla son almacenados y enviados posteriormente al resto de los nodos del entorno cada ciertos intervalos de tiempo y se pueden realizar modificaciones a los datos sin conexión de red.

Forma de capturar y almacenar 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*.

Forma de comportamiento de la réplica de datos atendiendo a los gestores de administración instalados en cada nodo:

- Homogénea: Se puede replicar los datos entre los nodos con Sistema Gestor de Bases de Datos (SGBD) iguales sin importar el sistema operativo (*MySQL* en *Windows* o *MySQL* en *Linux* u *Oracle* en *Windows* u *Oracle* en *Linux*).

- Heterogénea: Se puede replicar los datos entre los nodos con SGBD distintos sin importar el sistema operativo (*MySQL* en *Windows* u *Oracle* en *Windows* o *MySQL* en *Windows* u *Oracle* en *Linux*).

Cuando se desarrolla una solución de réplica de datos las principales tareas que se realizan son las de configuración, dirigidas a la creación e inicialización de los nodos. Con ellas se define la forma en que serán sincronizados y transformado los datos y cómo se tratarán los errores que pueden surgir en el envío

o captura de los datos. Los elementos anteriores no constituyen aval suficiente para realizar un seguimiento de la actividad y el estado de todos los procesos vinculados a la replicación, siendo de gran importancia las tareas de administración y monitorización.

1.2. Administración y monitorización de las soluciones de réplica de datos

La administración y monitorización de las soluciones de réplica de datos constituyen procesos continuos que ayudan a detectar y resolver problemas antes de que se manifiesten, garantizando su correcto funcionamiento. La administración de forma general es el proceso de planificar actividades que se realicen con eficiencia para lograr los objetivos propuestos.

En las soluciones de réplica de datos la administración permite efectuar una correcta configuración de los escenarios de sincronización y resolver errores que pueden surgir en el proceso de sincronización entre las base de datos. Detectar estos errores es una tarea que se puede cumplir si se ejecutan acciones de monitorización en la réplica de datos, asegurando que se efectúe un exhaustivo seguimiento y control en los procesos que se efectúan en las soluciones de réplica de datos.

1.2.1. 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 [5] :

- 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, los routers 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.

1.2.2. Monitorización de las soluciones de réplica de datos

Las tareas de monitorización permiten controlar el estado de la réplica de datos, analizando si el envío y captura de los datos se realiza de forma correcta o incorrecta y gestionando la detección de conflictos en la sincronización, fragmentación o enrutamiento de los datos.

Para poder monitorear una solución de réplica de datos se deben considerar aspectos tales como [6]:

- Latencia: el tiempo necesario para realizar los cambio entre nodos en una solución de réplica de datos.
- Rendimiento: la cantidad de actividad de replicación (medida en comandos suministrados durante un período de tiempo) que un sistema puede mantener en el tiempo.
- Simultaneidad: el número de procesos de replicación que pueden operar simultáneamente en un sistema.

- Duración de sincronización: el tiempo que tarda en finalizar una sincronización determinada.
- Utilización de recursos: recursos de hardware y de red utilizados como resultado del procesamiento de replicación.

1.3. Herramientas de réplica de datos

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 las consultadas en la literatura existe.

El sistema *Pyreplica* [7] permite el monitorización de las conexiones persistentes (*KeepAlive* son mecanismos que permiten múltiples peticiones sobre la misma conexión TCP) y conexiones directas a las etapas finales (*backends* que referencia al estado final de un proceso). Advierte al detectar conflictos de actualización/eliminación y falla en conflictos de inserción o errores de integridad de datos ofreciendo notificaciones vía e-mail, pero sin corregirlos. Además el control de fallo no es automático y no existe soporte para objetos grandes.

En la aplicación *DBReplicator* [8] los administradores de bases de datos requieren una participación mínima en la detección y resolución de conflictos, ya que muchas de sus versiones poseen algoritmos para realizar estas tareas. Estas versiones incluyen el uso de banderas para indicar el estado de replicación de los registros de una tabla. Los estados se clasifican en N (cuando no se ha replicado), P (cuando está pendiente) y S (cuando está sincronizado). También algunas de ellas manejan automáticamente los campos UTF8 (Formato de codificación de caracteres Unicode.) o caracteres Unicode (Estándar de codificación de caracteres) para facilitar el tratamiento informático, transmisión y visualización de textos de múltiples lenguajes y disciplinas técnicas. Además el sistema puede programarse para realizar operaciones en distintos intervalos de tiempo que pueden ser entre horas, minutos o segundos. También permite crear automáticamente las tablas que no existen en las bases de datos subscritas para que coincidan con las bases de datos originales.

Reko [9] es 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 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, HTTP o por ficheros de forma manual. Otra de las herramientas de réplica de datos es *SymmetricDS* la cual será descrita en la siguiente sección.

1.4. Herramientas de réplica de datos SymmetricDS

SymmetricDS [10] es un programa desarrollado en el lenguaje Java 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, Oracle, SQL Server, PostgreSql, DB2, Firebird, HSQLDB y H2*) y puede ser utilizado tanto en el sistema operativo *Linux* como en *Windows*.

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 [9]. Además, para realizar el transporte de datos utiliza el Protocolo de Transferencia de Hipertexto (HTTP por sus siglas en inglés) o Protocolo Seguro de Transferencia de Hipertexto (HTTPS por sus siglas en inglés) [11].

Atendiendo a la descripción anterior sus principales características se resumen en [10]:

- Compatibilidad: Posee soporte para 12 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 [10]:

- Desarrollada en el lenguaje Java en su versión 5 requiere *Java SE Runtime Environment* (JRE) o *Java SE Development Kit* (JDK) versión 5.0 o superior.

- Utiliza cualquier base de datos que almacene los datos haciendo uso de disparadores (*triggers*) y un controlador JDBC.

Como se expone en los epígrafes anteriores las tareas de configuración aplicadas a las soluciones de réplica de datos son imprescindibles en el proceso de replicación porque se encargan de dar vida a la réplica de datos, aunque también son importantes las acciones administrativas y de monitorización.

En *el sistema SymmetricDS* las funciones de administración se exponen a través de JMX y se puede acceder desde el *JConsole* Java o a través de un servidor de aplicaciones. Estas funciones incluyen:

- Carga de datos: este proceso se le determina Enrutamiento, que no es más que la acción de organizar y registrar las decisiones sobre hacia donde enviar los datos de una tabla, llamado *DATA_EVENT y OUTGOING_BATCH*.

- Depuración de datos antiguos: Esta función permitirá limpiar los datos capturados que ya no se necesitan en las tablas de tiempo de ejecución de la aplicación *SymmetricDS*. Esto solo se realizará sobre los datos que han sido sincronizados correctamente. Aunque la modificación realizada a los datos no enviados de un nodo discapacitado o fallido también se purga.

- Visualización de los lotes: Esta función permitirá realizar un seguimiento al envío y recepción de lotes, los cuales posibilitará el seguimiento y envío de los cambios de datos ocurridos en la sincronización entre los nodos.

- Gestión de nodos: Esta función permitirá añadir, modificar y eliminar un nodo, además de poder realizar el registro de nodos clientes a un nodo servidor y el ajuste de carga inicial a los nodos.

La aplicación *SymmetricDS* también proporciona una funcionalidad para enviar eventos de *SQL* a través del mismo mecanismo de sincronización que se utiliza para enviar datos. La carga útil de datos puede ser cualquier sentencia *SQL*. Este evento se procesa y se reconoce como cualquier otro tipo de evento.

1.5. Sistema para la administración y monitorización de la herramienta SymmetricDS

SymmetricDS es una herramienta de replicación que presenta un acelerado desarrollo debido al constante cambio de sus versiones. Según la literatura consultada se determinó que el único sistema dedicado a la administración y monitorización de la aplicación *SymmetricDS* encontrado hasta el momento es la herramienta conocida como: *SymmetricDS Pro*. En este epígrafe se describirán brevemente cada uno de los módulos que la componen.

SymmetricDS Pro

La herramienta SymmetricDS Pro [12] presenta una interfaz web [13] encargada de facilitar al usuario la configuración y gestión de la solución de réplica de datos que implementa y está compuesto por cinco módulos: Configuración, Gestión, Monitorización o Tablero de instrumentos (*Dashboard*), Explorador y la Ayuda, que apoyan y hacen que el proceso de desarrollo de una solución de réplica de datos utilizando la aplicación SymmetricDS sea más cómodo para los administradores. Seguidamente se muestra una breve descripción de los módulos que conforman este sistema donde se exponen las tareas que se pueden realizar con cada uno:

- Configuración: Permite realizar la configuración inicial de un escenario de sincronización.
- Gestión: Permite que se puedan añadir, modificar y eliminar nuevos nodos y nuevos escenarios de sincronización utilizando los mismos parámetros de la configuración inicial, donde las instancias del nodo cliente y el nodo servidor están montados en una sola instalación de *SymmetricDS* o por separado. Además se puede visualizar el estado en que se encuentran los lotes salientes y entrantes

que surgen en el proceso de envío y captura de los datos. Permite visualizar todos los procesos que son ejecutados cuando se crea una solución de réplica de datos.

- Monitorización: Permite la visualización del estado en que se encuentran los nodos, mostrando los parámetros de configuración correspondientes a cada nodo. Además permite visualizar los elementos de *hardware* y *software* necesarios para poder trabajar con el sistema *SymmetricDS*. Explorador: Se pueden visualizar cada una de las tablas de la base de dato *SymmetricDS*.

- Ayuda: Brinda al usuario información de la aplicación *SymmetricDS*, por ejemplo, se puede visualizar la versión que se está utilizando, los diferentes entornos con los que es compatible la herramienta, así como una guía para trabajar con la aplicación que permitirá configurar, administrar, y monitorear de forma correcta un escenario de sincronización.

A partir de las funcionalidades incorporadas en la herramienta *SymmetricDS,* se identificaron elementos esenciales que contribuyeron en la propuesta de solución tales como la configuración del conjunto réplica de datos, la gestión de nodos y la verificación de errores. A pesar de que *SymmetricDS* es una herramienta de código abierto, la solución *SymmetricDS* Pro posee una licencia de *software* privativo, lo que limita de esta forma su utilización. Dicha limitante fomenta la necesidad de crear una interfaz web que permita gestionar de forma adecuada a la aplicación *SymmetricDS* sin altos costos económicos.

1.6. Metodologías, herramientas y tecnologías propuestas

El uso de metodologías de desarrollo, así como las herramientas y tecnologías destinadas al desarrollo de *software* facilita obtener productos con mayor calidad que satisfagan las necesidades de los clientes a quienes van dirigidos. Al existir gran variedad de materiales que incluyen desde herramientas y tecnologías que apoyan el proceso de ingeniería, hasta los lenguajes de programación, entornos de desarrollo, servidores y marcos de trabajo. En este epígrafe se refleja una breve descripción de las herramientas y tecnologías que serán utilizadas en todo el proceso de desarrollo de la solución.

1.6.1. Metodología de desarrollo

Las metodologías para el desarrollo de *software* surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental que permitan obtener un producto de informático con calidad. Las características de cada proyecto conjuntamente con su equipo de desarrollo,

recursos, y requisitos exigen que se escoja una metodología que se adapte en mayor medida a sus particulares.

Muchas son las metodologías que existen para el desarrollo de *software* y entre las más utilizadas se encuentran *SCRUM* [14], Proceso Unificado para el desarrollo de Procesos (*RUP* por sus siglas en inglés) [15], *Microsoft Solution Framework* (MSF) [16] y Programación Extrema (*XP* por sus siglas en inglés) [14]. La metodología ágil *XP* está centrada en potenciar las relaciones interpersonales como clave para el éxito en el proceso de desarrollo de *software*, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Además brinda simplicidad en las soluciones implementadas y se define como especialmente adecuada para proyectos con requisitos imprecisos, muy cambiantes, y donde existe un alto riesgo técnico.

El equipo encargado de desarrollar la aplicación que dará solución al problema planteado en este trabajo de diploma está compuesto por dos personas y cuenta con un tiempo limitado para el desarrollo de todas las funcionalidades que requiere el sistema. Además sus integrantes no tienen la experiencia necesaria en el uso de la herramienta *SymmetricDS*, es de gran importancia establecer una estrecha relación entre el cliente y los desarrolladores para un mejor entendimiento de la misma, porque en caso de detectarse algún error en el proceso de desarrollo atrasaría la entrega a tiempo del producto.

Atendiendo a las condiciones existentes para el desarrollo de la solución que requieren de una rápida implementación de las funcionalidades, contando con un equipo de desarrollo pequeño y resultando limitado el tiempo para realizar la entrega de una primera versión funcional de la aplicación, se decide escoger la metodología *XP* para realizar todo el proceso de diseño, implementación y prueba de la solución.

1.6.2. Lenguaje de Modelado

Lenguaje Unificado de Modelado (*UML* por sus siglas en inglés) [17] se define como un lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software orientados a objetos. Se utiliza para entender, diseñar, hojear, configurar, mantener y controlar la información sobre tales sistemas en todo el ciclo de vida del desarrollo de *software*. Es un lenguaje de modelado consolidado universalmente, capaz de abstraer cualquier tipo de sistema informático. Permite expresar de una forma

gráfica cualquier aplicación informática para que lo puedan entender personas ajenas o no al equipo de desarrollo.

Con el objetivo de realizar el diseño de los diagramas que ayudaran a interpretar mejor la necesidad del desarrollo de la solución y modelar su estructura y comportamiento se emplea este lenguaje de modelado. Además es uno de los más usados en todo el mundo y con él se puede efectuar el diseño de los prototipos de interfaz que ayudaran a los programadores posteriormente en la implementación. Estos beneficios que brinda UML también permiten aumentar la calidad del producto y realizar la entrega en tiempo y con todos los requisitos cumplidos en tiempo de la primera versión de la solución.

1.6.3. Herramientas Ingeniería de Software Asistida por Computadora

Visual Paradigm - 8.0 [18] es una herramienta que soporta el ciclo de vida completo del desarrollo de *software.* Está orientada al desarrollo de aplicaciones utilizando modelado *UML*, ideal para ingenieros de *software*, analistas de sistemas y arquitectos que están interesados en la construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Utiliza un lenguaje estándar común al equipo de desarrollo que facilita la comunicación

Las ventajas que proporciona *Visual Paradimg* para *UML* son [19]:

- Facilita el modelado de *UML*, proporcionando herramientas específicas para ello. Esto también permite la estandarización de la documentación ajustándose al estándar soportado por la herramienta.
- Controla que el modelado con *UML* sea correcto.
- Al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades.
- Permite el trabajo en grupo, proporcionando herramientas de compartición de trabajo.
- Facilita la reutilización, ya que se dispone de una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos.

- Permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del programa.

Se decide utilizar *Visual Paradigm - 8.0* por que brinda soporte al lenguaje de modelado *UML*. Además apoya el trabajo en equipo y como permite la reutilización de modelos de proyectos similares es más cómodo para realizar el modelado. También reduce el tiempo en implementación y los errores en la codificación al incorporar entre sus ventajas la generación de código.

1.6.4. Lenguajes de Programación

Los lenguajes de programación son herramientas que permiten crear programas y software facilitando la tarea de programación ya que disponen de formas adecuadas que permiten ser leídas y escritas por personas que resultan independientes del modelo de computadora a utilizar.

El lenguaje de Programación Orientado a Objetos (*POO* por sus siglas en inglés) Java [20] fue ideado por un equipo de desarrollo de *Sun Microsystems* (adquirida por *Oracle Corporation*). Es un lenguaje de propósito general con gran aceptación en la web, especialmente en las aplicaciones Cliente -Servidor y el uso de *Java Server Pages* (*JSP* por sus siglas en inglés). Entre sus principales características tenemos:

- Orientado a Objeto: Java fue diseñado partiendo de cero, no es derivado de otro lenguaje y no tiene compatibilidad con ninguno de los lenguajes anteriores. En Java el concepto de objeto resulta sencillo y fácil de ampliar pues se conservan elementos como números, caracteres y otros tipos de datos simples.

- Robusto: Java verifica su código al mismo tiempo que lo escribe, y una vez más antes de ejecutarse, de manera que se consigue un alto margen de codificación sin errores.

- Multiplataforma: Java está diseñado para que un programa escrito en este lenguaje sea ejecutado correctamente independientemente de la plataforma en la que se esté actuando (*Macintosh, PC, UNIX*).

Teniendo en cuenta que *SymmetricDS* es un sistema desarrollado con el lenguaje de programación Java, se escoge también para desarrollar la interfaz web que permitirá su administración y monitorización, garantizando la compatibilidad entre ambos sistemas. Además es un lenguaje que dispone de un amplio conjunto de librerías documentadas, tiene facilidad para adicionar nuevas funcionalidades, puede ser

utilizado en cualquier sistema operativo (Multiplataforma) y puede usarse tanto para aplicaciones de tipo de escritorios como *Web.*

El lenguaje de programación JavaScript [21] que se utiliza principalmente para crear páginas *web* dinámicas, una de las principales ventajas que presenta sobre el *HTML*. Además es muy fácil de aprender y técnicamente es un lenguaje de programación interpretado por lo que no es necesario compilar los programas para ejecutarlos.

Los programas escritos en este lenguaje se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios y a pesar de que su nombre no guarda ninguna relación directa con el lenguaje de programación Java. Algunas de las principales características son las siguientes:

- Es interpretado por el navegador del cliente.
- Está basado en objetos por tanto emplea herencia, típicas de la *POO*.
- Su código se integra en las páginas *HTML* incluido en las propias páginas.
- No es necesario declarar los tipos de variables que van a utilizarse.
- Las referencias a objetos se comprueban en tiempo de ejecución, por lo tanto no se compila.

Este lenguaje será utilizado en la implementación de la interfaz *web* para la administración y monitorización de *SymmetricDS* porque permitirá brindarle al usuario una vista amigable y más dinámica. Además es compatible con distintos navegadores *web* y con él es posible realizar tareas que ayuden a ejecutar funciones de validación del lado del cliente que impidan la sobrecarga del servidor.

1.6.5. Entorno de Desarrollo Integrado (IDE)

Eclipse 4.2 [22] es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios *web*, programas en C++ o aplicaciones Java. Es un entorno de desarrollo integrado (*IDE*) que presenta una atractiva interfaz que lo hace fácil y agradable de usar. Además cuenta con un editor de texto donde se puede observar el contenido del fichero en el que se trabaja, una lista de tareas, y otros módulos similares. También proporciona herramientas para la gestión

de espacios de trabajo, escribir, desplegar, ejecutar y depurar aplicaciones y se caracteriza por ser un sistema [23]:

- Basado en perspectivas, editores y vistas.
- Permite la gestión de proyectos.
- Incluye un depurador de código.
- Almacena una extensa colección de *plug-ins.*

En esta versión nueva de Eclipse (Eclipse Juno 4.2) [24] entre otras mejoras, destacan las que proporcionan un mejor y más rápido acceso a recursos, como el campo de búsqueda rápida con el que podemos abrir nuevas vistas.

Eclipse es particularmente exigente en recursos del sistema especialmente cuando se utiliza en combinación con los servidores de aplicaciones. Si estos requisitos del sistema se aplican a la mayoría de las configuraciones del *IDE*, para los mejores resultados se aseguran de que el sistema cumpla al menos los requisitos de sistema recomendados para desarrollar aplicaciones *Java EE*.

Por tanto para el desarrollo de la interfaz para la administración y monitorización de *SymmetricDS* se utilizará el Eclipse 4.2, herramienta que es compatible con el servidor *Apache Tomcat*, el *framework Spring* y puede ser usada en entornos como *MS Windows, MacOSX* y *GNU/Linux*. Además la herramienta para la administración *SymmetricDS Pro* también fue desarrollada utilizando Eclipse como *IDE*.

1.6.6. Servidores de aplicaciones

Apache [25] es uno de los servidores *web* más utilizado en Internet. Se trata de una aplicación libre robusta y modular con una impresionante colección de funciones, pudiendo servir tanto para contenido estático como para prácticamente cualquier contenido dinámico.

El servidor *web* se encarga de la comunicación a través de la red con el navegador del usuario. Es capaz de atender muchas peticiones de forma concurrente pudiendo realizar diferentes funciones habituales entre las que se encuentran [25]:

- Registro de actividad y errores.

- Control de acceso basado en la dirección del cliente, contenido o usuario/contraseña.
- *Virtual Hosts,* para mantener diferentes *webs* (por ejemplo *http://elpuig.xeill.net y http://blog.elpuig.xeill.net*).
- *Proxy*, para reenviar las peticiones a otro servidor.
- Reescritura de *URLs*.
- Alias o mapeados de rutas.

El *Apache Tomcat* es un software de código abierto implementado para las tecnologías *Java Servlet* y *Java Server Pages*. Es desarrollado en un entorno abierto y participativo, publicado bajo la licencia del software de *Apache* y se ejecuta en varios sistemas operativos. Es un servidor configurable de diseño modular, con una diversidad que permite garantizar una elevada seguridad y buenas prestaciones. Además, brinda soporte para la plataforma de desarrollo *Java EE.* Existe bastante documentación asociada al mismo y cuenta con una gran comunidad de desarrollo siendo así uno de los más usados internacionalmente.

Su uso en el desarrollo de la solución propuesta es de vital importancia porque entre las opciones de implementación que propone *SymmetricDS* se encuentra el desarrollo de Aplicaciones de Archivado *Web* (*WAR*) implementadas en un servidor de aplicaciones como *Tomcat*, que permita compilar las clases de java *web*.

1.6.7. Plataformas de desarrollo

Una plataforma de desarrollo o marco de trabajo (*framework)* es un producto que se utiliza como base para la programación avanzada de aplicaciones que aporta una serie de funciones o códigos para realizar tareas habituales. Lo conforman un conjunto de librerías de código que contienen procesos o rutinas ya listos para usar. Los programadores lo utilizan para no tener que desarrollar manualmente las tareas básicas porque ya incorpora implementaciones que están probadas, funcionan y no se necesitan volver a programar. Además permiten desarrollar sistemas en menos tiempo, más robustos y con menos cantidad de errores.

Spring 3.0.4 [26]**:** *E*s uno de los *framework* más usados actualmente y aunque se encuentra dividido en distintos módulos, cada uno se encarga de partes diferentes de nuestra aplicación y pueden ser utilizados en caso que se necesite. Además es tan grande que para el desarrollo de aplicaciones pequeñas o medianas no sería necesario el uso de todos sus módulos. Precisamente como la interfaz web que se desea implementar cumple estas características solo serán utilizados los siguientes módulos:

- *Core*: Es el modulo central de todo el framework y es el que provee las funcionalidades principales. Contiene el *BeanFactory* el cual es el contenedor fundamental de *Spring* y la base en la que se basa la Inyección de Dependencias de *Spring*.

- *Context*: Este módulo extiende el concepto de *BeanFactory*, añadiendo soporte para los mensajes de internacionalización (utilizando, por ejemplo, los paquetes de recursos), evento de propagación, carga de recursos y la creación transparente de contextos, por ejemplo, un contenedor de *servlets*. El *ApplicationContext* interfaz es el punto central del módulo de contexto.

- Módulo *web* de *Spring*: Proporciona funciones de integración básicas de web como la funcionalidad de carga de archivos de varias partes y la inicialización del contenedor *IoC* usando oyentes *servlet* y un contexto de aplicación orientado a la web. También contiene las partes relacionadas con la web de soporte de la comunicación remota de Spring.

- Módulo *Web-Servlet*: Contiene el modelo-vista-controlador (MVC en lo adelante) para la implementación de aplicaciones *web*. *Framework* MVC de Spring proporciona una separación limpia entre el código del modelo de dominio y los formularios *web*, y se integra con todas las otras características de *Spring Framework*.

- Módulo *Web de portlets*: Proporciona la implementación MVC para ser utilizado en un entorno de módulo de función y refleja la funcionalidad del módulo *Web-Servlet*.

Entre las características más importantes que presenta y que lo distinguen de otros *framework* se encuentran [27]:

- Inyección de dependencia: consiste en lugar de que sean las clases las encargadas de crear (instanciar) los objetos que van a usar (sus atributos), los objetos se inyectaran mediante los métodos *setters* o mediante el constructor en el momento en el que se cree la clase.

- Programación orientada a aspectos que incluye la gestión de transacciones declarativa de *Spring.*
- Aplicación web *Spring MVC* y marco de servicios *web Restfull*.
- Apoyo fundacional para *JDBC, JPA, JMS*.

Spring es un *framework* de código abierto y fue creado para el desarrollo de aplicaciones hechas en Java pero por incluir muchas extensiones y mejoras para construir aplicaciones basadas en web por encima de la plataforma empresarial de Java se utilizará para la implementación de la interfaz web de administración y monitorización de *SymmetricDS*.

jQuery 1.3.4 [28]**:** Generalmente cuando un desarrollador utiliza *Javascript*, tiene que preocuparse por hacer *scripts* compatibles con varios navegadores (*Internet Explorer, Firefox, Opera*) y para ello debe incorporar mucho código. *jQuery* es un *framework* para el lenguaje *Javascript* que implementa una serie de clases (de programación orientada a objetos) que facilitan la implementación del navegador que utilice el usuario, ya que funcionan de forma exacta en todas las plataformas más habituales.

Este *framework JavaScript*, ofrece una infraestructura que permite crear aplicaciones complejas del lado del cliente con mayor facilidad. Por ejemplo, con *jQuery* se pueden realizar proyectos que incluyan la creación de interfaces de usuario y efectos dinámicos. También incluye librerías que brindan al programador obtener productos en menos tiempo y libre de errores. Además es un *framework* desarrollado bajo las licencias de software libre y esta característica es vital para obtener la solución que propone este trabajo de diploma.

jQuery es un producto con una aceptación muy buena por parte de los programadores y un grado de penetración en el mercado muy amplio, lo que hace suponer que es una de las mejores opciones. Además, es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del *framework*.

Bootstrap 2.3.2 [29]**:** *E*s un *framework* diseñado para simplificar el proceso de creación de diseños web. Para ello ofrece una serie de plantillas *CSS* y de ficheros *JavaScript* que permiten conseguir interfaces que funcionen correctamente en los navegadores actuales y no tan actuales. Esta herramienta brinda:

- Un diseño que pueda ser visualizado de forma correcta en distintos dispositivos y a distintas escalas y resoluciones.

- Una mejor integración con las librerías que utilizan habitualmente, como por ejemplo *jQuery.*
- Un diseño sólido basado en estándares como *CSS3/HTML5*.

Adicionalmente, si se desea que el diseño sea capaz de adaptarse a los distintos navegadores, configuraciones o incluso dispositivos, la selección de un *framework* de apoyo puede garantizar el ahorro de mucho tiempo y esfuerzo. Además el *Bootstrap* [29] es un software de código abierto por lo que puede ser utilizado sin pagar por su uso.

El uso de este marco de trabajo en el desarrollo de la interfaz para la administración y monitorización de *SymmetricDS* garantizará que el diseño de la aplicación pueda ser visualizado en cualquier navegador donde se ejecute y se ahorrará tiempo y esfuerzo en su implementación.

1.7. Consideraciones del capítulo

Una vez sistematizados los principales conceptos utilizados en la administración y monitorización de las soluciones de réplica de datos, analizados *SymmetricDS* para ver su comportamiento en la gestión de soluciones de réplica de datos, *SymmetricDS Pro* como sistema de gestión para la aplicación *SymmetricDS* y las herramientas y tecnologías a utilizar para el desarrollo de la solución propuesta se puede concluir que:

- Las soluciones de réplica de datos son identificadas como sistemas de replicación de datos, que tiene como objetivo fundamental mantener sincronizadas varias fuentes de datos optimizando tanto recursos de red como de hardware.
- Además SymmetricDS Pro es un sistema que permite la gestión de la herramienta SymmetricDS, pero está liberado bajo licencia privativa limitando su libre utilización.
- Las herramientas y tecnologías escogidas para el desarrollo de la solución fueron Visual Paradigm - 8.0 para el lenguaje de modelado UML, IDE de desarrollo Eclipse 4.2, servidor web Apache Tomcat 7.0.3, framework de desarrollo Spring, jQuery 1.3.4 y Bootstrap 2.3.2, como lenguajes de programación del lado del cliente JavaScript y del lado del servidor Java. Todas cumplen con la característica de presentar licencias libres, lo que garantiza su utilización con un mínimo de recursos materiales y financieros, así como la no dependencia tecnológica con otros países desarrollados.

- Se seleccionó XP como metodología para realizar todo el proceso de diseño, implementación y prueba de la solución atendiendo a que se requiere de una rápida implementación de las funcionalidades para realizar la entrega de una primera versión funcional de la aplicación en el tiempo establecido.

Capítulo 2: Análisis y diseño de la solución propuesta

Al concluir con el estudio del marco teórico relacionado con la administración y monitorización de las herramientas de réplica de datos, el análisis de *SymmetricDS Pro* como herramienta de administración de *SymmetricDS*, la definición de la metodología de desarrollo y las herramientas para la implementación de la solución se procede a realizar un análisis de los módulos que componen la solución propuesta por este trabajo de diploma.

Además se efectuará una exposición de los artefactos generados atendiendo a las fases de la metodología *XP* (Exploratoria, Planificación de la entrega) donde los clientes plantean las Historias de Usuario y se establece la prioridad de cada una, seguidamente los programadores realizan la estimación del esfuerzo que se empleará en el desarrollo de las Historias de Usuarios y se define el plan de que engloba las iteraciones que divide la implementación de la solución propuesta.

También se confeccionan las tarjetas CRC y se define el diseño arquitectónico de la solución identificando los patrones de diseño que debe cumplir la implementación. Todo lo anteriormente expuesto debe garantizar obtener en el tiempo estimado y con la calidad esperada una primera versión del sistema.

2.1. Descripción de la solución

La instalación de una solución de réplica de datos se realiza junto a la de *SymmetricDS* en el mismo centro de datos, esto trae como consecuencia que el sistema a implementar para dar cumplimiento al objetivo de este trabajo de diploma también se instale en el mismo centro de datos y su interfaz pueda ser accesible desde cualquier punto de red, de esta forma los administradores de la réplica de datos no necesitan realizar configuraciones a SymmetricDS obligatoriamente desde la PC servidor.

Considerando estas condiciones que debe cumplir el entorno donde se despliegue la solución se determinó que la interfaz para la administración y monitorización de *SymmetricDS* se construirá bajo los principios del desarrollo web que proponen varias ventajas como [30]:

- Las aplicaciones web son más sencillas y económicas de costear que las aplicaciones de escritorio porque para su desarrollo no hacen falta instalar ningún programa en los ordenadores de los

usuarios, solamente deben tener un navegador web (Chrome, Internet Explorer, Firefox) y además poseer conexión a internet.

- Pueden ejecutarse en cualquier sistema operativo como Windows o Linux sin restricciones.
- Su costo de mantenimiento no es elevado porque solo se necesita actualizar la aplicación en el servidor.

Para garantizar lo antes expuesto la aplicación se divide en varios módulos que agruparán todas las funcionalidades del sistema.

El módulo de *Instalación* será el encargado de garantizar la correcta instalación de *SymmetricDS*. Para lograrlo el sistema debe:

- Comprobar que el equipo (PC) cuenta con la Máquina Virtual de Java (*JDK* por sus siglas en inglés).
- Permitir al usuario extraer la carpeta con la versión de *SymmetricDS* que se usará y guardarla en el directorio que desee.
- Permitir al usuario el registro de una cuenta Admin para poder acceder al sistema.

El módulo *Gestión de Nodo* será el encargado de gestionar los nodos de la solución de réplica de datos y realizar el registro de nodos clientes para que puedan recibir la carga inicial.

- Para los nodos el sistema debe permitir: Adicionar un nuevo nodo, Editar configuración de un nodo, Registrar un nodo, Visualizar listado de nodos existentes.

El módulo *Gestión de Set de réplica* será el encargado de realizar la gestión de un escenario de réplica de datos además de permitir exportarlos e importarlos. Para los Set de réplica el sistema debe permitir:

- Para la gestión de grupos de nodos: Adicionar nuevos grupos, Editar configuración de los grupos existentes, Visualizar listado de grupos creados.
- Para la gestión de enlaces entre grupos de nodos: Adicionar un nuevo enlace, Editar configuración de los enlaces existentes, Visualizar listado de los enlaces creados.

- Para la gestión de canales: Adicionar un nuevo canal, Editar configuración de los canales existentes, Visualizar listado de canales creados.
- Para la gestión de disparadores: Adicionar un nuevo disparador, Editar configuración de los disparadores existentes, Vincular disparador a un router, Visualizar listado de disparadores creados.
- Para la gestión de router: Adicionar un nuevo router, Editar configuración de los routers existentes, Visualizar listado de router creados.

El módulo de *Monitorización* es el encargado de controlar los procesos de la réplica de datos notificando los errores que pueden surgir en ellos.

- Las funcionalidades que formarán parte de él serán: Visualizar los logs de SymmetricDS, verificar los lotes erróneos, verificar el estado de los nodos, reiniciar SymmetricDS.

El módulo de *Ayuda* es el encargado de brindar al usuario la información necesaria para que pueda interactuar con el sistema. Para ello debe:

- Permitir la visualización de un Manual de usuario que contiene información de cómo y cuándo utilizar cada una de las funcionalidades del sistema.

2.2. Modelo de Dominio

XP como metodología para el desarrollo de *software* gira más su atención a los procesos que guiarán la implementación del sistema. Entre los artefactos que genera no es visible el diagrama “Modelo del Dominio”, pero se utiliza con el objetivo de lograr un mejor entendimiento de los principales conceptos que se manejan en el negocio. A continuación se puede observar en la “Fig. 1. Modelo de Dominio” la representación de la necesidad de desarrollar la solución propuesta, en un diagrama de dominio.

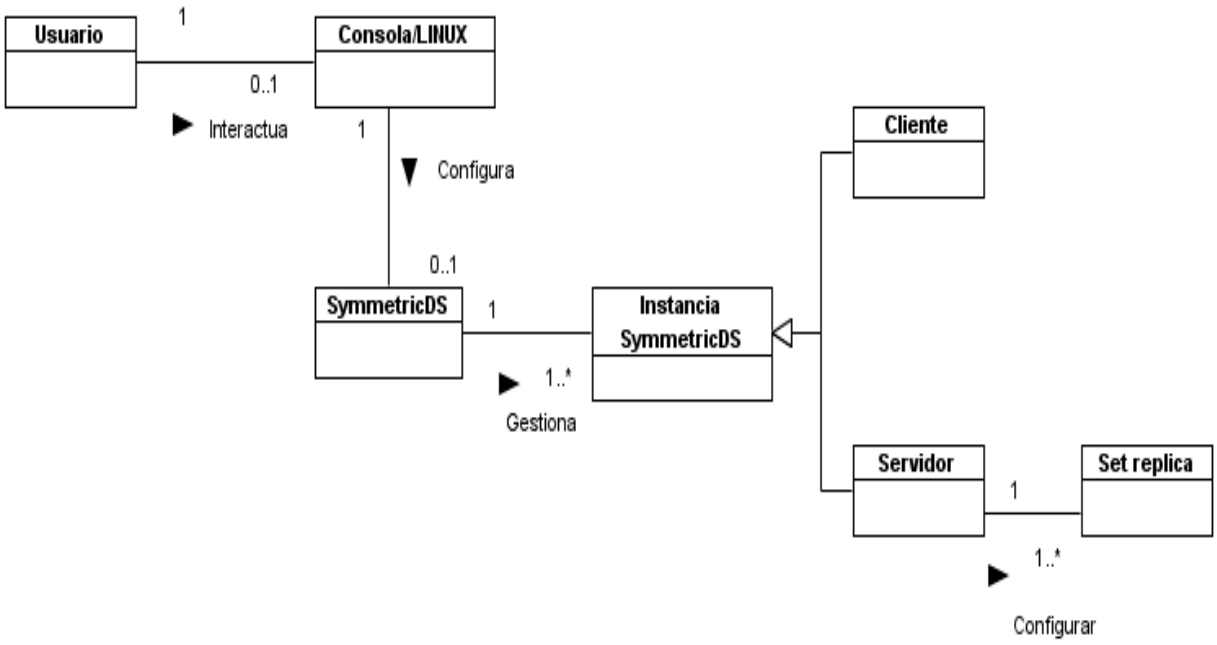


Fig. 1. Modelo de Dominio

- Usuario: Persona encargada de realizar la configuración a la solución de réplica de datos.
- Consola *GNU/Linux*: Sistema que permitirá la interacción del usuario con herramienta SymmetricDS para realizar la configuración de la solución de réplica de datos.
- *SymmetricDS*: Herramienta de réplica de datos que permite la gestión de la solución de réplica de datos.
- Instancia-*SymmetricDS*: Nodo creado usando como herramienta de réplica de datos a SymmetricDS.
- Nodo Servidor: Instancia de un nodo de tipo Servidor que hereda el comportamiento de Instancia-*SymmetricDS*.
- Nodo Cliente: Instancia de un nodo de tipo Cliente que hereda el comportamiento de Instancia-*SymmetricDS*.

- Set de Réplica: Escenario de sincronización que se configura para darle vida a la réplica de datos, donde se define el tipo de réplica, la forma de capturar y almacenar los cambios y con cuales plataformas de bases de datos será compatible la réplica.

2.3. Historias de usuario

Las historias de usuario [31] constituyen la técnica empleada por la metodología *XP* para representar una breve descripción de los requisitos del sistema y se redactan por los clientes en la fase de exploración. Luego se dividen en funcionalidades del sistema y se calcula el tiempo necesario para su implementación. También difiere de los casos de uso porque son escritos por el cliente, no por los programadores. Las historias de usuario transcurren por tres etapas fundamentales [31]:

- Tarjetas: en ella se almacena suficiente información para identificar y detallar la historia.
- Conversación: cliente y programadores discuten la historia para ampliar los detalles (verbalmente cuando sea posible, pero documentada cuando se requiera confirmación).
- Pruebas de Aceptación: permite confirmar que la historia ha sido implementada correctamente.

Luego del análisis realizado, con el objetivo de determinar los requisitos funcionales que corresponden a la aplicación que se necesita desarrollar, se determinaron 32 Historias de Usuario. A continuación se muestra la “Tabla 1. Historia de Usuario Adicionar Nodo” que contiene la Historia de Usuario “Adicionar Nodo¨, las restantes pueden ser consultadas en la “Planilla de Historias de Usuario” que se encuentra en el Expediente de Proyecto adjunto a este trabajo de diploma.

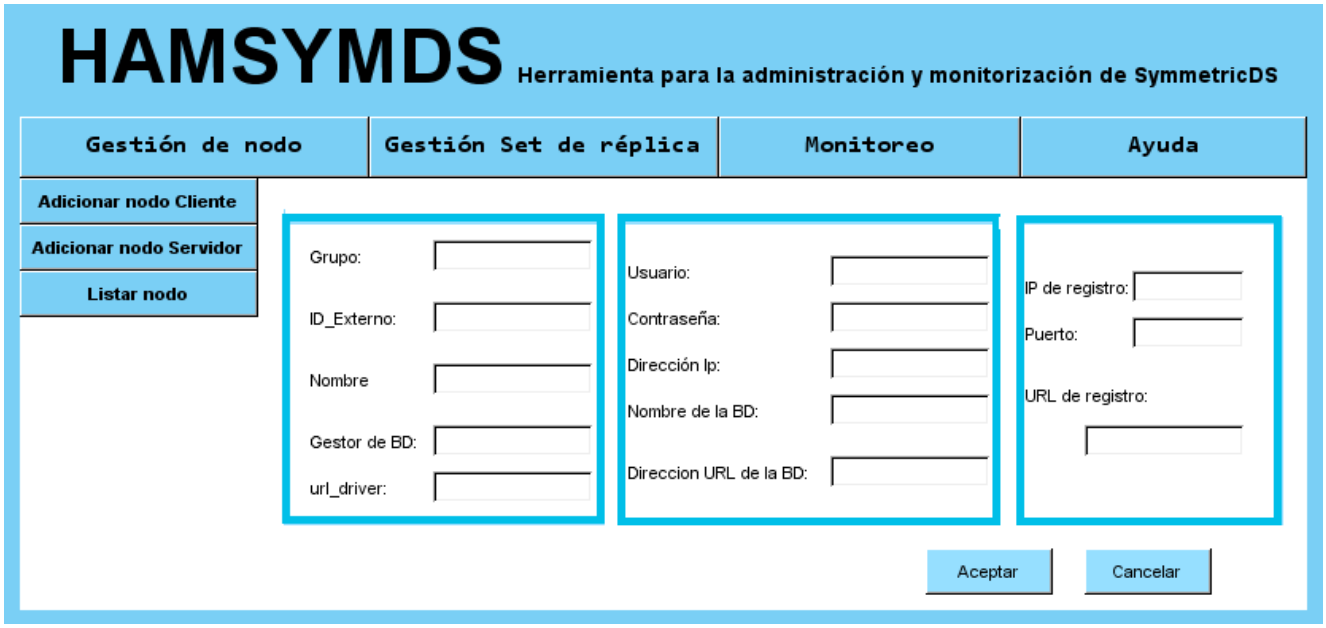
Tabla 1. Historia de Usuario Adicionar Nodo

Historia de Usuario	
Número: 6	**Nombre:** Adicionar Nodo
Cantidad de modificaciones: 1	
Usuario: Adrián Misael Peña Montero	**Iteración asignada:** 1
Prioridad en negocio: Muy Alta	**Puntos estimados:** 0.8 semanas
Riesgo en desarrollo: Muy Alto	**Puntos reales:** 0.6 semanas

Descripción: Permite adicionar un nodo a la réplica ya sea “Cliente” o “Servidor” insertando el nombre del nodo, grupo de nodos al que pertenece, dirección url, dirección de registro, usuario y contraseña de la base de datos. Si se inserta correctamente el nodo el sistema mostrará un mensaje.

Observaciones: El usuario solo podrá adicionar un solo nodo Servidor y cuantas instancias de nodo Cliente necesite para la réplica. En caso de no insertarse el nuevo nodo el sistema debe mostrar un mensaje indicando.

Prototipo no funcional de interfaz web:



2.3.1. Lista de reserva del producto

La lista de reserva del producto es una tabla que brinda información relacionada con los requisitos del sistema ya sean funcionales o no funcionales. Los primeros se organizan según la prioridad que establece el cliente para cada historia de usuario (Muy Alta, Alta, Media, Baja) de acuerdo con el valor que aportan para el negocio y la estimación del esfuerzo en el desarrollo que realizan los programadores. Los segundos se agrupan según su clasificación. Además se muestra el tiempo estimado para la

implementación de cada funcionalidad y quien realizó la estimación. Este artefacto se genera durante la planificación de la entrega, en la planificación de cada iteración y cuando sea necesario reconducir el proyecto.

A continuación se muestra la lista de reserva del producto como se puede observar en la “Tabla 2. Lista de reserva del producto” perteneciente a la solución:

Tabla 2. Lista de reserva del producto

Item *	Descripción	Estimación	Estimado por
Prioridad: Muy Alta			
1	Comprobar propiedades del sistema	0.4 semanas	Analista
2	Cargar versión de SymmetricDS	0.4 semanas	Analista
3	Seleccionar directorio	0.4 semanas	Analista
4	Registrar Usuario	0.4 semanas	Analista
5	Autenticar Usuario	0.4 semanas	Analista
6	Adicionar un nuevo nodo	0.8 semanas	Analista
7	Editar configuración de un nodo	0.6 semanas	Analista
8	Visualizar listado de nodos existentes	0.2 semanas	Analista
9	Adicionar nuevos grupos	0.8 semanas	Analista
10	Editar configuración de los grupos	0.6 semanas	Analista
11	Visualizar listado de grupos existentes	0.4 semanas	Analista
Prioridad: Alta			
12	Adicionar un nuevo enlace	0.6 semanas	Analista
13	Editar configuración de los enlaces	0.4 semanas	Analista
14	Visualizar listado de los enlaces existentes	0.2 semanas	Analista
15	Adicionar un nuevo canal	0.6 semanas	Analista
16	Editar configuración de los canales	0.4 semanas	Analista
17	Visualizar listado de canales creados	0.2 semanas	Analista
18	Adicionar un nuevo disparador	0.6 semanas	Analista
19	Editar configuración de los disparadores	0.4 semanas	Analista

20	Visualizar listado de disparadores creados	0.2 semanas	Analista
21	Adicionar un nuevo router	0.6 semanas	Analista
22	Editar configuración de los routers	0.4 semanas	Analista
23	Visualizar listado de routers creados	0.2 semanas	Analista
24	Poblar base de datos	0.6 semanas	Analista
25	Identificar nodo	0.4 semanas	Analista
Prioridad: Media			
26	Registrar un nodo	0.4 semanas	Analista
27	Visualizar los logs de SymmetricDS	0.6 semanas	Analista
28	Verificar los lotes erróneos	0.6 semanas	Analista
29	Verificar estado de los nodos	0.6 semanas	Analista
30	Reiniciar el servicio SymmetricDS	0.6 semanas	Analista
Prioridad: Baja			
31	Vincular disparador a un router	0.4 semanas	Analista
32	Mostrar la Ayuda del sistema	0.4 semanas	Analista
Requisitos No Funcionales			
1	**Usabilidad:** El usuario debe tener conocimiento previo del trabajo con la herramienta SymmetricDS para interactuar de forma correcta con el sistema.		Analista
2	**Seguridad:** Garantizar que el sistema solicite usuario y contraseña como credencial para la autenticación en el sistema. El sistema validará la contraseña para un usuario Admin y será almacenada en un fichero cifrado. En el caso del usuario y contraseña de las bases de datos de cada instancia de SymmetricDS también serán almacenados en ficheros cifrados.		Analista

3	**Software:** El equipo PC servidor debe tener instalado PostgreSQL v 9.1, la Máquina Virtual de Java (JDK) y el servidor de aplicaciones Apache Tomcat v 7.0.3, y estar montado en el sistema operativo Linux o Windows. El equipo PC cliente debe contar con el navegador web Mozilla Firefox v 20.		Analista
4	**Hardware:** El equipo PC servidor debe tener como mínimo 2 GB de memoria RAM, 4 GB de espacio libre en el Disco y microprocesador de 1.5 GHz o superior. Debe estar conectado a una unidad de red. El equipo PC cliente debe tener como mínimo 1 GB de memoria RAM, 1 GB de espacio libre en el Disco y microprocesador de 1.5 GHz o superior. Debe estar conectado a una unidad de red.		Analista
5	**Apariencia o interfaz externa:** La interfaz debe resultar fácil de utilizar por parte del usuario, intuitiva y cumplir con la resolución 1024x768.		Analista
6	**Disponibilidad:** El sistema solo estará disponible si existen conexiones de red.		Analista

2.3.2. Plan de iteraciones

Cuando se concluye el proceso de definir las Historias de usuario en la fase de planificación para guiar la etapa de implementación de la solución se procede a desarrollar el plan de iteraciones. Este plan se construye en la fase de Iteración y consiste en un documento que muestra la cantidad de Historias de

Usuario definidas por el cliente en cada iteración y el tiempo estimado para completar las iteraciones, con el objetivo de planificar el periodo de implementación del sistema.

Para el desarrollo de la primera versión del sistema se estiman 15 semanas para la implementación. En la "

Tabla 3. Plan de **iteraciones**" que se muestra a continuación se evidencia el plan de iteraciones propuesto:

Tabla 3. Plan de iteraciones

No	Descripción de la iteración	Historias a implementar	Duración total
1	Se desarrollarán 11 Historias de usuario de prioridad Muy Alta por su importancia para el negocio, relacionadas con la gestión de nodos y la gestión de grupos de nodos.	1, 2, 3, 4, 5, 6, 7, 9, 10, 11 y 12	5.0 semanas
2	Se desarrollarán 14 Historias de usuario de prioridad Alta relacionadas con la gestión de enlaces, canales, router y la resolución de conflicto, acciones que permitirán establecer la comunicación entre los nodos y los grupos de nodos.	13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26 y 27	5.8 semanas
3	Se desarrollaran 5 Historias de usuario de prioridad Media relacionadas con las acciones de monitorización de réplica de datos y una breve ayuda que permitirá a los usuarios un mejor trabajo con la herramienta.	8, 29,30,31 y 32	2.8 semanas
4	Se desarrollaran 2 Historias de Usuario de prioridad Baja que permitirán desarrollar acciones como vincular los disparadores a	25 y 33	0.8 semanas

	los routers, importar y exportar las configuraciones realizadas a los Set de réplica de datos de datos de datos de datos de datos de datos de datos de datos de datos de datos de datos.		

2.4. Tarjetas CRC

La utilización de tarjetas Clase-Responsabilidad-Colaboración (CRC por sus siglas en inglés) [32] es una técnica de diseño que utiliza la metodología XP para la representación del sistema y son confeccionadas y mejoradas por los programadores en la fase de Iteración. El objetivo de la misma es hacer, mediante tarjetas, un inventario de las clases que se necesitarán para implementar el sistema y las relaciones entre ellas. De esta forma se pretende facilitar el análisis y discusión de las mismas por parte de varios actores del equipo de proyecto con el fin de garantizar que el diseño sea lo más simple posible verificando las especificaciones del sistema. Para confeccionar estas tarjetas se tienen en cuenta el nombre de la clase, su responsabilidad; en este caso que acciones se pueden realizar haciendo uso de los métodos que implementa, y la colaboración o relación que presenta con otras clases del sistema.

En la “Tabla 4. Tarjeta CRC-Clase Interfaz_” se muestra la Tarjeta CRC correspondiente a la clase Interfaz_Administracion del módulo Gestión de Nodos, quien se relaciona con las clases Interfaz_Principal, NodoCliente, NodoServidor, Controlador_Administracion es la encargada de brindar al usuario la posibilidad de gestionar los nodos. Las tarjetas restantes que corresponden a las demás clases de la aplicación se encuentran en la “Planilla de Tarjeta CRC” dentro del expediente de proyecto adjunto a este Trabajo de Diploma.

Tabla 4. Tarjeta CRC-Clase Interfaz_Administracion

Tarjeta CRC	
Clase: controlador_Administracion (Módulo Gestión de Nodos)	
Responsabilidades	**Colaboraciones**
• Gestionar los nodos, garantiza adicionar, editar, listar los nodos. Es la encargada de interactuar	• nodoCliente

con el modelo o base de datos	• nodoServidor • interfaz_Administracion • DAOService

2.5. Arquitectura base de la aplicación

La Arquitectura de Software es la forma de estructurar el sistema teniendo en cuenta un conjunto de patrones que proporcionan una guía a través del ciclo de desarrollo del software, permitiendo a analistas, diseñadores, y programadores seguir una línea de trabajo en común para poder alcanzar los objetivos del sistema. De acuerdo Roger S. Pressman [33], la arquitectura de software no es otra cosa que "(…) una descripción de los subsistemas y los componentes de un sistema informático y las relaciones entre ellos. De igual manera, la arquitectura de software de tres niveles, incluye todos estos aspectos, y además, brinda mejores opciones para proyectos informáticos de gran alcance y complejidad".

2.5.1. Patrones de arquitectura

Para la creación del sistema se utilizó el *framework Spring* el cual propone una arquitectura Modelo - Vista - Controlador (MVC) [34], para una mejor organización y manejo de las vistas (.jsp). Este patrón separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Se puede observar frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página; el modelo es la representación específica de la información con la cual el sistema opera y la lógica de negocio; y el controlador es el responsable de recibir los eventos de entrada desde la vista. Se fundamenta en separar cada uno de estos tres componentes en elementos independientes y claramente diferenciados. De esta manera, el diseño gana en cuanto a reutilización, ya que el desacople de elementos garantiza poder modificarlos con mayor facilidad y de forma independiente.

La "Fig 2. Diagrama de clases (Patrón MVC)" muestra el diagrama de clases correspondiente al proceso del negocio "Adicionar Nodo" diseñado siguiendo el patrón MVC, donde se puede observar las relaciones entre cada una de sus clases.

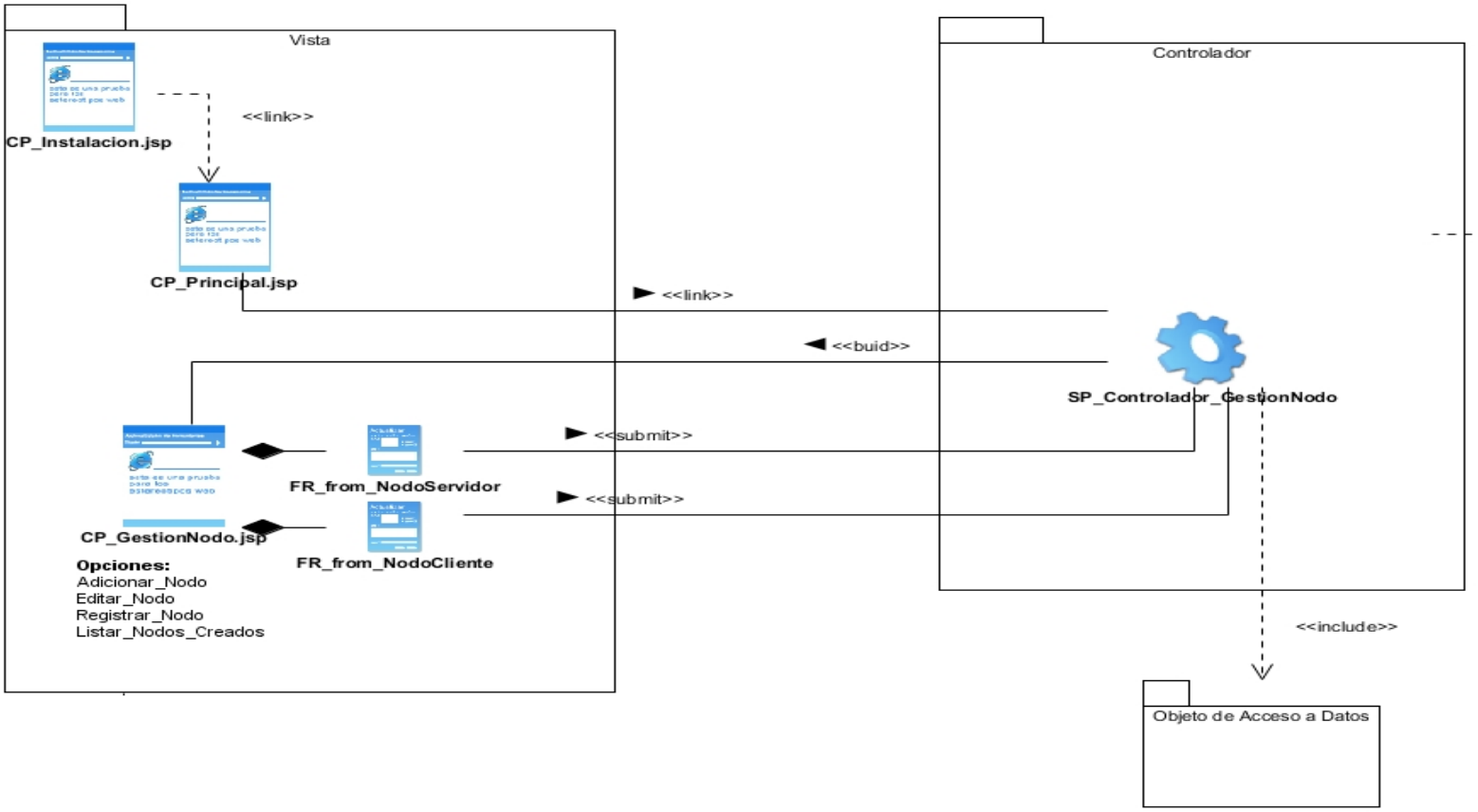


Fig 2. Diagrama de clases (Patrón MVC)

2.5.2. Patrones de diseño

Los patrones de diseño [35] constituyen el esqueleto de las soluciones a problemas comunes en el desarrollo de software y facilitan la reutilización de arquitecturas y diseños de sistemas exitosos. La aplicación de un patrón de diseño no tiene efectos sobre la estructura fundamental del sistema (arquitectura), pero puede tener una fuerte influencia sobre la arquitectura de un subsistema.

Patrones GRASP: Describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

- Controlador: El patrón controlador permite facilitar la centralización de actividades, delegar las actividades en otras clases con las que mantiene un modelo de alta cohesión. El uso de este patrón se pone de manifiesto en la utilización de diversos controladores en el sistema como por ejemplo controladorAdministracion, controladorConfiguracion. Los controladores mencionados anteriormente son los encargados de interactuar con el modelo y enviar una respuesta a la vista. En la “Fig 3 Patrón Controlador”

```
public class ControladorConfiguracion {
    @RequestMapping(value="configuracion.htm", method=RequestMethod.GET)
    protected String mostrarPaginaConfiguracion(HttpServletRequest request, HttpServletResponse hsr1)  {
        request.setAttribute("module_config", true);
        return "configuracion";
    }
```

Fig 3 Patrón Controlador

- Creador: La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. Guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental del patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. La “Fig 4 Patrón Creador” muestra la utilización del patrón en el sistema:

```
public class FileManager {
    private List<String> res;
    public static String URL_BASE = System.getProperty("user.home") + File.separator;
    private static final String EXTENCION = ".properties";
    public static List<String> getDirectorio(String directorio) {
        List<String> res = new ArrayList<String>();
```

Fig 4 Patrón Creador

- Experto: Para el desarrollo de un sistema se pueden definir muchas clases de software, y una aplicación tal vez requiera el cumplimiento de varias responsabilidades. Si estas se asignan en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y da la oportunidad de reutilizar los componentes en futuras aplicaciones. Este patrón se usa para asignar responsabilidades a clases que cuentan con la información necesaria para cumplirlas. Un ejemplo de cómo se evidencia en el desarrollo de la aplicación se muestra en la “Fig 5 Patrón Experto”:

```
public class DAOService {
    public static Connection getConnection() {...}
    public static void adicionarGrupoDeNodo(){...}
    public static List<GrupoNodo> obtenerGruposDeNodosDe() {...}
    public static boolean actualizarGrupoDeNodoDe() {..}
    public static void adicionarCanal() {...}
    public static List<Canal> obtenerCanalesDe() {...}
    public static boolean actualizarCanalDe(){...}
    public static void adicionarEnlace() {...}
    public static List<Enlace> obtenerEnlacesDe() {...}
    public static boolean actualizarEnlaceDe(){...}
    public static void adicionarTrigger(){...}
    public static List<Trigger> obtenerTriggersDe(ParametrosConexion cxn) {...}
    ...
```

Fig 5 Patrón Experto

- Bajo acoplamiento: Significa que una clase no depende de muchas clases. Este tipo de patrones no se afectan por cambios de otros componentes, son fáciles de entender por separado y fáciles de reutilizar.

- Alta Cohesión: Caracterizan a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Mejoran la claridad y facilidad con que se entiende el diseño, se simplifica el mantenimiento y las mejoras de funcionalidad. Genera un bajo acoplamiento y soporta mayor capacidad de reutilización.

Patrones GoF: Son utilizados para marcar un diseño en la implementación de la solución fueron escogidos atendiendo al framework Spring:

- DAO: Este patrón cuenta con diversas fuentes de datos (base de datos, archivos, servicios externos), de tal forma que se encapsula la forma de acceder a los datos. Se trata de que el software cliente se centre en los datos que necesita y se olvide de cómo se realiza el acceso a los datos o de cuál es la fuente de almacenamiento. En la “Fig 6 Patrón DAO” se muestra un ejemplo:

```
import com.web.symetric.bus.dao.DAOService;
public class ConfiguracionServicioAjax {
@RemoteMethod
    public static void adicionarGrupoNodo(String artefacto, String id,
            String desc) {
        ParametrosConexion cxn;
        try {
            cxn = FileManager.obtenerConexionDeArtefacto(artefacto);
            System.out.println(cxn.toString());
            DAOService.adicionarGrupoDeNodo(cxn, id, desc);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Fig 6 Patrón DAO

- Inyección de dependencia: Es un patrón de diseño orientado a objetos, en el que se inyectan objetos a una clase en lugar de ser la propia clase quien cree el objeto. Es no instanciar las dependencias explícitamente en su clase, sino declarativamente expresarlas en la definición de la clase. La esencia de la inyección de dependencias es contar con un componente capaz de obtener instancias válidas de las dependencias del objeto y pasárselas durante la creación o inicialización del objeto. La “Fig 7 Patrón Inyección de dependencia” muestra cómo se evidencia el patrón en el desarrollo del sistema:

```
<bean id="addGestionarSetReplica"
    class="com.web.symetric.bus.controllers.ControladorSetReplica">
    <property name="successView" value="/WEB-INF/jsp/SetReplica.jsp" />
    <property name="formView "value="/WEB-INF/jsp/SetReplica.jsp"/>
    <property name="SetReplica" ref="beanSetReplica"/>

</bean>
<bean id="beanSetReplica"
    class="com.web.symetric.bus.vo.GestionarGrupoNodo">
        <property name="DaoService" value="DaoService" />
</bean>
<bean id="DAOService"
    class="com.web.symetric.bus.dao.DaoService">
        <property name="GrupoNodo" value="GrupoNodo" />
</bean>
```

Fig 7 Patrón Inyección de dependencia

2.6. Consideraciones del capítulo

A partir de efectuar las actividades propias del análisis y diseño de la solución interfaz web para la administración y monitorización de *SymmetricDS* y de generar los artefactos correspondientes, se arriba a las siguientes conclusiones:

- Se analizaron los procesos fundamentales vinculados al desarrollo de la interfaz web para la administración y monitorización de SymmetricDS, y se describió la solución propuesta.
- Para facilitar el entendimiento del proceso de desarrollo de la solución se adoptaron artefactos de otras metodologías como el diagrama de dominio y el diagrama de clases.
- Identificaron 32 historias de usuario y con ellas 32 requisitos funcionales, agrupados junto a los requisitos no funcionales en la lista de reserva del producto, determinando la prioridad y el tiempo estimado para la implementación de cada uno. Además se definió el plan de iteraciones delimitando un tiempo de desarrollo de 15 semanas, y las tarjetas CRC para definir las responsabilidades de cada una de las clases del sistema y sus relaciones con otras clases.
- Se empleó Modelo-Vista-Controlador como patrón arquitectónico, y entre los patrones de diseño se utilizaron los patrones GRASP: Experto, Creador, Alta cohesión y Bajo acoplamiento, además de los patrones GoF: Inyección de dependencia, DAO

Capítulo 3: Implementación y validación de la solución

Una vez concluido el modelo del diseño, se dispone de los detalles suficientes para proceder a la construcción del sistema, y una vez concluido este, se procede a la verificación del cumplimiento de los requisitos funcionales mediante las pruebas de software. En el presente capítulo se exponen las tareas de ingeniería diseñadas con el objetivo de dividir las Historias de Usuarios en tareas de implementación y realizar una mejor planificación del trabajo de los programadores y se sintetizan los estándares de codificación que permitirán a los programadores un proceso de implementación más cómodo, rápido y con la menor cantidad de errores. Además se determinan los tipos de pruebas a realizar y los casos de pruebas que serán aplicados al sistema.

3.1. Tareas de ingeniería

Las Tareas de ingeniería surgen de un proceso de revisión y análisis realizado por parte del equipo de desarrollo a las Historias de Usuarios definidas con anterioridad, con el objetivo de dividir las funcionalidades que engloba cada Historia de Usuario en tareas más pequeñas, tributando al correcto funcionamiento de la funcionalidad y una mejor implementación.

Cada una de las tareas están compuestas por su número, el número de Historia de Usuario de la que se deriva, el nombre de la tarea, tipo, la fecha de inicio y fecha de fin, el tiempo estimado, quién es el responsable de implementarla y una breve descripción. Por ejemplo en la “Tabla 5 Tarea de Ingeniería: Implementar adicionar nodo Cliente” y “**¡Error! No se encuentra el origen de la referencia.**” se muestran las tareas de ingeniería correspondiente a la Historia de Usuario “Adicionar nodo”.

Tabla 5 Tarea de Ingeniería: Implementar adicionar nodo Cliente

Tarea de Ingeniería	
Numero de tarea: 1	**Numero de Historia de Usuario:** 6
Nombre de la tarea: Diseñar interfaz gráfica para adicionar un nodo.	
Tipo de Tarea : Desarrollo	**Estimados:** 0.4 semanas

Fecha de Inicio: 24/1/2014	**Fecha de Fin:** 28/1/2014
Programador responsable: Alfredo Sabina Diez	
Descripción: Se realiza un diseño de la interfaz para que el usuario pueda introducir los datos necesarios para adicionar un nodo a la réplica.	

Tabla 6. Tarea de Ingeniería "Implementar nodo servidor"

Tarea de Ingeniería	
Numero de tarea: 2	**Numero de Historia de Usuario:** 6
Nombre de la tarea: Implementar formulario nodo servidor	
Tipo de Tarea: Desarrollo	**Estimados:** 0.4
Fecha de Inicio: 28/1/2014	**Fecha de Fin:** 29/1/2014
Programador responsable: Alfredo Sabina Diez	
Descripción: Se implementará un formulario para recoger la información que el usuario inserte relacionada con el nodo Servidor, y contará con los siguientes datos: engine.name, JDBC Driver, JDBC URL, db.user, db.password, registration.url, sync.url, group.id, external.id.	

3.2. Estándares de codificación

El propósito fundamental de los estándares de codificación [36] es establecer una arquitectura y un estilo consistente, independiente del programador, para que el sistema resulte fácil de entender y por supuesto fácil de mantener. El mayor problema que trata de enfrentar esta práctica es tratar de entender el formato y el estilo utilizado en el código escrito por otros desarrolladores.

Los estándares de codificación son un complemento a la programación por pares y es importante tener un buen estándar de codificación que permita clarificar el código más que confundir, promover la necesidad

del código, permitir que los programas se acerquen lo mejor posible al lenguaje natural, incorporar las mejores prácticas de la codificación para un mejor entendimiento entre los programadores.

3.2.1. Estándares de codificación para el lenguaje Java

Los estándares de codificación [37] incluyen un conjunto de convenciones de código que son importantes para los programadores por la ayuda que les brinda. A continuación se proponen las siguientes especificaciones orientadas al lenguaje de programación Java que son utilizadas en el código de la solución propuesta e evidenciado en la “Fig 8 Ejemplo de código 1 aplicando el estándar de codificación de Java” y “Fig 9 Ejemplo de código 2 aplicando el estándar de codificación de Java”:

- El primer componente del nombre de un paquete único se escribe siempre en minúsculas con caracteres ASCII y debe ser uno de los nombres de dominio de último nivel, actualmente com, edu, gov, mil, net, org.
- Las variables de clase (estáticas) deben seguir el siguiente orden: Primero las variables de clase públicas, después las protegidas, después las de nivel de paquete (sin modificador de acceso), y después las privadas.
- Las variables de instancia: Primero las públicas, después las protegidas, después las de nivel de paquete (sin modificador de acceso), y después las privadas.
- Evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas
- Comentarios: Los programas Java pueden tener dos tipos de comentarios: comentarios de implementación y comentarios de documentación
- Se usan dos líneas en blanco entre las secciones de un fichero fuente y las definiciones de clases e interfaces.
- Se usa una línea en blanco entre métodos, las variables locales de un método y su primera sentencia, antes de un comentario de bloque o de un comentario de una línea y entre las distintas secciones lógicas de un método para facilitar la lectura.

```java
package com.web.symetric.bus.dao;

import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class DAOService {

// Este metodo permitira listar los vinculos entre los router y los triggers
    public static List<TriggerRouter> obtenerTriggersRoutersDe(
            ParametrosConexion cxn) {
        List<TriggerRouter> res = new ArrayList<TriggerRouter>();
```

Fig 8 Ejemplo de código 1 aplicando el estándar de codificación de Java

```java
        Connection con = getConnection(cxn);
        try {
// Se crea un objeto de Statement class para ejecutarlo en el SQL statement
            Statement stmt = con.createStatement();
// Se ejecuta el SQL statement y se obtienen los resultados en rs de tipo ResultSet
            ResultSet rs = stmt
                    .executeQuery("select * from sym_trigger_router");
            while (rs.next()) {
                String trigger_id = rs.getString("trigger_id");
                String router_id = rs.getString("router_id");
                int initial_load = rs.getInt("initial_load_order");

                res.add(new TriggerRouter(trigger_id, router_id, initial_load));
                System.out.println("Trigger Router= " + router_id
                        + " initial_load= " + initial_load);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return res;
```

Fig 9 Ejemplo de código 2 aplicando el estándar de codificación de Java

3.2.2. Estándares de codificación para el lenguaje JavaScript

Las siguientes especificaciones de los estándares de codificación para leguaje JavaScript fueron utilizadas durante la implementación de la solución propuesta [21] evidenciadas en la Fig 10 Ejemplo de código aplicando el estándar de codificación de Java:

- Los comentarios inician con el símbolo // si son de una línea o empezar con /*y finalizar con */ si son de varia líneas.

- Las líneas de código terminan con el punto y coma (;).
- Las llaves ({ }) son utilizadas para agrupar código.
- El nombre de las variables debe empezar con una letra que puede ir seguida de otra letra, un número o el signo "_" y se asignan valores como cadenas de textos, valores numéricos o valores booleanos.
- La declaración de una variable puede realizarse de cualquiera de las siguientes formas:

- Var variable = valor;
- Variable = valor;

```
   /* Esta funcion permite construir el nombre de un nodo, para realizar
    * esta accion el usuario debe introducir el grupo del nodo y
    * el identificador del nodo */
function conformarNombreDeNodo(){
    var group = $("#group_nodo_serv").val();
    var id_externo = $("#id_nodo_serv").val();
    // permite mostrar un mensaje de error si los campos estan vacios
    if(group.trim() === '' &&  id_externo.trim() === ''){
        alertify.error("Debe llenar  los campos vacios");
        return;
    }
    var pretext =  group + "-" + id_externo;

    $("#name_nodo_serv").val(pretext);
}
```

Fig 10 Ejemplo de código aplicando el estándar de codificación de Java

3.3. Despliegue de la solución propuesta

Los diagramas de despliegue [38] muestran la configuración de elementos de proceso (el despliegue de procesadores, periféricos, comunicaciones, etc.) y los componentes software (programas, procesos, etc.) que residen en ellos en tiempo de ejecución. En un diagrama de despliegue no pueden aparecer componentes que no existen como entidades en tiempo de ejecución, como los ficheros fuente.

A continuación se puede observar la ''Fig 11 Diagrama de despliegue de la solución propuesta'' que contiene el modelo de despliegue de la solución. El nodo PC Servidor hace referencia al servidor usado donde debe ser instalada la herramienta SymmetricDS, la interfaz para la administración y monitorización

de SymmetricDS, el servidor de aplicaciones Apache Tomcat, y la Máquina Virtual de Java. El nodo PC Cliente representa el lugar de acceso del usuario, la conexión entre la PC Cliente y la PC Servidor se realiza utilizando el protocolo HTTPS. Los nodos PC Base Datos PostgreSQL 1 y PC Base Datos PostgreSQL 2 representan las bases de datos donde se crean las instancias de SymmetricDS, la comunicación con el servidor se realiza por el protocolo TCP/IP.

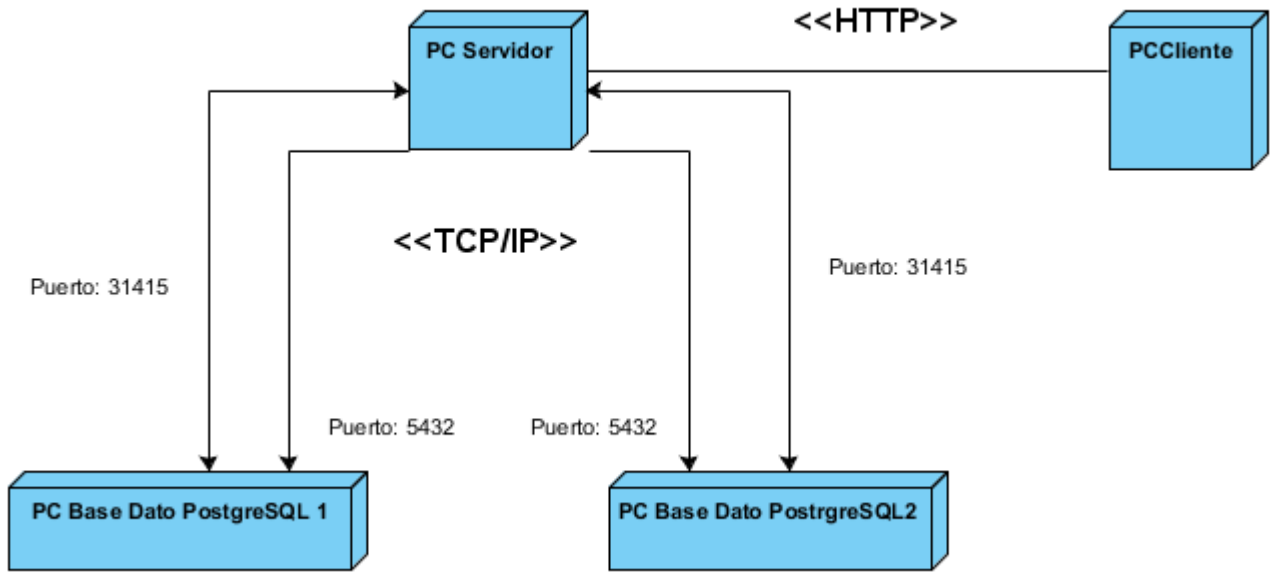


Fig 11 Diagrama de despliegue de la solución propuesta

3.4. Interfaces de la aplicación

En este epígrafe se muestran ejemplos de las interfaces del sistema desarrollado. En la ‘’Fig 12 Interfaz Adicionar Nodo Servidor’’ se puede observar en la parte superior izquierda el menú Adicionar Nodo Servidor que permitirá al usuario introducir los datos que se muestran en el centro de la página. En la parte inferior derecha se pueden observar los botones Probar Conexión, Aceptar y Cancelar.

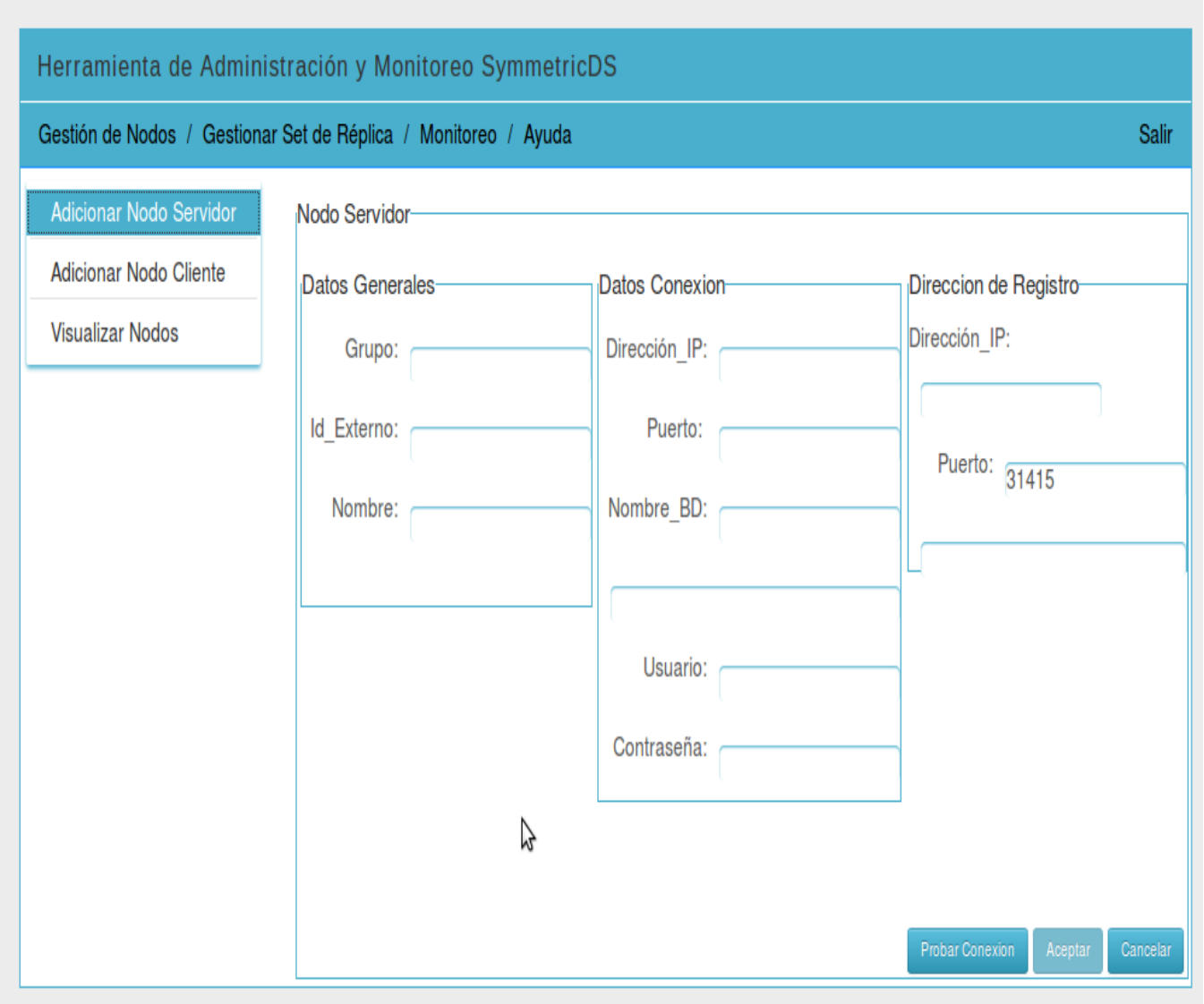


Fig 12 Interfaz Adicionar Nodo Servidor

En la ‘’Fig 13 Interfaz Registrar Nodo’’ se puede observar en la parte superior izquierda el menú Registrar Nodo que permitirá al usuario introducir y escoger los datos que se muestran en el centro de la página junto a los botones Aceptar y Cancelar.

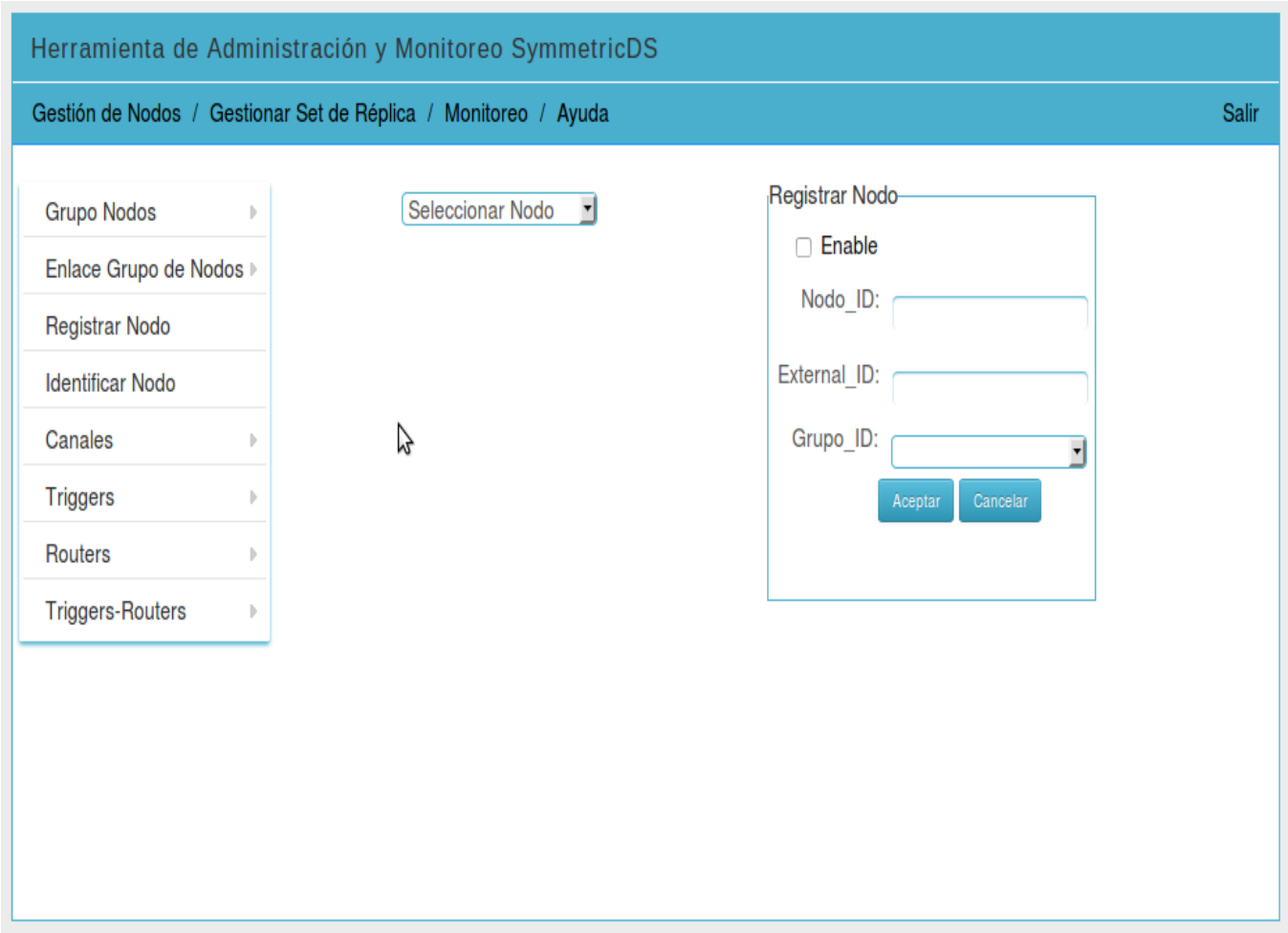


Fig 13 Interfaz Registrar Nodo

3.5. Proceso de pruebas de la metodología XP

Uno de los pilares de la metodología *XP* es el proceso de pruebas [39]. Esta metodología anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final.

Las metodologías para el desarrollo de software no constituyen la única ayuda para determinar qué tipo de prueba aplicar al software, se debe conocer a fondo que aspectos serán probados. Por esta razón además de tener en cuenta los tipos de pruebas mencionadas anteriormente se analizarán las pruebas de regresión. Estas pruebas se encargan de verifican que se detecten nuevos errores al añadir nuevas funcionalidades o corregir algún error, pues es frecuente que de manera involuntaria se creen problemas que ya se solucionaron [40].

3.5.1. Pruebas de unidad

Prueba de unidad [41]: Las pruebas de unidad se concentran en la lógica del procesamiento interno y en las estructuras de datos dentro de los límites de un componente. Este tipo de prueba se puede aplicar en paralelo a varios componentes.

3.5.2. Pruebas del sistema

Se denominan pruebas de sistema [42] a las pruebas de software que tienen por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados. También se les denomina pruebas de comportamiento o pruebas de caja negra, ya que los analistas de pruebas, no enfocan su atención a como se generan las respuestas del sistema, básicamente el enfoque de este tipo de prueba se basa en el análisis de los datos de entrada y en los de salida, esto generalmente se define en los casos de prueba preparados antes del inicio de las pruebas.

Método de prueba de caja negra

La partición equivalente [43] es un método de prueba que se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de prueba que deben desarrollarse. El diseño de casos de prueba para partición equivalente se basa en la evaluación de un conjunto de estados válidos y no válidos para las condiciones de entrada. Por lo general, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana.

Para realizar las pruebas al sistema se confeccionaron 32 casos de prueba. En la “Tabla 7. Caso de Prueba "Adicionar Nodo Servidor"” se puede observar un ejemplo de caso de prueba correspondiente a la Historia de Usuario “Adicionar Nodo”, las restantes pueden ser consultadas en la “Planilla de Casos de prueba” que se encuentra en el Expediente de Proyecto adjunto a este trabajo de diploma.

Condiciones de ejecución: El usuario debe haberse autenticado previamente en el sistema y debe existir un Nodo insertado con anterioridad.

Tabla 7. Caso de Prueba "Adicionar Nodo Servidor"

Escenario	Descripción	Variables	Respuesta del sistema	Flujo central
C 1.1 Adicionar un nodo servidor introduciendo todos los datos correctamente	El nodo es adicionado correctamente	**Variable 1:** corp-000(Servidor)	El sistema muestra el mensaje " El nodo se insertó correctamente"	
		Variable 2: org.postgres.Driver		
		Variable3: jdbc:postgres:10.53.13.12:5432 /corp-000		Seleccionar el menú Gestión de Nodo. 2- Seleccionar el tipo de nodo (Cliente o Servidor). 3- Seleccionar la opción de menú Adicionar nodo.
		Variable 4: postgres		
		Variable 5: postgres		
		Variable 6: http:// 10.53.13.12:8080/sync/corp-000		
		Variable 7: http:// 10.53.13.12:8080/sync/corp-000		
		Variable 8: corp		
		Variable 9:-000		

				4- Introducir los datos 5- Seleccionar el botón Aceptar
EC 1.2 Adicionar un nodo servidor introduciendo datos incorrectos	El nodo no es adicionado correctamente	**Variable 1:** l(**store-000** / **corp** / **000**)	El sistema muestra el mensaje "El nodo no pudo insertarse correctamente"	1- Seleccionar el menú Gestión de Nodo. 2- Seleccionar el tipo de nodo (Cliente o Servidor). 3- Seleccionar la opción de menú Adicionar nodo 4- Introducir los datos 5- Seleccionar
		Variable 2: l(org.Postgres.Driver / org.postgres / org.mysql.Driver)		
		Variable 3: l(postgres: ip/5432/ corp-000)2:5432 /corp-000		
		Variable 4: mybaseDato		
		Variable 5: mybaseDato123		
		Variable 6: l(http:// 10.53.13.12:8080/sync / ftp:// 10.53.13.12:8080/sync)		
		Variable 7: l(http:// 10.53.13.12:8080/sync / ftp:// 10.53.13.12:8080/sync)		
		Variable 8: store		
		Variable 9: id		

				el botón Aceptar
EC 1.3 Actualizar datos de nodo servidor sin especificar el identificador del grupo, ni el identificador externo.	La categoría no es actualizada.	**Variable 1:** corp-000(Servidor) **Variable 2:** org.postgres.Driver **Variable 3:** jdbc:postgres:10.53.13.12:5432 /corp-000 **Variable 4:** postgres **Variable 5:** postgres **Variable 6:** http:// 10.53.13.12:8080/sync/corp-000 **Variable 7:**http:// 10.53.13.12:8080/sync/corp-000 **Variable 8:** l( ) **Variable 9:** l( )	El sistema muestra el mensaje "El nodo no pudo insertarse correctamente. Existen campos sin completar "	1- Seleccionar el menú Gestión de Nodo. 2- Seleccionar el tipo de nodo (Cliente o Servidor). 3- Seleccionar la opción de menú Adicionar nodo. 4- Introducir los datos 5- Seleccionar el botón Aceptar

Cada uno de los escenarios con los que cuentan los casos de prueba identificados, tienen variables específicas que permiten validar su funcionamiento En la “Tabla 8. Variables para el caso de prueba

Adicionar Nodo Servidor" se puede observar un ejemplo de variables para el caso de prueba "Adicionar Nodo".

Tabla 8. Variables para el caso de prueba Adicionar Nodo Servidor

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	nombre	Campo de texto	No	Campo alfanumérico con longitud mayor o igual a 3 que inicia con letras, se pueden entrar hasta 10 letras ya sea mayúscula o minúscula seguido de un guión (-) y terminar en número de tres cifras, la longitud máxima es de 14.
2	jdbc.Driver	Campo de texto	No	Campo de texto con longitud mayor a 20. Se pueden entrar letras mayúsculas o minúsculas y debe seguir el siguiente formato ***org.nombre de la base de datos.Driver.*** La longitud máxima es de 30.
3	db.url	Campo de texto	No	Campo alfanumérico con longitud mayor a 20. Se pueden entrar letras mayúsculas o minúsculas y debe seguir el siguiente formato ***jdbc: postgres: / / localhost / nombre de la base de datos*.** La longitud máxima es de 30.
4	db.user	Campo de texto	No	Campo de texto con longitud mínima de 5 letras, con longitud máxima de 10 letras, ya sean mayúsculas o minúsculas.
5	db.password	Campo de texto	No	Campo alfanumérico con longitud mínima a 5, con longitud máxima de 10 letras, ya sean mayúsculas o minúsculas.

6	registration.url	Campo de texto	No	Campo alfanumérico con longitud mayor a 20 donde se pueden entrar letras mayúsculas o minúsculas y debe seguir el siguiente formato ***http:// {nombre de host}: {puerto} / sincronización / {} engine.name.***
7	sycn.url	Campo de texto	No	Campo alfanumérico con longitud mayor a 20 donde se pueden entrar letras mayúsculas o minúsculas y debe seguir el siguiente formato ***http:// {nombre de host}: {puerto} / sincronización / {} engine.name.***
8	grupo.id	Campo de texto	No	Campo de texto con longitud máxima 10 donde se pueden entrar letras mayúsculas o minúsculas.
9	external.id	Campo de texto	No	Campo numérico con longitud máxima de 3.

Es importante conocer que las pruebas se realizan con el objetivo de encontrar la mayor cantidad de errores en el software y no para verificar que se desarrolló el sistema perfecto. La realización de las pruebas arrojó algunas Historias de Usuarios satisfactorias, es decir, no presentaban errores, pero también se detectaron errores en otras. Estas deficiencias unidas a las Historias de Usuarios pasaron a ser No Conformidades y se registraron en la Planilla de No Conformidades correspondiente al expediente de proyecto adjunto a este trabajo de diploma. En la “Tabla 9. Tabla de ejemplo de No Conformidades.” se puede observar un listado de no conformidades para la Historia de Usuario “Adicionar Nodo”.

Tabla 9. Tabla de ejemplo de No Conformidades.

Elemento	No	No C	Aspecto C	Etapa de detección	Clasif	Estado No C	Respuesta de Equipo Desarrollo
Aplicación	1	Se puede adicionar un nodo aun	Adicionar nodo	Prueba	S	015/05/2014 PD	Se corrigió el error encontrado. El sistema no acepta la entrada de

		cuando los datos son incorrectos.				017/05/2014 RA	caracteres inválidos.
Aplicación	2	El sistema muestra un mensaje indicando que los datos fueron insertados correctamente cuando no es así.	Adicionar nodo	Prueba	S	015/05/2014 PD 017/05/2014 RA	Se corrigió el error permitiendo solamente mostrar el mensaje cuando se adicionaran correctamente los nodos.

Resultado de las pruebas de sistema

Las pruebas de sistema realizadas brindarán al cliente conformidad y seguridad ante las funcionalidades definidas, para su ejecución se generó un caso de prueba por cada Historia de Usuario y se efectuaron 2 iteraciones de pruebas. En la “Fig 14. Resultados de las pruebas” se puede observar el total de No conformidades encontradas y las que fueron resueltas en cada iteración de prueba.

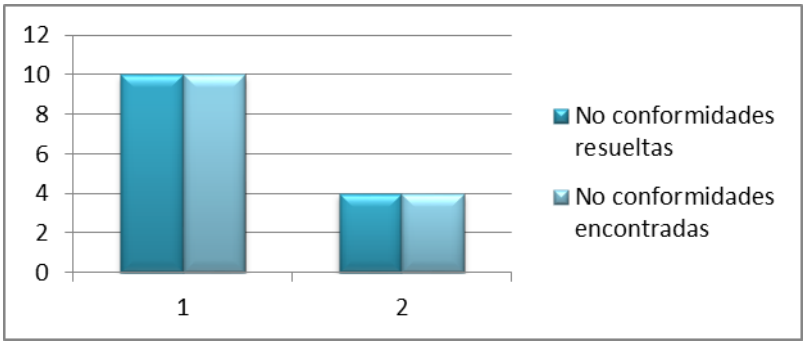


Fig 14. Resultados de las pruebas de sistema

3.5.3. *Pruebas de aceptación*

Las pruebas de aceptación [44] se realizan para obtener la aceptación final del cliente antes de la entrega del producto para su paso a producción. La planificación de estas pruebas se realiza en etapas tempranas del desarrollo para utilizar sus resultados como indicadores de validez. Su ejecución es facultativa de los clientes y requiere de un entorno de pruebas que simule el entorno real de producción. Se realizan luego de efectuar las pruebas del sistema donde se corrigen la mayoría de los defectos, para evitar la realización de un proceso exhaustivo y tener que volver a las pruebas del sistema.

Para avalar la realización de esta prueba se efectuó la firma por parte del cliente y los desarrolladores de la interfaz web para la administración y monitorización de SymmetricDS, de un Acta de aceptación que se encuentra anexada a este documento. De esta forma queda demostrada la aprobación del cliente por la solución desarrollada.

3.6. Consideraciones del capítulo

Al efectuar las acciones de las etapas de implementación y prueba que propone la metodología XP y que fueron descritas en este capítulo e incluyendo los artefactos necesarios para su correcta realización se concluye que:

- La implementación fue guiada por los estándares de codificación para el lenguaje Java y JavaScript lo que permitió realizar una entrega funcional al terminar cada iteración planificada en la etapa de análisis y cumplir con la fecha de entrega definido por el cliente.
- Se diseñaron un total de 57 tareas de ingeniería para desarrollar las Historias de Usuarios y cumplir con la implementación de cada una de las funcionalidades que debe cumplir el sistema desarrollado para satisfacer las necesidades del cliente.
- Se realizaron pruebas de aceptación y de regresión para garantizar el correcto funcionamiento del software utilizando el método partición de equivalente. Estas pruebas se realizaron a través de la interfaz del sistema y se detectaron inconformidades como campos no validados y errores no tratados; todas ellas solventadas rápidamente mediante la corrección de los objetos correspondientes.

Conclusiones Generales

Con la realización de este trabajo de diploma se obtuvo una propuesta que da cumplimiento al objetivo general planteado, al lograr desarrollar una interfaz web para la administración y monitorización de la herramienta de réplica de datos SymmetricDS. Al llegar a este resultado se puede concluir lo siguiente:

- Se realizó un estudio del estado del arte basado en los procesos de administración y monitorización de las herramientas de réplica de datos de datos de datos de datos de datos de datos de datos de datos de datos de datos, identificando cuales son las tareas de administración y monitorización que utiliza la herramienta de réplica de datos SymmetricDS y cómo las realiza al crear una soluciones de réplica de datos.
- Se realizó la implementación de la Interfaz Web para la administración y monitorización de SymmetricDS cumpliendo con los 33 requisitos funcionales identificados y en la etapa de diseño se obtuvieron los artefactos y diagramas necesarios para cumplir con el desarrollo de la solución.
- El diseño y ejecución de pruebas del sistema permitieron comprobar el correcto funcionamiento de la Interfaz para la administración y monitorización de la herramienta de réplica de datos SymmetricDS.

Recomendaciones

Con el objetivo de mejorar y aumentar el alcance de la interfaz para la administración y monitorización de la herramienta SymmetricDS se recomienda:

- Extender el uso de la aplicación a otros sistemas gestores de base de datos, además de PostgreSQL.
- Realizar la implementación de otras funcionalidades de SymmetricDS que fueron identificadas durante el desarrollo de la interfaz como gestionar disparadores de archivos y la gestión de Jobs.

Referencias Bibliográficas

1. Moreno, G.R.M., *Replicación en PostgreSQL 9.0*, in *Facultad de Ciencias Exactas Base de Datos II*. 2012, Universidad Nacional de Salta

2. *"Replicación de SQL Server"*. 2014 [cited 1/6/2014; Available from: http://msdn.microsoft.com.

3. Pérez, A.P., *Desarrollo de la funcionalidad para la programación de la captura y envío de datos en el Replicador REKO*. 2010, Universidad de las Ciencias Informáticas: La Habana.

4. López, R.S., *Propuesta de solución de réplica de datos con fragmentación horizontal.* 2010, Universidad de las Ciencias Informáticas: La Habana.

5. SL, M. *Replicación Solución* 2014 [cited 3/6/2014; Available from: http://www.solucionesit.com.ve.

6. Microsoft. *Practicas recomendadas para la Administración de replicación*. 2014 [cited 4/6/2014; Available from: http://technet.microsoft.com.

7. Reingart, M. *Sistema de replicación simple para PostgreSQL programado en Python*. [cited 9/2/2014; Available from: http://code.google.com/p/pyreplica/.

8. *DBreplicator.* . [cited 11/2/2014; Available from: http://dbreplicator.org/.

9. Angela Gloria Gomez Peña, J.A.V., *Desarrollo para el modulo de transmicion de datos de gran tamaño para el sistema Reko*. 2010, Universidad de las Ciencias Inform'aticas: La Habana.

10. Eric Long, C.H., Mark Hanes, Greg Wilmer, *SymmetricDS User Guide v3.5*. 2007 – 2013.

11. Community, S. *SymmetricDS*. 2013; Available from: http://www.SymmetricDS.org/.

12. Eric Long, C.H., Mark Hanes, Greg Wilmer, Austin Brougher, *SymmetricDS Pro Quick Start*. 2007 – 2013.

13. Fernández, A. *El diseño de la interfaz*. 2010 [cited 5/6/2014; Available from: http://www.lawebera.es.

14. Canós, J.H., *Metodologías Ágiles en el Desarrollo de Software*. 2008, Universidad Politécnica de Valencia Camino de Vera: Valencia.

15. Ibarra, A.F. *Rational Unified Process*. Available from: https://www.u-ursos.cl/ingenieria/2004/2/CC62C/1/material_docente/objeto/44990.

16. Sanchez, M.A.M. *Metodologías De Desarrollo De Software*. Available from: http://www.informatizate.net.

17. Cornejo, J.E.G.; Available from: http://www.docirs.cl/uml.htm.30.

18. *Visual Paradigm para UML* Copyright © Targetware 2007-2013:[Available from: http://www.software.com.ar/visual-paradigm-para-uml.html.

19. *Guión Visual Paradigm for UML*. Available from: http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.

20. Lorenzo, H.Y., *Sistema de Gestión de Información de las Coordinaciones Regionales de Prevención del Delito de la República Bolivariana de Venezuela*. 2011, Universidad de la Ciencias Informáticas: La Habana

21. Eguíluz Pérez, J.I.a.J.A.f.h.w.l.e. *Introducción a JavaScript*. [cited 1/3/2014; Available from: http://www.librosweb.es

22. *Eclipse IDE*. 2011 [cited 4/3/2014; Available from: http://www.genbetadev.com/herramientas/eclipse-ide

23. Ramón, A.B. *Eclipse Juno, la versión 4.2 de Eclipse*. **6/3/2014**.

24. Eclipse. *Eclipse.* [cited 12/3/2014; Available from: http://www.eclipse.org..

25. *Apache Tomcat.* . [cited 14/3/2014; Available from: http://www.tomcatapache.org.

26. *Spring.* . [cited 21/3/2014; Available from: http://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/html/overview.html.

27. *Spring.* 2009 [cited 24/3/2014; Available from: http://docs.spring.io.

28. Alvarez, M.A., *Manual de jQuery*.

29. *Bootstrap*. [cited 26/3/2014; Available from: http://www.genbetadev.com/desarrollo-web/disenando-tu-nuevo-proyecto-web-con-bootstrap-2-0..

30. Mora, S.L., *Historia, principios básicos y clientes web.*

31. *Historias de Usuario*. [cited 28/3/2014;Available from: http://www.genbetadev.com/metodologias-de-programacion/historias-de-usuario-una-forma-natural-de-analisis-funcional32. *Targetas CRC.*

32. Targetas CRC. [cited 28/3/2014;Available from: http://jummp.wordpress.com/2012/01/10/desarrollo-de-programa-tarjetas-crc/

33. Pressman, R.S., *Ingeniería de software. Un enfoque práctico. Capítulo 13.*

34. *Patron MVC.* [cited 29/3/2014;Available from: http://www.e-continua.com.mx/index.php/15-springmvc

35. *Patrones de Diseño*. Available from: http://arlethparedes.wordpress.com/2012/08/27/patrones-de-arquitectura-vs-patrones-de-diseno/.

36. *Guía para Implementar Estándares de Codificación v 1.0*. [cited 30/3/2014;Available from: http://www.inf.utfsm.cl/~visconti/xp/Guia_>Estandares_Codificacion_2.doc

37. Molpeceres., S.H.-S.M.I.A. *Convenciones de código para el lenguaje de programación Java*. [cited 2/4/2014; Available from: http://www.javahispano.com.

38. Maden., Y.P.J.M.A., *Módulo para la edición de flujos de procesos de medias*. 2012

Universidad de las Ciencias Informáticas: La Habana.

39. J.J Gutierrez, M.J.e., M. Mejías, J. Torres, *Pruebas del sistema en Programación Extrema.*: Departamento de lenguajes y sistemas. University of Sevilla.

40. Carceler, V., *Prueba de aplicaciones web y documentación*.

41. Pressman, R.S., *Ingeniería de Software. Un enfoque Práctico*.

42. B, A.O. *FUNCTIONAL TESTING - PRUEBAS FUNCIONALES*. 2009; Available from: http://www.calidadysoftware.com.

43. Pressman, R.S., *Ingeniería de Software. Un enfoque Práctico.Capítulo 14.*

44. . Jorge luis mendoza valdez, G.A.V.H., otros, *Pruebas del sistema y pruebas de aceptación. .* , in *Facultad de ingeniería de sistema Ciclo IX*. 2013, FIS-UNICA.

Bibliografías

- Moreno, G.R.M., *Replicación en PostgreSQL 9.0*, in *Facultad de Ciencias Exactas Base de Datos II*. 2012, Universidad Nacional de Salta.
- *"Replicación de SQL Server"*. 2014 [cited 1/6/2014; Available from: http://msdn.microsoft.com.
- Pérez, A.P., *Desarrollo de la funcionalidad para la programación de la captura y envío de datos en el Replicador REKO*. 2010, Universidad de las Ciencias Informáticas: La Habana.
- López, R.S., *Propuesta de solución de réplica de datos con fragmentación horizontal.* 2010, Universidad de las Ciencias Informáticas: La Habana.
- SL, M. *Replicación Solución* 2014 [cited 3/6/2014; Available from: http://www.solucionesit.com.ve.
- Microsoft. *Practicas recomendadas para la Administración de replicación*. 2014 [cited 4/6/2014; Available from: http://technet.microsoft.com.
- Reingart, M. *Sistema de replicación simple para PostgreSQL programado en Python*. [cited 9/2/2014; Available from: http://code.google.com/p/pyreplica/.
- *DBreplicator.* . [cited 11/2/2014; Available from: http://dbreplicator.org/.
- Angela Gloria Gomez Peña, J.A.V., *Desarrollo para el modulo de transmicion de datos de gran tamaño para el sistema Reko*. 2010, Universidad de las Ciencias Inform'aticas: La Habana.
- Eric Long, C.H., Mark Hanes, Greg Wilmer, *SymmetricDS User Guide v3.5*. 2007 – 2013.
- Community, S. *SymmetricDS*. 2013; Available from: http://www.SymmetricDS.org/.
- Eric Long, C.H., Mark Hanes, Greg Wilmer, Austin Brougher, *SymmetricDS Pro Quick Start*. 2007 – 2013.
- Fernández, A. *El diseño de la interfaz*. 2010 [cited 5/6/2014; Available from: http://www.lawebera.es.
- Canós, J.H., *Metodologías Ágiles en el Desarrollo de Software*. 2008, Universidad Politécnica de Valencia Camino de Vera: Valencia.
- Ibarra, A.F. *Rational Unified Process*. Available from: https://www.u-ursos.cl/ingenieria/2004/2/CC62C/1/material_docente/objeto/44990.
- Sanchez, M.A.M. *Metodologías De Desarrollo De Software*. Available from: http://www.informatizate.net.
- Cornejo, J.E.G.; Available from: http://www.docirs.cl/uml.htm.30.
- *Visual Paradigm para UML* Copyright © Targetware 2007-2013:[Available from: http://www.software.com.ar/visual-paradigm-para-uml.html.

- *Guión Visual Paradigm for UML*. Available from: http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.
- Lorenzo, H.Y., *Sistema de Gestión de Información de las Coordinaciones Regionales de Prevención del Delito de la República Bolivariana de Venezuela*. 2011, Universidad de la Ciencias Informáticas: La Habana.
- Eguíluz Pérez, J.I.a.J.A.f.h.w.l.e. *Introducción a JavaScript*. [cited 1/3/2014; Available from: http://www.librosweb.es.
- *Eclipse IDE*. 2011 [cited 4/3/2014; Available from: http://www.genbetadev.com/herramientas/eclipse-ide
- Ramón, A.B. *Eclipse Juno, la versión 4.2 de Eclipse*. [cited 6/3/2014; Available from: http://www.eclipse.org.
- Eclipse. *Eclipse.* [cited 12/3/2014; Available from: http://www.eclipse.org.
- *Apache Tomcat.* . [cited 14/3/2014; Available from: http://www.tomcatapache.org.
- *Spring.* . [cited 21/3/2014; Available from: http://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/html/overview.html.
- *Spring.* 2009 [cited 24/3/2014; Available from: http://docs.spring.io.
- Alvarez, M.A., *Manual de jQuery*.
- *Bootstrap*. [cited 26/3/2014; Available from: http://www.genbetadev.com/desarrollo-web/disenando-tu-nuevo-proyecto-web-con-bootstrap-2-0.
- Mora, S.L., *Historia, principios básicos y clientes web.*
- *Historias de Usuario*. [cited 28/3/2014;Available from: http://www.genbetadev.com/metodologias-de-programacion/historias-de-usuario-una-forma-natural-de-analisis-funcional.
- *Targetas CRC.* [cited 28/3/2014;Available from: http://jummp.wordpress.com/2012/01/10/desarrollo-de-programa-tarjetas-crc/
- Pressman, R.S., *Ingeniería de software. Un enfoque práctico*
- *Patron MVC.* [cited 29/3/2014;Available from: http://www.e-continua.com.mx/index.php/15-springmvc
- *Patrones de Diseño*. [cited 30/3/2014; Available from: http://arlethparedes.wordpress.com/2012/08/27/patrones-de-arquitectura-vs-patrones-de-diseno/.
- *Guía para Implementar Estándares de Codificación v 1.0*. [cited 30/3/2014;Available from: http://www.inf.utfsm.cl/~visconti/xp/Guia_>Estandares_Codificacion_2.doc

- Molpeceres., S.H.-S.M.I.A. *Convenciones de código para el lenguaje de programación Java*. [cited 2/4/2014; Available from: http://www.javahispano.com.
- Maden., Y.P.J.M.A., *Módulo para la edición de flujos de procesos de medias*. 2012 Universidad de las Ciencias Informáticas: La Habana.
- J.J Gutierrez, M.J.e., M. Mejías, J. Torres, *Pruebas del sistema en Programación Extrema.*: Departamento de lenguajes y sistemas. University of Sevilla.
- Carceler, V., *Prueba de aplicaciones web y documentación*.
- Pressman, R.S., *Ingeniería de Software. Un enfoque Práctico*.
- B, A.O. *FUNCTIONAL TESTING - PRUEBAS FUNCIONALES*. 2009; Available from: http://www.calidadysoftware.com.
- Pressman, R.S., *Ingeniería de Software. Un enfoque Práctico.*
- Jorge luis mendoza valdez, G.A.V.H., otros, *Pruebas del sistema y pruebas de aceptación. .* , in *Facultad de ingeniería de sistema Ciclo IX*. 2013, FIS-UNICA.