Universidad de las Ciencias Informáticas

Facultad 4

Módulo para la configuración y monitorización de réplicas con la herramienta SymmetricDS para la arquitectura Xalix.

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

Autor:

Reiman Alfonso Azcuy

Tutores:

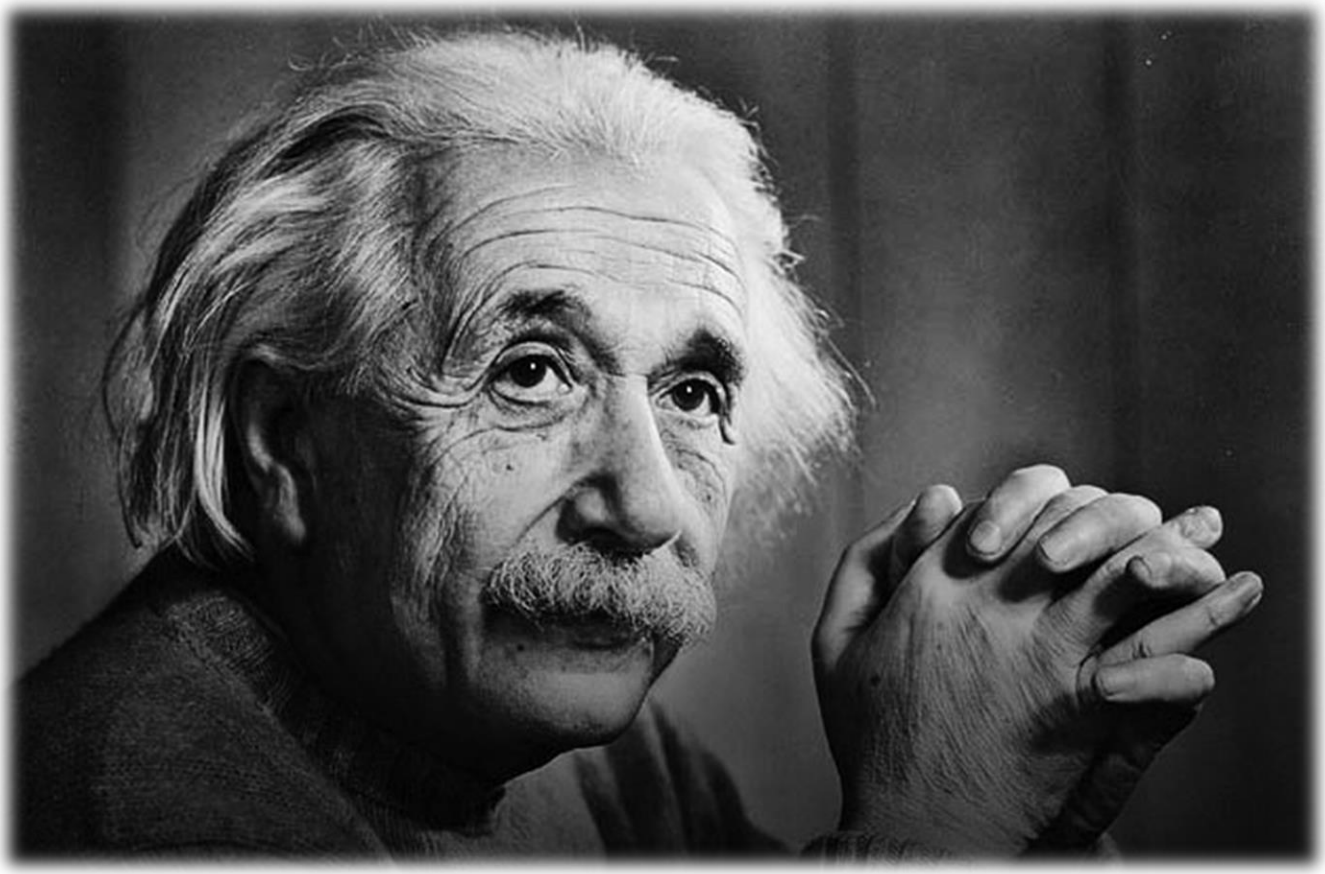
Ing. Arcel Labrada Batista

Ing. Jorge Luis Piña González

Ing. Arley Enrique Cera Rojas

La Habana, junio de 2017

“Año 58 de la Revolución”



"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad".

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 4 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____
Reiman Alfonso Azcuy

_____
Ing. Arcel Labrada Batista

_____
Ing. Jorge Luis Piña González

_____
Ing. Arley E. Cera Rojas

AGRADECIMIENTOS

A Dios que sé que ha obrado en mi vida y me acompaña en cada paso importante que doy en mi vida. A Fidel y la revolución por darme la oportunidad de estudiar en una universidad como esta. A mi mamá Elvirita que nunca perdió la fe en mi incluso cuando yo mismo la había perdido. A mi hermano Pedro que siempre ha estado a mi lado para todo, lo bueno y lo malo que ha sido siempre mi mejor amigo.

A mis abuelas María y Elena por todo su amor incondicional y su preocupación A mi papá por su cariño excepcional, y por la educación que me dio forjándome con los valores que debe tener un buen ser humano. A mi hermano el Chino, a su esposa Midialis y sus hijos por todas las atenciones que tuvieron conmigo durante mi año de preparación de pruebas de ingreso. A mi hermana Yusimí y sus hijos por todo su cariño incondicional y por estar pendiente todo el tiempo de mi aquí en la UCI.

A Marién que yo digo cariñosamente que es mi prima pero que es mucho más que eso, es mi amiga de la infancia, mi hermana, que también ha estado conmigo en las buenas y en las malas. A toda mi familia, no los menciono a todos porque solamente tengo una hoja y poco tiempo. A todas mis amistades de la biblioteca que me dieron todo su apoyo durante la etapa de las pruebas de ingreso.

A todas mis amistades de aquí que hice durante los cinco años , especialmente a todos los que fueron mis compañeros de aula o apartamento Daniel, Liován ,David, Naylín, Jennifer, Leo, Diosvel, Norlys, Yusniel, Mayara, Flavia, Caridad, Dairon, Sobrino, Maydalis, Josué, los dos Ernestos, Adrián, Rormery ,Duniel, Lisy, Gleidys, Jorjito y su familia, Amado,el Luisy, Iris, Elí, la gente de Manzanillo, la gente del IPI, la gente de la CUJAE, los que entrenaron conmigo en el equipo de Karate, el secretariado de la FEU, en fin a todos. A Taire por toda la ayuda que me dio en quinto año, y por defenderme a muerte contra quién fuera. A Greter y sus padres Orquídea y Oscar por su amistad sincera y valiosa en todo momento. A Danilo y su grupo que me sacaron de mis casillas varias.

A todos mis profesores por la formación que me dieron, muy especialmente para aquellos me impactaron con su ejemplo al no limitarse a una clase sino al pensar en el futuro como profesional y como persona, profesores como Yadilka, Vázques, Mercedes, Rafael, Yinimary, Yudanis, Greilan, Eduardo, Maritsa, Yisel, Yosleidy, Leduán, Dosague, Ordoqui, Mailin, Karenia, Bettys, Tatiana, Yudisney, Yorgelys, Leonardo…

A las profes Yuleisy y Graciela, que son un auténtico ejemplo de que enseñar puede cualquiera, pero educar solo quién sea un evangelio vivo. En su trabajo como profesoras guías fueron madres y amigas

de cada miembro del grupo dos, son una prueba de amor y cariño incondicional que un profesor debe tener a sus estudiantes, muchas gracias por todo.

A los profes Tomás y Yordanquis que me permitieron trabajar con ellos como alumno ayudante desde el segundo año de la carrera, y verdaderamente más que profesores me demostraron ser muy buenos amigos.

A la profe Rosalba Carralero Medina que a pesar de la distancia ha mantenido su amistad y me ha ayudado mucho con las cosas de la Tesis.

A la profe Yasiris y a Carlos Montenegro que igualmente en más de una ocasión me ayudaron con las revisiones del documento, y sus consejos me fueron de mucha utilidad.

A la profe Dunia por darme la oportunidad de pertenecer a su grupo de investigación.

A la profe Yirka por su ayuda incondicional siempre con amabilidad y una sonrisa.

Al profe Coca que además de ser uno de los profesores más geniales que he tenido en la carrera, como presidente del tribunal me dio consejos muy útiles.

A todo el personal de FORTES en especial a todos los del lab 205 por toda la ayuda que me dieron durante el desarrollo de la aplicación.

A todos los que fueron mis compañeros de concurso de Bases de Datos: Katherine, Orlando, Julio, Daniel, y Arianna…

Especialmente a Arianna que se ha convertido en una persona muy especial en mi vida, y espero que no salga de ella,

A mis tutores, el profe Piña que me ayudó muchísimo y que, aunque de vez en cuando discutíamos considero que fue un gran tutor, además de un gran profesional de la informática; a Arley que era como un hermano mayor pendiente de todo en todo momento dispuesto a aclarar cualquier duda;

Y por supuesto a Arcel, el mejor profe de Bases de Datos del país, excelente tutor y persona, que verdaderamente me ha enseñado mucho como tutor, de Tesis, y de trabajó como alumno ayudante.

Al oponente y de forma general al tribunal por toda la ayuda y los consejos.

A todos los amigos, compañeros, profesores que no menciono pero que en algún momento me acompañaron en mi paso por la Universidad

A todas los que en algún momento te preguntaban:¿Oye cómo va la Tesis?

DEDICATORIA

A mi abuelo Fidel que siempre creyó en mí y que donde quiera que esté debe estar orgulloso.

A mi abuelo Marciano, del que recuerdo poco, pero cada recuerdo es bueno y en vida no le pude regalar nada.

A mi familia, por su apoyo incondicional, en especial a mi mamá y a mi hermano Pedro que siempre han confiado en mí.

A mis abuelas María y Elena que han estado pendiente de mi durante toda la carrera.

RESUMEN

Con los avances de las Tecnologías de la Información y las Comunicaciones aumenta el almacenamiento masivo de información en bases de datos, convirtiéndose las mismas en el soporte de todo tipo de procesos y servicios en los diferentes sectores de la sociedad. Es común que las entidades funcionen de forma distribuida, y requieran que un punto se actualice con información proveniente de otro haciéndose necesario utilizar mecanismos de replicación. El Centro de Tecnologías para la Formación de la Universidad de las Ciencias Informáticas desarrolla bajo la Arquitectura Xalix un grupo de productos que deben funcionar de forma distribuida, y utiliza para la replicación la herramienta SymmetricDS. Esta herramienta posee varias funcionalidades y se caracteriza por ser efectiva, sin embargo, no cuenta con una interfaz visual libre por lo que trabajar con ella es complejo, al requerir ficheros de configuración, código SQL y comandos a través de la consola. Por ello, se desarrolló un módulo que permite realizar las operaciones de configuración y monitorización sobre SymmetricDS para la Arquitectura Xalix, de tal forma que sea usable en todos los proyectos implementados bajo la misma. Para el desarrollo del módulo se utilizó la metodología AUP en su variante UCI enmarcada en el escenario 4 de Historias de Usuario, generando los artefactos que define cada una de sus fases. Se utilizaron los lenguajes HTML v5, CCS v3, JavaScript y PHPv7 así como el framework Symfony 2.7.1, y el entorno de desarrollo Netbeans v8.0. Se validó el funcionamiento del módulo a través de la aplicación de pruebas de software detectando y corrigiendo las no conformidades existentes a lo largo de cuatro iteraciones. Se comprobó el valor práctico de la investigación para los usuarios finales a través del método de Iadov.

PALABRAS CLAVES

SymmetricDS, bases de datos, réplica, replicación, Xalix.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En la última década, el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) se ha enfocado en la informatización de los distintos sectores de la sociedad, incluyendo los procesos en diferentes entidades y empresas, lo que ha conllevado al almacenamiento masivo de información en Bases de Datos (BD).

Las BD se han convertido en una fuente confiable de almacenamiento de grandes cantidades de información para cualquier sector, sirviendo de soporte para diversos procesos vinculados a la toma de decisiones en sistemas empresariales y a la consulta rápida de información. Las mismas son utilizadas en los servidores administrativos de todo el mundo en apoyo a servicios como correo electrónico y Sistemas de Nombre de Dominio (DNS según sus siglas en inglés), siendo el centro del funcionamiento en los sistemas de dichos servidores por lo que se han hecho imprescindibles. Sin embargo, con dichos avances se hace necesario que la información almacenada en una BD sea accesible desde otros Centros de Información (CI).

La necesidad de acceso a información que se encuentra en diferentes BD, se hace visible en diversas empresas que se distribuyen en varias sucursales y se necesita que la información registrada en cada una de esas sucursales se actualice en el servidor central de la empresa. Un ejemplo de ello es la red de Universidades del Ministerio de Educación Superior que poseen sucursales distribuidas a lo largo del país.

También ocurren los casos en los que por el volumen de la BD y las propiedades del hardware con que cuenta el servidor de BD el mismo no puede soportar las operaciones de lectura y escritura. Debido a esto es necesario separar la BD en dos o más servidores para lectura y escritura, manteniéndose, la necesidad de la actualización de los datos en diferentes servidores mencionada anteriormente. Generalmente se requiere tener una BD centralizada, y hacer copias de la información desde ella hacia otras BD secundarias. La forma adecuada para solucionar esta situación es lo que se denomina replicación de datos o réplicas, lo cual permitiría el envío de información entre BD distribuidas geográficamente a través de la red, haciendo uso de algún mecanismo o herramienta de replicación.

La replicación proporciona redundancia y aumenta la disponibilidad de los datos. En algunos casos, se puede usar la replicación para aumentar la capacidad de lectura. Los clientes tienen la posibilidad de enviar las operaciones de lectura a diferentes servidores. También puede mantener copias en diferentes centros de datos para aumentar la localidad y la disponibilidad de datos para las aplicaciones distribuidas (MongoDB, 2011).

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

En Cuba existen varias entidades que utilizan replicación para garantizar la accesibilidad de la información. Un ejemplo de ello son las aplicaciones que se desarrollan en varios centros de producción de la Universidad de las Ciencias Informáticas (UCI).

En el Centro de Tecnologías para la Formación FORTES de la UCI que realiza productos para la educación, se desarrollan plataformas bajo la arquitectura Xalix. Dentro de estas plataformas se encuentran Zera y el Sistema de Gestión de Ingreso a la Educación Superior (SIGIES). Dichas plataformas deben funcionar de forma distribuida, manteniendo actualizada la información en un nodo central según las modificaciones realizadas en los diferentes servidores, por lo que se requiere la utilización de réplicas. Sin embargo, para cumplir con las políticas de migración a software libre que se están llevando a cabo en el país, se decide utilizar por parte del equipo de desarrollo la herramienta libre SymmetricDS.

La herramienta SymmetricDS es efectiva para la configuración y monitorización de réplicas (Eric Long C. H., 2014) sin embargo, configurarla es un proceso bastante engorroso, debido a que se deben modificar, de forma manual, una serie de ficheros correspondientes a cada una de las BD entre las cuales se desea establecer un sistema de réplicas. Además de que todas las instrucciones le deben ser escritas por medio de la consola, y la información que la herramienta requiere para replicar le debe ser entrada directamente a la BD utilizando consultas SQL.

Existe una herramienta desarrollada por la compañía Oracle llamada SymmetricDS-Pro la cual funciona como interfaz gráfica para la herramienta SymmetricDS, permitiendo realizar de forma simple la planificación, configuración, administración y monitorización de réplicas. Esta herramienta, tiene la agravante de que es un producto liberado bajo licencia privativa. Dicha agravante implica que, si es instalado sin pagar la licencia correspondiente, la utilización de la misma no puede ser superior a quince días. Esto significa que para poder utilizar este producto el país tendría que incurrir en gastos económicos significativos para el pago de la licencia antes mencionada.

A partir de la situación problémica descrita anteriormente, se plantea el siguiente **problema a resolver:** ¿Cómo facilitar la configuración y monitorización de réplicas con la herramienta SymmetricDS, en los productos desarrollados en FORTES sin que el país tenga que incurrir en gastos económicos por el pago de una licencia?

Objeto de estudio:

Herramientas para la configuración y monitorización de réplicas.

Campo de acción:

Sistemas que permitan la configuración y monitorización en forma visual de réplicas sobre herramientas de replicación.

Objetivo general de la investigación:

Desarrollar un módulo que permita la planificación y configuración de réplicas con la herramienta SymmetricDS para la Arquitectura Xalix utilizada en los productos desarrollados en el centro FORTES.

En función del objetivo planteado se formulan las siguientes **preguntas científicas:**

- ¿Qué son las réplicas y cuáles son las principales herramientas para configurarlas?
- ¿Cómo funciona la herramienta SymmetricDS y cómo se configura?
- ¿Qué es la Arquitectura Xalix, que lenguajes la componen?
- ¿Cómo está estructurada la solución propuesta?
- ¿Cómo diseñar e implementar la propuesta de solución?
- ¿Cómo validar la propuesta de solución?

Tareas de la investigación:

1. Análisis de lo referente a réplica, sistemas de réplicas en las BD, así como las principales herramientas de réplicas, características, clasificaciones, ventajas y desventajas.
2. Configuración y pruebas con la herramienta SymmetricDS.
3. Estudio de los lenguajes, tecnologías y metodología definidas para el marco de trabajo Xalix.
4. Descripción de la propuesta de solución.
5. Extracción de requisitos funcionales y no funcionales.
6. Diseño del modelo de datos de la aplicación y los prototipos de interfaz.
7. Implementación de las funcionalidades de la aplicación.
8. Aplicación de las pruebas de software para validar el funcionamiento del sistema.

Los **métodos de trabajo científico** utilizados fueron los siguientes:

Métodos teóricos

Histórico lógico: fue utilizado para el estudio crítico de trabajos anteriores. Así como comprobar la evolución del fenómeno investigado y el comportamiento de este en una secuencia temporal. Empleado para asumir el conocimiento de antecedentes, causas y otras evidencias históricas en que se aplican métodos y técnicas de replicación de datos.

Analítico-sintético: utilizado al descomponer el problema de investigación en elementos por separado

y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución de la propuesta. Quedando como elementos la complejidad de utilización de la herramienta SymmetricDS, el costo económico que implica usar una herramienta privativa, y como crear una solución en el marco de trabajo Xalix.

Inducción-deducción: fue utilizado para definir criterios específicos a partir de los conocimientos generales, conceptos del fenómeno investigado que fue la replicación, y factores de alta influencia en las etapas de la investigación a partir del análisis de la bibliografía que se consulta.

Análisis documental: fue utilizado en la consulta de la literatura especializada, para extraer la información necesaria que responda a las características distintivas del problema.

Métodos empíricos

Observación: posibilitó la observación de los procesos de réplica, así como el desarrollo de los eventos y la interacción de los mismos durante el proceso de validación.

Experimentación: fue utilizado para analizar el funcionamiento de las herramientas de réplica, y muy específicamente SymmetricDS, así como para obtener los resultados del funcionamiento del módulo desarrollado.

La **estructura del trabajo** quedó constituida en tres capítulos, conclusiones, anexos y bibliografía.

- Capítulo 1: Fundamentación teórica. Se plantean los principales conceptos relacionados al tema de la investigación, así como una descripción de las tecnologías y herramientas utilizadas.
- Capítulo 2: Descripción de la propuesta de solución. Análisis y diseño.
- Capítulo 3: Implementación y aplicación de los diferentes tipos de pruebas de software

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se abordan los elementos referidos a los temas teóricos que se analizaron en el proceso de dar solución al problema planteado para el trabajo de diploma, esencialmente ligado a las réplicas, sistemas de réplicas y herramientas de configuración de réplicas. Además, se describe la posible solución, así como las tecnologías y herramientas utilizadas para desarrollar dicha propuesta haciendo referencia a la arquitectura Xalix.

1.1. Replicación y réplicas. Características y principales ventajas.

Para poder implementar un módulo que permita configurar réplicas primeramente se debe tener conocimiento de que es una réplica, lo que representa, cuáles son sus ventajas y desventajas. Las réplicas constituyen una forma más rápida y confiable de esparcir datos entre distintas bases de datos en un sistema que trabaja de forma distribuida. Varios autores la definen como:

"La replicación copia y mantiene los objetos de las BD en las múltiples bases de datos que levantan un sistema distribuido. Puede mejorar el funcionamiento y proteger la disponibilidad de las aplicaciones porque alterna opciones de acceso a los datos existente" (Monge, 2005).

"La replicación 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 rea extensa, conexiones de acceso telefónico, conexiones inalámbricas e Internet" (Moreno, 2012).

"La replicación de datos consiste en el transporte de datos entre dos o más servidores, permitiendo que ciertos datos de las BD estén almacenados en más de un sitio y así aumentar la disponibilidad de los datos y mejorar el rendimiento de las consultas globales" (PlanBSur Solutions, 2010).

"El sistema conserva varias copias o réplicas idénticas de una tabla. Cada réplica se almacena en un nodo diferente" (Morales, 2011).

A partir de los conceptos anteriores se puede concluir que, la replicación es un recurso que permite, intercambiar información entre bases de datos geográficamente distribuidas, de forma segura. Una

réplica, se puede definir como una acción del concepto de la replicación. Algunas características generales de la replicación son las siguientes (Martin, 2012):

Efectividad: depende de la forma en la que los datos son distribuidos y almacenados. A mayor efectividad, mayor será la disponibilidad de datos para ejecutar procesos paralelos.

Alta Disponibilidad: es la razón de tiempo prudente en la que un servicio puede ser accedido. En el mejor de los casos puede ser de un 100%, a pesar de los fallos que se puedan presentar en el servidor, ya que debe existir un servidor adicional que posea alguna técnica de replicación que lo pueda suplantar en caso de ser necesario.

Tolerancia a fallos: garantiza un comportamiento correcto, donde efectivamente pueden existir un número finito de fallos y tipos de fallos.

Coordinación: cada una de las partes que conforman la BD distribuida acuerda un consenso para realizar las invocaciones de los servicios a los objetos, que al final de la transacción debe realizarse tal y como fue solicitada, para lo cual debe utilizar algún tipo de ordenamiento.

Como las principales ventajas de utilizar la replicación se tienen las siguientes (Replicación de SQL Server):

Alta disponibilidad: se puede incrementar la disponibilidad de una BD 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. Esta es una técnica usada para compartir el trabajo a realizar entre varias computadoras.

Soporte para aplicaciones de alto consumo: se pueden satisfacer las necesidades de ciertos clientes que requieren un alto consumo en consultas, que sería muy costoso 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.

1.1.1. Tipos de réplicas

Las réplicas son implementadas atendiendo a que reciben diferentes clasificaciones según sus características. A continuación, se sintetizan las características de la réplica según la forma de transmisión de los datos (Pérez, 2013) y (López, 2011):

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 cierto intervalo de tiempo y se pueden realizar modificaciones a los datos sin conexión de red.

Forma de capturar y almacenar los cambios a replicar:

Para entender esta clasificación primeramente es necesario conocer cuáles son los objetos o mecanismos de BD utilizados. Dichos objetos son:

Trigger (disparador)**:** es un objeto de base de datos con nombre que se asocia a una tabla, y se activa cuando ocurre un evento en particular para la tabla. Algunos usos para los disparadores es verificar valores a ser insertados o llevar a cabo cálculos sobre valores involucrados en una actualización.

Logs: se usa el término *log* o historial de *log* a la grabación secuencial en un registro, ya sea en un archivo o en una base de datos.

Una vez analizados estos objetos entonces se define lo siguiente:

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*).

La implementación de estas réplicas, constituyen soluciones a problemas sobre las BD distribuidas. En dependencia de dichos problemas se realiza la configuración de las mismas. Las características de la configuración van en dependencia de la herramienta para replicación utilizada. Es por ello que en dependencia de las características del problema se seleccionen las herramientas que más se adecuan. En el siguiente epígrafe, se describe algunas de las herramientas que se pueden utilizar para llevar a cabo este proceso.

1.2. SymmetricDS como herramienta para la configuración de réplicas.

SymmetricDS es un software de replicación de datos asíncrona que permite subscriptores múltiples y sincronización bidireccional. Utiliza tecnologías web y de BD para replicar tablas entre los BD relacionales asincrónicamente, casi en tiempo real. Fue diseñado para escalar a un gran número de BD, trabajar con conexiones de bajo ancho de banda, resistir a espacios de tiempo de inoperatividad de la red y se comporta como múltiples-maestros. Puede funcionar con los siguientes gestores de BD: MySQL, Oracle, SQL Server, PostgreSQL, DB2, Firebird, HSQLDB, H2, y Apache Derby. La aplicación SymmetricDS permite que los cambios de entrada y de salidas sean sincronizadas con otras BD.

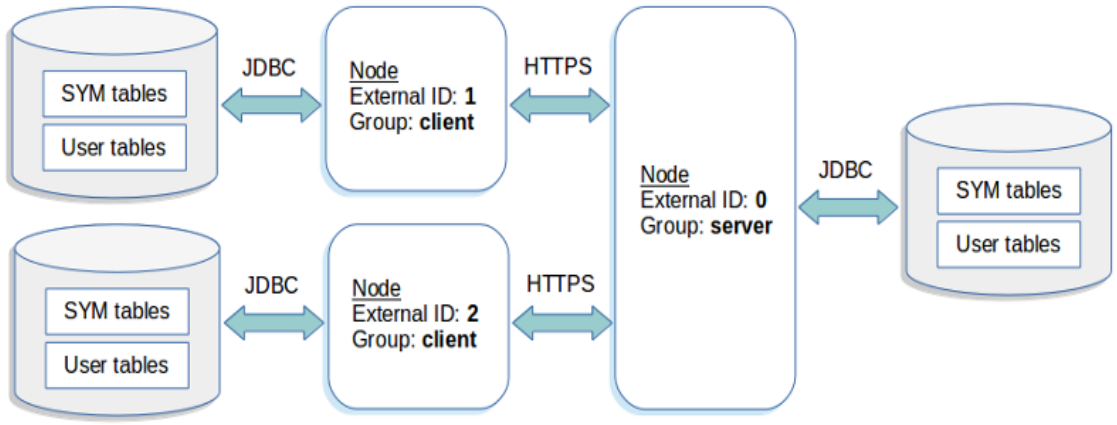


Ilustración 1. Arquitectura de SymmetricDS

El nodo que inicia la conexión es el cliente y el nodo que la recibe es el servidor. Teniendo en cuenta que la sincronización está configurada para extraer o enviar en cualquier dirección, el mismo nodo puede actuar a la vez como cliente o como servidor en diferentes circunstancias.

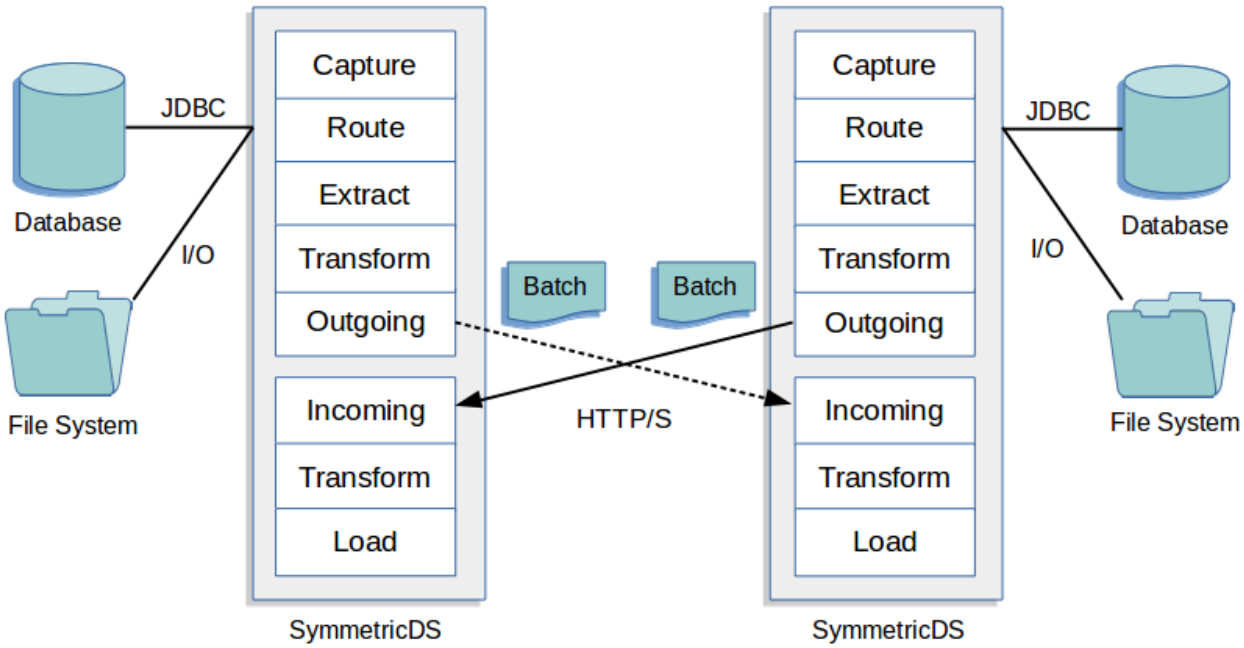


Ilustración 2. Flujo de trabajo de SymmetricDS

SymmetricDS es independiente del gestor de base de datos que se utilice, siempre y cuando soporte la tecnología de *trigger* y driver JDBC 1. Es una herramienta en constante evolución con una amplia comunidad, fácil de aprender y con más de una posibilidad para su despliegue, lo que permite usar la

forma más adecuada según las características del hardware que se posea (Community, S. SymmetricDS. 2013).

1.3. Arquitectura Xalix

La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema. Una Arquitectura de Software, también denominada Arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan un marco definido y claro para interactuar con el código fuente del software (Wilson, 1999).

Una arquitectura de software se selecciona y diseña con base en objetivos (requerimientos) y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, adaptabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos. Toda arquitectura debe ser implementada en un marco de trabajo físico, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea.

La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar, un cierto número de elementos arquitectónicos de forma adecuada, para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad.

El centro FORTES de la UCI desarrolla una serie de proyectos enmarcados en la línea Xauce dedicada a la educación, dentro de ellos se encuentran la plataforma educativa Zera y el Sistema de Gestión de ingreso a la Educación Superior, a los cuales se hacía referencia en la introducción del documento. Estas plataformas deben ser desplegadas a lo largo del territorio nacional. Esto implica que se requiera replicación para la actualización de los mismos dado a que tendrán un servidor central y/o nodo central y varios hijos que dependan de él. Para con la utilización de algún mecanismo de replicación se puede garantizar la actualización de cada uno de los hijos, así como su retroalimentación con el servidor central.

Para el desarrollo de estos proyectos fue utilizada la Arquitectura Xalix que es una arquitectura definida por el centro de producción para sus proyectos utilizando Symfony 2 de conjunto con varios *bundles* de

terceros u administrativos. Estos se encuentran ya definidos de forma genérica para la implementación de cualquier nuevo proyecto que requiera funcionalidades que ellos brinden (Manso Guerra, 2016).

Dado que tanto los proyectos Zera y SIGIES como la plataforma en desarrollo Félix Varela, pueden, en algún momento determinado requerir réplicas se requiere la implementación de un *bundle*, que permita incluir dentro de la arquitectura Xalix servicios de réplicas. Por lo que el producto al que el documento hace referencia debe estar enmarcado en las tecnologías que implementa dicha arquitectura, o sea una versión del *framework* Symfony 2 o superior, y basarse en la metodología de desarrollo que se utiliza en dichos proyectos la cual es la variación AUP UCI de la metodología AUP.

1.4. Sistemas similares

Hasta el momento se ha establecido la definición de los principales conceptos referentes a la replicación, y cuáles son las herramientas que pueden ser utilizadas para replicar, sin embargo, para dar solución al problema planteado, se debe hacer un estudio de sistemas que funcionen como interfaz gráfica para herramientas de planificación y configuración de réplicas. A continuación, se hace mención de algunos de sistemas similares.

1.4.1. SymmetricDS-Pro

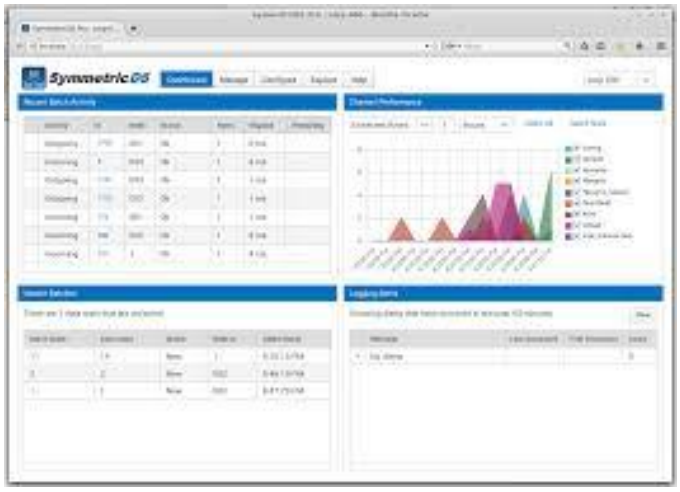


Ilustración 3. Visual de SymmetricDS-Pro

La herramienta SymmetricDS Pro (Eric Long, 2007 – 2013) presenta una interfaz web 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.

1.4.2. Herramienta administrativa para mecanismo de replicación SymmetricDS.

Es una herramienta libre, que funciona sobre la web 2.0 y administra con eficiencia todos los objetos del mecanismo de replicación SymmetricDS. Está desarrollada en lenguaje Python y permite realizar tanto monitorización como configuración, evitando el tecleo de consultas SQL sin embargo no hace

ningún cambio en los comandos por consola que recibe SymmetricDS (Figueroa, 2011). La versión de Symmetric utilizada para el desarrollo es la 3.6.18 mientras que la última versión estable es la 3.8.17 por lo que la solución no es estable para el FORTES. Además, las tecnologías en que está desarrollada la herramienta administrativa no son compatibles con la Arquitectura Xalix por lo que se puede deducir que este sistema no es una solución fiable para el proyecto.

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

Es una aplicación web desarrollada en la Facultad de Ciencias y Tecnologías Computacionales de la UCI. El lenguaje utilizado fue Java con el *framework Spring* java. Permite la configuración de réplicas con SymmetricDS al igual que la monitorización. Tiene como requerimiento inicial, que debe mediante el terminal haberse configurado los nodos en cada uno de los servidores, y es utilizarlo (Arianne Ferrer Carcacés, 2014). Está desarrollada para la versión 3.7.12 de SymmetricDS la cual no es una versión estable. Las tecnologías que sustentan esta solución no son integrables con la Arquitectura por lo que este sistema no es una solución fiable para los productos desarrollados por FORTES.

1.4.4. Comparación entre los sistemas estudiados

En base a los sistemas caracterizados anteriormente se establece una comparación mostrada en la tabla 1. En función de lo observado se puede afirmar que no existe una aplicación que permita la configuración y monitorización de réplicas con SymmetricDS que incluya dentro de sus funcionalidades tareas relacionadas al trabajo con los comandos y que al mismo tiempo sea integrable con la Arquitectura Xalix.

Para establecer la comparación se tuvieron en cuenta las tecnologías que integran Xalix las cuales fueron explicadas anteriormente, así como las tecnologías en que fueron desarrolladas las soluciones investigadas como es el caso del lenguaje de programación. También se tiene en cuenta si permite las opciones de configuración y monitorización, así como la existencia de licencias privativas.

Ninguna de las herramientas estudiadas permite una integración fluida con Xalix. además de que no contemplan los comandos necesarios para la configuración de SymmetricDS, por ello debe ser desarrollado un módulo para Xalix que si incluya todas las opciones para configurar y monitorizar y que tenga en cuenta una mejora para los comandos ejecutados a través de la consola.

Tabla 1. Comparación entre sistemas similares

Sistema	Licencia privativa o costeable	Interfaz web	Permite configuración de Symmetric	Permite monitorización de Symmetric	Lenguaje desarrollado	Es integrable con Xalix
SymmetricDS-Pro	Sí	Sí	Sí	Sí	Java	No
Herramienta administrativa para mecanismo de replicación SymmetricDS	No	Sí	Sí	Sí	Python	No
Interfaz web para la administración y monitorización para la herramienta de réplica de datos SymmetricDS	No	Sí	Sí	Sí	Java	No

1.5. Lenguajes de programación y framework de desarrollo

Un lenguaje de programación es un lenguaje formal diseñado para realizar procesos que pueden ser llevados a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana (O'Reilly Media, 2010).

Está formado por un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura, el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura,

se compila (de ser necesario) y se mantiene el código fuente de un programa informático se le llama programación. Como se mencionó anteriormente la aplicación se rige por la Arquitectura Xalix que define como lenguaje de programación PHP con el *framework* de desarrollo Symfony en su versión 2.7.1 debido a que es la que se utiliza en los proyectos actuales. A continuación, se hace una breve descripción de los mismos.

1.5.1 Lenguaje PHP

Lenguaje de programación, interpretado, diseñado originalmente para la creación de Páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (*server-side scripting*), pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.PHP es un acrónimo recursivo que significa PHP *Hypertext Pre-processor* (inicialmente PHP Tools, o, *Personal Home Page Tools*) (Palomo & Pérez, 2010).

Fue creado originalmente por Rasmus Lerdorf en 1994; sin embargo, la implementación principal de PHP es producida ahora por *The PHP Group* y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la licencia PHP, la *Free Software Foundation* considera esta licencia como software libre.

1.5.2. Symfony 2.7.1

Un *framework* simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un *framework* facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las Aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación (Eguiluz, 2012).

Dentro de las características de Symfony se pueden enumerar las siguientes (Eguiluz, 2012):

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y Linux.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

1.5.3 Doctrine para mapeo relacional de objetos

Doctrine es un mapeador de objetos relacional (ORM) que proporciona persistencia para objetos PHP. Está por encima de la capa de abstracción a la base de datos. Una de sus características es la posibilidad de escribir consultas a la base de datos a partir del tratamiento con objetos en PHP, esta capacidad se denomina *Doctrine Query Language* (DQL) (The Symfony Book, 2016).

Otra de las características de Doctrine es el bajo nivel de configuración que se necesita para comenzar un proyecto. Doctrine puede generar clases a partir de una base de datos creada, y el programador puede especificar relaciones y agregar funcionalidades comunes para las clases generadas. No hay necesidad de generar o mantener esquemas complejos en XML, como vistos en otros *frameworks* (The Symfony Book, 2016).

Otra característica de Doctrine es la habilidad de escribir consultas a la base de datos a partir de la programación orientada a objetos, llamada DQL (*Doctrine Query Language*). Estas clases proporcionan a los desarrolladores alternativas de gran alcance para SQL de mantener la flexibilidad y todavía permiten el cambio de base de datos de back-ends, sin necesidad de la duplicación de código. Además cuenta con las siguientes propiedades (Eguiluz, 2012):

- Soporte para jerarquía (árbol de estructura) de datos.
- Soporte para los ganchos (métodos que puede validar o modificar la base de datos de entrada y salida) y los eventos a la estructura lógica de negocios relacionados con ellos.
- Herencia columna de agregación (objetos similares se pueden almacenar en la tabla de base de datos, con un tipo de columna que especifica el subtipo del objeto en particular - la subclase correcta siempre se devuelve cuando se hace una consulta).
- Un marco de almacenamiento en caché, haciendo uso de varios *backends* como *memcached*, *SQLite* o *APC*.
- El ácido transacciones.
- Comportamientos del modelo (Sluggable, Timestampable, conjunto anidado, la internacionalización, registro de auditoría, el índice de búsqueda).
- Migraciones de bases de datos.

- Una "compilación" la función de combinar muchos archivos PHP del marco en una sola, para evitar el impacto en el rendimiento efectuados por lo general mediante la inclusión de los muchos archivos PHP de un marco.

1.5.4 Lenguajes del lado cliente

HTML

El HTML (*Hyper Text Markup Language*) es el lenguaje con el que se escriben las páginas web. Es un lenguaje de hipertexto, es decir, un lenguaje que permite escribir texto de forma estructurada, y que está compuesto por etiquetas, que marcan el inicio y el fin de cada elemento del documento (Gauchat, 2015).

Un documento hipertexto no sólo se compone de texto, puede contener imagen, sonido, vídeo, etc., por lo que el resultado puede considerarse como un documento multimedia. Los documentos HTML deben tener la extensión html o htm, para que puedan ser visualizados en los navegadores (programas que permiten visualizar las páginas web) (Eguíluz J. , 2013).

Los navegadores se encargan de interpretar el código HTML de los documentos, y de mostrar a los usuarios las páginas web resultantes del código interpretado (Gauchat, 2015).

Twig

Es un motor de plantillas para el lenguaje de programación PHP. Su sintaxis se origina a partir de Jinja y plantillas de Django. El producto de código abierto se distribuye bajo licencia BSD y desarrollado por Fabien Potencier. El *framework* Symfony2 viene con un soporte incluido para Twig como su motor de plantillas por defecto (Eguiluz, 2012).

Características

- **Rápido**: Twig compila las plantillas a código PHP sencillo optimizado. La sobrecarga en comparación con el ordinario de código PHP se ha reducido a la mínima expresión.

- **Seguro**: Twig tiene un modo de recinto para evaluar código de la plantilla no es de confianza. Esto permite Twig para ser utilizado como un lenguaje de plantillas para aplicaciones donde los usuarios pueden modificar el diseño de la plantilla.

- **Flexible**: Twig es impulsado por un léxico flexible y analizador. Esto permite al desarrollador definir sus propias etiquetas y filtros personalizados, y crear su propio DSL.

JavaScript

Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico (Eguíluz J. P., 2013).

Características de Javascript

- Es simple, no hace falta tener conocimientos avanzados de programación para poder hacer un programa en JavaScript (Gauchat, 2015).

- Maneja objetos dentro de nuestra página Web y sobre ese objeto podemos definir diferentes eventos. Dichos objetos facilitan la programación de páginas interactivas, a la vez que se evita la posibilidad de ejecutar comandos que puedan ser peligrosos para la máquina del usuario, tales como formateo de unidades y modificación de archivos (Eguíluz J. P., 2013).

- Es dinámico, responde a eventos en tiempo real. Eventos como presionar un botón, pasar el puntero del mouse sobre un determinado texto o el simple hecho de cargar la página o caducar un tiempo. Con esto podemos cambiar totalmente el aspecto de nuestra página al gusto del usuario, evitándonos tener en el servidor un página para cada gusto, hacer cálculos en base a variables cuyo valor es determinado por el usuario (Gauchat, 2015).

1.6. Gestor de Bases de Datos PostgreSQL

Siguiendo lo que plantea el marco de trabajo Xalix se utiliza PostgreSQL como gestor de BD. A continuación, se hace una descripción del mismo mediante una selección de diferentes autores:

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarles a otras bases de datos comerciales (web oficial de Postgre SQL, 20013).

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (Reingart, 2016).

A continuación, se explican de manera general los componentes más importantes en un sistema PostgreSQL (Sotolongo León & Vazquez , 2017):

Aplicación cliente: Esta es la aplicación cliente que utiliza PostgreSQL como administrador de bases de datos. La conexión puede ocurrir vía TCP/IP o sockets locales.

Demonio postmaster: Este es el proceso principal de PostgreSQL. Es el encargado de escuchar por un puerto/socket por conexiones entrantes de clientes. También es el encargado de crear los procesos hijos que se encargaran de autentificar estas peticiones, gestionar las consultas y mandar los resultados a las aplicaciones clientes.

Ficheros de configuración: Los tres ficheros principales de configuración utilizados por PostgreSQL, postgresql.conf, pg_hba.conf y pg_ident.conf.

Procesos hijos postgres: Procesos hijos que se encargan de autentificar a los clientes, de gestionar las consultas y mandar los resultados a las aplicaciones clientes.

PostgreSQL share buffer cache: Memoria compartida usada por PostgreSQL para almacenar datos en caché.

Write-Ahead Log (WAL): Componente del sistema encargado de asegurar la integridad de los datos (recuperación de tipo REDO).

Kernel disk buffer cache: Caché de disco del sistema operativo.

Disco: Disco físico donde se almacenan los datos y toda la información necesaria para que PostgreSQL funcione.

La última serie de producción es la 9.3. Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado. Su desarrollo comenzó hace más de 16 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema (Web oficial de PostgreSQL, 20013).

A continuación se muestran algunas de las características más importantes y soportadas por PostgreSQL (Vasyliev, 2017):

- Integridad referencial.

- Tablespaces.
- Nested transactions (savepoints).
- Replicación asincrónica/sincrónica / Streaming replication - Hot Standby.
- Two-phase commit.

Tiene otras ventajas las cuales se podrán encontrar en los anexos.

1.7. Entorno de desarrollo integrado (IDE)

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solos o pueden ser parte de aplicaciones existentes. El lenguaje Visual Basic, por ejemplo, puede ser usado dentro de las aplicaciones de Microsoft Office, lo que hace posible escribir sentencias Visual Basic en forma de macros para Microsoft Word (Rodríguez Sala , 2013).

Los IDE ofrecen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto, como es el caso de Smalltalk u Objective-C (Rodríguez Sala , 2013).

Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. Este es el caso de Eclipse, al que mediante *plugins* se le puede añadir soporte de lenguajes adicionales.

Un IDE debe tener las siguientes características:

- Multiplataforma.
- Soporte para diversos lenguajes de programación.
- Integración con Sistemas de Control de Versiones.
- Reconocimiento de Sintaxis.
- Extensiones y Componentes para el IDE.

- Integración con *framework* populares.
- Depurador.
- Importar y Exportar proyectos.
- Múltiples idiomas.
- Manual de Usuarios y Ayuda.

A continuación, se muestran algunos IDE que fueron investigados con objetivos de identificar el más adecuado para el desarrollo de la aplicación.

Eclipse

IDE de código abierto multiplataforma para desarrollar proyectos. Esta plataforma ha sido usada para desarrollar entornos de desarrollo integrados, como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). También se puede usar para otros tipos de aplicaciones cliente, como *BitTorrent* o *Azureus*. En Eclipse se pueden usar diferentes lenguajes de programación como: Java, ANCI C, C++, JSP, *sh, perl, php, sed* (Eclipse, 2014) .

NetBeans

Programa que sirve como IDE que permite programar en distintos lenguajes, es ideal para trabajar con el lenguaje de desarrollo JAVA (y todos sus derivados), además ofrece un excelente entorno para programar en PHP. También se puede descargar una vez instalado NetBeans, los complementos para programar en C++. La IDE de NetBeans es perfecta y muy cómoda para los programadores. Tiene un excelente balance entre una interfaz con múltiples opciones y un aceptable completamiento de código.

Existe además un número importante de módulos para extender el IDE NetBeans, es un producto libre y gratuito sin restricciones de uso. Esta plataforma es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones. NetBeans ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación (Netbeans, 2015).

Entre las características de la plataforma están (Netbeans, 2015):

1. Administración de las interfaces de usuario (ej. menús y barras de herramientas).
2. Administración de las configuraciones del usuario.
3. Administración del almacenamiento (guardando y cargando cualquier tipo de dato).
4. Administración de ventanas.
5. *Framework* basado en asistentes.

Geanny

Es un IDE que hasta hace bien poquito sólo estaba disponible para sistemas Linux, Mac OS X y BSD, pero ya está disponible para Windows. Este entorno es muy sencillo, pero proporciona las funcionalidades necesarias para desarrollar aplicaciones sin problemas. Su interfaz está dividida en tres zonas: panel lateral con el árbol de carpetas y documentos abiertos, sección principal para el código y panel inferior para los mensajes de la aplicación, compilación, etc. Este IDE permite programar en diferentes lenguajes como: C, C++, Java, Python, Pascal, SQL o HTML (Geany, 2014).

CodeRun

CodeRun: Es un IDE que te permitirá programar en línea varios lenguajes, entre ellos PHP, Ajax, C#, CSS, JavaScript y HTML. Funciona perfectamente, aunque está en inglés, es útil para quién no disponga de un buen editor a mano (Ealasur, 2013).

IDE seleccionado

Tras el análisis realizado se decide que el IDE más adecuado para efectuar el desarrollo del sistema por todas las ventajas que brinda es NetBeans en su versión 8.0.

1.8. Metodología de desarrollo

Las metodologías de desarrollo de software son un conjunto de procedimientos y pasos que deben ser seguidos para desarrollar un software. O sea que permiten guiar el ciclo de vida del software dividiéndolo en etapas. En dependencia de las características de un determinado proyecto, la correcta selección de la metodología puede representar el éxito o el fracaso de dicho proyecto (Pressman R., 2010).

Una metodología define una estrategia global para enfrentarse con el proyecto. Entre los elementos que forman parte de una metodología se pueden destacar:

Fases: tareas a realizar en cada fase.

Productos: E/S de cada fase, documentos.

Procedimientos y herramientas: apoyo a la realización de cada tarea.

Criterios de evaluación: del proceso y del producto. Saber si se han logrado los objetivos.

Una metodología de desarrollo de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información. Una gran variedad de estos marcos de trabajo ha evolucionado durante los años, cada uno con sus propias fortalezas y debilidades. Una metodología de desarrollo de sistemas no tiene que ser necesariamente adecuada para usarla en todos los proyectos. Cada una de las metodologías disponibles es más adecuada para tipos específicos de proyectos, basados en consideraciones técnicas, organizacionales, de proyecto y de equipo (Sommerville, 2007).

1.8.1 Metodología seleccionada

La metodología seleccionada para el desarrollo se basa en la metodología utilizada para los proyectos productivos de la UCI para lograr una estandarización en los productos con propósito de la certificación de la norma CMMI en su nivel dos. La metodología que se propone es la versión ágil de *Rational Unified Process* (RUP) que es el Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP, por sus siglas en inglés), pero con la variación realizada en la UCI, AUP-UCI. Esta metodología combina los artefactos de RUP con el ciclo de vida de XP *(Extrem Programming)*.

1.8.1.1 Metodología AUP

El AUP es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo (Edeki, 2013):

- Desarrollo Dirigido por Pruebas (test driven development - TDD en inglés).
- Modelado ágil.
- Gestión de Cambios ágil.
- Refactorización de Base de Datos para mejorar la productividad.
- Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva.

1.8.1.2 Metodología AUP en su variante UCI

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI (Rodríguez T., 2015).

Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, para ello se sigue el Modelo CMMI-DEV v1.3. Dicho modelo constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad. Por lo descrito anteriormente para seguir con lo establecido por los el proceso productivo de la UCI se decide utilizar esta metodología para el desarrollo del módulo en específicamente en el escenario 4 asociado a Historias de Usuario.

1.9. Lenguaje de modelado

El UML, por sus siglas en inglés, Unified Modeling Language: es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el Object Management Group (OMG). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables (Oestereich y Weilkiens, 2007).

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional), pero no especifica en sí mismo qué metodología o proceso usar (Oestereich y Weilkiens, 2007).

UML no puede compararse con la programación estructurada, pues UML significa (Lengua de Modelación Unificada), no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la orientación a objetos, sin embargo, la orientación a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas (Oestereich y Weilkiens, 2007).

1.10. Herramienta CASE para el modelado.

Cumputer Aided Software Engineering (CASE). Conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de Software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. Este puede ser generalmente aplicado a cualquier sistema o colección de herramientas que ayudan a automatizar el proceso de diseño y desarrollo de software.

Algunas de las herramientas CASE estudiadas fueron las siguientes:

1.10.1 ArgoUML

Es una herramienta libre de modelado sencilla de utilizar, que incluye soporte para los diagramas del estándar UML y se puede utilizar tanto para realizar los diagramas de apoyo a la ingeniería de software como aplicar la Ingeniería inversa a proyectos ya terminados. (CollabNet, 2016)

Sin embargo, desde la versión 0.20, ArgoUML está incompleto. No es conforme completamente a los estándares UML y carece de soporte completo para algunos tipos de diagramas de secuencia y los de colaboración (CollabNet, 2016).

Características

- La mayoría de las funciones ahora soportan la selección múltiple de los elementos del modelo.
- Se puede arrastrar y soltar desde el árbol de exploración al diagrama y dentro del árbol de exploración.
- Construido en diseños críticos, suministra una revisión no obstructiva del diseño y sugerencias para mejoras.
- Restricciones OCL para clases.
- Soporte para el lenguaje de generación de código: Java, PHP, Python, C++ y C Sharp.
- Ingeniería inversa.
- Disposición (layout) automática del diagrama de clases.
- Todos los diagramas 1.4 están soportados.

Ventajas

- Genera código automáticamente.
- Propone soluciones a algunos errores.
- Incluye un panel de propiedades y de tareas pendientes bastante útil.

Desventajas

- Instalación costosa.

- Poco amigable.
- Los modelos a veces no pueden ser re-abiertos.
- Se debe seleccionar una clase para crear un diagrama de secuencia.

1.10.2 POSEIDON para UML

Es una herramienta para modelar cualquier clase de sistema que esté o no relacionada con programación. Poseidon para UML puede simplificar la compleja tarea de desarrollo de software ayudando a estructurar pensamientos, a clarificar la comunicación, y a encontrar la correcta abstracción. La incorrecta implantación de la herramienta UML, le sumergirá en detalles llenos de funciones extrañas y excesivamente complicadas, lo que le evitará el ahorro de tiempo y esfuerzo (Poseidon for UML, 2015).

La intuitiva interfaz hace de Poseidon una de las herramientas más rápidas de UML para dominar el análisis orientado a objetos, liberando al diseñador para centrase solamente en su modelo (Poseidon for UML, 2015).

Características

- Soporta diagramas UML.
- Opciones avanzadas de impresión.
- Soporta gráficos en la mayoría de los formatos.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.

Ventajas

- Herramienta hecha completamente en Java, por lo que es independiente de la plataforma.

- Interfaz de usuario muy bien diseñada, fácil de aprender a usar e intuitiva.

1.10.3 Visual Paradigm

Visual Paradigm es una de las herramientas UML CASE del mercado, considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones (Visual Paradigm, 2014).

Fue creada para el ciclo vital completo del desarrollo de software que lo automatiza y acelera, permitiendo la captura de requisitos, análisis, diseño e implementación. Tiene la capacidad de crear el esquema de clases a partir de una base de datos y crear la definición de base de datos a partir del esquema de las clases (Visual Paradigm, 2014).

Permite invertir código fuente de programas, archivos ejecutables y binarios en modelos UML al instante, creando de manera simple toda la documentación. Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto, además apoya los estándares más recientes de las notaciones de Java y de UML. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros (Visual Paradigm, 2014).

Características

- Producto de calidad.
- Soporta aplicaciones Web.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.
- Se integra con las siguientes herramientas Java: Eclipse/IBM WebSphere, Jbuilder, NetBeans IDE, Oracle Jdeveloper, BEA Weblogic.

Ventajas

- Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes. Tiene apoyo adicional en cuanto a generación de artefactos automáticamente.
- Genera modelos VP-UML instantáneamente a partir de código binario .Net.
- Generación de documentación en formatos HTML y PDF.
- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XP, o Vista), Linux, Mac OS X, Solaris o Java.
- Brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones como por ejemplo Visio y Rational Rose.
- Generación de código e ingeniería inversa: brinda la posibilidad de generar código a partir de los diagramas, para las plataformas como .Net, Java y PHP, así como obtener los diagramas a partir del código.
- Generación de documentación: brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.

Desventajas

- Las imágenes y reportes generados, no son de muy buena calidad.

1.10.4 Herramienta seleccionada

Después del análisis realizado se decide que la herramienta más adecuada es la herramienta Visual Paradigm ya que ofrece mejor cobertura para el lenguaje UML y todos los artefactos que el mismo genera. Dado que se tiene una mayor experiencia de trabajo y tomando en cuenta las ventajas analizadas se decide utilizar Visual Paradigm en su versión 8.0 como herramienta de modelado.

1.11. Consideraciones del capítulo

En el presente capitulo fue realizada una descripción de los principales conceptos, herramientas y tecnologías asociados a la solución a defender. Después de ello se concluye lo siguiente:

- La replicación es un mecanismo que permite el intercambio de información entre BD geográficamente distribuidas.
- Las tecnologías a utilizarse son las regidas por la arquitectura Xalix del centro FORTES que implica como lenguaje de programación PHP en su versión 7.0 con el *framework* de desarrollo Symfony en su versión 2.7.1.
- Se utilizará UML como lenguaje de modelado con la herramienta *CASE* Visual Paradigm.
- El entorno de desarrollo a utilizar es Netbeans en su versión 8.0.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN. ANÁLISIS Y DISEÑO

Las características del sistema a desarrollar y su descripción detallada son los elementos que se expondrán en el capítulo. Se utiliza modelo de dominio para describir los principales conceptos asociados al dominio del problema y sus relaciones. Se diseña y describe la propuesta de solución al problema planteado. Se realiza la captura de los requisitos funcionales y no funcionales del producto, las historias de usuario para la descripción de los mismos y los artefactos generados durante las fases de análisis y diseño.

2.1. Propuesta de solución

La configuración de forma manual de la herramienta SymmetricDS consta de tres entradas fundamentales: un fichero de configuración, comandos que se deben ejecutar desde la consola y consultas formuladas mediante código SQL. Como se explicaba anteriormente este proceso se puede tornar complejo y largo, principalmente para especialistas con poca experiencia de trabajo directo con bases de datos distribuidas. Además, Xalix no cuenta hasta el momento con ningún mecanismo que le permita simplificar estas configuraciones.

Es por ello que como propuesta de solución se plantea el desarrollo de un módulo que permita unificar las tres entradas de SymmetricDS bajo la Arquitectura de Xalix de tal forma que dicha arquitectura quede completamente integrada con SymmetricDS y permita a través de la web hacer las configuraciones necesarias, y que recoja los comandos que brinda la herramienta como comandos del marco de trabajo siendo reconocidos por la interfaz web para comandos que posee cada proyecto de tipo plataforma que es desarrollado por FORTES.

2.2. Modelo de dominio

El modelo del dominio captura, comprende y describe los objetos más importantes dentro del contexto de un sistema. Se describe mediante diagramas de UML, especialmente mediante diagramas de clases. Los objetos del dominio o clases pueden obtenerse a partir de una especificación de requisitos o mediante la entrevista con los expertos del tema. Las clases que lo componen suelen aparecer en tres formas típicas (Jacobson, Booch y Rumbaugh, 2000):

- Objetos del negocio que representan los elementos que se manipulan en el negocio.

- Objetos del mundo real y conceptos de los que el sistema debe hacer un seguimiento.
- Sucesos que ocurrirán o que han ocurrido.

El modelo de dominio que a continuación se describe constituye una representación gráfica de los principales conceptos que forman parte del módulo para la configuración y monitorización de réplicas con la herramienta SymmetricDS para la Arquitectura Xalix.

2.2.1. Descripción de los conceptos del modelo de dominio

A continuación, se esclarecen los conceptos asociados al dominio, definiendo los términos más importantes que serán de utilidad para la posterior confección del modelo de dominio.

- **Administrador**: Es la persona que interactúa con el sistema y le indica al mismo que acciones debe realizar.
- **SymmetricDS**: Es la herramienta de replicación descrita en el capítulo 1, que el sistema utilice para replicar.
- **Interfaz web**: Componente visual de una aplicación, en el sistema permite configurar y monitorizar la herramienta SymmetricDS.
- **Ficheros de configuración**: Ficheros de texto que utilice la herramienta SymmetricDS para su funcionamiento en los cuales se especifican un grupo de parámetros necesarios para el sistema.
- **Nodo**: Es la representación física de una base de datos en un servidor para establecer réplicas.
- **Servidor:** Componente físico de hardware donde se almacena la base de datos, y en donde se configuran los nodos.
- **Base de Datos**: Conjunto de información organizado en tablas asociados al sistema. Es lo que se busca replica con el uso de la herramienta SymmetricDS.
- **Plataforma**: Es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o de software con los que es compatible. Dicho sistema está definido por un estándar alrededor del cual se determina una arquitectura de hardware y una plataforma de software (Morales, 2011).

- **FORTES**: Centro de desarrollo de software, varios de sus productos constituyen plataformas que requieren la utilización de un sistema de réplicas, lo que lo convierte en el principal cliente de la aplicación.

- **Arquitectura Xalix**: Es el marco de trabajo utilizado para las plataformas desarrolladas por FORTES.

2.2.2. Diagrama del modelo de dominio

El siguiente diagrama muestra la relación entre los conceptos identificados del problema descrito anteriormente.

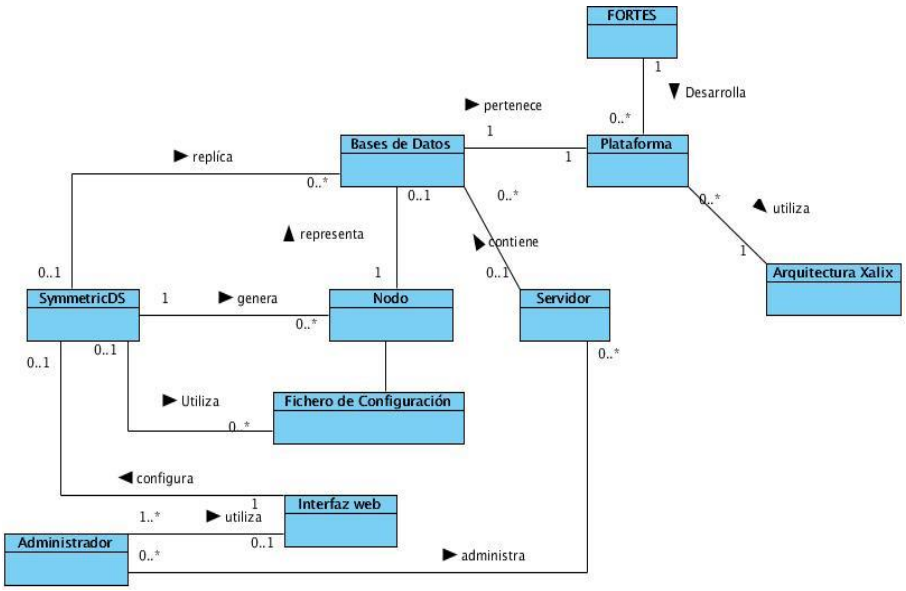


Ilustración 4 Diagrama del modelo de dominio

2.3. Requisitos funcionales y no funcionales

El proceso de desarrollo de software comprende en sus etapas tempranas la definición de tareas orientadas a captar las necesidades o características para satisfacer el sistema que se vaya a crear o modificar (Pressman R., 2010).

2.3.1. Requisitos funcionales

Los requisitos funcionales representan las funcionalidades del sistema y se modelan mediante Historias

de Usuario (HU). Para la propuesta de solución se definen los siguientes requisitos *funcionales:*

RF1: Iniciar servicio Symmetric.

RF2: Detener servicio Symmetric.

RF3: Consultar estado del servicio Symmetric.

RF4: Instalar SymmetricDS.

RF5: Desinstalar SymmetricDS.

RF6: Abrir registro para nodos desde el servidor.

RF7: Registrar nodos en la configuración del servidor enviando carga inicial.

RF8: Cambiar configuración de puertos de conexión para el servidor.

RF9: Crear fichero de Configuración de Symmetric.

RF10: Editar fichero de Configuración de Symmetric.

RF11: Listar ficheros de Configuración de Symmetric.

RF12: Borrar fichero de Configuración de Symmetric.

RF13: Ver fichero de Configuración de Symmetric.

RF14: Generar estructura de tablas de Symmetric.

RF15: Iniciar Symmetric en el Nodo.

RF16: Iniciar Symmetric para todas las configuraciones disponibles.

RF17: Incluir un Canal.

RF18: Editar un Canal.

RF19: Borrar un Canal.

RF20: Ver un Canal.

RF21: Listar Canales.

RF22: Incluir un Grupo de Nodos.

RF23: Editar un Grupo de Nodos.

RF24: Borrar un Grupo de Nodos.

RF25: Ver un Grupo de Nodos.

RF26: Listar Grupos de Nodos.

RF27: Incluir un enlace entre Grupo de Nodos.

RF28: Editar un enlace entre Grupo de Nodos.

RF29: Borrar un enlace entre Grupo de Nodos.

RF30: Ver un enlace entre Grupo de Nodos.

RF31: Listar enlaces entre Grupos de Nodos.

RF32: Incluir un Disparador.

RF33: Editar un Disparador.

RF34: Borrar un Disparador.

RF35: Ver un Disparador.

RF36: Listar Disparadores.

RF37: Incluir un Enrutador.

RF38: Editar un Enrutador.

RF39: Borrar un Enrutador.

RF40: Ver un Enrutador.

RF41: Listar Enrutador.

RF42: Gestionar Disparadores-Enrutadores.

RF43: Incluir relación entre un Disparador y un Enrutador.

RF44: Editar relación entre un Disparador y un Enrutador.

RF45: Borrar relación entre un Disparador y un Enrutador.

RF46: Ver relación entre un Disparador y un Enrutador.

RF47: Listar relaciones entre Disparadores y un Enrutadores.

RF48: Incluir un Nodo.

RF49: Editar un Nodo.

RF50: Borrar un Nodo.

RF51: Ver un Nodo.

RF52: Listar Nodo.

RF53: Definir seguridad en un Nodo.

RF54: Actualizar seguridad en un Nodo.

RF55: Crear Identidad de un Nodo en el servidor.

Para monitorizar:

RF56: Listar lotes de salida.

RF57: Ver un lote de salida.

RF58: Listar los lotes de entrada.

RF59: Ver lote de entrada.

2.3.2. Requisitos no funcionales

Los requerimientos no funcionales son requisitos que imponen restricciones en el diseño o implementación. Son propiedades o cualidades que el producto debe tener (Pressman R., 2010). Seguidamente se muestran los requisitos no funcionales para la solución propuesta.

De usabilidad:

- Cumplir con las pautas de diseño establecidas en la Estrategia Marcaria de la Universidad. Cuando se crea/actualiza/elimina un elemento, así como el cancelar, se muestra un mensaje con el resultado de la acción.

De portabilidad:

- Java SE Runtime Environment 7 o superior.
- Memoria - 64 (MB).
- Disco - 256 (MB) disponible.
- Para ejecutar las acciones de configuración el usuario debe tener permiso *root.*
- Deben estar habilitados los puertos a través de los cuales se configuran las réplicas.
- Debe tener creados en la BD un usuario para que sea el que utilice Symmetric.

Los requisitos de memoria, disco y CPU aumentan con el número de clientes conectados y la cantidad de datos que se están sincronizando. La mejor manera de dimensionar un servidor es simular la sincronización en el entorno inferior y la carga de datos de referencia. Sin embargo, una regla general para los servidores es una CPU de clase servidor con 2 GB de memoria por cada 500 MB / hora de transferencia de datos y 350 clientes. Pueden utilizarse varios servidores como un clúster detrás del equilibrador de carga para lograr un mejor rendimiento y disponibilidad (C. H. Eric Long, 2014).

2.3.3. Historias de Usuario

Las HU son la forma en que se especifican en las metodologías ágiles los requisitos del sistema, las

mismas no deben ser descritas en más de tres líneas e idealmente es el cliente quien las redacta y prioriza. Por tanto, serán descripciones cortas y escritas en el lenguaje del usuario, sin terminología técnica (Cruz Castro, 2011).

Las HU deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Cuando llegue el momento de la implementación, los desarrolladores dialogarán directamente con el cliente para obtener todos los detalles necesarios. Las HU deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia (Joskowics, 2008).

En el cuarto escenario de AUP, en su variante UCI, los requisitos se administran utilizando historias de usuario, por lo que para la solución propuesta se describieron un total de 57 Historias de Usuario (Ver Anexo), de las cuales se presenta la correspondiente al RF19:

Número: 19	**Nombre del requisito:** Editar datos del canal
Programador: Reiman Alfonso Azcuy	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:**3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días

Descripción:

Permite al administrador editar los datos de un canal.

1- Objetivo:

Permitir editar datos de un canal.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para modificar los datos de una categoría se debe cumplir que:

- Debe existir en el sistema al menos un canal.
- Tener en cuenta los siguientes datos: identificador del canal, descripción, tamaño máximo del lote, si está habilitado y orden de procesamiento.
- Estar autenticado en el sistema con el rol Administrador.

3- Comportamientos válidos y no válidos (flujo central y alternos):

El campo identificador es obligatorio.

Nombre: campo de texto que admite caracteres alfabéticos y tiene un máximo de hasta 100 caracteres

Descripción: campo de texto que permite cualquier carácter

4- Flujo de la acción a realizar:

- El sistema debe permitir modificar un canal, esta acción puede realizarse seleccionando la opción editar en el listado de canales o desde la vista previa del canal.
- Cuando el usuario modifica de forma correcta los datos necesarios y selecciona la opción Actualizar, se muestra un mensaje de información de que el canal fue modificado de forma correcta.
- Si los datos están incompletos o incorrectos se señalarán los campos en cuestión dando la posibilidad al usuario de realizar nuevamente la acción en cuestión.
- Si selecciona la opción Cancelar regresará a la vista previa.

Observaciones: los canales son independientes del resto de las tablas sin embargo deben estar configurados dado que las acciones de los disparadores se desencadenan desde los canales.

Prototipo de interfaz:

Tabla 2. Historia de usuario correspondiente al RF Editar canal

2.4. Análisis y Diseño.

El análisis tiene como propósito fundamental conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea más fácil de mantener y que ayude a estructurar el sistema completo, incluyendo la arquitectura (JACOBSON, 2000). El diseño permite analizar los requerimientos refinándolo y estructurándolos con el objetivo de dar soporte a todos los requisitos funcionales y no funcionales que se incorporan en el mismo (JACOBSON, 2000). La concepción de los modelos de análisis y diseño es una de las tareas más complejas e importantes dentro del desarrollo de productos de software, cuando se requiere que los sistemas puedan ser interpretados por terceros ajenos al negocio, lo cuál es el objetivo fundamental del presente sub-epígrafe.

2.4.1. Diagramas de clases del análisis.

Las clases del análisis constituyen el artefacto fundamental del modelo de análisis (Pressman R., 2010). Los diagramas de clases del análisis (DCA) se utilizan para mostrar las clases y sus relaciones, además, presentar los subsistemas e interfaces. Una clase de análisis representa una abstracción de una o varias clases y/o subsistemas del diseño de un sistema, las mismas se representan mediante los siguientes estereotipos estandarizados en UML lo que permite a los desarrolladores, distinguir el ámbito

de las diferentes clases (JACOBSON, 2000).

Clases de Interfaz (CI): se utilizan para modelar las interacciones entre el sistema y sus actores. Representan abstracciones de ventanas, formularios, paneles, interfaces de comunicaciones, de impresoras, sensores y terminales.

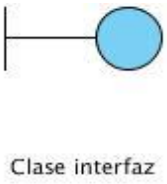


Ilustración 5. Estereotipo de clases interfaz

Clases de Entidad (CE): se utilizan para modelar la información que posee larga vida y que es a menudo persistente. Suelen mostrar una estructura de datos lógica y contribuyen a comprender de qué información depende el sistema.

Ilustración 6. Estereotipo de clases de entidad

Clases de Control (CC): representan coordinación, secuencia, transacciones y control de otros objetos y se usan con frecuencia para encapsular el control de un caso de uso en concreto, además, el manejo y coordinación de las acciones a otros objetos, es decir, objetos de interfaz y de entidad.

Ilustración 7. Estereotipo de clases controladoras

A continuación, se muestra el DCA correspondiente a los RF 18,19,20 y 21 los cuales representan las operaciones que son realizadas sobre la entidad Canal. Los restantes diagramas se encuentran en los anexos.

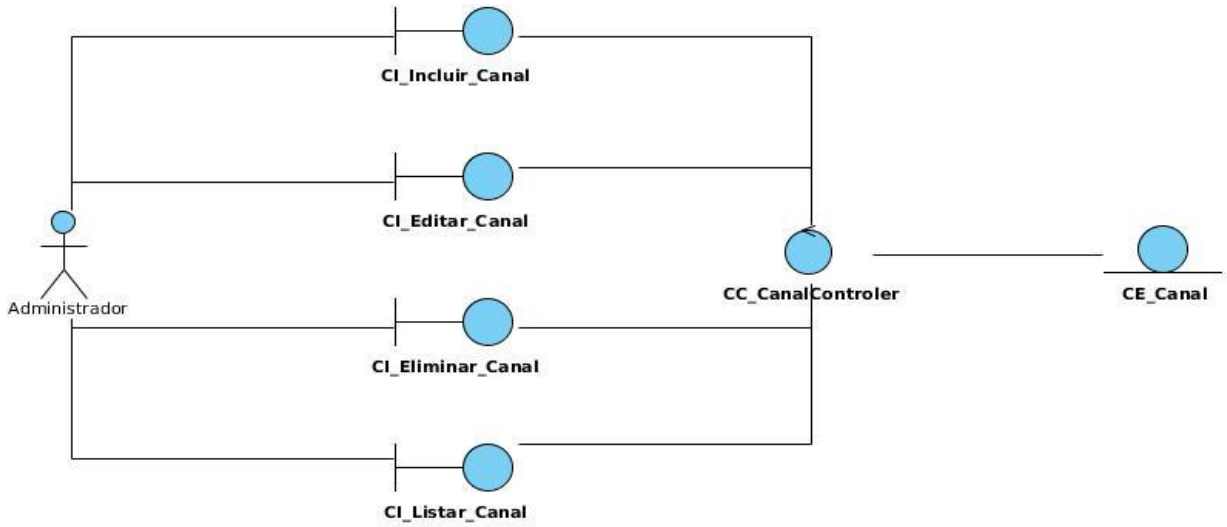


Ilustración 8. DCA para RF18, RF19, RF20 y RF21

2.4.2. Diagramas de colaboración.

Los diagramas de colaboración (DC) son los artefactos que se generan durante el flujo de trabajo Análisis y Diseño, los mismos permiten mostrar las interacciones entre los objetos del análisis, creando enlaces y añadiendo mensajes entre ellos. A continuación, se muestran los DC correspondientes a los RF 19,20 y 21. El resto de los diagramas se encuentran en los anexos.

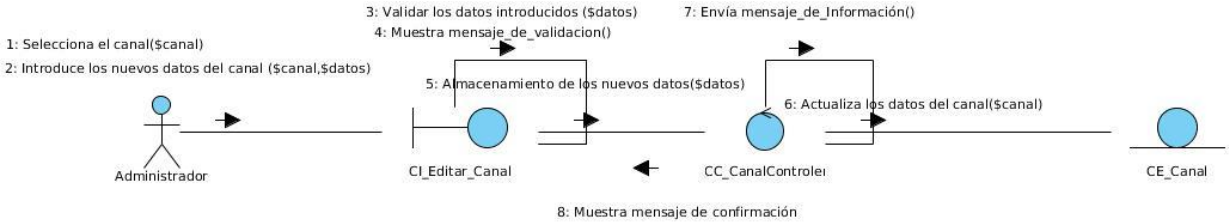


Ilustración 9. Incluir un canal

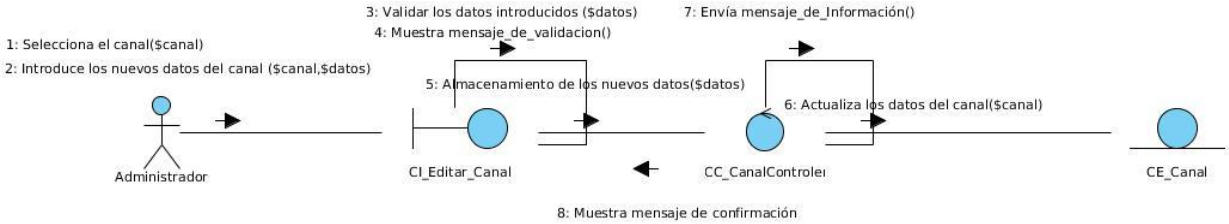


Ilustración 10. Editar un canal

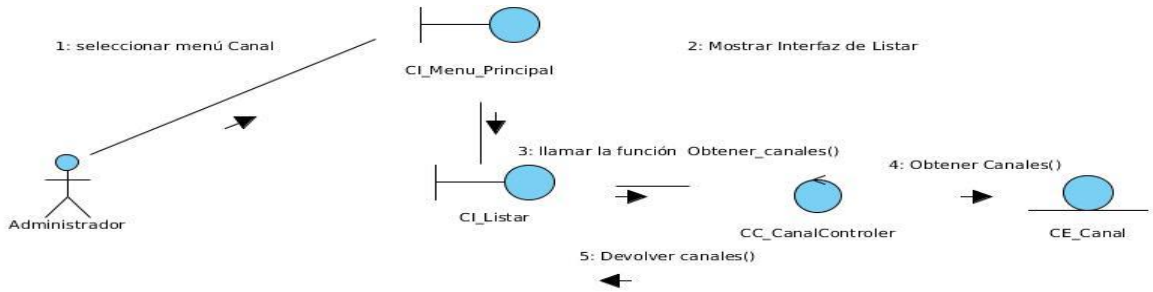


Ilustración 11. Listar Canales

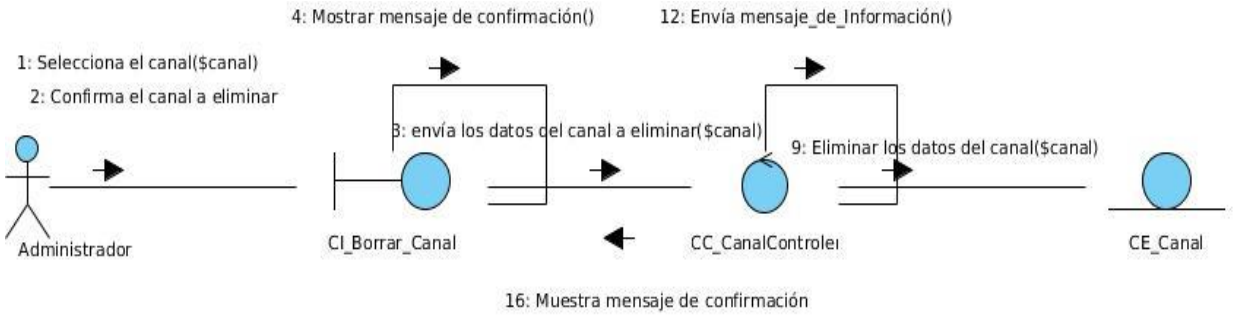


Ilustración 12. Borrar un canal

2.5. Modelo de Diseño

El modelo del diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en los requisitos funcionales y no funcionales, junto a otras restricciones relacionadas con el entorno de implementación (Sommerville, 2007).

La disciplina de modelado de la metodología AUP en su variante UCI comprende las fases de modelado del negocio, requisitos, análisis y diseño. Este modelado es colaborativo, generando los artefactos necesarios para la comprensión del sistema con un mayor nivel de abstracción y una vez obtenidos dichos elementos se da paso a la implementación y las pruebas. Se procede al modelado de diseño del sistema debido a que el equipo de trabajo y el cliente poseen una total claridad de los requerimientos, por lo que se generarán los diagramas de clases y diagramas de secuencia.

2.5.1. Arquitectura

En la construcción de software existen diversos componentes que proporcionan una filosofía de trabajo para los equipos de desarrollo, creando una visión que unifica la forma en que los miembros del equipo ven el sistema y además impone una transformación del mismo. Entre los mencionados componentes se encuentran los estilos arquitectónicos, patrones arquitectónicos y patrones de diseño, los cuales se encuentran estrechamente relacionados, conformando la base de la arquitectura de un software.

Existen diversas definiciones en lo que respecta a arquitectura de software, pero todas se ven reflejadas en las siguientes ideas fundamentales:

- La arquitectura de software como un proceso particular dentro del ciclo de vida.
- La arquitectura de software como la topología o la forma de articular distintos componentes en

una solución.

- La disciplina académica y profesional de la arquitectura de software.

Por ello la arquitectura de software se entiende como la organización fundamental de un sistema representada en sus componentes, las relaciones existentes entre ellos y con el entorno, y los principios que orientan su diseño y evolución (IEEE Computer Society, 2000).

Un estilo arquitectónico es una transformación impuesta al diseño de todo un sistema, cumpliendo el objetivo de establecer una estructura para todos los componentes del mismo. Presentan un alcance global dentro de los sistemas ya que se encuentra enfocado en toda la arquitectura del software (Pressman R., 2010).

Por la forma en que se encuentra concebido el sistema, fueron aplicados dos estilos arquitectónicos: la arquitectura de llamada y retorno y la arquitectura orientada a objetos.

Arquitectura de llamada y retorno:

Este estilo fue aplicado por la necesidad existente de separar las funciones de la aplicación, principalmente las que se utilizan entre las interfaces y el modelo de datos, para lograr optimizar las peticiones que se realicen por parte de los usuarios. Además, permite que una interfaz de usuario pueda mostrar múltiples vistas de los mismos datos simultáneamente.

Arquitectura orientada a objetos:

Los componentes del estilo se basan en principios Orientados a Objetos: encapsulamiento, herencia y polimorfismo. Son asimismo las unidades de modelado, diseño e implementación y los objetos y sus interacciones, el centro de las responsabilidades en el diseño de la arquitectura y en la estructura de la aplicación (Reynoso y Kicillof, 2004).

En sentido general se aplica esta arquitectura para encapsular aquellos atributos que pertenecen a un concepto o entidad específico, haciendo posible su reutilización a través de llamadas a métodos donde intervienen dentro de la lógica del negocio.

2.5.2. Patrón Arquitectónico Modelo-Vista-Controlador

El *framework* de desarrollo utilizado Symfony en su versión 2.7.1 propone la utilización del patrón arquitectónico Modelo Vista Controlador (MVC). En las ilustraciones 15 y 16 se muestra la descripción

de forma general de este patrón arquitectónico, así como su aplicación en el *framework*.

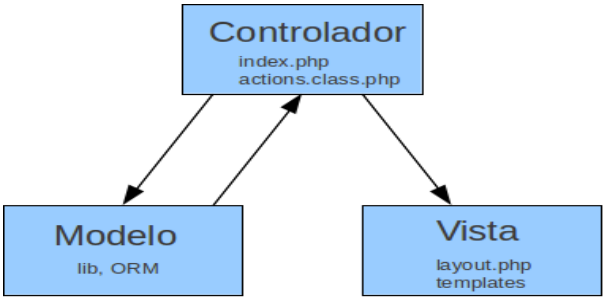


Ilustración 13. Forma general del Patrón Arquitectónico Modelo-Vista-Controlador

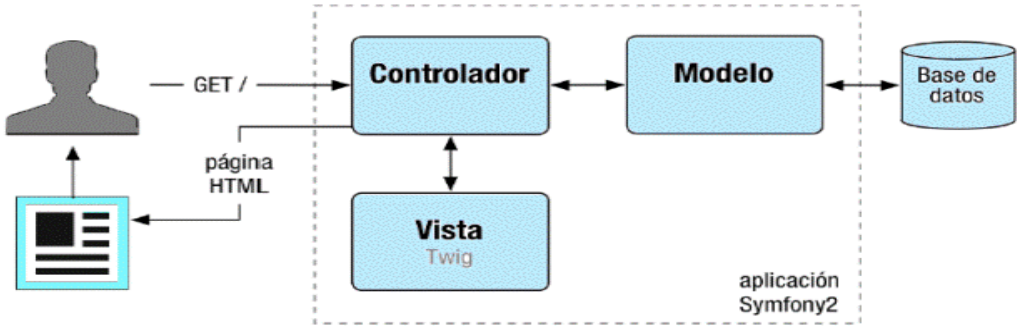


Ilustración 14. Patrón Modelo-Vista-Controlador según el sitio web oficial de Symfony

Una de las razones por las que se utiliza el patrón Modelo – Vista – Controlador (MVC) es por el nivel de organización que provee en las aplicaciones informáticas que, conjuntamente con la estructura interna que plantea el *framework* Symfony2, hace posible que la envergadura con que se desarrolla un sistema informático fluya de forma organizada. Este patrón separa la arquitectura de una aplicación en tres capas: modelo, vista y controlador. Cada una de estas capas tiene su funcionamiento específico, y una depende de otra para cumplir con sus responsabilidades dentro de la arquitectura.

El Modelo representa la información con la que trabaja la aplicación web, representando un intermediario entre la base de datos y el resto del sistema. Es el responsable del acceso directo a los datos, a través de funciones y restricciones que especifica la lógica del negocio. Esta capa hace uso del ORM Doctrine dentro de la arquitectura que plantea el *framework* Symfony2, el cual define funcionalidades y estructura para realizar las consultas y extracción de datos.

La Vista es la representación visual de los datos que provee el sistema, permitiendo que el usuario interactúe con el mismo de forma dinámica. Symfony2 plantea una estructura denominada Tres Capas, para el sistema de interfaces de usuario, utilizando Twig, el cual se presenta como un lenguaje de plantillas moderno, seguro, rápido y con el que se puede crear plantillas concisas y muy fáciles de mantener (Eguiluz, 2012).

El Controlador es quien ejecuta las acciones dentro de la lógica del sistema. Funge como un intermediario entre las peticiones del usuario y lo que el sistema debe mostrar. Esta capa basa su funcionamiento en las solicitudes del usuario, obteniendo datos del modelo y visualizándolos nuevamente al usuario en una plantilla perteneciente a la vista.

2.5.3. Patrones de diseño

Los patrones de diseño representan soluciones a problemas que surgen cuando se desarrolla un software en un contexto particular. “Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular” (Pressman R., 2010). Para realizar el diseño de la aplicación se han utilizado algunos de los patrones existentes, los cuales se mencionan a continuación:

Patrones GRASP

Los Patrones Generales de Software para Asignar Responsabilidades (GRASP, por sus siglas en inglés), son parejas de problema y solución, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Los patrones GRASP utilizados en la investigación son:

Experto: permite asignar una responsabilidad al experto en información, a la clase que cuenta con la información necesaria para cumplir la responsabilidad (Larman, 1999). Symfony2 incluye la librería ORM Doctrine como interfaz de comunicación con las clases del modelo, lo que permite encapsular toda la lógica de los datos y generar clases para manipular la información de las entidades de la base de datos. Las clases generadas contienen toda la información necesaria de la entidad que representan y sus funcionalidades, además estas generalmente se encuentran bajo el nombre entity_nameRepository. Cada entidad tiene asociado un repositorio, en el módulo se evidencia, en la entidad ConfigFile con su respectivo ConfigFileRepository.

Creador: Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento (Larman, 2004). Este patrón se evidencia en el constructor del *EasyAdminBundle* utilizado en el desarrollo de la aplicación, específicamente en la creación de los objetos $entity,$fields y $newform.

Bajo Acoplamiento: Este patrón plantea que la dependencia entre las clases que conforman la arquitectura del sistema debe ser mínima, debido a que, a la hora de realizar modificaciones en una clase, no afecte la estructura de las restantes. En la aplicación se concibió mantener separadas las clases pertenecientes al modelo de datos, las vistas de usuario y las clases controladoras.

Alta cohesión: las responsabilidades que almacena una clase deben ser pequeñas y enfocadas. Los métodos y atributos deben ser de ser sencillos para implementar dichas responsabilidades (Pressman R., 2010). En el sistema se aplica en la mayoría de las classes, de las que se utilizan en cada un atributo simple, ejemplo de ello son las clases SymTrigger, SymRouter y SymTriggerRouter.

Controlador: Sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es el controlador quien recibe los datos del usuario y quien los envía a las distintas clases según el método llamado. El *framework* Symfony2 incluye un *bundle*, denominado *FrameworkBundle*, el cual presenta una serie de clases controladoras, encargadas de las principales operaciones dentro del funcionamiento de Symfony2. Además, se crean aquellas clases controladoras que darán solución a los principales requerimientos a partir del *EasyAdminBundle*, facilitando la centralización de las actividades.

Patrones Gang of Four(GoF)

Patrón singleton (The Singleton Pattern): asegura que una clase tiene solo una instancia, y proporciona un punto de acceso global a la misma (Freeman and Freeman, 2004). Un ejemplo de esto en la aplicación se evidencia en las clases ConfigFile y ConfigServerPorts.

Patrón observador (The Observer Pattern): define de una a muchas dependencias entre objetos de forma que cuando un objeto cambia de estado, todos sus dependientes son notificados y actualizadas de forma automática (Freeman and Freeman, 2004). En la aplicación este patrón se evidencia en las clases SymNode,SymNodeGroup, SymNodeGroupLink SymTrigger, SymRouter y SymTriggerRouter.

Patrón inyección de dependencia (Dependency Injection Pattern): es donde los componentes dan

sus dependencias a través de sus constructores, métodos, o directamente en los campos. Además, la mayoría de los frameworks PHP modernos utilizan inyección de dependencias para proporcionar un conjunto de componentes desacoplados pero cohesionados (Potencier, 2009). En la aplicación está presente en cualquiera de las clases *manager*, los formularios, o los *subscriber.*

2.5.4. Patrones de diseño de Bases de Datos.

Estos patrones de diseño permiten crear una base de datos más fortalecida a partir de una guía (Blaha, 2010). Los utilizados en la solución propuesta son:

Llave subrogada: Este patrón es muy utilizado pues se decide generar una llave primara única para cada entidad en vez de usar un atributo identificador en el contexto dado. Normalmente se usa enteros en columnas identity o GUID (Global Unique Identifier) que están demostrados que no se repiten o con una probabilidad extremadamente baja. Permite que las tablas sean más fáciles de consultar por el identificador dado que se conoce el mismo tipo de todos en cada tabla. Un ejemplo de su aplicación es en la tabla SymNodeGroupLink dado que el identificador de la misma es compuesto por otros dos identificadores y se le agrega un identificador numérico para facilitar el trabajo con el mapeo relacional de objetos utilizado ORM.

Árbol fuertemente codificado: El árbol fuertemente codificado es utilizado para representar jerarquías donde es bien conocida la estructura, es importante representar la correspondencia, por ejemplo las estructuras organizacionales. Debe utilizarse sólo en los casos en que los cambios en la estructura a representar sean poco probables. El patrón admite tantos niveles como requiera la jerarquía que se vaya a representar (Silberschatz y Korth, 2002). En la aplicación esto puede aplicarse en la relación existente entre las clases SymNodeGroupLink, SymNodeGroup y SymNode.

2.5.5. Diagramas de clases del Diseño

El diagrama de clases del diseño es una descripción gráfica las especificaciones de las clases de software y de las interfaces en una aplicación (Larman, 2004). A continuación, se presenta el diagrama de clases del diseño correspondiente a RF19, RF20, RF21.

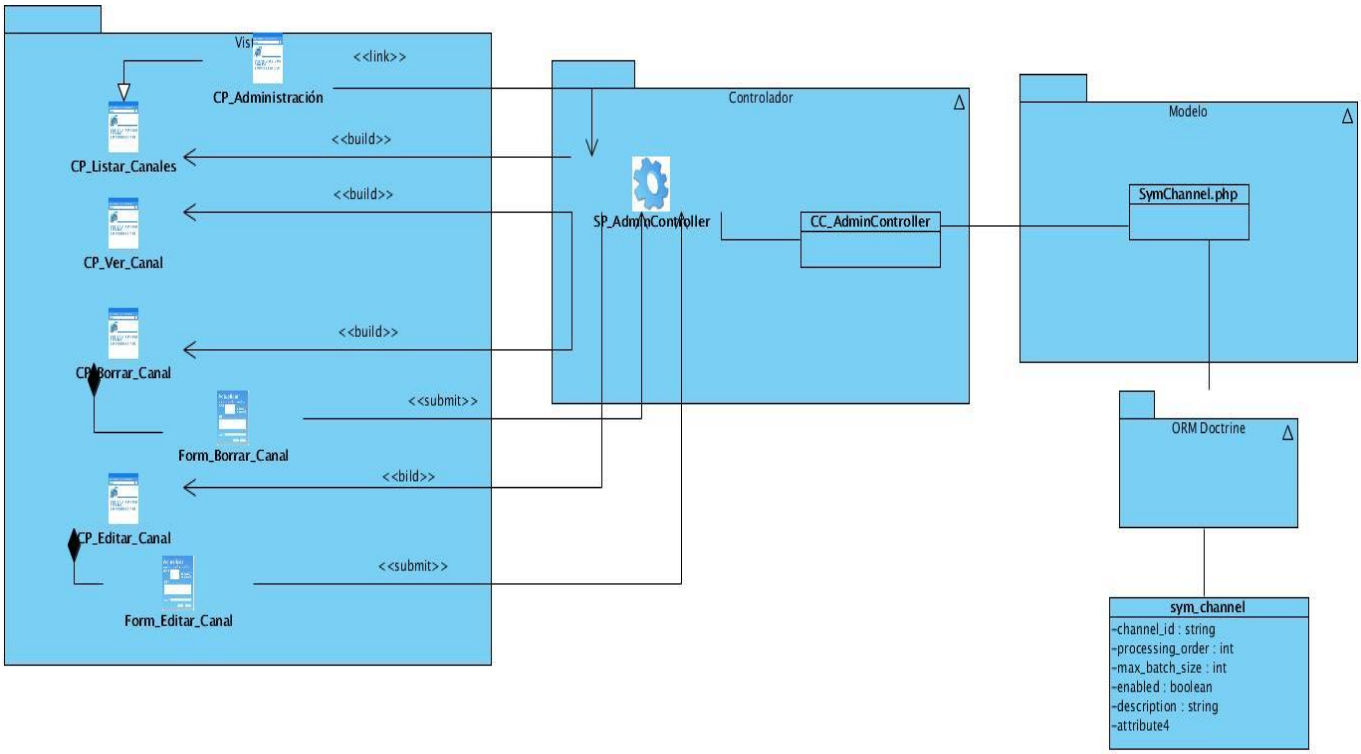


Ilustración 15. Diagrama de clases del diseño de RF19, RF20 y RF21

2.5.6. Diagramas de secuencia del Diseño

Los diagramas de secuencia ilustran las interacciones entre objetos con el transcurso del tiempo. Estos muestran los objetos participes de la interacción y la secuencia de los mensajes intercambiados (Larman, 2004). A continuación, se muestran los diagramas de secuencia correspondientes a los requisitos funcionales 19,20 y 21.

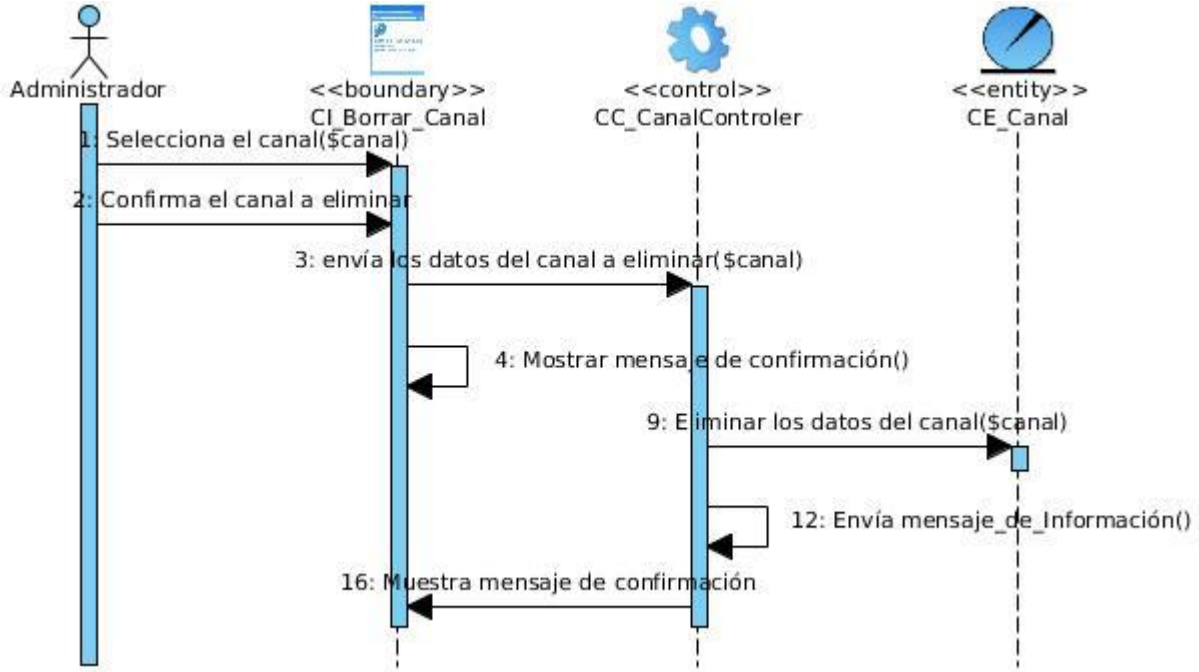


Ilustración 16. Diagrama de secuencia de RF 20 Borrar Canal

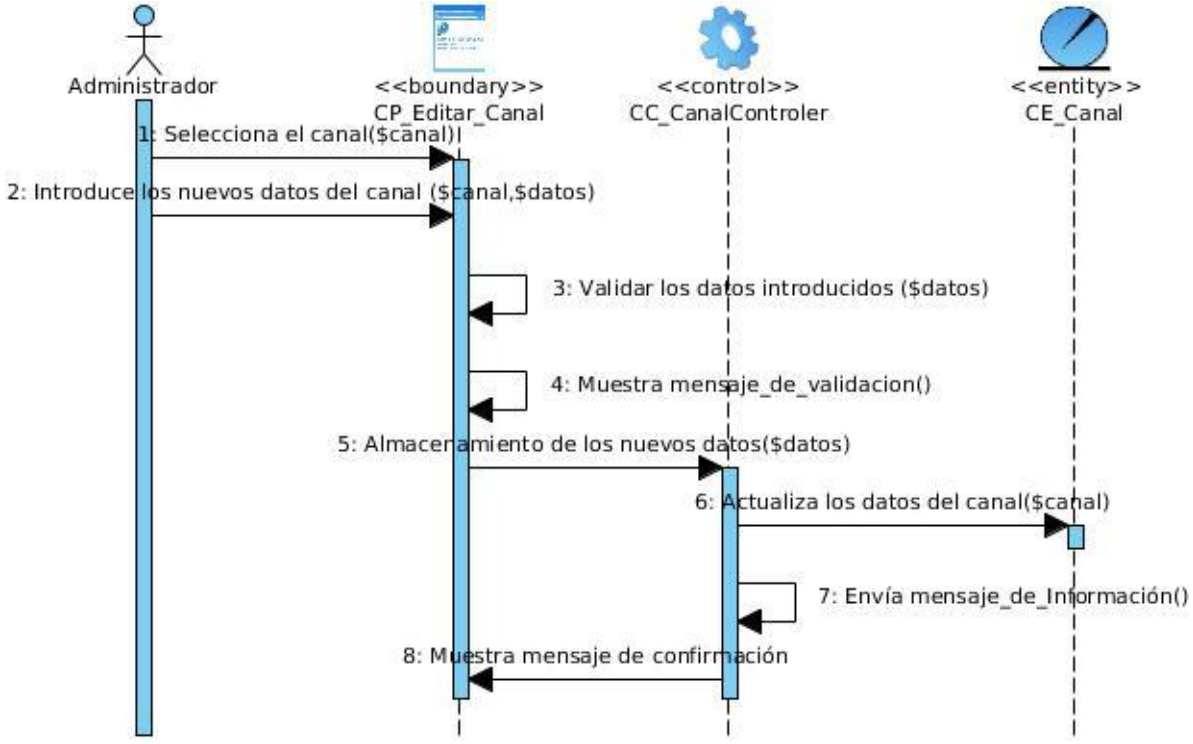


Ilustración 17. Diagrama de secuencia de RF19 Editar Canal

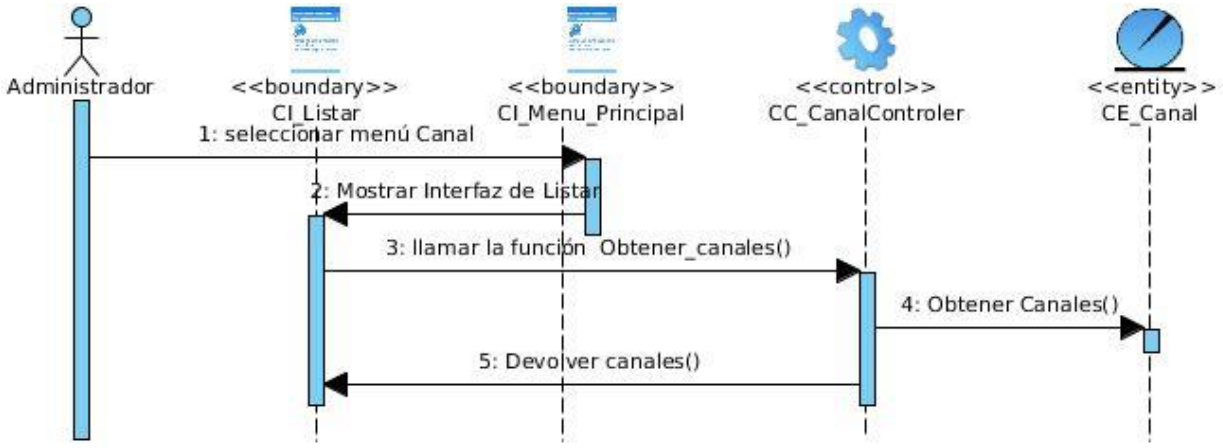


Ilustración 18. Diagrama de secuencia de RF21 Listar Canales

2.5.7. Modelo de datos

Un modelo de datos es un modelo abstracto que organiza elementos de datos y estandariza cómo se relacionan entre sí y con las propiedades del mundo real (Silberschatz y Korth, 2004). El mismo tiene el propósito de garantizar que los datos persistentes sean almacenados de manera coherente y eficaz, así como definir el comportamiento que debe ser implementado en la base de datos (Pressman R., 2010).

En el modelo de datos para la aplicación se modela un total de 43 tablas incluyendo las tablas generadas por la herramienta SymmetricDS y otras dos tablas para el almacenamiento de las configuraciones, a continuación, se muestra una selección del modelo de datos donde se recogen las tablas que inciden directamente en las funcionalidades de la aplicación de configuración y monitorización.

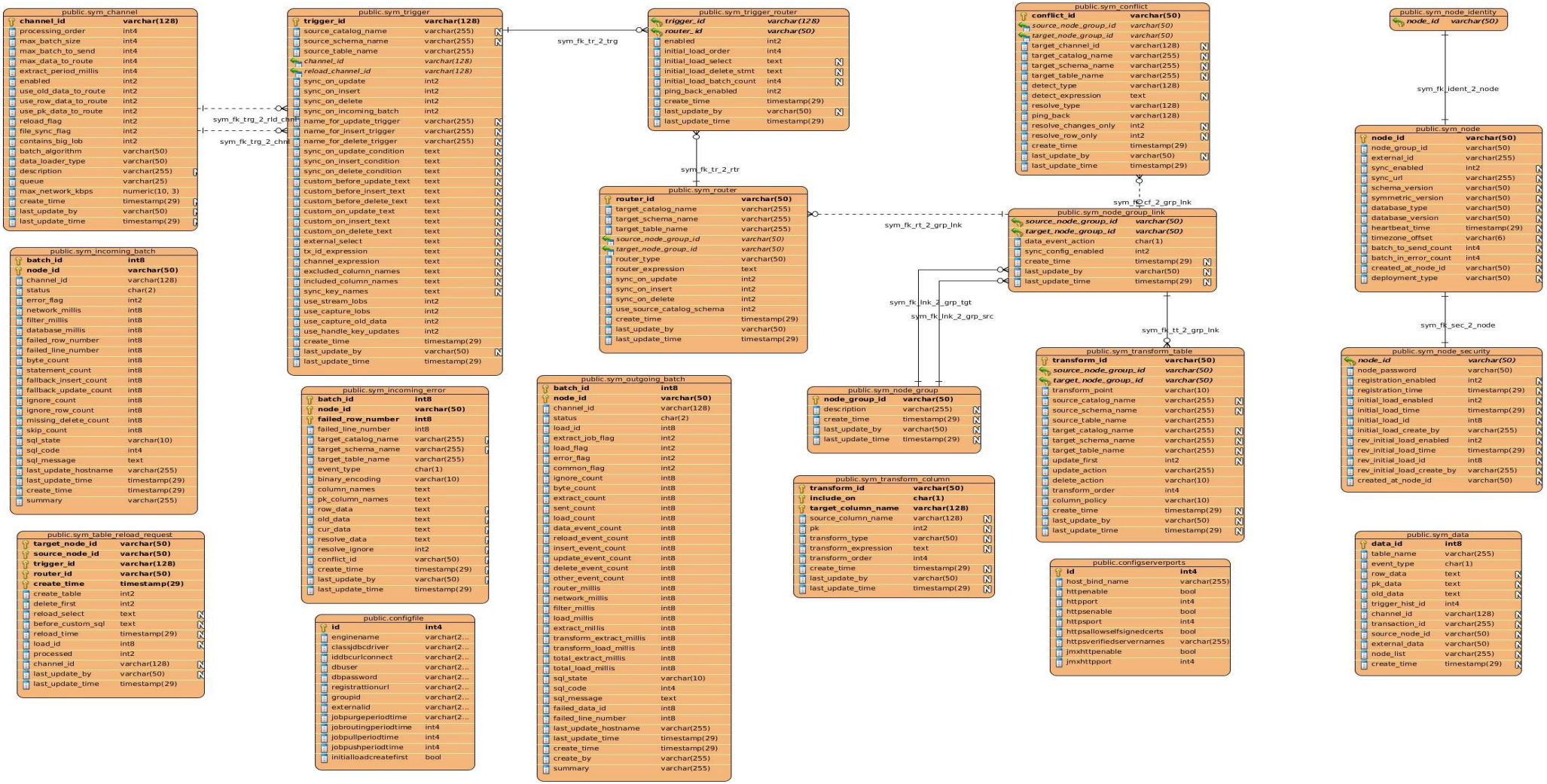


Ilustración 19. Modelo Entidad Relación de la aplicación descrita

Se muestra también la descripción de la tabla canal con sus atributos fundamentales.

Sym_Channel		
Se corresponde con la clase entidad canal descrita en el análisis. En ella se guardan los canales creados para las réplicas.		
Atributo	Tipo	Descripción
CHANNEL_ID (canal_id)	VARCHAR	Un identificador único, generalmente llamado algo significativo, como 'ventas' o 'inventario'.
PROCESSING_ORDER(orden de procesamiento)	INTEGER	Orden de secuencia para procesar datos de canal.
MAX_BATCH_SIZE (tamaño máximo del lote)	INTEGER	Número máximo de eventos de datos a procesar en un lote para este canal.
ENABLED (habilitado)	TINYIN	Indica si el canal está habilitado o no.
DESCRIPTION (descripción)	VARCHAR	Descripción del tipo de datos transportados en este canal

Tabla 3. Descripción de la tabla Canal de la Base de Datos

2.6. Consideraciones del capítulo

Durante la fase de Análisis y Diseño propuesto por la metodología de desarrollo AUP-UCI, se crearon como principal artefacto las HU descritos en el presente capítulo. Se obtuvieron los diagramas correspondientes al modelo del análisis y del diseño, se define la arquitectura MVC como arquitectura a aplicar por las facilidades que proporciona, se utilizaron varios patrones para el diseño que facilitan la reutilización y mejor organización del sistema como Experto, Creador, Controlador u Observador. Además, se diseñó el diagrama entidad-relación que define la estructura de la base de datos.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

En este capítulo se documenta el proceso de implementación de los elementos identificados durante la realización del diseño. Para ello se modela el diagrama de componentes. Además, se incluyen los resultados de las pruebas y las validaciones realizadas al módulo, permitiendo de esta manera que las funcionalidades desarrolladas cumplan con los requisitos establecidos y puedan ser integrados con los productos desarrollados bajo la Arquitectura Xalix.

3.1. Modelo de implementación

El modelo de implementación describe cómo los elementos del diseño se implementan en componentes. El propósito principal de este flujo de trabajo es desarrollar la arquitectura y el sistema como un todo. Describe también la organización de los componentes según los mecanismos de estructuración y modularización disponibles en el entorno de desarrollo, el lenguaje de programación utilizada, y la dependencia entre componentes. Como parte del modelo de implementación se obtienen los diagramas de componentes que a continuación se presentan (López Sanz, 2009).

3.1.1. Diagrama de componentes

El diagrama de componentes representa la estructura física de la implementación. Este diagrama permite visualizar la estructura de alto nivel del sistema y el comportamiento del servicio que estos componentes proporcionan y usan a través de interfaces. (Microsift, 2016) En resumen, muestra los elementos del diseño de un sistema de software y se usa para modelar la estructura del software, incluyendo las dependencias entre los componentes: de software, de código binario y los ejecutables.

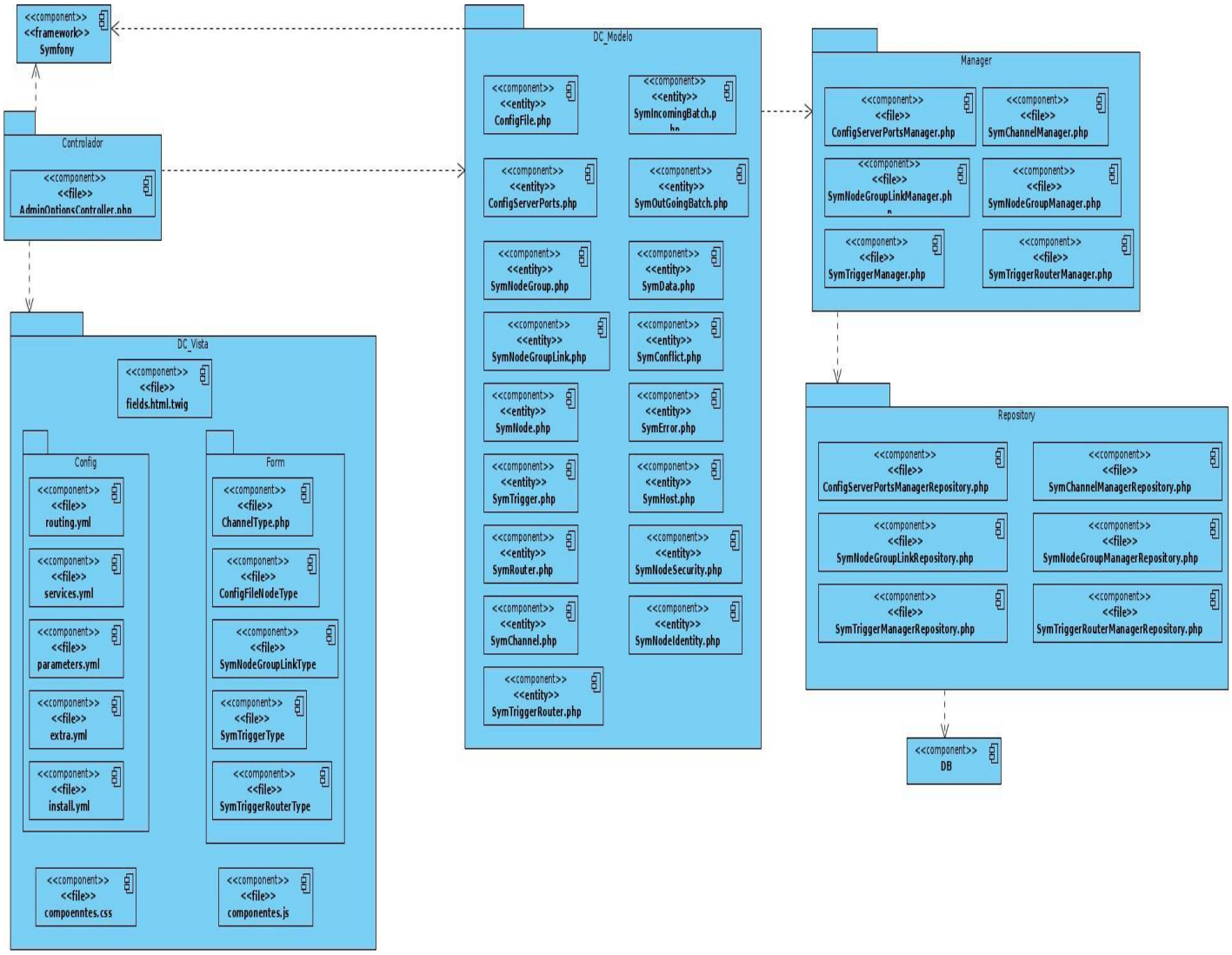


Ilustración 20. Diagrama de componentes

De manera general, el diagrama de componente antes mostrado permite modelar la vista de implementación del sistema compone la aplicación. Modela además la asignación de clases y otros elementos que facilitan la toma de decisiones respecto a las tareas de implementación.

3.1.2. Estándares de codificación

Un estándar de codificación completo, comprende todos los aspectos de la generación de código. Al inicio de un desarrollo de software, es necesario establecer un estándar de codificación, para que todos los desarrolladores trabajen sobre un código similar. Si fuera necesario incorporar código fuente previo,

o bien realizar mantenimiento a un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con el código ya existente (Pereda, 2013).

En el presente trabajo se tomó como referencia el estándar de codificación definido por el Centro FORTES para el marco de trabajo Xalix, el cual establece una serie de parámetros para el trabajo de codificación (Pereda, 2013).

Para el caso de las tablas de la base de datos relacionadas directamente con Symmetric se agrega el prefijo ***sym,*** ejemplo sym_trigger, sym_router, entre otras. Para los servicios relacionados con el modulo se establece el rpefijo symmetric y para los comandos se finaliza con el sufijo command, tal es el caso symmetricInstallCommand. Además, para los parámetros que requieren los ya mencionados comandos se establece que deben comenzar con la palabra symmetric.

```
fortes.symmetric.tablesdatabaseorder:
    class: FORTES\SymmetricConfigBundle\Library\TablesDataBaseOrderUtils
    arguments:
        - '@doctrine.orm.entity_manager'
        - '@service_container'
```

Ilustración 21. Ejemplo de un servicio

```
symmetric.root_path: %kernel.root_dir%/../src/FORTES/SymmetricConfigBundle/Library/sym
symmetric.command.start: %symmetric.root_path%bin/sym_service start
symmetric.command.stop: %symmetric.root_path%bin/sym_service stop
symmetric.command.status: %symmetric.root_path%bin/sym_service status
symmetric.command.install: 'sudo %symmetric.root_path%bin/sym_service install'
symmetric.command.uninstall: 'sudo %symmetric.root_path%bin/sym_service uninstall'
symmetric.command.register: 'sudo %symmetric.root_path%bin/symadmin  --engine %s reloa
symmetric.command.startAll: 'sudo %symmetric.root_path%bin/sym'
symmetric.command.createTables: 'sudo %symmetric.root_path%bin/symadmin --engine %s cr
symmetric.command.openRegistration: 'sudo %symmetric.root_path%bin/symadmin --engine %
symmetric.command.startNode: 'sudo %symmetric.root_path%bin/sym %s'
```

Ilustración 22. Ejemplo de parámetros de Symmetric

3.2. Pruebas de software

"Pruebas de Software: es la ejecución del código usando combinaciones de entradas, en un determinado estado, para revelar defectos. Proceso de ejecutar un programa con el fin de encontrar errores" (Martínez, 2016).

Las pruebas son un elemento importante dentro del ciclo de vida de un proyecto que proporciona una medida de la calidad del software. Estas son actividades en las cuales un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente (Alemán Llamo, 2011).

Las pruebas realizadas a todo producto de software constituyen un elemento crucial para verificar la calidad e integridad del mismo, pues permiten a los desarrolladores identificar, documentar y corregir un conjunto de no conformidades antes de que el producto sea liberado, garantizando de esta forma que el producto final funcione de acuerdo a los fines para los cuales fue diseñado e implemente de manera correcta los requerimientos identificados.

3.2.1. Niveles de Pruebas

La estrategia que se ha de seguir a la hora de evaluar dinámicamente un sistema de software debe permitir comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto. Más concretamente, los pasos a seguir son: pruebas unitarias, pruebas de integración, pruebas del sistema y pruebas de aceptación.

Pruebas Unitarias: centra el proceso de verificación en la menor unidad del diseño del software: el componente de software o módulo. Se prueba la interfaz del módulo para asegurar que la información fluye de forma adecuada hacia y desde la unidad de programa que está siendo probada (Ruiz, 2010).

Pruebas de Integración: es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es tomar los módulos probados mediante la prueba de unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño (Ruiz, 2010).

Pruebas del Sistema: está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas (Ruiz, 2010).

1.2.1.2 Tipos de pruebas

Dentro de las pruebas de sistema cabe señalar otras pruebas con vista a evaluar las funcionalidades del sistema, así como el control sobre la herramienta de réplicas SymmetricDS. A continuación, se explican dichas pruebas.

Pruebas de funcionalidad: son aquellas que se realizan con el objetivo de probar que el sistema cumpla con las funciones específicas para los cuales ha sido creado.

Pruebas de instalación: se realizan para comprobar específicamente los requisitos referidos a la instalación de SymmetricDS.

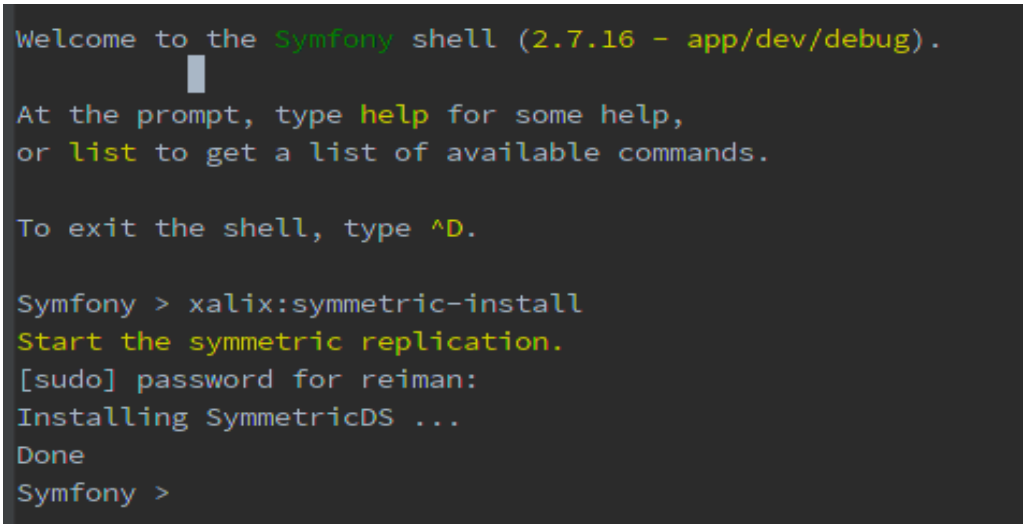


Ilustración 23. Prueba de instalación

Pruebas de replicación: se realizan para probar el funcionamiento de la aplicación ante fenómenos como caída de la red, fallos eléctricos, o fallos en el hardware.

3.2.2. Métodos de Pruebas

Pruebas de caja blanca o estructural: se basan en un minucioso análisis de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa. Su uso posibilita la obtención de casos de prueba que garantizan, al menos una vez, que sean ejecutados todos los caminos independientes de cada módulo. Posibilita ejercitar todas las decisiones lógicas en sus vertientes verdaderas y falsas. Permite, además, la ejecución de cada bucle con sus límites operacionales.

Pruebas de caja negra: *"Las pruebas de caja negra son las que se aplican a la interfaz del software. Una prueba de este tipo examina algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica interna del software"* (Martínez, 2016).

Pruebas que se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene (no se ve el código).

3.2.3 Estrategia de Casos de Prueba

"Conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo en particular" (Martínez, 2016). En los casos de prueba (CP) se incluyen la descripción de los principales escenarios, actores, posibles entradas, variables que intervienen en el proceso y flujo central donde se realiza el procedimiento.

A continuación, se muestra el caso de prueba correspondiente al RF18 que permite al administrador editar un canal.

Descripción general:
Permitir modificar datos de un canal en la configuración de SymmetricDS
Condiciones de ejecución
Para modificar datos de un canal debe cumplirse que: - Estar autenticado en el sistema con el rol de administrador. - Debe existir en el sistema al menos un canal.
SC1. Modificar datos de un Canal.

Escenario	**Descripción**	**Identificador del canal**	**Descripción**	**Tamaño máximo del lote**	**habilitado**	**Orden de procesamiento**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Opción de modificar los datos	Selecciona la opción de modificar los datos del canal	N/A	N/A	N/A	N/A	N/A	Muestra los datos del canal: seleccionada, permitiendo modificar los valores: - Identificador - Descripción - - Tamaño máximo del lote - Si está habilitado -Orden de procesamiento Permite además: - Actualizar los datos. - Cancelar la operación en	Administración/Configuración/Configuración de SymmetricDS/CanalesEditar Administración/Configuración/Configuración de SymmetricDS/Canale/Ver/Editar/

							cualquier momento.	
EC 1.2 Opción de actualizar los datos	Modifica los datos que necesite y selecciona la opción de actualizar los datos del canal.	V	N/A	N/A	N/A	N/A	*Valida los datos. Actualiza los datos del canal.* Muestra el listado de los canales y un mensaje de información.	Administración/Configuración/ Configuración de SymmetricDS/CanalesEditar/ Actualizar Administración/Configuración/ Configuración de SymmetricDS/Canale/Ver/Editar/Actualizar
EC 1.3 Opción de cancelar.	Selecciona la opción de Cancelar.	N/A	N/A			N/A	*Elimina los datos creados.* Regresa al listado de canales y muestra un mensaje de información.	Administración/Configuración/ Configuración de SymmetricDS/Canales//Editar /Cancelar Administración/Configuración/ Configuración de SymmetricDS/Canales/Ver/Editar/Cancelar
EC 1.4 Datos incompletos	Existen datos incompletos.	I	V			N/A	Muestra un mensaje de información. Muestra un indicador	Administración/Configuración/ Configuración de SymmetricDS/CanalesEditar/ Actualizar Administración/Configuración/ Configuración de SymmetricDS/Canale/Ver/Editar/Actualizar

							sobre los campos vacíos. Regresa al EC 1.1.	
		V	I			N/A		
EC 1.5 Datos incorrectos	Existen datos incorrectos.	I	V			N/A	Muestra un mensaje de información. Muestra un indicador sobre los campos incorrectos. Regresa al EC 1.1.	Administración/Configuración/ Configuración de SymmetricDS/CanalesEditar/ Actualizar Administración/Configuración/ Configuración de SymmetricDS/Canale/Ver/Editar/Actualizar
		V	I			N/A		
		V	V					

Tabla 4. Modelo de caso de prueba correspondiente al RF18.

3.3. Resultados de las pruebas de software

Con la aplicación de las pruebas de software se logra corregir los problemas presentes en las funcionalidades. A continuación, se muestran los resultados de dichas pruebas.

3.3.1 Resultados de las pruebas unitarias

Las pruebas unitarias permitieron ir comprobando el correcto funcionamiento de determinadas funciones implementadas durante el desarrollo del sistema. Para llevar a cabo estas pruebas se empleó el método de caja blanca utilizando además el *framework* de pruebas PHPUnit. Estas no se planificaron ni fueron registradas ya que fueron realizadas a medida que avanzaba el proceso de desarrollo de la aplicación.

3.3.2 Resultados de las pruebas del sistema

Con el objetivo de verificar el cumplimiento de los requisitos funcionales establecidos para la presente investigación se hace uso de las Pruebas de Caja Negra, teniendo en cuenta la técnica de partición por equivalencia. Dicha técnica permite examinar los valores válidos e inválidos de las entradas existentes en el software y descubrir de forma inmediata los errores presentes en el sistema. La partición equivalente se basa en la definición de casos de pruebas que descubran diferentes tipos de errores, reduciendo así en número de clases de prueba que hay que desarrollar.

Además, se hace uso de los casos de prueba generados durante este flujo de trabajo con el fin de detectar la mayor cantidad de no conformidades posibles en las funcionalidades del sistema realizándose cuatro iteraciones de prueba. Para el seguimiento de todo el proceso de corrección de no conformidades se realiza una tabla, la misma contará la cantidad de no conformidades determinadas en cada iteración clasificadas de acuerdo a su nivel significación alto, medio, o bajo. Se define nivel de significación alto para aquellas no conformidades que impactan directamente en el éxito de funcionalidades importantes para el trabajo del sistema; medio para aquellas no afectan la lógica del sistema, pero si la forma en que el usuario interactúa con ellas, y bajo para aquellas referidas a problemas de diseño, redacción, ortografía o traducción.

Significación de la no conformidad	Primera Iteración	Segunda Iteración	Tercera Iteración	Cuarta Iteración
Alta	16	11	4	0
Media	8	6	2	2
Baja	11	8	3	2
Total	35	25	9	4

Tabla 5. Resumen de no conformidades en las pruebas de funcionalidad.

Las no conformidades encontradas durante la cuarto iteración fueron solucionadas posteriormente. Dado que eran problemas de significación baja y media muy puntuales no fue requerido documentar una quinta iteración.

La información se muestra de forma más detallada en el siguiente gráfico.

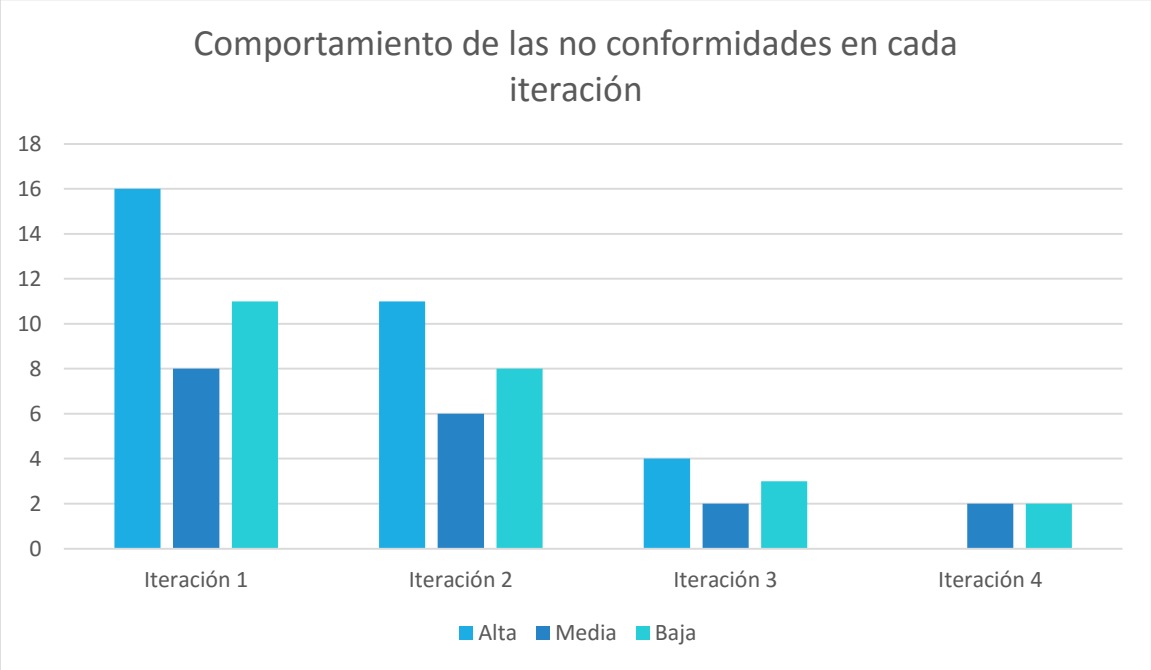


Ilustración 24. Comportamiento de las no conformidades

Para las pruebas relacionadas directamente con las réplicas fueron utilizados dos ambientes de trabajo en función de maestro y esclavo.

PC	Dirección IP	Memoria RAM
Maestro	10.35.8.246	4 GB
Esclavo	10.55.9.210	4 GB

Tabla 6. Medios utilizados para las pruebas de la replicación en red

Para estas pruebas los resultados fueron favorables ya que las tablas seleccionadas se replicaron con éxito.

3.4. Validación de la investigación mediante el método de Iadov

Para validar la presente investigación fue utilizado el método de Iadov el cual permite obtener mediante la aplicación de encuestas un índice de satisfacción que determina el valor real de la investigación para usuarios potenciales. La encuesta fue aplicada a trece usuarios obteniendo para ellos un índice de satisfacción de 0.65 lo cual ajustado a los valores significa que los usuarios tienen alta satisfacción con la investigación lo cual permite consolidar el valor práctico del módulo para la configuración y monitorización de réplicas con la herramienta SymmetricDS. La encuesta aplicada, así como la tabla utilizada para obtener los valores se encuentran en los anexos.

3.5. Consideraciones del capítulo

Durante esta etapa se generaron los artefactos correspondientes al modelo de implementación incluyendo dentro de ellos el diagrama de componentes. Fueron aplicadas pruebas unitarias y de sistema con el fin de validar el funcionamiento de cada uno de los requerimientos generando también un total de 59 modelos de casos de prueba. Además, se validó mediante el método de Iadov la investigación realizada en función de las opiniones de usuarios potenciales obteniéndose Un índice de satisfacción de 0.65 lo cual representa un alto grado de conformidad y valor práctico.

CONCLUSIONES GENERALES

Luego de realizar el análisis, el diseño, la implementación y las pruebas al módulo desarrollado para la Arquitectura Xalix que utilizan los productos de FORTES, se arribó a las siguientes conclusiones:

- La elaboración del marco teórico de la investigación permitió caracterizar los medios para establecer réplicas entre diferentes puntos geográficamente distribuidos en los productos desarrollados por FORTES bajo la Arquitectura Xalix así como hacer una selección de las tecnologías y herramientas para el desarrollo de la solución.

- En la fase de análisis y diseño quedó documentado cada uno de los artefactos definidos por la metodología AUP-UCI, los cuales facilitaron el desarrollo del módulo para la configuración y monitorización de réplicas.

- Se implementó un módulo que integra la herramienta SymmetricDS con la Arquitectura Xalix a través de una interfaz web que sustituye la creación manual de ficheros de configuración y las consultas SQL, así como un total de 11 comandos simples que siguen los estándares de la arquitectura que sustituyen los comandos originales de la herramienta.

- Las correctas ejecuciones de las pruebas aplicadas al módulo permitieron detectar y corregir las 35 no conformidades presentes a lo largo de 4 iteraciones hasta obtener resultados satisfactorios.

RECOMENDACIONES

Tras la investigación realizada para dar solución al problema planteado, se obtuvo un módulo para la configuración de réplicas de utilizando la herramienta SymmetricDS bajo el marco de trabajo Xalix. De este trabajo se hacen las siguientes recomendaciones:

- Aplicar otros tipos de prueba para hacer una mejor validación de la propuesta.
- Utilizar técnicas de Minería de Datos para hacer un estudio del comportamiento de las réplicas y sugerir mejores formas para configurar, así como prevenir fallos a causa de conflictos.

REFERENCIAS BIBLIOGRÁFICA

1. Arianne Ferrer Carcacés, A. S. (2014). Interfaz web para la administración y monitorización de la herramienta de réplica de datos SymmetricDS. La Habana.
2. Bourque, P., & Dupuis, R. (2004). Guide to the Software Engineering Body of Knowledge.
3. CollabNet. (3 de abril de 2016). ArgoUML, Web Oficial. Obtenido de Tigris.org : http://argouml.tigris.org/
4. Community, S. SymmetricDS. 2013. (2013). Recuperado el 8 de 10 de 2016, de http://www.SymmetricDS.org/
5. Costal, D. C. (2010). Introducción al diseño de base de datos.
6. Craig, L. (1999). UML y patrones.Una introducción al análisis y el diseño orientado.
7. DBreplicator. (2013). Recuperado el 9 de 12 de 2016, de http://dbreplicator.org/
8. Ealasur. (2013). Obtenido de CodeRun: http://www.ealsur.com.ar/wp/index.php/2012/04/09/coderun/
9. Eclipse. (2014). Obtenido de Eclipse: https://eclipse.org/ide/
10. Edeki, C. (2013). Agile Unifed Process. Obtenido de http://www.ijcsma.com/publications/september2013/V1I304.pdf
11. Eguíluz, J. (2009). CSS avanzado.
12. Eguíluz, J. (2013). Introducción a XHTML.
13. Eguíluz, J. P. (2013). Introducción a JavaScript.
14. Eguiluz, J. (2012). Desarrollo web ágil con Symfony2.
15. Eric Long, C. H. (2014). SymmetricDS User Guide. JumpMind, Inc.
16. Eric Long, C. M. (2007 – 2013). SymmetricDS Pro Quick Start. .
17. Fernández, Y. R., & Díaz, Y. G. (2012). Patrón Modelo-Vista-Controlador.
18. Figueroa, I. R. (2011). Herramienta administrativa para mecanismo de replicación SymmetricDS. Santa Clara: Universidad Central “Marta Abreu” de las Villas.
19. Garlan, D., & Perry, D. (s.f). Software Architecture: Practice, Potential, and Pitfalls.
20. Gauchat, J. D. (2015). El gran libro de HTML5, CSS3 y Javascript.
21. Geany. (2014). Obtenido de Geany: https://www.geany.org/

22. Grupo ISSI, I. d. (2003). Metodologías Ágiles en el Desarrollo de Software. Alicante.

23. Guerrero, C. A., & Suárez, J. M. (2013). Patrones de diseño G (The Gang of Four) en el contexto de procesos de desarrollo de aplicaciones orientadasa la web. Scielo. Obtenido de www.scielo.com

24. Jacobson, I., & Rumbaugh, J. y. (2000). El Lenguaje Unificado de Modelado.Manual de Referencia.

25. Jacobson, I., Booch, G., & Rumbaugh, J. (1999). The Unified Software Development.

26. Kaur, M., & Shaik, B. (2016). PostgreSQL Development Essentials.

27. Lawrence-Pfleeger, & Shari. (1998). Software Engineering: Theory and Practice.

28. López Sanz, M. (2009). Proceso Unificado: Implementación.

29. Manso Guerra, Y. (16 de 11 de 2016). Arquitectura Xalix. (R. Alfonso Azcuy, Entrevistador)

30. Martin, R. C. (2003). Agile Software Development, Principles, Patterns, and Practices.

31. Martínez, I. C. (2016). Pruebas de software. .

32. Microsift. (s.f.). Diagrama de Componentes UML. Recuperado el 11 de abril de 2017, de https://msdn.microsoft.com/es-es/library/dd409390.aspx.

33. Monge, R. (2005). Sistemas Distribuidos de Computación. Trabajo Investigativo "Base de Datos Distribuidas:Replicación". Valparaiso.

34. Morales, V. T. (2011). Bases de Datos Distribuidas.

35. Moreno, G. (2012). Replicación en PostgreSQL 9.0. Universidad Nacional de Salta .

36. Netbeans. (2015). Obtenido de Netbeans: https://netbeans.org/

37. Palomo, M. D., & Pérez, M. I. (s.f). Programación en PHP a travéz de ejemplos.

38. PGFoundry. (2015). Recuperado el 10 de 12 de 2016, de http://pgfoundry.org/projects/pgcluster.

39. PlanBSur Solutions. Alta Disponibilidad. (2011). Recuperado el 10 de 12 de 2016, de http://www.planbsur.cl/soluc_altadisponibilidad.htm.

40. Poseidon for UML. (abril de 2015). Obtenido de Poseidon for UML: http://www.software.com.co/p/poseidon-for-uml

41. PostgreSQL-es. (s.f.). Recuperado el 27 de 11 de 2016, de http://www.postgresql.org.es

42. Potencier, F. (s.f.). Obtenido de http://fabien.potencier.org/article/49/what-is-symfony2

43. Potencier, F. (23 de 8 de 2011). Symfony2. Recuperado el 16 de 12 de 2016, de http://symfony.com/

44. Pressman, R. (2010). Ingenería de software.Un enfoque práctico. (5 ed.).

45. Reingart, M. (2016). Sistema de replicación simple para PostgreSQL programado en Python. . Recuperado el 9 de 12 de 2016, de http://code.google.com/p/pyreplica/.

46. Replicación de SQL Server. (2014). Recuperado el 24 de 11 de 2016, de http://msdn.microsoft.com.

47. Rodríguez, T. S. (2015). Metodología de desarrollo para la actividad productiva de la UCI. Universidad de las Ciencias Informáticas, La Habana.

48. Rodríguez, T. S. (2015). Metodología de desarrollo para la actividad productiva de la UCI. Universidad de las Ciencias Informáticas. La Habana.

49. Rodríguez Sala , J. J. (2013). Introducción a la programación. Teoría y práctica. San Vicente (Alicante(: Editorial Club Universitario.

50. Ruiz, R. T. (2010). Las pruebas de software y su importancia en las organizaciones.

51. Software, L. N. (2009). Ingenería de software:metodologías y ciclos de vida.

52. Sommerville, I. (2007). Ingenería de software (8 ed.).

53. Sotolongo León, A. R., & Vazquez , Y. O. (2017). PL/pgSQL y otros lenguajes procedurales en PostgreSQL.

54. The Symfony Book. (2016).

55. Vasyliev, A. (2017). Working with PostgreSQL: configuration and scaling.

56. Visual Paradigm. (2014). Obtenido de Visual Paradigm: https://www.visual-paradigm.com/

57. web oficial de Postgre SQL. (20013). Recuperado el 15 de enero de 2017, de Postgre SQL: http://www.postgresql.org.es/

58. Wilson, S. F. (1999). Analyzing Requirements and Defining Solution Architectures. Redmond: Microsoft Press.

GLOSARIO

Bundle: es similar a un complemento en otro software, pero aún mejor. La diferencia clave es que todo es un paquete en Symfony, que incluye tanto la funcionalidad del *framework* principal como el código escrito para su aplicación. Los paquetes son ciudadanos de primera clase en Symfony.

Plugin: un complemento o plug-in es una aplicación (o programa informático) que se relaciona con otra para agregarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la interfaz de programación de aplicaciones. Complemento y plugin se diferencian en que los plug-in son desarrollados por empresas reconocidas y tienen certificado de seguridad y los complementos pueden ser desarrollados por cualquiera.

Framework: es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Bundle de tercera: son *bundles* que implementan funcionalidades muy propias dentro de Symfony asociadas a todo tipo de servicios administrativos o funcionalidades generales.

Licencia: es un contrato entre el licenciante (autor/titular de los derechos de explotación/distribuidor) y el licenciatario (usuario consumidor/usuario profesional o empresa) del programa informático, para utilizar el software cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas, es decir, es un conjunto de permisos que un desarrollador le puede otorgar a un usuario en los que tiene la posibilidad de distribuir, usar y/o modificar el producto bajo una licencia determinada. Además, se suelen definir los plazos de duración.

Sistema de nombre de dominio DNS: es un sistema de nomenclatura jerárquico descentralizado para dispositivos conectados a redes IP como Internet o una red privada. Este sistema asocia información variada con nombre de dominio asignado a cada uno de los participantes. Su función más importante es "traducir" nombres inteligibles para las personas en identificadores binarios asociados con los equipos conectados a la red, esto con el propósito de poder localizar y direccionar estos equipos mundialmente.

Trigger: son objetos que se asocian con tablas y se almacenan en la base de datos. Su nombre se deriva por el comportamiento que presentan en su funcionamiento, ya que se ejecutan cuando sucede

algún evento sobre las tablas a las que se encuentra asociado. Los eventos que hacen que se ejecute un trigger son las operaciones de inserción (INSERT), borrado (DELETE) o actualización (UPDATE), ya que modifican los datos de una tabla.

Logs: grabación secuencial en un archivo o en una base de datos de todos los acontecimientos (eventos o acciones) que afectan a un proceso particular (aplicación, actividad de una red informática, etc.). De esta forma constituye una evidencia del comportamiento del sistema. Por derivación, el proceso de generación del log se le suele llamar guardar, registrar o loguear (un neologismo del inglés logging) y al proceso o sistema que realiza la grabación en el log se le suele llamar logger o registrador.

Anexo 1: Listado de ventajas de PostgreSQL

Ahorros considerables de costos de operación: *PostgreSQL* ha sido diseñado para tener un mantenimiento y ajuste menor que los productos de proveedores comerciales, conservando todas las características, estabilidad y rendimiento.

Estabilidad y confiabilidad: No se han presentado caídas de la base de datos.

Extensible: El código fuente está disponible de forma gratuita, para que quien necesite extender o personalizar el programa pueda hacerlo sin costes.

Multiplataforma: Está disponible en casi cualquier *Unix*, con 34 plataformas en la última versión estable, además de una versión nativa de *Windows* en estado de prueba.

Diseñado para ambientes de alto volumen: Utilizando una estrategia de almacenamiento de filas llamada MVCC, consigue mejor respuesta en grandes volúmenes. Además, MVCC permite a los accesos de solo lectura continuar leyendo datos consistentes durante la actualización de registros, permitiendo copias de seguridad en caliente

Herramientas gráficas de diseño y administración de bases de datos.

Soporta los tipos de datos, cláusulas, funciones y comandos de tipo estándar *SQL92/SQL99* y extendidos propios de *PostgreSQL.*

Puede operar sobre distintas plataformas, incluyendo *Linux, Windows, Unix, Solaris y MacOS X.*

Buen sistema de seguridad mediante la gestión de usuarios, grupos de usuarios y contraseñas.

Gran capacidad de almacenamiento.

Buena escalabilidad ya que es capaz de ajustarse al número de CPU y a la cantidad de memoria disponible de forma óptima, soportando una mayor cantidad de peticiones simultáneas a la base de datos de forma correcta.

Anexo 2: Historias de usuario

RF1: Iniciar servicio Symmetric.

Número: 1	**Nombre del requisito:** Iniciar servicio Symmetric
: **Programador:** Reiman Alfonso Azcuy	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: ninguno	**Tiempo Real:** 2 días

Descripción:

Permite al administrador iniciar el servicio Symmetric.

1- Objetivo:

Permite al administrador iniciar el servicio Symmetric

2- Acciones para lograr el objetivo (precondiciones y datos):

Para incluir una categoría hay que:

- Estar autenticado en el sistema con el rol administrador.

-Debe estar instalada la herramienta SymmetricDS.

3- Comportamientos válidos y no válidos (flujo central y alternos):

El servicio no debe estar activo cuando se ejecuta la instrucción.

4- Flujo de la acción a realizar:

- El sistema debe permitir incluir al usuario seleccionar el comando a ejecutar.

-El sistema muestra el proceso de iniciado y da una notificación de que el servicio ya está iniciado

Observaciones: no hay observaciones

Prototipo de interfaz:

Es un comando por lo que no cuenta con una interfaz de usuario

RF2: Detener servicio Symmetric.

Número: 2	**Nombre del requisito:** Detener servicio Symmetric
: **Programador:** Reiman Alfonso Azcuy	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: ninguno	**Tiempo Real:** 2 días

Descripción:

Permite al administrador detener el servicio Symmetric.

1- Objetivo:

Permite al administrador detener el servicio Symmetric

2- Acciones para lograr el objetivo (precondiciones y datos):

Para incluir una categoría debe cumplirse que:

- Estar autenticado en el sistema con el rol administrador.

-Debe estar instalada la herramienta SymmetricDS.

-El servicio debe haber sido iniciado

3- Comportamientos válidos y no válidos (flujo central y alternos):

El servicio debe estar activo cuando se ejecuta la instrucción.

4- Flujo de la acción a realizar:

- El sistema debe permitir incluir al usuario seleccionar el comando a ejecutar.

-El sistema muestra el proceso de iniciado y da una notificación de que el servicio ya está fue desinstalado

Observaciones: no hay observaciones

Prototipo de interfaz:

Es un comando por lo que no cuenta con una interfaz de usuario

RF3: Consultar estado del servicio Symmetric.

Número: 3	**Nombre del requisito:** Consultar estado del servicio Symmetric
: **Programador:** Reiman Alfonso Azcuy	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: ninguno	**Tiempo Real:** 2 días

Descripción:

Permite al administrador consultar el estado del servicio Symmetric.

1- Objetivo:

Permite al administrador consultar el estado del servicio Symmetric.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para incluir una categoría debe cumplirse que:

- Estar autenticado en el sistema con el rol administrador.

3- Comportamientos válidos y no válidos (flujo central y alternos):

-Deben cumplirse todos los requisitos no funcionales planteados en la extracción de requisitos.

4- Flujo de la acción a realizar:

- El sistema debe permitir incluir al usuario seleccionar el comando a ejecutar.

-El sistema muestra el proceso de iniciado y da una notificación de que el servicio ya está fue desinstalado

Observaciones: no hay observaciones

Prototipo de interfaz:

Es un comando por lo que no cuenta con una interfaz de usuario

RF4: Instalar SymmetricDS.

Número: 4	**Nombre del requisito:** Instalar Symmetric
:	**Iteración Asignada:** 1era

Programador: Reiman Alfonso Azcuy	
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: ninguno	**Tiempo Real:** 2 días

Descripción:

Permite al administrador Instalar Symmetric.

1- Objetivo:

Permite al administrador Instalar Symmetric.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para instalar Symmetric debe cumplirse:

- Estar autenticado en el sistema con el rol administrador.

-Deben cumplirse todos los requisitos no funcionales planteados en la extracción de requicitos.

3- Comportamientos válidos y no válidos (flujo central y alternos):

En caso de fallar la instalación muestra una notificación al usuario.

4- Flujo de la acción a realizar:

- El sistema debe permitir al usuario seleccionar el comando a ejecutar.

-El sistema muestra el proceso de iniciado y da una notificación de que el servicio ya está iniciado

Observaciones: no hay observaciones

Prototipo de interfaz:

Es un comando por lo que no cuenta con una interfaz de usuario

RF5: Desinstalar SymmetricDS.

Número: 5	**Nombre del requisito:** Desinstalar Symmetric
: **Programador:** Reiman Alfonso Azcuy	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días

Riesgo en Desarrollo: ninguno	**Tiempo Real:** 2 días

Descripción:

Permite al administrador Instalar Symmetric.

1- Objetivo:

Permite al administrador Instalar Symmetric.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para instalar Symmetric debe cumplirse:

- Estar autenticado en el sistema con el rol administrador.

-Deben cumplirse todos los requisitos no funcionales planteados en la extracción de requicitos.

-Debe estar instalado Symmetric.

3- Comportamientos válidos y no válidos (flujo central y alternos):

En caso de que la herramienta no esté instalada el sistema debe mostrar una notificación.

4- Flujo de la acción a realizar:

- El sistema debe permitir al usuario seleccionar el comando a ejecutar.

-El sistema muestra el proceso de desinstalación y da una notificación de que el servicio ya está iniciado

Observaciones: no hay observaciones

Prototipo de interfaz:

Es un comando por lo que no cuenta con una interfaz de usuario

RF6: Abrir registro para nodos desde el servidor.

Número: 6	**Nombre del requisito:** Abrir registro para nodos desde el servidor
: **Programador:** Reiman Alfonso Azcuy	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 2 días
Riesgo en Desarrollo: ninguno	**Tiempo Real:** 2 días

Descripción: Permite al administrador abrir registro en el nodo maestro para un nodo esclavo determinado a partir de su grupo y su identificador externo. **1- Objetivo:** Permite al administrador abrir registro en el nodo maestro para un nodo esclavo determinado a partir de su grupo y su identificador externo. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para instalar Symmetric debe cumplirse: - Estar autenticado en el sistema con el rol administrador. -Deben cumplirse todos los requisitos no funcionales planteados en la extracción de requicitos. -Debe estar instalado Symmetric. -Debe estar configurado Symmetric. -Se requieren los datos identificador del maestro, identificador del esclavo, identificador externo del esclavo. **3- Comportamientos válidos y no válidos (flujo central y alternos):** En caso de que los datos entrados no sean correctos el sistema debe mostrar una notificación. En caso de que el sistema no esté configurado debe mostrar una notificación. **4- Flujo de la acción a realizar:** - El sistema debe permitir al usuario seleccionar el comando a ejecutar. -El sistema muestra el proceso de desinstalación y da una notificación de que el servicio ya está iniciado
Observaciones: no hay observaciones
Prototipo de interfaz: **Es un comando por lo que no cuenta con una interfaz de usuario**

RF7: Registrar nodos en la configuración del servidor enviando carga inicial.

Número: 7	**Nombre del requisito:** Registrar nodos en la configuración del servidor enviando carga inicial

: **Programador:** Reiman Alfonso Azcuy	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 2 días
Riesgo en Desarrollo: ninguno	**Tiempo Real:** 2 días

Descripción:

Permite al administrador enviar una carga inicial de datos desde al maestro hasta el esclavo.

1- Objetivo:

Permite al administrador enviar una carga inicial de datos desde al maestro hasta el esclavo.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para instalar Symmetric debe cumplirse:

- Estar autenticado en el sistema con el rol administrador.

-Deben cumplirse todos los requisitos no funcionales planteados en la extracción de requicitos.

-Debe estar instalado Symmetric.

-Debe estar configurado Symmetric.

-Se requieren los datos id identificador del maestro, identificador del esclavo, identificador externo del esclavo.

-Se debe haber abierto registro en el servidor para el nodo al que se quiere enviar la carga inicial.

3- Comportamientos válidos y no válidos (flujo central y alternos):

En caso de que los datos entrados no sean correctos el sistema debe mostrar una notificación.

En caso de que el sistema no esté configurado debe mostrar una notificación.

4- Flujo de la acción a realizar:

- El sistema debe permitir al usuario seleccionar el comando a ejecutar.

-El sistema muestra el proceso de desinstalación y notifica sobre el proceso realizado hasta que termina

Observaciones: no hay observaciones

Prototipo de interfaz:

Es un comando por lo que no cuenta con una interfaz de usuario

RF8: Cambiar configuración de puertos de conexión para el servidor.

RF9: Crear fichero de Configuración de Symmetric.

RF10: Editar fichero de Configuración de Symmetric.

RF11: Listar ficheros de Configuración de Symmetric.

RF12: Borrar fichero de Configuración de Symmetric.

RF13: Ver fichero de Configuración de Symmetric.

RF14: Generar estructura de tablas de Symmetric.

RF15: Iniciar Symmetric en el Nodo.

Número: 15	**Nombre del requisito:** Iniciar Symmetric en el Nodo.
: **Programador:** Reiman Alfonso Azcuy	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 2 días
Riesgo en Desarrollo: ninguno	**Tiempo Real:** 2 días

Descripción:

Permite al administrador iniciar Symmeric en un nodo específico.

1- Objetivo:

Permite al administrador iniciar Symmeric en un nodo específico

2- Acciones para lograr el objetivo (precondiciones y datos):

Para instalar Symmetric debe cumplirse:

- Estar autenticado en el sistema con el rol administrador.

-Deben cumplirse todos los requisitos no funcionales planteados en la extracción de requicitos.

-Debe estar instalado Symmetric.

-Debe estar configurado Symmetric.

-Se requieren los datos id identificador del nodo y el puerto.

3- Comportamientos válidos y no válidos (flujo central y alternos):

En caso de que los datos entrados no sean correctos el sistema debe mostrar una notificación.

En caso de que el sistema no esté configurado debe mostrar una notificación.

4- Flujo de la acción a realizar:

- El sistema debe permitir al usuario seleccionar el comando a ejecutar.

-El sistema muestra el proceso de desinstalación y notifica sobre el proceso realizado hasta que termina

Observaciones: no hay observaciones

Prototipo de interfaz:

Es un comando por lo que no cuenta con una interfaz de usuario

RF16: Iniciar Symmetric para todas las configuraciones disponibles.

Número: 16	**Nombre del requisito:** Iniciar Symmetric para todas las configuraciones disponibles
: **Programador:** Reiman Alfonso Azcuy	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 2 días
Riesgo en Desarrollo: ninguno	**Tiempo Real:** 2 días

Descripción:

Permite al administrador iniciar Symmeric en cargando todas las configuraciones disponibles.

1- Objetivo:

Permite al administrador iniciar Symmeric en cargando todas las configuraciones disponibles para cada nodo registrado en la Base de Datos.

2- Acciones para lograr el objetivo (precondiciones y datos): Para instalar Symmetric debe cumplirse: - Estar autenticado en el sistema con el rol administrador. -Deben cumplirse todos los requisitos no funcionales planteados en la extracción de requicitos. -Debe estar instalado Symmetric. -Debe estar configurado Symmetric. -Se requieren los datos id identificador del nodo y el puerto. **3- Comportamientos válidos y no válidos (flujo central y alternos):** En caso de que los datos entrados no sean correctos el sistema debe mostrar una notificación. En caso de que el sistema no esté configurado debe mostrar una notificación. **4- Flujo de la acción a realizar:** - El sistema debe permitir al usuario seleccionar el comando a ejecutar. -El sistema muestra el proceso de desinstalación y notifica sobre el proceso realizado hasta que termina
Observaciones: no hay observaciones
Prototipo de interfaz: **Es un comando por lo que no cuenta con una interfaz de usuario**

RF17: Incluir un Canal.

Número: 17	**Nombre del requisito:** Crear canal
: **Programador:** Reiman Alfonso Azcuy	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: ninguno	**Tiempo Real:** 2 días
Descripción: Permite al administrador crear un canal.	

1- Objetivo:

Permitir incluir nuevas canales para en el sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para incluir una categoría hay que:

- Tener en cuenta los siguientes datos: identificador del canal, descripción, tamaño máximo del lote, si está habilitado y orden de procesamiento.
- Estar autenticado en el sistema con el rol administrador.

3- Comportamientos válidos y no válidos (flujo central y alternos):

El campo descripción no es obligatorios, el resto si.

Nombre: campo de texto que admite caracteres alfabéticos y tiene un máximo de hasta 100 caracteres

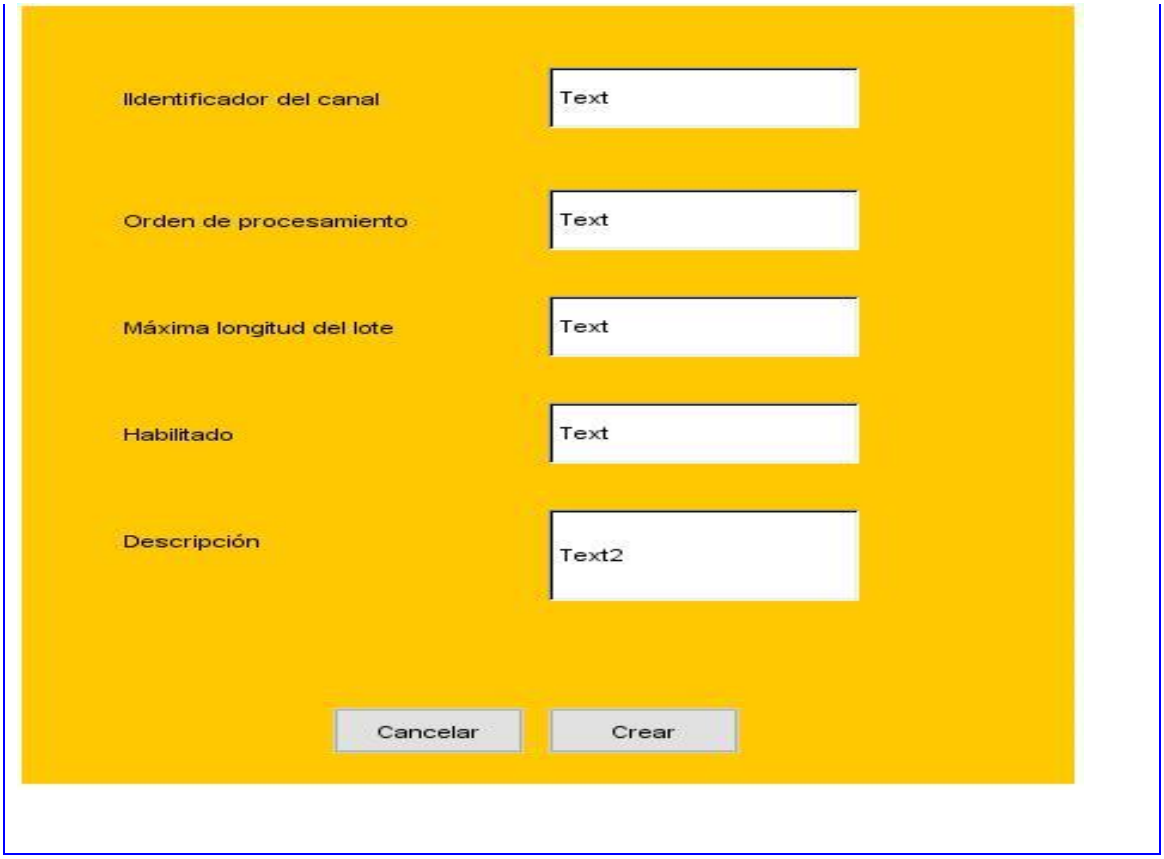
Descripción: campo de texto que permite cualquier carácter

4- Flujo de la acción a realizar:

- El sistema debe permitir incluir y/o seleccionar los datos para incluir un nuevo canal.
- Cuando el usuario incluye y/o selecciona correctamente los datos necesarios para incluir un canal y selecciona la opción Guardar, se crea un nuevo elemento y el sistema muestra un mensaje de información.
- Si los datos están incompletos o incorrectos se señalarán los campos en cuestión dando la posibilidad al usuario de realizar nuevamente la acción en cuestión.
- Si selecciona la opción Cancelar regresará a la vista previa.

Observaciones: no hay observaciones

Prototipo de interfaz:



RF18: Editar un Canal.

Número: 18	**Nombre del requisito:** Editar datos del canal
Programador: Reiman Alfonso Azcuy	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:**3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días

Descripción:

Permite al administrador editar los datos de un canal.

1- Objetivo:

Permitir editar datos de un canal.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para modificar los datos de una categoría se debe cumplir que:

- Debe existir en el sistema al menos un canal.
- Tener en cuenta los siguientes datos: identificador del canal, descripción, tamaño máximo del lote, si está habilitado y orden de procesamiento.
- Estar autenticado en el sistema con el rol Administrador.

3- Comportamientos válidos y no válidos (flujo central y alternos):

El campo identificador es obligatorio.

Nombre: campo de texto que admite caracteres alfabéticos y tiene un máximo de hasta 100 caracteres

Descripción: campo de texto que permite cualquier carácter

4- Flujo de la acción a realizar:

- El sistema debe permitir modificar un canal, esta acción puede realizarse seleccionando la opción editar en el listado de canales o desde la vista previa del canal.
- Cuando el usuario modifica de forma correcta los datos necesarios y selecciona la opción Actualizar, se muestra un mensaje de información de que el canal fue modificado de forma correcta.
- Si los datos están incompletos o incorrectos se señalarán los campos en cuestión dando la posibilidad al usuario de realizar nuevamente la acción en cuestión.
- Si selecciona la opción Cancelar regresará a la vista previa.

Observaciones: los canales son independientes del resto de las tablas sin embargo deben estar configurados dado que las acciones de los disparadores se desencadenan desde los canales.

Prototipo de interfaz:

RF19: Borrar un Canal.

RF20: Ver un Canal.

Número: 18	**Nombre del requisito:** Ver datos del canal
Programador: Reiman Alfonso Azcuy	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:**3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días

Descripción:

Permite al administrador editar los datos de un canal.

1- Objetivo:

Permitir ver datos de un canal.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para modificar los datos de una categoría se debe cumplir que:

- Debe existir en el sistema al menos un canal.
- Estar autenticado en el sistema con el rol Administrador.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Debe permitir la opción de direccionar a un Editar

4- Flujo de la acción a realizar:

- El sistema debe permitir ver un canal, esta acción puede realizarse seleccionando la opción ver en el listado de canales o desde la vista previa del canal.
- Cuando el usuario modifica de forma correcta los datos necesarios y selecciona la opción Actualizar, se muestra un mensaje de información de que el canal fue modificado de forma correcta.

- Si selecciona la opción Cancelar regresará a la vista previa.

Observaciones: los canales son independientes del resto de las tablas sin embargo deben estar configurados dado que las acciones de los disparadores se desencadenan desde los canales.

Prototipo de interfaz:

RF21: Listar Canales.

Número: 21	**Nombre del requisito:** Listar canales
Programador: Reiman Alfonso Azcuy	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:**3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días

Descripción:

Permite al administrador visualizar un listado con todos los nodos.

1- Objetivo:

Permite al administrador visualizar un listado con todos los nodos.

2- Acciones para lograr el objetivo (precondiciones y datos):

Permite al administrador visualizar un listado con todos los nodos.

3- Comportamientos válidos y no válidos (flujo central y alternos):

Se visualizan los nodos que están presentes en la base de datos.

En caso de que el usuario presione la opción eliminar el canal pasa a ser eliminado automáticamente de la base de datos.

4- Flujo de la acción a realizar:

- El sistema debe permite entrar a la opción e canales, donde se visualiza un listado con todos los canales disponibles. Se muestran además las opciones de eliminar y editar.

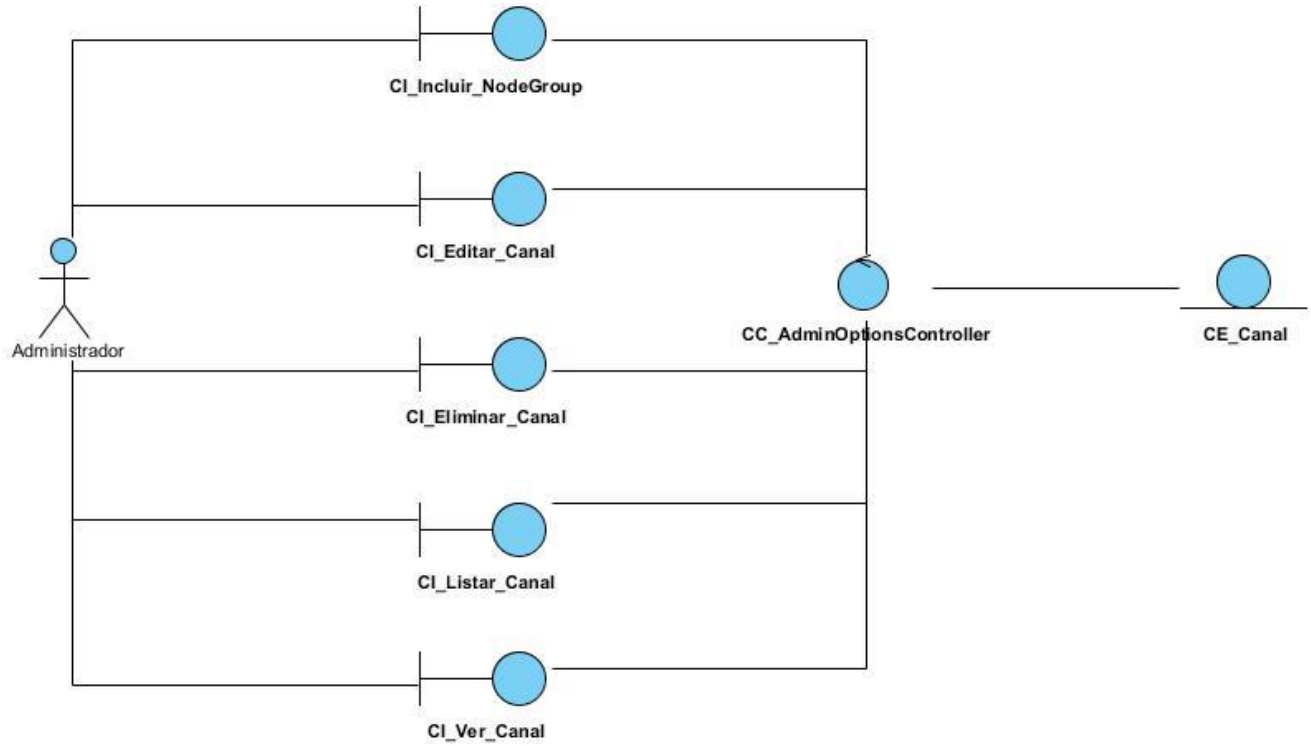
-En caso de que el usuario presione la opción editar se ejecuta el procedimiento de Editar un canal del requicito descrito anteriormente.

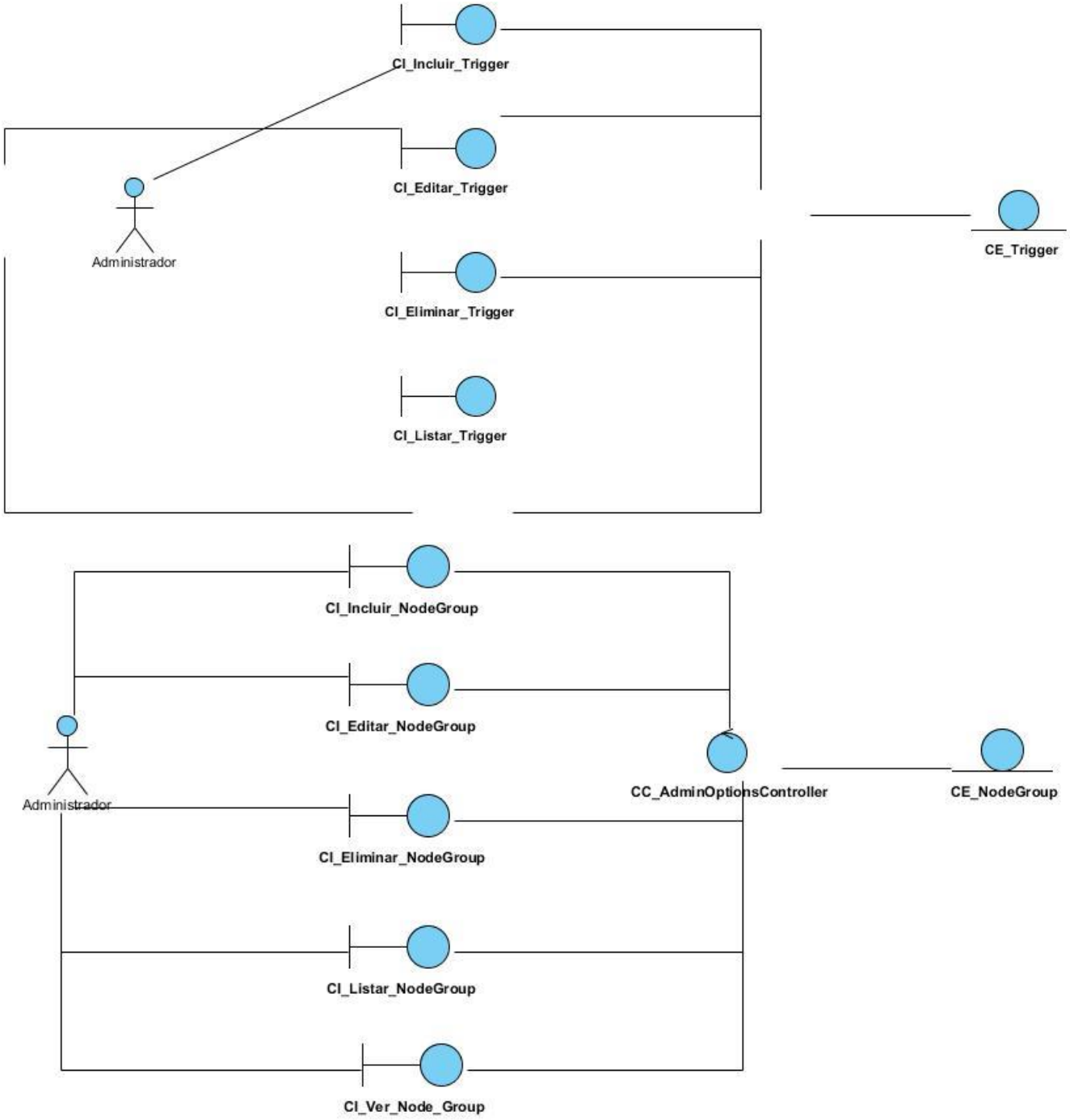
- Si selecciona la opción Cancelar regresará a la vista previa.

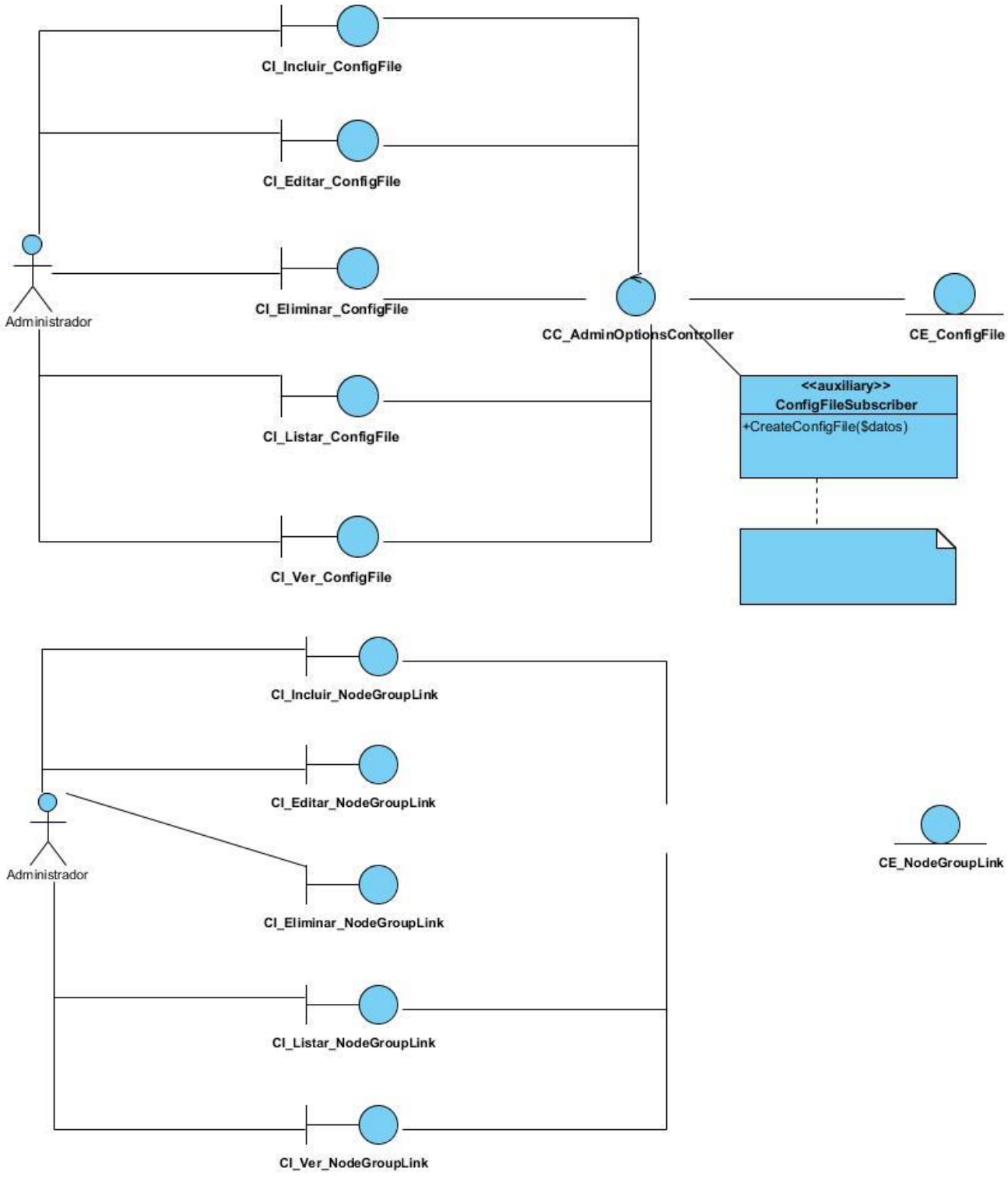
Observaciones: los canales son independientes del resto de las tablas sin embargo deben estar configurados dado que las acciones de los disparadores se desencadenan desde los canales.

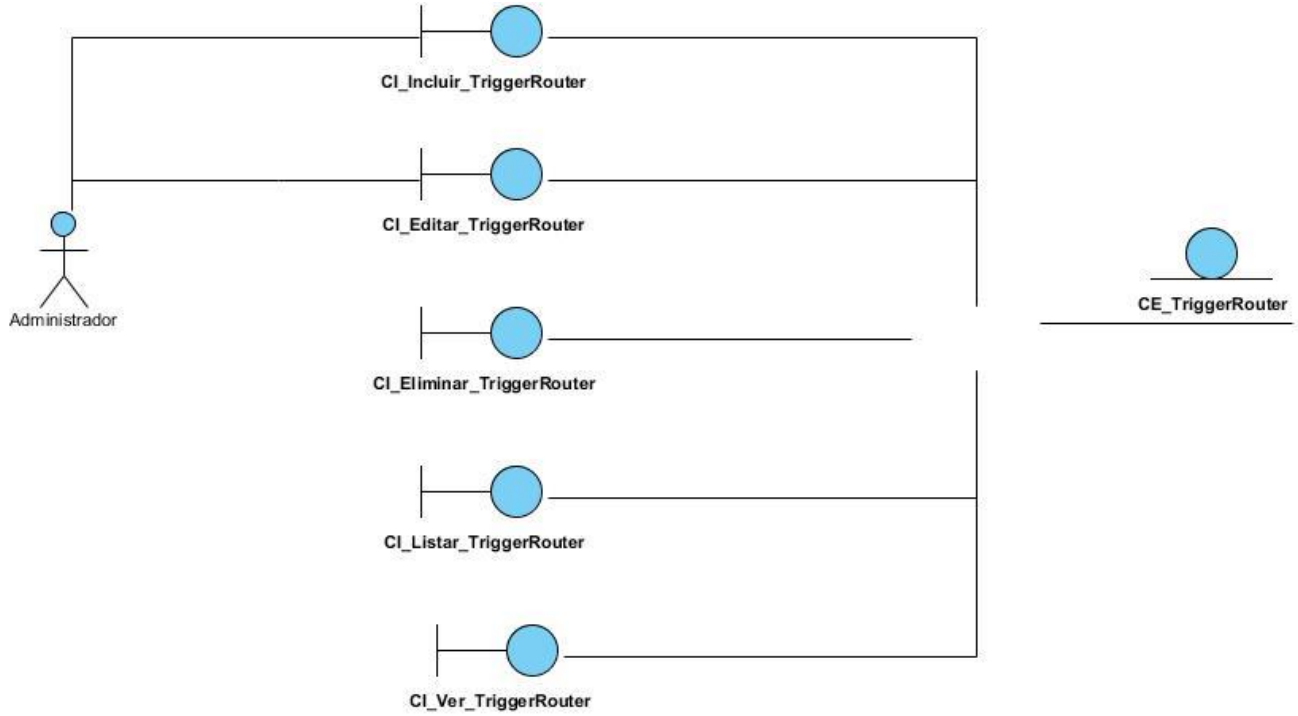
Prototipo de interfaz:

Anexo 3: Diagramas de clases de análisis

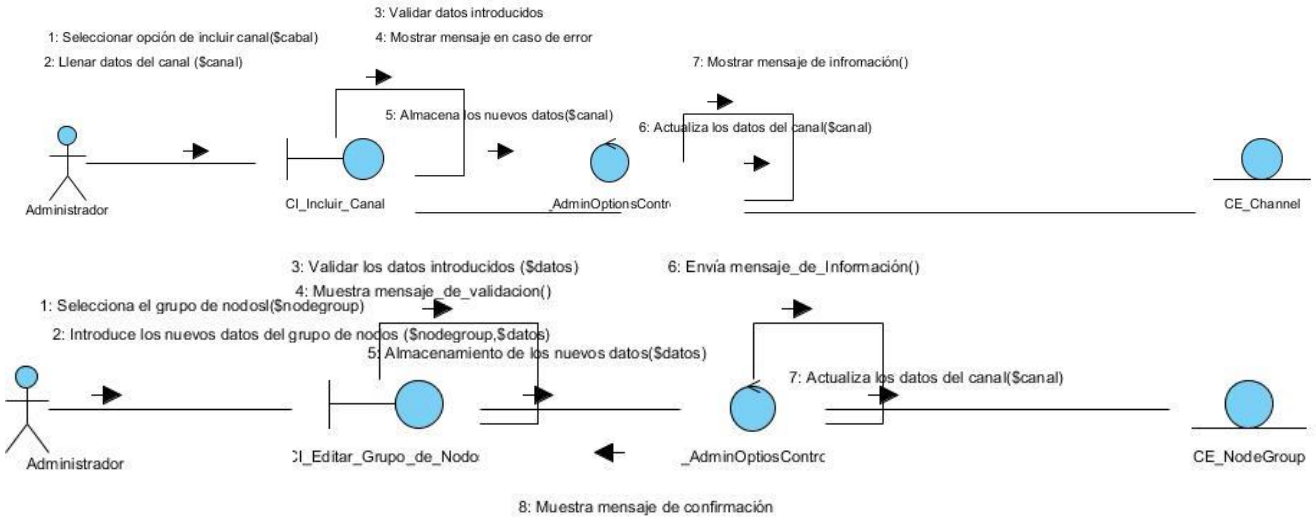


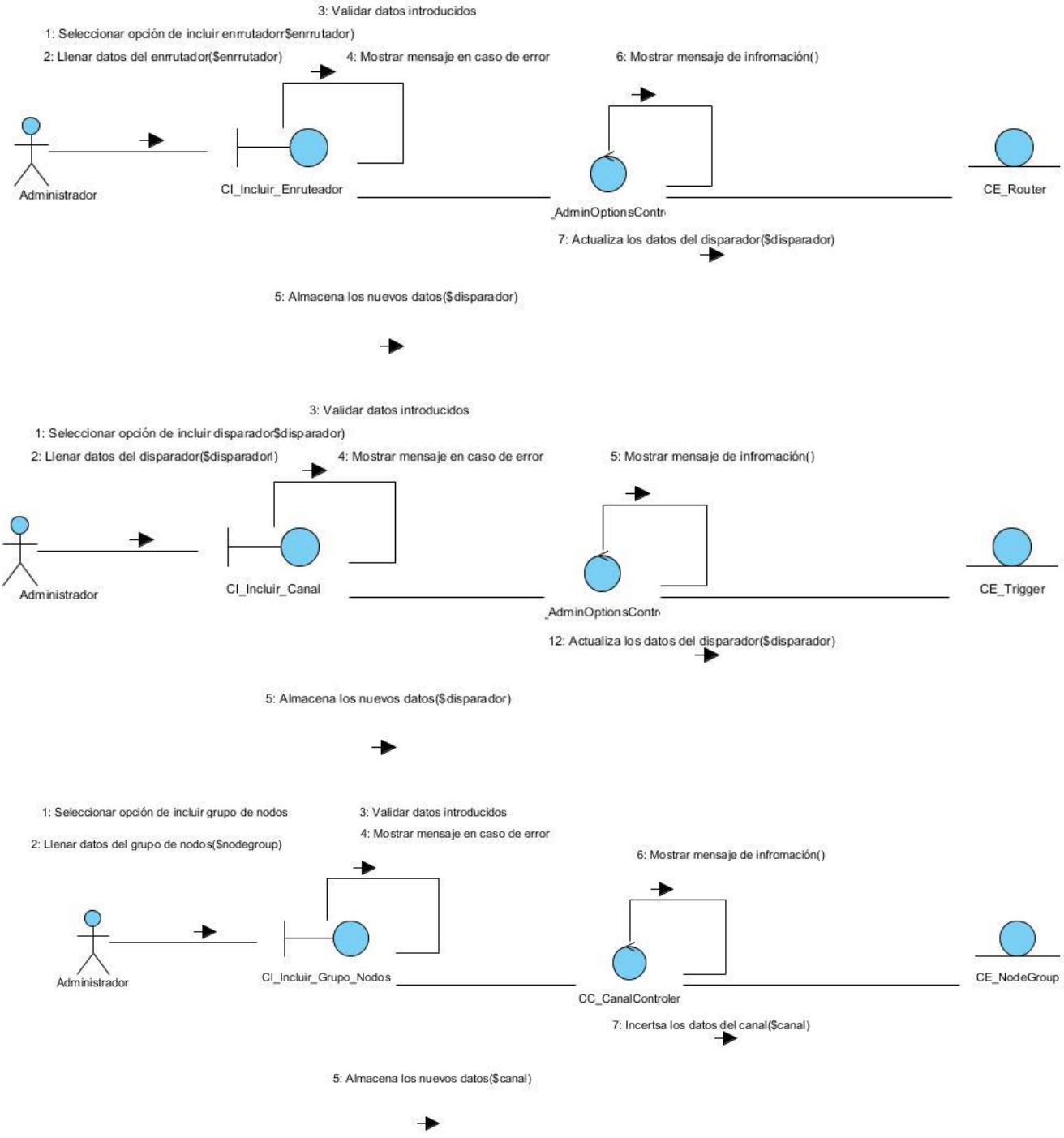

CI_Incluir_Trigger
CI_Editar_Trigger
Administrador
CE_Trigger
CI_Eliminar_Trigger
CI_Listar_Trigger
CI_Incluir_NodeGroup
CI_Editar_NodeGroup
Administrador
CC_AdminOptionsController
CE_NodeGroup
CI_Eliminar_NodeGroup
CI_Listar_NodeGroup
CI_Ver_Node_Group

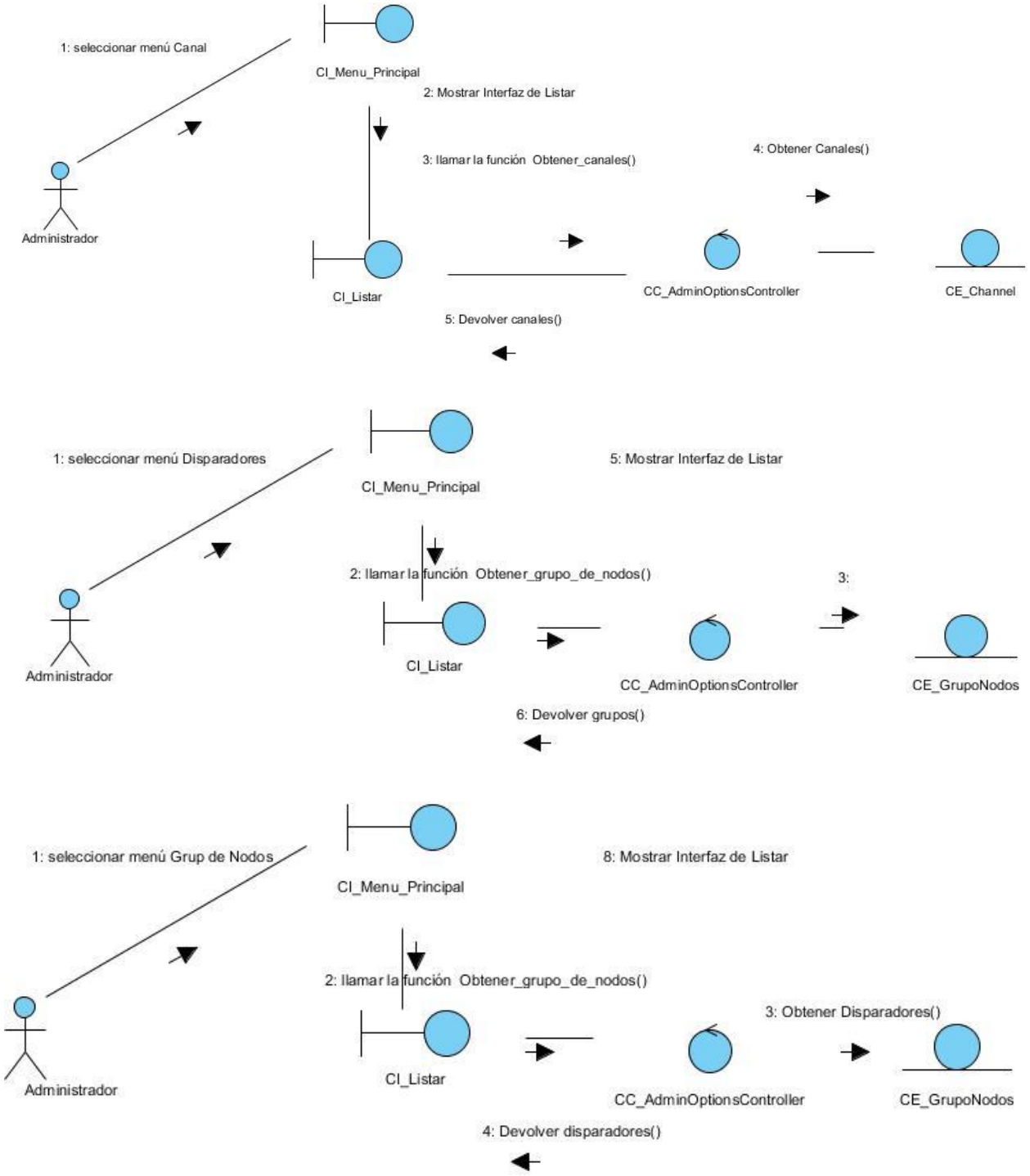

CI_Incluir_ConfigFile
CI_Editar_ConfigFile
CI_Eliminar_ConfigFile
CI_Listar_ConfigFile
CI_Ver_ConfigFile
Administrador
CC_AdminOptionsController
CE_ConfigFile
<<auxiliary>>
ConfigFileSubscriber
+CreateConfigFile($datos)
CI_Incluir_NodeGroupLink
CI_Editar_NodeGroupLink
CI_Eliminar_NodeGroupLink
CI_Listar_NodeGroupLink
CI_Ver_NodeGroupLink
Administrador
CE_NodeGroupLink

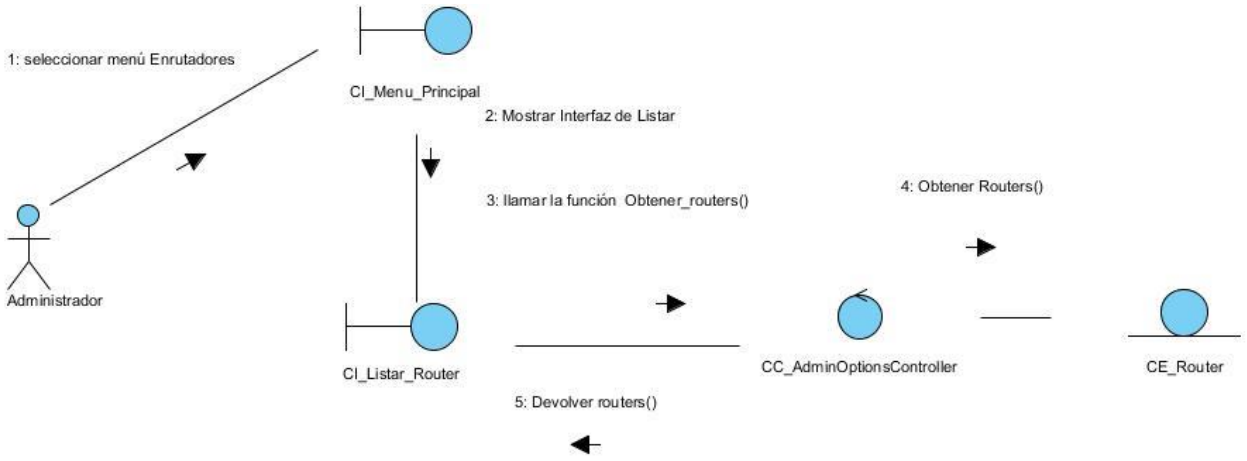


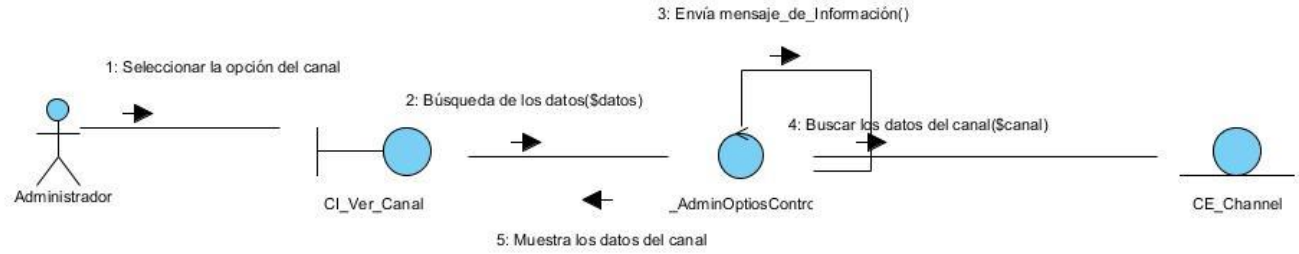
Anexo 4: Diagramas de comunicación

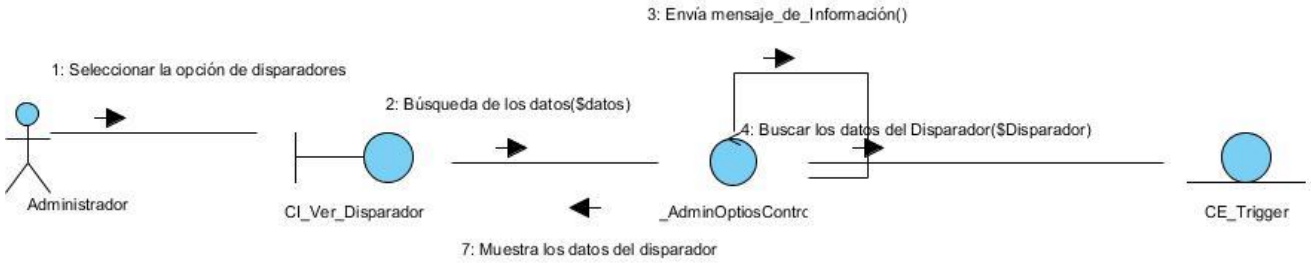


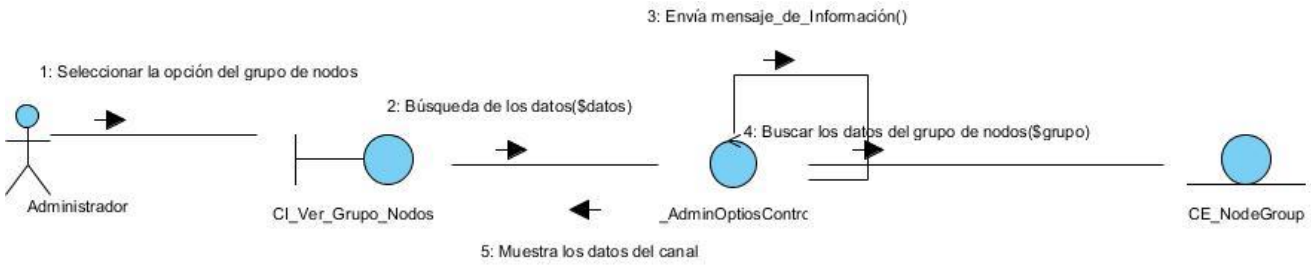

3: Validar datos introducidos
1: Seleccionar opción de incluir enrrutadorr$enrrutador)
2: Llenar datos del enrrutador($enrrutador)
4: Mostrar mensaje en caso de error
6: Mostrar mensaje de infromación()
Administrador
CI_Incluir_Enruteador
_AdminOptionsContr
CE_Router
7: Actualiza los datos del disparador($disparador)
5: Almacena los nuevos datos($disparador)
3: Validar datos introducidos
1: Seleccionar opción de incluir disparador$disparador)
2: Llenar datos del disparador($disparador)
4: Mostrar mensaje en caso de error
5: Mostrar mensaje de infromación()
Administrador
CI_Incluir_Canal
_AdminOptionsContr
CE_Trigger
12: Actualiza los datos del disparador($disparador)
5: Almacena los nuevos datos($disparador)
1: Seleccionar opción de incluir grupo de nodos
3: Validar datos introducidos
4: Mostrar mensaje en caso de error
2: Llenar datos del grupo de nodos($nodegroup)
6: Mostrar mensaje de infromación()
Administrador
CI_Incluir_Grupo_Nodos
CC_CanalControler
CE_NodeGroup
7: Incertsa los datos del canal($canal)
5: Almacena los nuevos datos($canal)

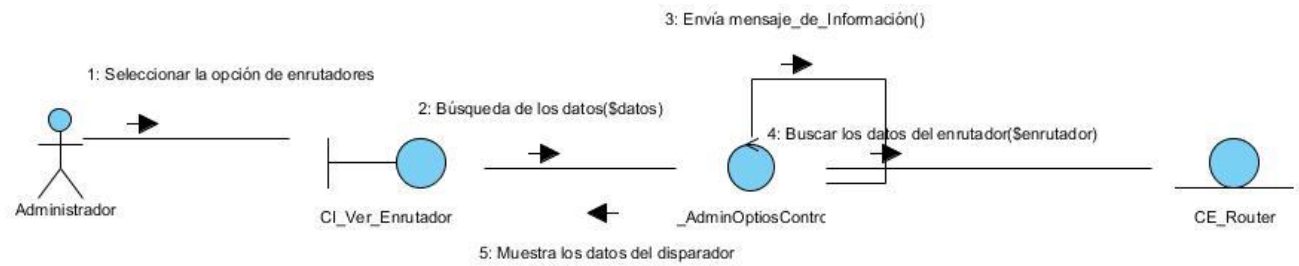

1: seleccionar menú Canal
Administrador
CI_Menu_Principal
2: Mostrar Interfaz de Listar
3: llamar la función Obtener_canales()
CI_Listar
4: Obtener Canales()
CC_AdminOptionsController
CE_Channel
5: Devolver canales()
1: seleccionar menú Disparadores
Administrador
CI_Menu_Principal
5: Mostrar Interfaz de Listar
2: llamar la función Obtener_grupo_de_nodos()
CI_Listar
3:
CC_AdminOptionsController
CE_GrupoNodos
6: Devolver grupos()
1: seleccionar menú Grup de Nodos
Administrador
CI_Menu_Principal
8: Mostrar Interfaz de Listar
2: llamar la función Obtener_grupo_de_nodos()
CI_Listar
3: Obtener Disparadores()
CC_AdminOptionsController
CE_GrupoNodos
4: Devolver disparadores()

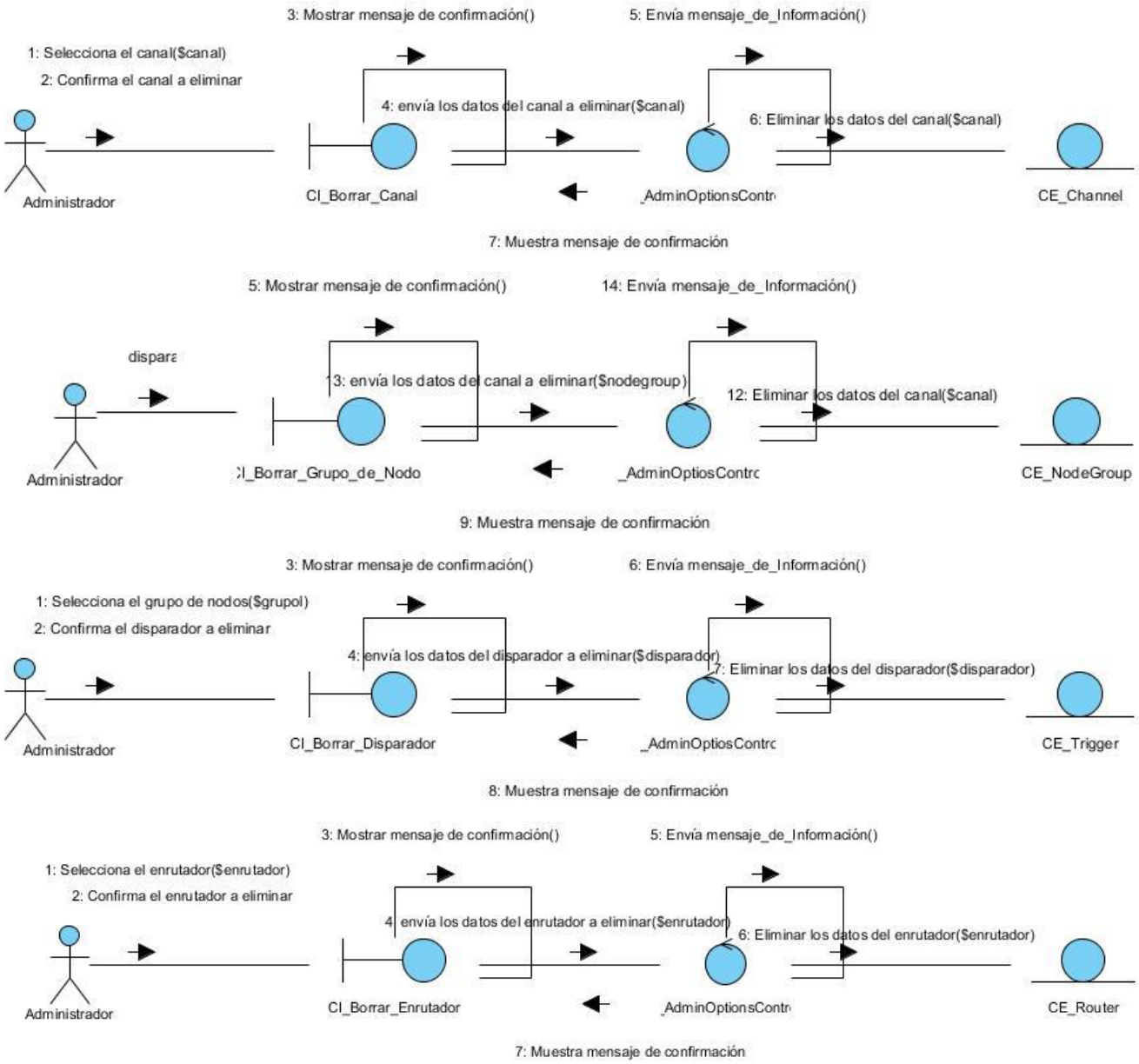

1: seleccionar menú Enrutadores
Administrador
CI_Menu_Principal
2: Mostrar Interfaz de Listar
3: llamar la función Obtener_routers()
CI_Listar_Router
4: Obtener Routers()
CC_AdminOptionsController
CE_Router
5: Devolver routers()

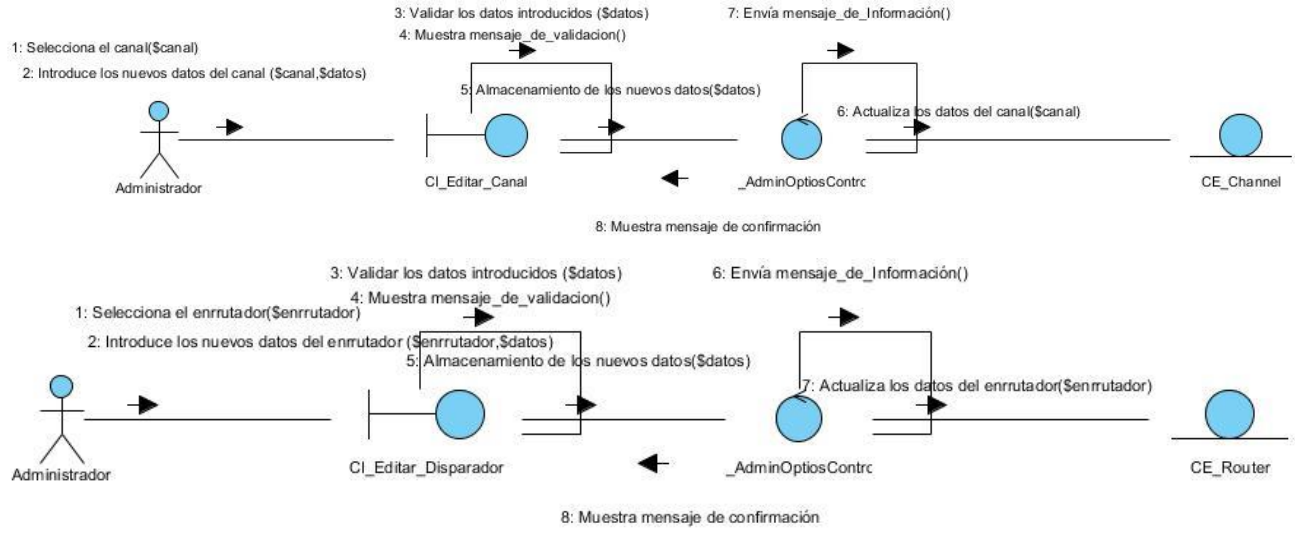

3: Envía mensaje_de_Información()
1: Seleccionar la opción del canal
Administrador
2: Búsqueda de los datos($datos)
CI_Ver_Canal
4: Buscar los datos del canal($canal)
_AdminOptiosContrc
CE_Channel
5: Muestra los datos del canal


3: Envía mensaje_de_Información()
1: Seleccionar la opción de disparadores
Administrador
2: Búsqueda de los datos($datos)
CI_Ver_Disparador
4: Buscar los datos del Disparador($Disparador)
_AdminOptiosContrc
CE_Trigger
7: Muestra los datos del disparador

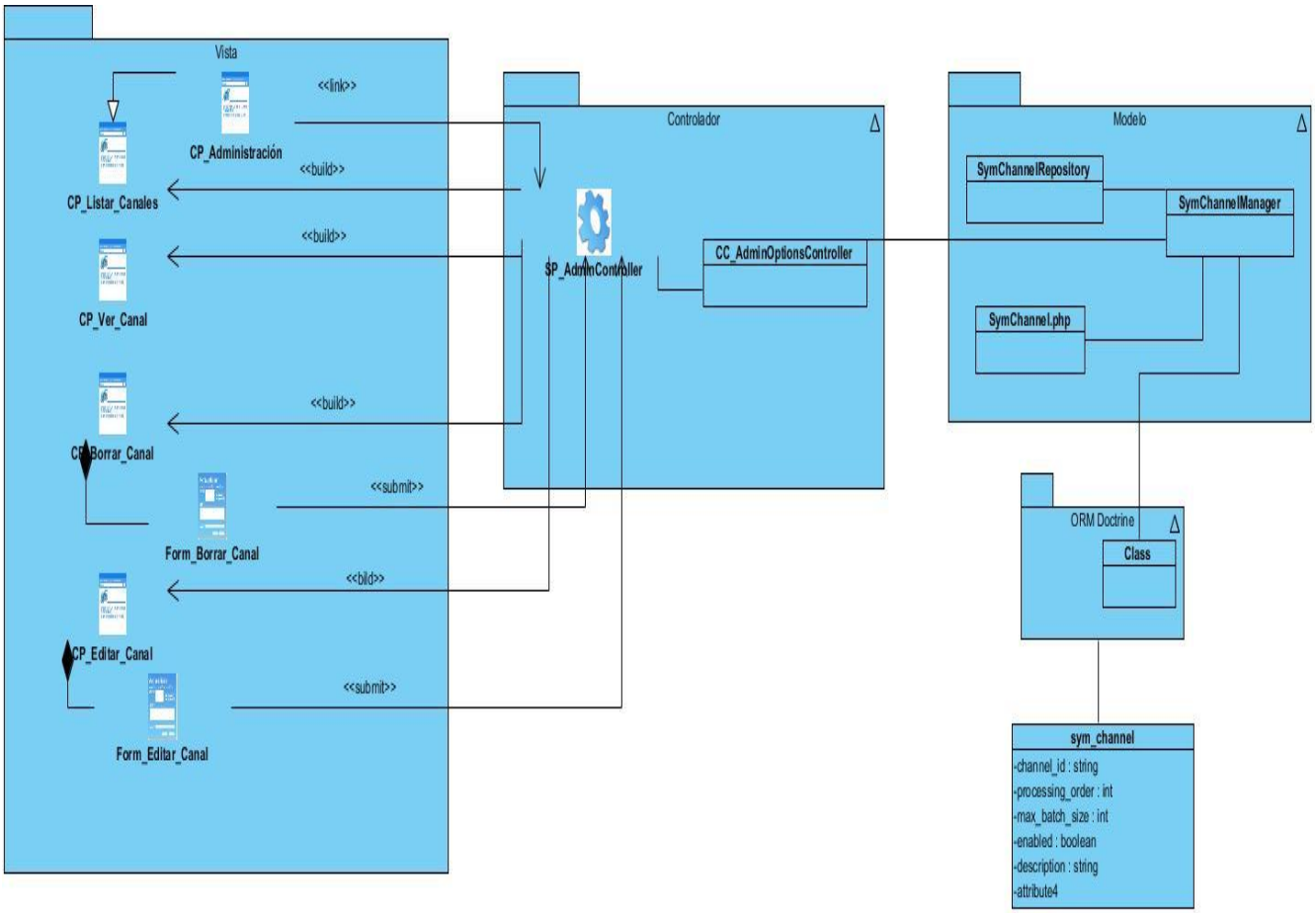

3: Envía mensaje_de_Información()
1: Seleccionar la opción del grupo de nodos
Administrador
2: Búsqueda de los datos($datos)
CI_Ver_Grupo_Nodos
4: Buscar los datos del grupo de nodos($grupo)
_AdminOptiosContrc
CE_NodeGroup
5: Muestra los datos del canal


3: Envía mensaje_de_Información()
1: Seleccionar la opción de enrutadores
Administrador
2: Búsqueda de los datos($datos)
CI_Ver_Enrutador
4: Buscar los datos del enrutador($enrutador)
_AdminOptiosContrc
CE_Router
5: Muestra los datos del disparador

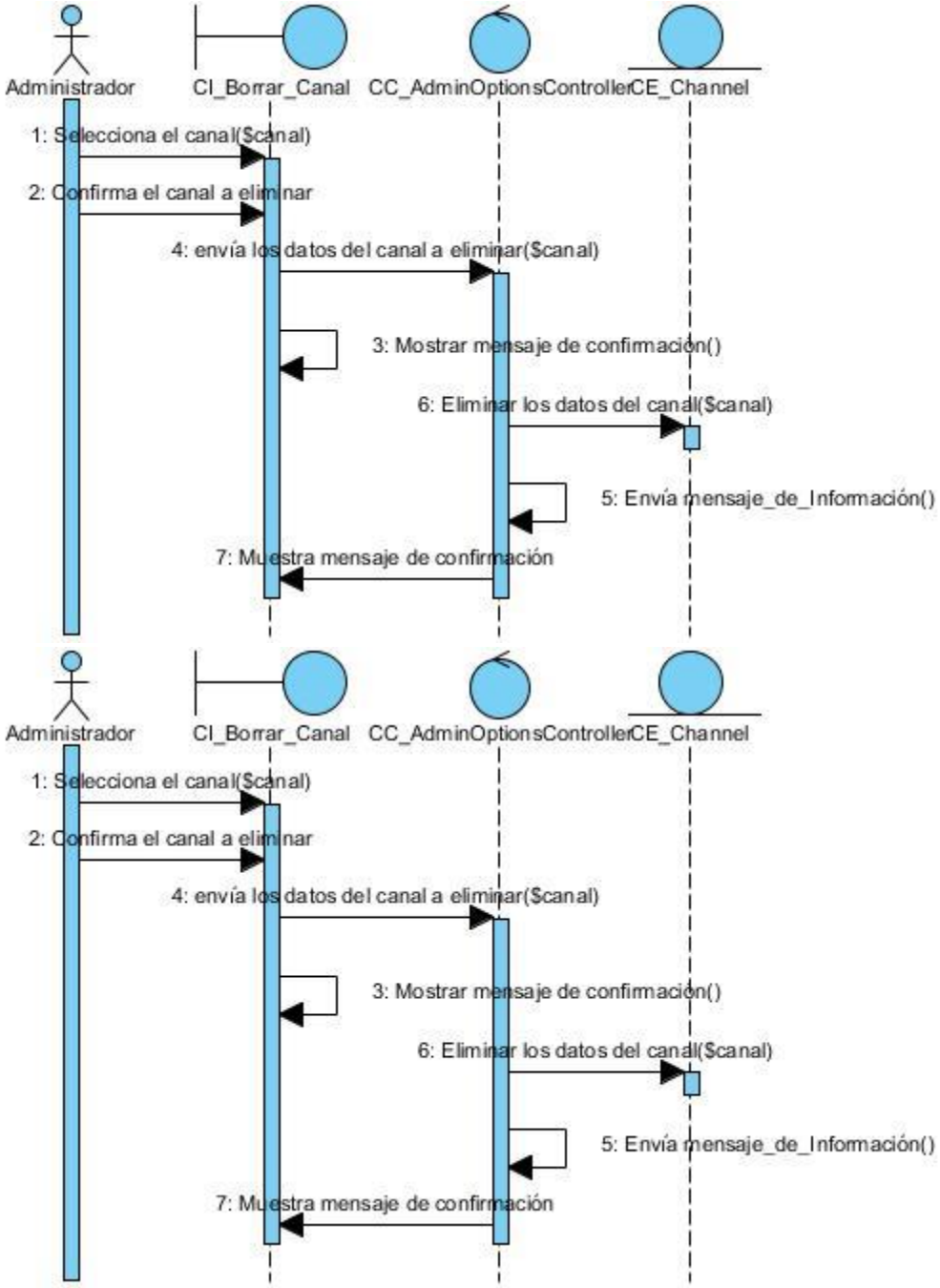

3: Mostrar mensaje de confirmación()
5: Envía mensaje_de_Información()
1: Selecciona el canal($canal)
2: Confirma el canal a eliminar
4: envía los datos del canal a eliminar($canal)
6: Eliminar los datos del canal($canal)
Administrador
Cl_Borrar_Canal
_AdminOptionsContr
CE_Channel
7: Muestra mensaje de confirmación
5: Mostrar mensaje de confirmación()
14: Envía mensaje_de_Información()
dispara
13: envía los datos del canal a eliminar($nodegroup)
12: Eliminar los datos del canal($canal)
Administrador
Cl_Borrar_Grupo_de_Nodo
_AdminOptiosContrc
CE_NodeGroup
9: Muestra mensaje de confirmación
3: Mostrar mensaje de confirmación()
6: Envía mensaje_de_Información()
1: Selecciona el grupo de nodos($grupol)
2: Confirma el disparador a eliminar
4: envía los datos del disparador a eliminar($disparador)
7: Eliminar los datos del disparador($disparador)
Administrador
Cl_Borrar_Disparador
_AdminOptiosContrc
CE_Trigger
8: Muestra mensaje de confirmación
3: Mostrar mensaje de confirmación()
5: Envía mensaje_de_Información()
1: Selecciona el enrutador($enrutador)
2: Confirma el enrutador a eliminar
4: envía los datos del enrutador a eliminar($enrutador)
6: Eliminar los datos del enrutador($enrutador)
Administrador
Cl_Borrar_Enrutador
_AdminOptionsContr
CE_Router
7: Muestra mensaje de confirmación

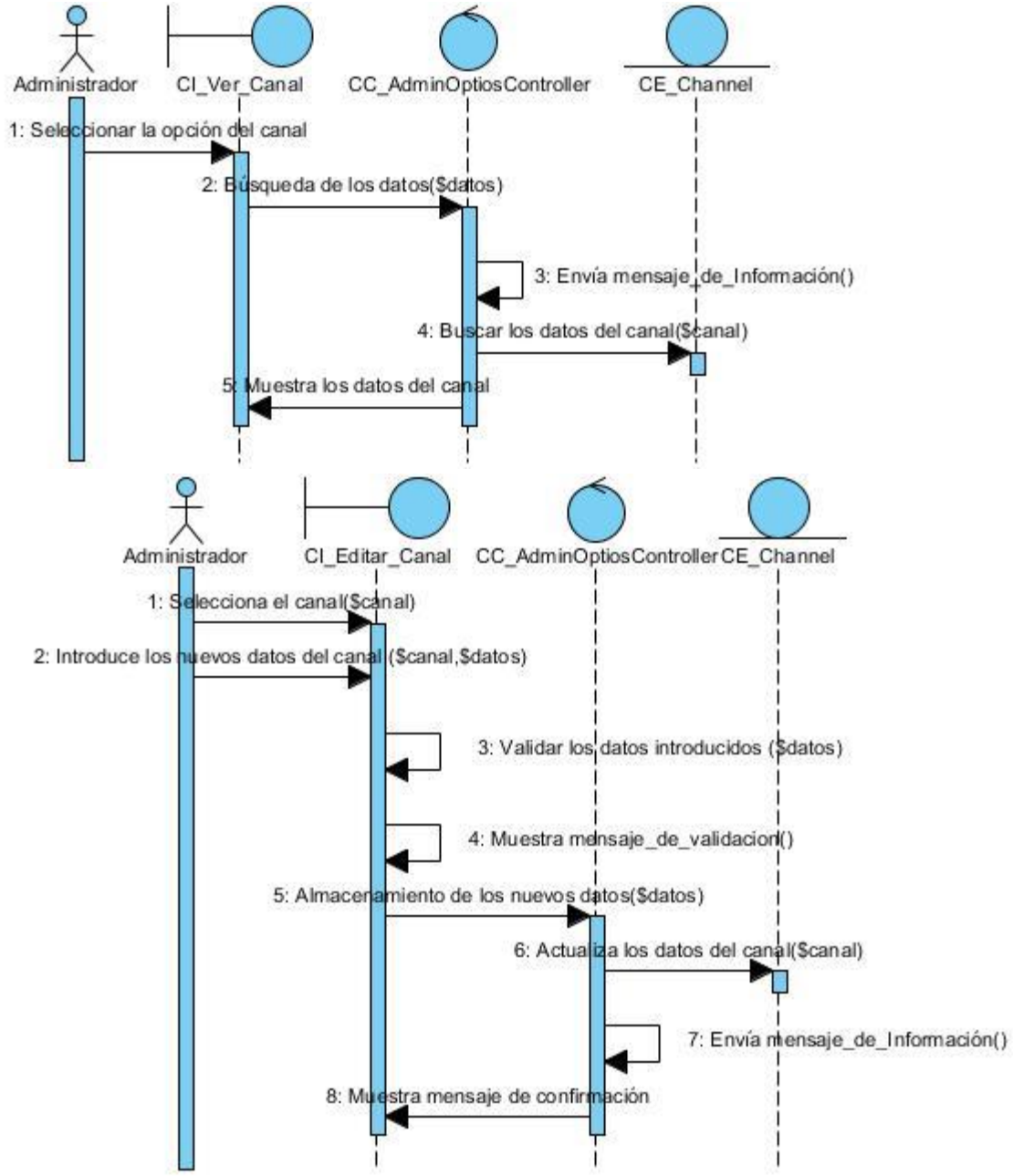


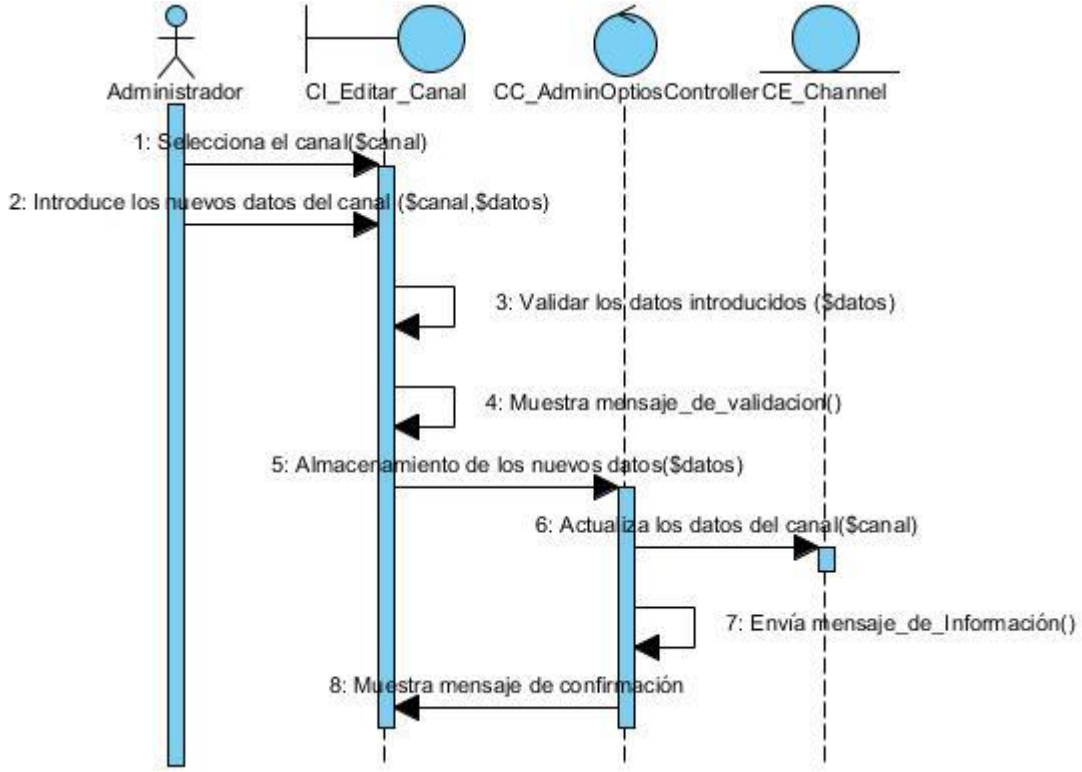
Anexo 4: Diagramas de clases del diseño

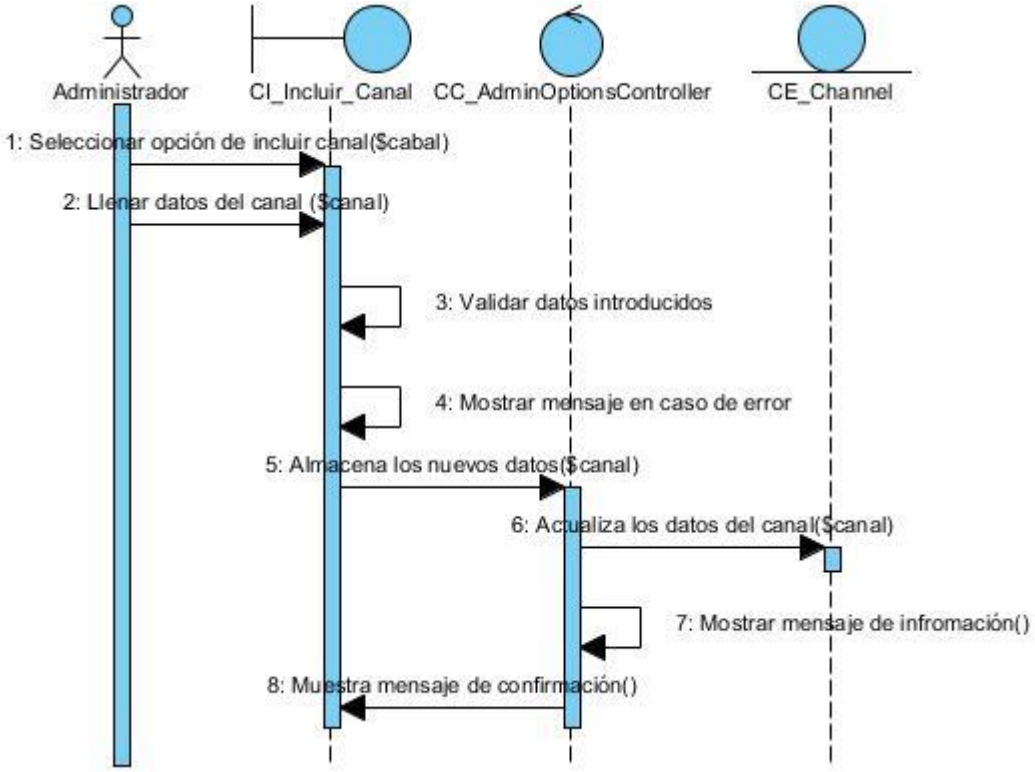


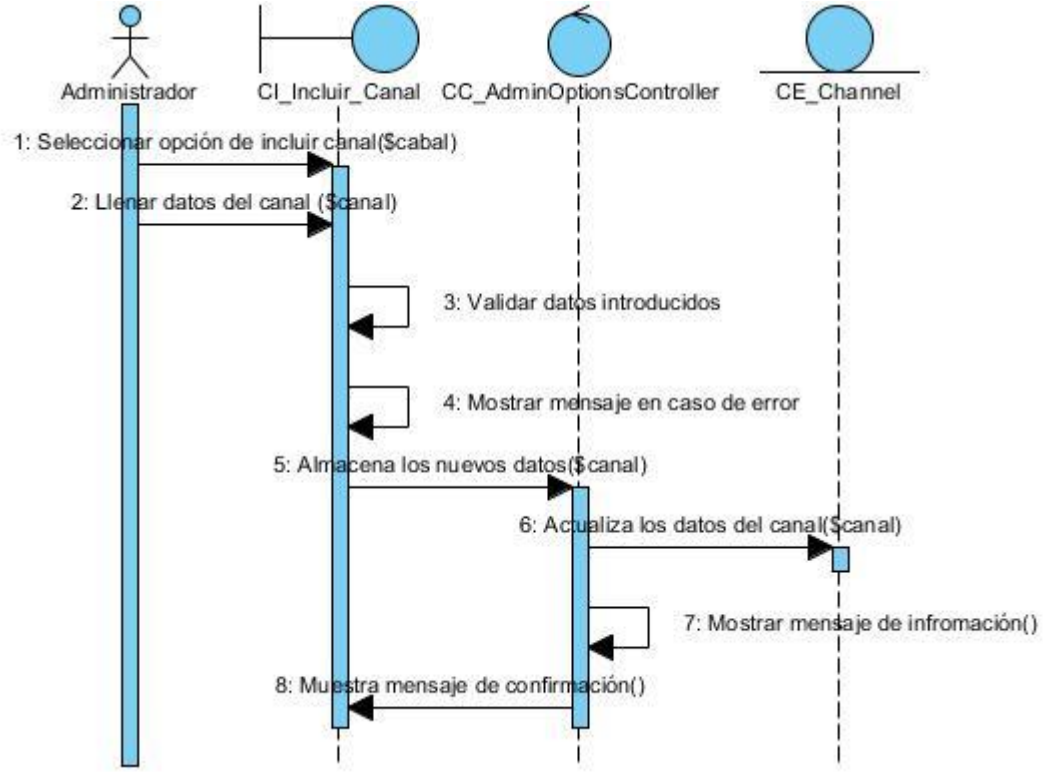
Anexo 5: Diagramas de secuencia del diseño

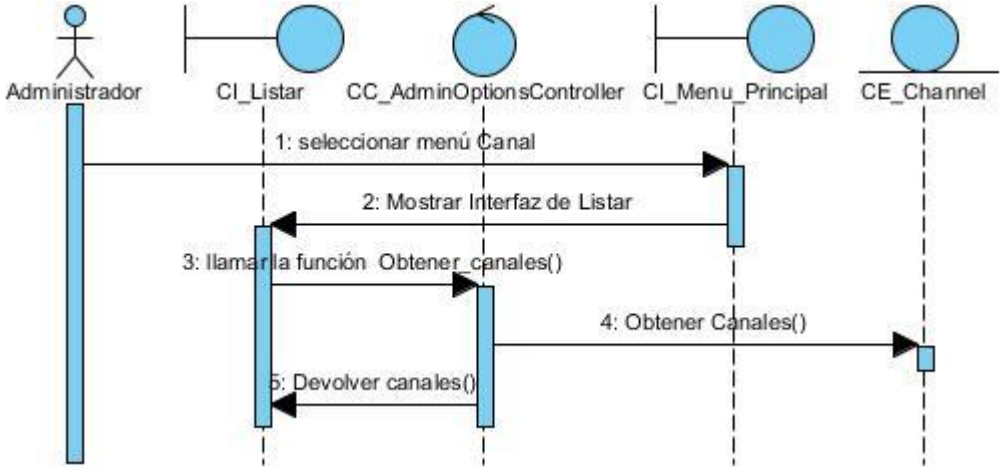


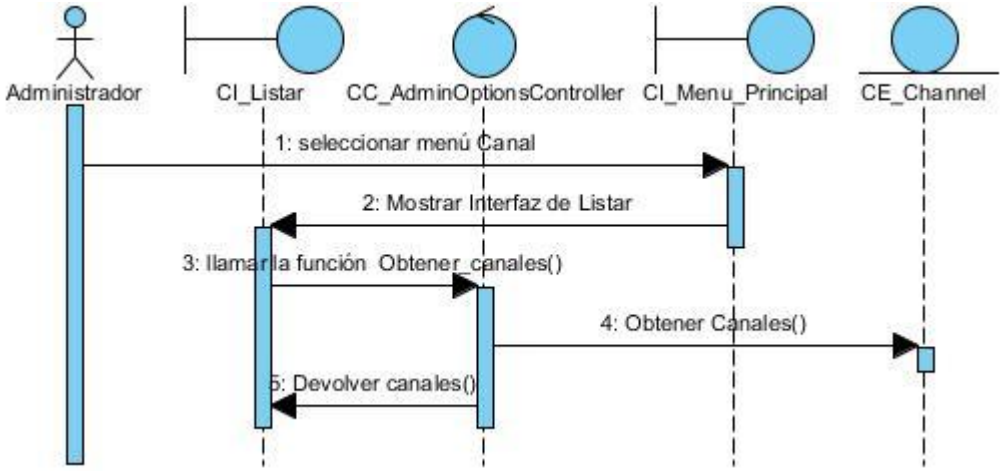

Administrador
CI_Ver_Canal
CC_AdminOptiosController
CE_Channel
1: Seleccionar la opción del canal
2: Búsqueda de los datos($datos)
3: Envía mensaje_de_Información()
4: Buscar los datos del canal($canal)
5: Muestra los datos del canal
Administrador
CI_Editar_Canal
CC_AdminOptiosController
CE_Channel
1: Selecciona el canal($canal)
2: Introduce los nuevos datos del canal ($canal,$datos)
3: Validar los datos introducidos ($datos)
4: Muestra mensaje_de_validacion()
5: Almacenamiento de los nuevos datos($datos)
6: Actualiza los datos del canal($canal)
7: Envía mensaje_de_Información()
8: Muestra mensaje de confirmación


Administrador
CI_Editar_Canal
CC_AdminOptiosController
CE_Channel
1: Selecciona el canal($canal)
2: Introduce los nuevos datos del canal ($canal,$datos)
3: Validar los datos introducidos ($datos)
4: Muestra mensaje_de_validacion()
5: Almacenamiento de los nuevos datos($datos)
6: Actualiza los datos del canal($canal)
7: Envía mensaje_de_Información()
8: Muestra mensaje de confirmación


Administrador
CI_Incluir_Canal
CC_AdminOptionsController
CE_Channel
1: Seleccionar opción de incluir canal($cabal)
2: Llenar datos del canal ($canal)
3: Validar datos introducidos
4: Mostrar mensaje en caso de error
5: Almacena los nuevos datos($canal)
6: Actualiza los datos del canal($canal)
7: Mostrar mensaje de infromación()
8: Muestra mensaje de confirmación()


Administrador
CI_Incluir_Canal
CC_AdminOptionsController
CE_Channel
1: Seleccionar opción de incluir canal($cabal)
2: Llenar datos del canal ($canal)
3: Validar datos introducidos
4: Mostrar mensaje en caso de error
5: Almacena los nuevos datos($canal)
6: Actualiza los datos del canal($canal)
7: Mostrar mensaje de infromación()
8: Muestra mensaje de confirmación()


Administrador
CI_Listar
CC_AdminOptionsController
CI_Menu_Principal
CE_Channel
1: seleccionar menú Canal
2: Mostrar Interfaz de Listar
3: llamar la función Obtener_canales()
4: Obtener Canales()
5: Devolver canales()



Anexo 10: Recursos usados para aplicar el método de Iadov.

	1. ¿Considera usted que la incorporación a la arquitectura Xalix de un módulo para configurar la herramienta de réplicas SymmetricDS puede ser más factible para los proyectos que utilizar puramente una herramienta de replicación?								
	Sí			No sé			No		
2. ¿Está usted satisfecho con el desarrollo de un módulo para la configuración de réplicas con la herrameinta SymmetricDS dentro de la arquitectura Xalix?	3. Si el sistema en el que usted trabaja por sus carácteristicas de despliegue requiere el uso de técnicas de replicación ¿Utilizaría el Módulo para la configuración de réplicas con SymmetricDS desarrollado para la arquitectura Xalix en lugar de algún otro mecanismo para replicación?								
	Sí	No sé	No	Sí	No sé	No	Si	No sé	No
Me satisface mucho	1	2	6	2	2	6	6	6	6
Más satisfecho que insatisfecho	2	2	3	2	3	3	6	3	6
Me es indiferente	3	3	3	3	3	3	3	3	3
Más insatisfecho que	6	3	6	3	4	4	3	4	4

satisfecho									
No me satisface	6	6	6	6	4	4	6	4	5
No sé qué decir	2	3	6	3	3	3	6	3	4

Tabla 7. . Cuadro de análisis de ladov modificado por el autor.

El número resultante de la interrelación de las tres preguntas nos indica la posición de cada evaluador en los siguientes niveles de satisfacción:

1. Clara satisfacción.
2. Más satisfecho que insatisfecho.
3. No definida.
4. Más insatisfecho
que satisfecho.
5. Clara insatisfacción.
6. Contradictoria

La cual se necesita para calcular el Índice de Satisfacción Grupal (ISG) mediante la fórmula:

ISG =[A (+ 1) + B (+ 0,5) + C (0) + D (-0,5) + E (-1)]/N

Donde: A, B, C, D, E, representan el número
de sujetos con índice individual 1; 2; 3 ó 6; 4;
5 y N representa el número total de sujetos del grupo.
Que permite reconocer las siguientes categorías grupales:

•Insatisfacción: desde (-1) hasta (-0,5).
•Contradictorio: desde (-0,49) hasta (+0,49).
•Satisfacción: desde (+0,5) hasta (1).

Anexo 11: Encuesta aplicada para método de ladov

Encuesta de comprobación de la necesidad de un módulo de réplicas para la arquitectura Xalix del Centro de Tecnologías para la Formación.

Marque con una X la opción que considere adecuada.

1. ¿Tiene usted experiencia trabajando con herramientas de réplica?

__Si __No

2. ¿Ha formado parte del equipo de desarrollo de algún proyecto que haya requerido el uso de réplicas?

   __Si __No

3. ¿Ha trabajado directamente en la configuración de réplicas?

   __Si __No

En caso de que su respuesta sea positiva escriba cuales: _____

4. ¿Considera usted que la incorporación a la arquitectura Xalix de un módulo para configurar la herramienta de réplicas SymmetricDS puede ser más factible para los proyectos que utilizar puramente una herramienta de replicación?

   _____Si _____No sé _____No

5. Si el sistema en el que usted trabaja por sus carácteristicas de despliegue requiere el uso de técnicas de replicación ¿Utilizaría el Módulo para la configuración de réplicas con SymmetricDS desarrollado para la arquitectura Xalix en lugar de algún otro mecanismo para replicación?

   _____Si _____No sé _____No

6. ¿Está usted satisfecho con el desarrollo de un módulo para la configuración de réplicas con la herrameinta SymmetricDS dentro de la arquitectura Xalix?

__Me satisface mucho.

__Más satisfecho que insatisfecho

__Me es indiferente

__Más insatisfecho que satisfecho

__No me satisface

__No se que decir