Universidad de las Ciencias Informáticas

Facultad 6

Título: *Desarrollo de la funcionalidad para la programación de captura y envío de datos en el Replicador Reko.*

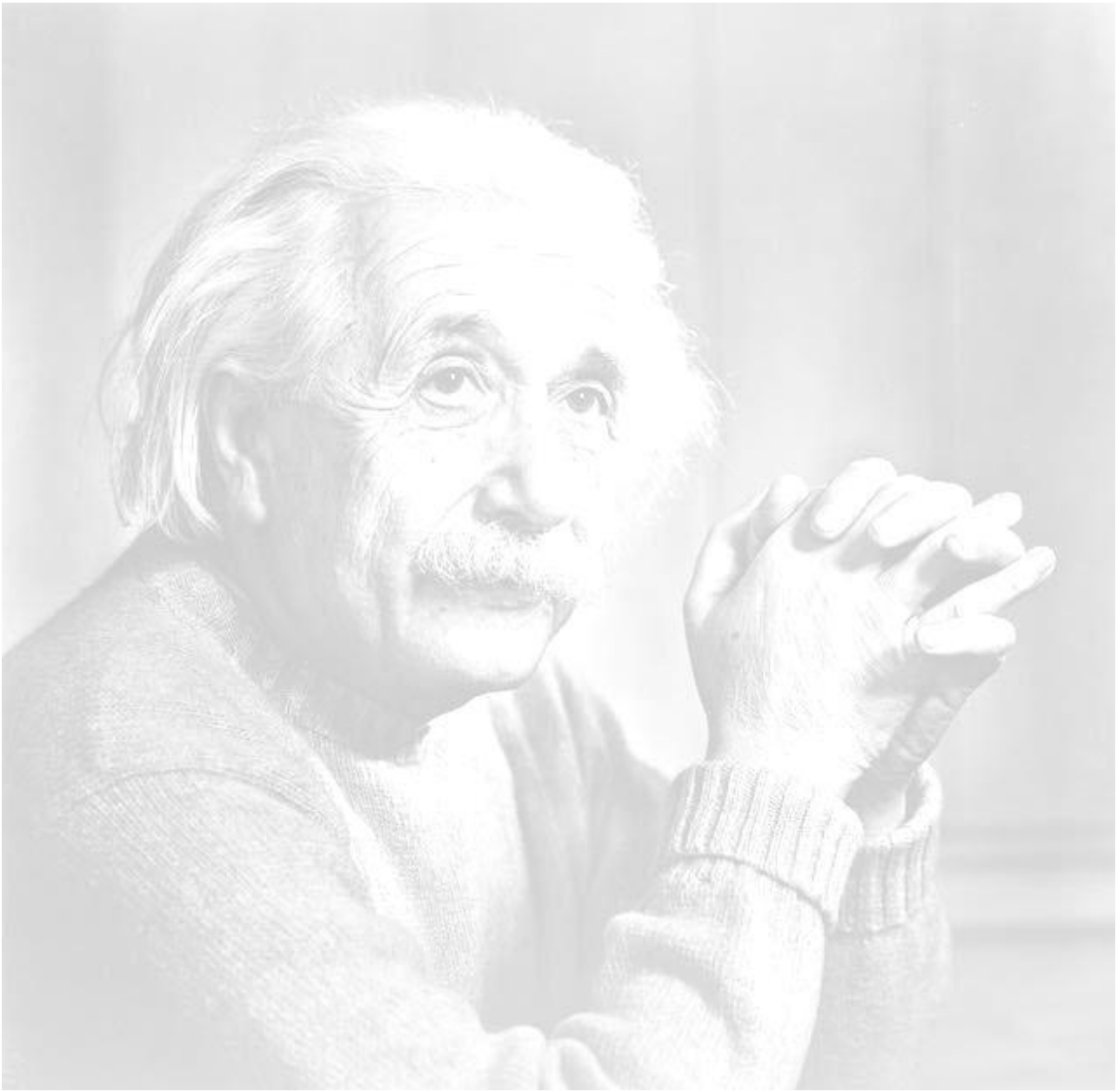
Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor(es): Anaís Peña Pérez
Anay Miranda Estrada

Tutor(es): Ing. Raciel Evelio Ramírez Romero
Lic.Yoemny González Almaguer

Ciudad de la Habana, Cuba.

Junio ,2010



"Nunca consideres el estudio como una obligación sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber."
Albert Einstein

DECLARACIÓN DE AUTORÍA

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

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

Anaís Peña Pérez

_____

Firma del Autor

Lic.Yoemny González Almaguer

_____

Firma del Tutor

Anay Miranda Estrada

_____

Firma del Autor

Ing. Raciel Evelio Ramírez Romero

_____

Firma del Tutor

DATOS DE CONTACTO

Autores:

Anaís Peña Perez **email:** apenap@estudiantes.uci.cu

Anay Miranda Estrada **email:** amestrada@estudiantes.uci.cu

Tutores:

Raciel Evelio Ramírez Romero

Ingeniero en Ciencias Informáticas

Email: reramirez@uci.cu

Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

Yoemny González Almaguer

Licenciado en Ciencias de la Computación

Email: yoemny@uci.cu

Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

AGRADECIMIENTOS

Anaís: *Agradezco a mis padres por la confianza que tuvieron en mi.*

A mi madre por tantas noches de desvelo cuando tuve algún problema y me encontraba lejos de mi hogar, por nunca fallarme y ayudarme a cumplir mis sueños.

A mi padre por darme tantas fuerzas cuando muchas veces creí que no podría terminar. Por ser mí mejor amigo.

A mi hermano por cubrirme en casa durante 5 largos años y llenar con amor mi ausencia.

A mis abuelas por darme tanto amor y apoyo incondicional.

A mis abuelos que aunque no estén físicamente presentes, siempre los llevo en mi corazón.

A mis tíos y primos por estar siempre pendiente a mí.

A mis amigos de estos 5 años, especialmente a Andrés y a Félix por haberme ayudado a llegar donde estoy hoy. Sin ustedes no lo hubiese logrado.

A mi segunda familia, la familia de Anay que siempre les estaré agradecidos por acogerme en su hogar como una hija más.

A todas las compañeras de apartamento por convivir tantos buenos momentos juntas.

A nuestros tutores por ayudarnos en la realización de la tesis.

A Anay por brindarme su Amistad por tantos años, por pasar los mejores momentos en la Universidad junto a ella….Amigas por siempre…

***Anay**: Agradezco a mi familia por haberme brindado su apoyo y confianza en estos 5 años de la carrera.*

A mi mamá y mi papá por estar siempre en los momentos más difíciles, sin ustedes no habría llegado este día, los amo.

A mi tío Macho, mi segundo papá, por ser un guía para mí.

A mis abuela Rina por darme tanto amor y dedicación.

A mi hermano y mi prima Gaby, por existir y recordarme que tengo a quienes dar el ejemplo.

A mi abuelo Elio, a mi tía Miriam, a todos mis tíos y primos que de alguna forma han estado pendientes de mí durante estos 5 años.

A mis amigos Félix y Andrés, gracias por brindarme su ayuda incondicional, sin ustedes hubiese no hubiese llegado al final.

A la familia de Anaís, a sus padres por acogerme en su casa y hacerme sentir como una hija más para ellos.

A mi novio Armando, por ser tan cariñoso y paciente conmigo.

A Figo por estar siempre dispuesto a ayudar.

A Sandor y a su familia por estar a mi lado en los años más difíciles de la carrera.

A todas mis compañeras de apartamento por los momentos que vivimos juntas.

A todas mis amistades con las que pasé momentos inolvidables.

A mis tutores y profesores que me ayudaron a realizar la tesis.

A mi mejor amiga Anaís, gracias dejarme estar a tu lado por tantos años, por entenderme y aconsejarme y por vivir juntas tantos momentos de felicidad que nunca voy a olvidar…espero que seamos amigas para siempre.

DEDICATORIA

A mis padres con todo mi amor.

Anaís.

A mi hermano, espero algún día aparecer en el mismo lugar.

Anay.

RESUMEN

En el presente trabajo se pretende mejorar la versión del Replicador Reko creado por la Comunidad Java (JEE) que pertenece al Centro de Consultoría ubicado dentro de de la Universidad de las Ciencias Informáticas (UCI). El software de réplica de datos implementado en la UCI surge como respuesta a la necesidad de mantener actualizadas un conjunto de bases de datos. Se fundamenta en la copia de información de una localización a otra. Pretende cubrir las principales necesidades relacionadas con la distribución de datos entre los gestores más populares como la protección, recuperación, sincronización de datos, transferencia de datos entre diversas localizaciones o la centralización de la información en una única localización. Para su desarrollo se realizó un amplio estudio de los procesos implicados en el tema y del diseño anteriormente propuesto, con el objetivo de garantizar un mejor funcionamiento de la funcionalidad que se implementará. Finalmente se hicieron pruebas para verificar la calidad, seguridad, confiabilidad, y disponibilidad, con el objetivo de obtener la aceptación del cliente.

Palabras clave: Replicador

ÍNDICE

INTRODUCCIÓN

Cuba en aras de lograr una cultura digital en la sociedad, debe aplicar la informatización en todas sus esferas, de ahí la necesidad de dominar e introducir las tecnologías de la información y las comunicaciones. A pesar de ser un país bloqueado económicamente, se apuesta por el desarrollo de la informática para el progreso de la economía y en este sentido se han creado los Politécnicos de Informática, los Joven Club de Computación y Electrónica y se han introducido las carreras de Ciencias Informáticas y de Computación en todas las universidades del país, surgiendo así la Universidad de las Ciencias Informáticas (UCI) como centro rector de la industria cubana de producción de software.

En el país comenzó a existir la necesidad de estimular la actividad de consultoría como un servicio generador de recursos financieros al Estado. El capital intelectual utilizado fue el de la Universidad de las Ciencias Informáticas. En la misma se crea el Centro de Consultoría Tecnológica e Integración de Sistemas que tiene como objetivo brindar servicios de consultoría en tecnologías avanzadas de la información a instituciones nacionales y extranjeras. Dentro del marco de expansión de dicho centro surge la Comunidad de tecnologías java (JEE) que tiene como objetivo el desarrollo de tecnologías sobre la plataforma Java como un paso fundamental para la estandarización de la producción del software en la UCI, contando con la experiencia de los arquitectos y desarrolladores de los proyectos productivos. Además de formar consultores altamente calificados, comprometidos con los principios de la Revolución Cubana y al servicio de esta. El centro fue construido sobre las tecnologías asociadas a las líneas de consultoría propias como la Arquitectura Orientada a Servicios (SOA).

Como respuesta a la necesidad de mantener actualizadas un conjunto de base de datos, la UCI en conjunto con esta Comunidad crea el software de réplica de datos Reko. El mismo cuenta con una primera versión. Se fundamenta en la copia de información de una localización a otra y pretende cubrir las principales necesidades relacionadas con la distribución de datos entre los gestores más populares como la protección, recuperación, sincronización de datos, transferencia de datos entre diversas localizaciones o la centralización de la información en una única localización. Permite además la representación de complejos escenarios de replicación con herramientas para la definición de

localizaciones o nodos y la selección de los datos de replicación. Cuenta con una herramienta Web para la administración y monitoreo de los datos de replicación.

Reko constituye hoy una solución práctica a las principales necesidades de replicación de datos en la mayoría de los sistemas actuales. Su carácter portable, teniendo en cuenta que el software es multiplataforma y consume pocos recursos, permiten que sea implantado en la mayoría de los escenarios actuales. Realizando un estudio de los diferentes replicadores existentes en la actualidad identificamos que muchos cuentan con un planificador integrado donde pueden programarse transferencias automáticas. Reko desarrolla actualmente sus tareas en tiempo real y no cuenta con una funcionalidad que permita definirle horarios a las tareas de captura y envío de datos.

Dado la situación anteriormente expuesta se obtiene el siguiente **problema:** ¿Cómo permitir que las tareas de captura y envío de datos puedan realizarse en un instante de tiempo determinado?

Objeto de estudio: Proceso de replicación de datos.

Campo de acción: Proceso de replicación de datos en el Replicador Reko.

Objetivo General: Desarrollar la funcionalidad que permita planificar horarios para la captura y envío de datos en el Replicador Reko.

Objetivos específicos:

- Diseñar la funcionalidad para la planificación de captura y envío de los datos en el Replicador Reko.
- Implementar la funcionalidad para la planificación de captura y envío de los datos en el Replicador Reko.
- Realizar pruebas de sistema a la funcionalidad obtenida.

Tareas a realizar:

- Revisión de la bibliografía existente, enfatizando en la referente al replicador Reko.
- Realización del diseño de la funcionalidad para la planificación de captura y envío de los datos del replicador Reko.
- Realización de la implementación de la funcionalidad para la planificación de captura y envío de datos en los horarios definidos.
- Realización de las pruebas de sistema a la funcionalidad obtenida.

Métodos Utilizados:

- **Sintético-Analítico:** Se consultaron diferentes replicadores y se analizaron sus funcionalidades, además se revisó la bibliografía referente al replicador Reko, identificando sus características y funciones, derivándose de dicho análisis sus insuficiencias.

- **Inductivo-Deductivo**: Esta investigación se inicia mediante una situación problémica de la que se pudo deducir la necesidad de hacer programables las tareas de captura y envío. Lo que permite formular el problema e inducir las tareas de la investigación en el camino del cumplimiento de los objetivos específicos, los cuales tributan al objetivo general que es implementar la funcionalidad que permita la programación automática de las tareas de captura y envío.

- **Consulta a expertos**: Se realizará a partir de presentar a especialistas, lo modelado respecto a la funcionalidad de la programación automática de las tareas de captura y envío, en función de recoger criterios que validen la propuesta como una mejora para la transferencia de datos con la utilización del Replicador Reko. Los especialistas deben ser programadores y analistas conocedores de la plataforma Java.

Posibles resultados: El software de réplica Reko contará con la posibilidad de programar de forma automática la captura y envío de los datos brindando así una solución más completa ante las necesidades de replicación.

El presente trabajo de diploma se estructura en introducción, cuatro capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía, anexos y glosario de términos.

En el Capítulo 1: Fundamentación Teórica, se analizan y describen, la metodología, los roles que se emplearán producto de las necesidades actuales, patrones de diseño, herramientas, lenguaje de programación definido y demás elementos asociados al estado del arte.

En el Capítulo 2: Características del sistema, en este capítulo se muestra el Modelo del dominio, así como la descripción de las clases persistentes. Se especifican los requisitos funcionales y los no funcionales, además se realizan los diagramas de casos de uso y las descripciones textuales de los mismos.

En el Capítulo 3: Diseño, se aborda el desarrollo de esta fase, se muestran, los detalles relacionados con el diseño de la funcionalidad propuesta y se utilizan para su modelado los diagramas de clases de diseño y los diagramas de iteración para los diferentes casos de uso, así como el diagrama de despliegue. Se tienen en cuenta aspectos de seguridad, prototipo de interfaz y validación.

En el Capítulo 4: Implementación y Pruebas, se muestra la implementación de la funcionalidad, se analizan los estilos de codificación, y se brinda una solución a los requisitos especificados. Se obtienen los diagramas de componentes, se muestra el código fuente de las principales clases así como Interfaces de la aplicación. Se realizan pruebas de sistema a la funcionalidad realizada.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción

En este capítulo se abordarán los principales conceptos referentes al proceso de replicación de datos. Se realizará un estudio de las herramientas y funcionalidades existentes actualmente para la planificación de horarios de una tarea. Se describen además las herramientas, tecnologías y metodologías utilizadas para la solución del problema planteado.

1.2 Proceso de Replicación de datos

La replicación es el proceso de copiar y mantener los objetos de base de datos, tales como tablas, en múltiples bases de datos que componen un sistema de bases de datos distribuidas. Los cambios aplicados en un sitio, son capturados y almacenados localmente antes de ser transmitidos y aplicados en cada uno de las ubicaciones remotas.

Existen cuatro tipos de clasificaciones para las réplicas de datos atendiendo a:

- Ambiente de replicación:
  - **Maestro-esclavo** (master-slave)**:** La replicación se realiza en un sólo sentido: desde un nodo maestro a uno o varios nodos esclavos.
  - **Multi-Maestro** (multi-master)**:** Todos los nodos se configuran como nodos maestros y pueden replicar 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. Requiere una alta disponibilidad de recursos de red. Una transacción no debe ser aceptada si todos los sitios intervinientes no están de acuerdo o disponibles.
  - **Asíncrona:** Los cambios ejecutados en una tabla son almacenados y enviados posteriormente al resto de los nodos del entorno cada ciertos intervalos de tiempo. La

replicación asincrónica es el modo por defecto para la replicación debido a que requiere menos recursos de red y hardware que la replicación sincrónica, resultando en mejor disponibilidad y desempeño.

- Forma de capturar y almacenar los cambios a replicar:
  - **Basada en triggers:** Se crean una serie de triggers en la base de datos, que permiten capturar las operaciones 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.

- Realización en línea o fuera de línea:
  - **Replicación de datos en línea *(On-line):*** La replicación de datos *On-line* consiste en copiar datos de un nodo a otro mediante una conexión. Se emplea en casos en que el tiempo de recuperación debe ser corto. Los datos en el nodo alterno deben estar replicados en tiempo real en todos los dispositivos de almacenamiento.
  - **Replicación de datos fuera de línea (*Off-line)*:** Involucra dos o más sitios de que almacenan su información en algún medio o lo envían de un nodo a otro vía mensajería. Se utiliza principalmente con aquellas aplicaciones en las cuales el tiempo de recuperación no es una prioridad para el negocio (1).

1.3 Herramientas para la planificación de tareas

¿Qué es un planificador de tareas?

Es una herramienta que permite ejecutar tareas planificadas de forma automática una sola vez, en un período de tiempo especificado tras el inicio del ordenador o mediante algunos eventos como la pulsación de una tecla de acceso directo.

Quartz

Es una herramienta de código abierto, con licencia Apache 2.0 para la planificación y gestión de tareas. Puede ser utilizada de forma individual o integrada con aplicaciones Java.

1. Válido para aplicaciones tanto Java 2 Enterprise Edition (J2EE) como Java Standar Edition (J2SE).

2. Planificación flexible de tareas, por ejemplo: el primer lunes de Enero de cada año a las 17:45.
3. Mantenimiento del estado de las tareas incluso en caso de fallos y reinicios de máquinas.
4. Posibilidad de trabajar en modo Clúster.

A continuación se describen sus componentes: Job, JobDetail, Trigger, Scheduler.

org.quartz.Job: Define una tarea o un trabajo.

org.quartz.JobDetail: Es una clase que almacena propiedades de una determinada tarea. Las tareas se clasifican en grupos y cada tarea tiene un nombre único dentro del grupo.

org.quartz.Trigger: Es una clase abstracta que define los instantes en que la tarea debe ser ejecutada. Existen varias implementaciones de esta clase pero las más usadas son:

1. **org.quartz.SimpleTrigger**: Permite especificar ejecuciones de tareas teniendo en cuenta los siguientes parámetros: fecha, hora, nº de repeticiones e intervalo entre repeticiones.

2. **org.quartz.CronTrigger**: Es el más utilizado, pues permite especificar mediante expresiones más complejas los instantes en los que deben ejecutarse las tareas.

org.quartz.Scheduler: Su funcionalidad es almacenar y planificar las tareas (Job) en base a los Triggers, recuperar tareas fallidas, realizar reintentos y gestionar el estado del sistema de planificación (2).

Clase java.util.Timer

Desde Java Development Kit (JDK) 1.3, en Java se incorporaron capacidades de temporizador, a través de las clases java.util.Timer y java.util.TimerTask. Con esta nueva capacidad se pueden programar tareas para un plazo de ejecución, o para ejecución repetida a intervalos regulares.

La forma de utilizarlo es creando una clase que implemente TimerTask y redefinir, con la tarea a ejecutar, el método run () que define esta clase. Este método run () es el que se ejecutará cuando llegue el momento indicado.

La clase Timer es la encargada de registrar las tareas y de definir el instante de tiempo en el cual se ejecutara la tarea registrada. Para registrar una tarea, la clase Timer utiliza el método scheduleAtFixedRate (), al cual se le pasan los siguientes parámetros: un objeto que implemente TimerTask(es la tarea a realizar), el instante de tiempo en que comienza dicha tarea y el intervalo de tiempo en el que se seguirá ejecutando la tarea.

De las herramientas para programar tareas en el tiempo se estudiaron el mecanismo temporizador de Java y la librería Quartz, se decidió utilizar la librería Quartz por las siguientes razones:

- El temporizador de Java no tiene mecanismo de persistencia.
- El temporizador posee un horario inflexible (solo son capaz de establecer un tiempo de inicio y un intervalo de repetición, nada basado en fechas, horas del día, entre otros) (3).

1.4 Programas que utilizan un planificador para realizar algunas de sus tareas.

Kaspersky

Es un software antivirus que contiene funcionalidades básicas para la protección de computadoras contra una amplia gama de amenazas informáticas. Incluye un planificador de horarios para la tarea de Actualización del software.

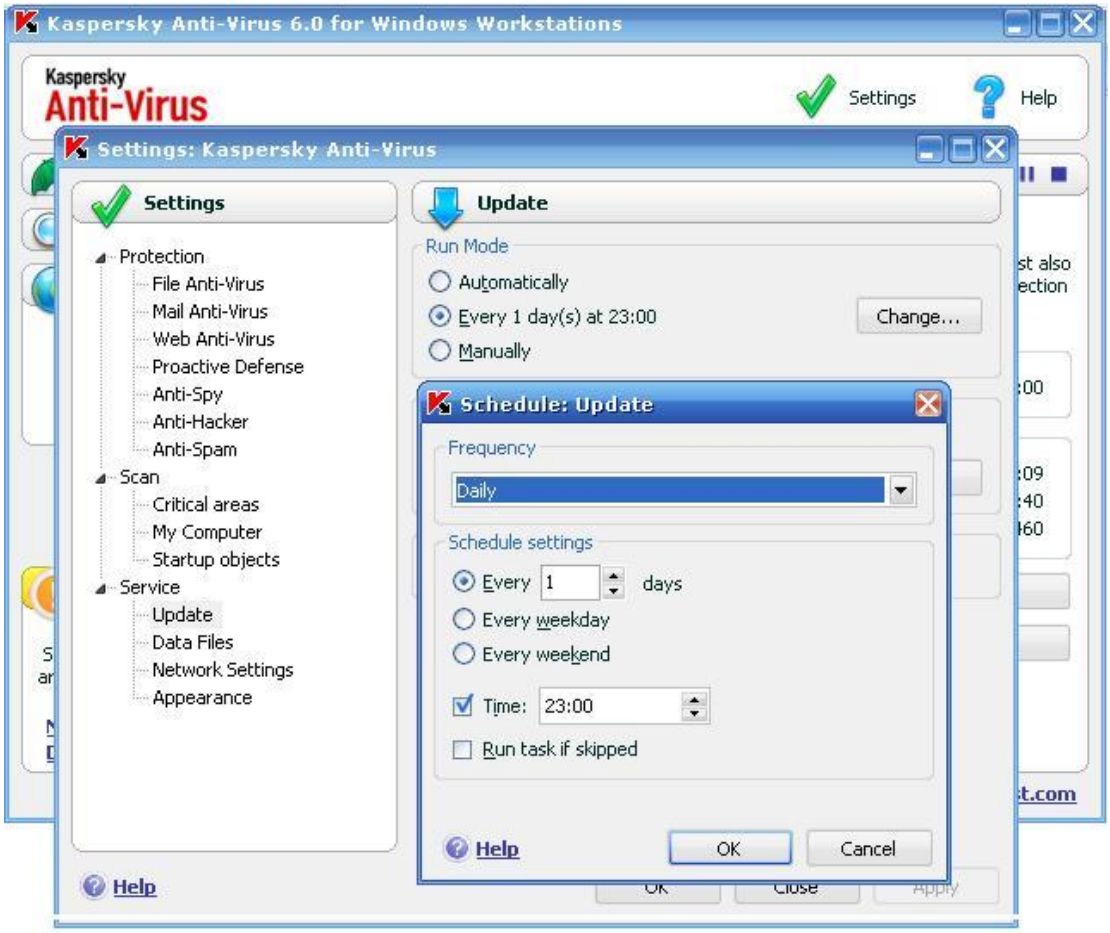


Figura1.Planificador de horarios para la Actualización del Kaspersky.

Advanced Task Scheduler 1.5 build 0442

Es un planificador de tareas multifuncional, lo cual permite la ejecución de tareas y archivos por lotes, abrir documentos y páginas de Internet, mostrar recordatorios emergentes, reproducir sonidos, apagar y reiniciar el ordenador, apagar solo el monitor, detener procesos en ejecución, establecer y cerrar conexiones de marcado, de forma automática (4).

Requerimiento de Sistema Operativo:

* Windows XP
* Windows 2000
* Windows ME
* Windows NT
* Windows 98
* Windows 95

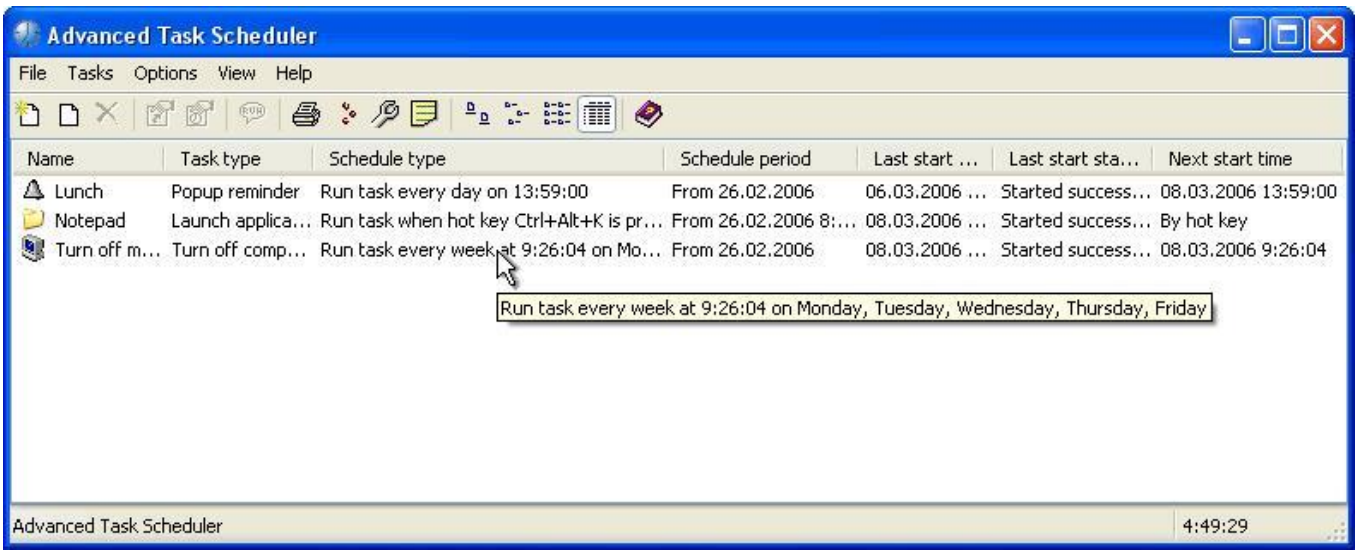


Figura 2.Planificador de tareas avanzado para Windows.

1.5 Replicadores existentes con planificador integrado

Un replicador de datos es un sistema que se encarga de mantener actualizada un conjunto de bases de datos. Existen algunos replicadores que cuentan con un planificador de tareas integrado. El cual permite realizar las tareas de forma automática.

LS Retail

Es una herramienta para controlar y mantener actualizados un conjunto de datos. LS Retail incluye un módulo de comunicación que le permite transmitir fácilmente datos. El módulo de comunicación consiste de tres partes: el LS Data Director (director de datos), Transaction Server (servidor de transacciones) y el LS Retail Scheduler (Planificador del LS Retail). LS Retail incluye un mecanismo de planificación que se puede utilizar para ejecutar trabajos por lote. Este mecanismo también se usa para programar transferencias de datos. LS Retail Scheduler (planificador) es una herramienta muy flexible que puede operar de acuerdo a determinados parámetros. Los trabajos se pueden programar para que se ejecuten en determinadas fechas o con determinados intervalos (5).

Karen's Replicator v3.6.5, para hacer copias de respaldo (backups) automatizadas

Karen's Replicator es una herramienta gratuita que realiza automáticamente copias de seguridad de tus archivos y directorios más importantes, o incluso de unidades completas. El programa copia los datos seleccionados de un directorio o unidad a otro, y tanto el origen como el destino pueden estar en tu propia máquina o en cualquier otra conectada en red local. Soporta ficheros de gran tamaño, permite realizar copias de acuerdo con un intervalo de tiempo personalizable y tiene capacidad para actualizar las copias de seguridad añadiendo únicamente aquellos archivos que hayan sido modificados (6).

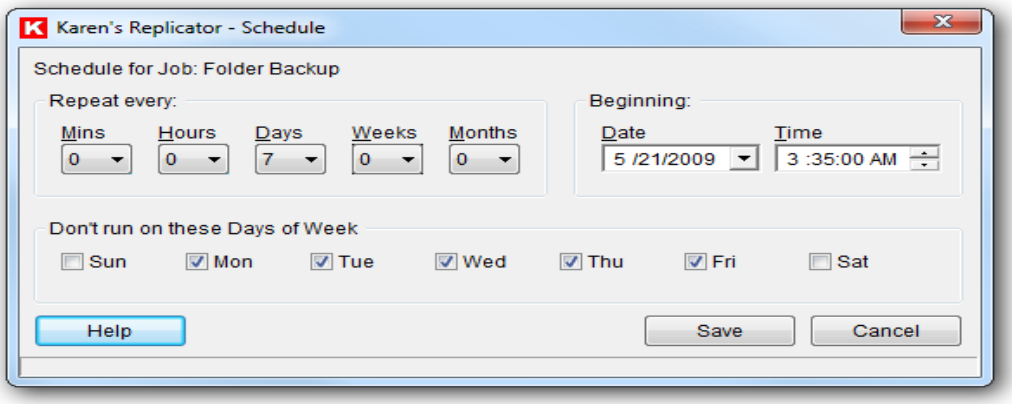


Figura 3. Planificador de Karen´s Replicator.

1.6 Características del Replicador Reko

Características.

El software de réplica de datos, Reko, fue diseñado y construido por un grupo de desarrolladores pertenecientes a la Universidad de las Ciencias Informáticas. El 2 de noviembre de 2009, fue liberada su versión 1.0, luego de ser aprobado por los Laboratorios de Pruebas y el grupo de Calisoft.

Tipos de replicación: Soporta la replicación de transacciones a través de *Hibernate* y la replicación de acciones a través de *triggers*.

Hibernate.

Es una herramienta de Mapeo objeto-relacional para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos Extensible Markup Language (XML) que permiten establecer estas relaciones.

Trigger: un trigger (o disparador) en una Base de datos, es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).

Gestores soportados: Se integra actualmente con los gestores de bases de datos *Oracle* y *PostgreSQL*.

Selección de datos: Provee la facilidad de seleccionar el subconjunto de tablas y datos a ser replicados de la base de datos mediante la definición de filtros aplicables a las tablas y la selección de usuarios propios del gestor.

Transmisión de datos: La transferencia de datos de replicación puede realizarse a soporte para replicación mediante protocolos de transferencia de archivos (TCP/IP, FTP, HTTP) o por ficheros de forma manual. Los archivos de gran tamaño son enviados por Protocolo de Transferencia de Archivos FTP permitiendo resumir la trasmisión en caso de interrupciones en la red.

Configuración entre nodos: El mecanismo de registro entre nodos de replicación se basa únicamente en un identificador (id) del nodo lo que permite la abstracción de los datos físicos de cada nodo como son el IP y el protocolo de comunicación. Este mecanismo permite que los nodos puedan moverse por distintas subredes y mantener sus datos sincronizados. Por ejemplo, mantener sincronizada la base de datos de una laptop con la base de datos central a través de Internet.

Seguridad: Los nodos de replicación manejan credenciales entre ellos para verificar la autenticidad de los datos transferidos. El envío de datos se realiza utilizando protocolos de comunicación seguros como son (HTTPS y SSL).

Monitoreo: Permite el conocer el estado de los datos transmitidos, puede realizarse en tiempo real a través de la Web así como dar un seguimiento al funcionamiento interno del mecanismo.

Interfaz visual Web: La administración y configuración de la réplica se realiza a través de una interfaz Web, por lo que es administrable vía remota usando solamente un navegador.

Independiente de la plataforma: El software de réplica puede ser instalado en cualquier sistema operativo y no necesita de un ambiente gráfico en el mismo para su funcionamiento.

Tolerancia a fallos: Detecta errores de conexión. Mantiene los datos de réplica en un estado estable en caso de desconexión. Al restablecerse la conexión, automáticamente sincroniza los datos entre las bases de datos.

En la figura 4 se muestra un posible escenario de replicación donde la información de todas las BD es enviada hacia un clúster de BD y a la vez el clúster puede enviar actualizaciones hacia los demás nodos. El nodo 5 se encuentra dentro del sistema de réplica como otro nodo más, con la particularidad de que transmite sus datos a través de internet mediante el protocolo HTTP o HTTPs. Reko permite que dos BD pueden replicar entre sí aunque se encuentren en subredes distintas y utilicen protocolos de transmisión diferentes, la única restricción es que todos deben estar conectados un servidor central JMS (Java Message Service) o en caso de un clúster de servidores JMS, estar conectados a uno de ellos .

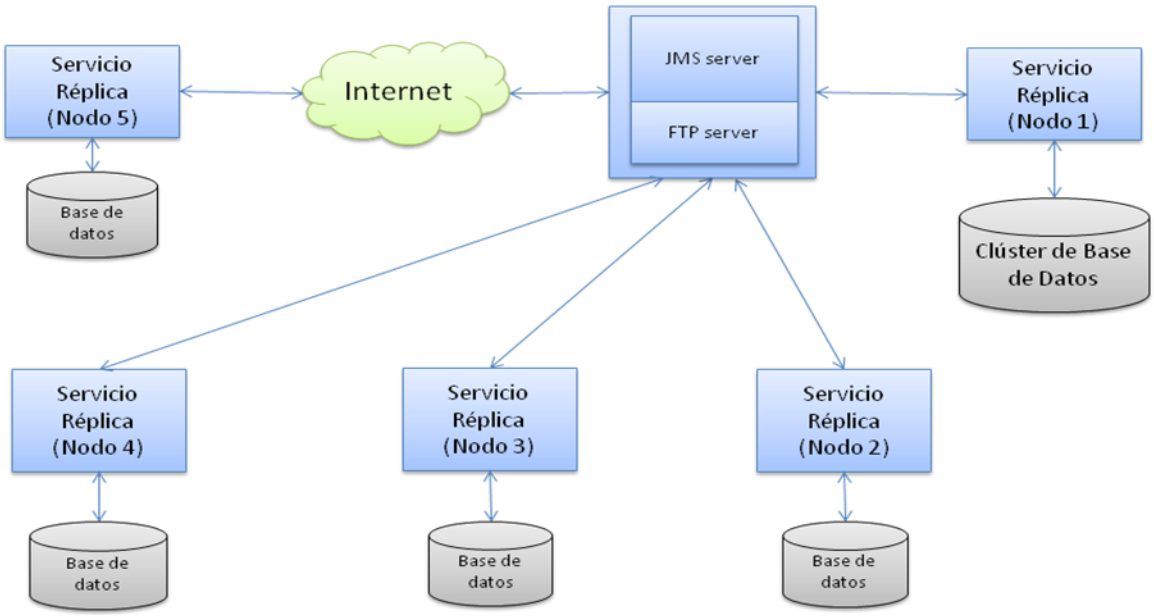


Figura 4. Escenario de replicación de Reko.

El estudio realizado permitió conocer las diferentes formas y procedimientos para programar una tarea, proporcionando conocimientos para el diseño e implementación de la funcionalidad requerida (7).

1.7 Herramientas y Metodologías

Para el desarrollo de la funcionalidad se continuó haciendo uso de las metodologías y herramientas definidas anteriormente para el desarrollo del software.

1.7.1 Lenguaje de programación

JAVA

Es un lenguaje de programación orientado a objeto (OO) desarrollado por Sun Microsystems, a principio de los años 90´s, e independiente de la plataforma. Es un lenguaje de propósito general, lo cual permite

crear diversos tipos de aplicación con él; radicando su mayor éxito en Internet, con el uso de los Applets, las aplicaciones cliente – servidor, o las páginas servidoras de java.
Se distingue por ser capaz de gestionar la memoria automáticamente; no permite el uso de técnicas de programación inadecuadas; posee mecanismos de seguridad incorporados, los cuales limitan el acceso a recursos de las máquinas donde se ejecuta; incorpora herramientas de documentación; es multihilos (multithreading). Todas estas características lo califican como un lenguaje de programación robusto (8).

1.7.2 Ambiente de desarrollo

Eclipse IDE

Eclipse es una plataforma de software de código abierto independiente de una plataforma para desarrollar "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Eclipse es más general para el desarrollo de aplicaciones en Java, en la que se ha especializado, también permite programar en PHP, aunque con muy pocas opciones de revisión y corrección de errores y sin completamiento de código. Es una herramienta que necesita de mucha ayuda del hardware para realizar la compilación del código fuente escrito, además de ser multiplataforma (9).

1.7.3 Framework

Un Framework no es más que “una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que puede añadírsele las últimas piezas para construir una aplicación concreta” (10).

Los principales objetivos de cada framework consisten en:

- Acelerar el proceso de desarrollo.
- Reutilizar código ya existente.
- Promover buenas prácticas de desarrollo como el uso de patrones.

Spring

Es un framework de código abierto (Open Source), desarrollado por la compañía Interface 2, para aplicaciones escritas en el lenguaje Java. Los desarrolladores de este framework, lograron combinar en él herramientas tales como: Java 2 Enterprise Editions (J2EE), incluyendo Enterprise JavaBeans (EJB), y Java Server Pages (JSP); esto facilitó brindar una estructura mucho más sólida y brindar un mejor soporte.

Este framework no necesita de muchos recursos para ser ejecutado. Está basado en inyección de dependencias (DI) y orientación a aspectos. Brinda abundante funcionalidad de infraestructura (integración con frameworks de persistencia, gestión de transacciones, etc.) y deja el desarrollo de la lógica de la aplicación al desarrollador. Contiene y gestiona el ciclo de vida y configuración de objetos de aplicación; por estos motivos es considerado un contenedor (11).

1.7.4 Sistema de Gestión de Base de Datos

PostgreSQL

PostgreSQL intenta ser un sistema de base de datos de mayor nivel que MySQL, a la altura de Oracle. Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido más tarde en otros sistemas de gestión comerciales. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de estas características, PostgreSQL no es un sistema de gestión de base de datos puramente orientado a objetos.

Principales características de PostgreSQL.

- Incorpora una estructura de datos array.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.

- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- Tiene mejor soporte para triggers y procedimientos en el servidor (12).

1.7.5 Herramienta CASE

Visual Paradigm

Visual Paradigm es una herramienta de modelado profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objeto (OO), construcción, pruebas y despliegue. Permite la elaboración de todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación en diferentes extensiones (.Pdf, .Doc.).

Facilita el modelado colaborativo con Sistema de Versiones Concurrentes (Concurrent Versions System - CVS) y Subversión (control de versiones), y la realización del proceso de ingeniería así como ingeniería inversa (proceso ingenieril en el que se obtienen modelos conceptuales a partir de los artefactos software como código fuente, ejecutables, binarios y ficheros intermedios). Brinda soporte para el mapeo de objeto relacional (Object-Relational Mapping – ORM) y facilidades para la generación de bases de datos. Soporta múltiples usuarios trabajando sobre el mismo proyecto. Presenta licencia gratuita y comercial. Es fácil de instalar y actualizar y compatible entre ediciones (13).

1.7.6 Metodología de desarrollo

Open Up

Es un framework de procesos de desarrollo de software de código abierto, construido sobre una donación realizada por IBM del Basic Unified Process y liberado por el Eclipse Process Framework (EPF). Permite abordar ágilmente el desarrollo de proyectos pequeños y tiene un enfoque centrado en el cliente con iteraciones cortas.

Este proceso de desarrollo unificado está basado en las mejores prácticas del Rational Unified Process (RUP), permitiendo de esta manera que se puedan desarrollar artefactos ligeros apropiados, utilizando el Unified Modeling Language (UML) e incrementando de esta forma las probabilidades de éxito en función de calidad, costo, tiempo y alcance.

Debido a que está basada en RUP, aplica un enfoque iterativo e incremental dentro de su estructura del ciclo de vida y puede adaptarse para desarrollar diversos tipos de proyectos. Por demás, es un proceso iterativo del desarrollo del software que es mínimo, completo, y extensible (14).

1.7.7 Lenguaje de Modelado

UML

El Lenguaje Unificado de Modelado (UML), es un lenguaje de modelado visual que se emplea para especificar, visualizar, construir y documentar artefactos de un sistema de software orientado a objetos.

Este lenguaje no define un proceso de desarrollo específico, tan solo se trata de una notación. UML es una parte esencial del Proceso Unificado. RUP emplea UML como base para el desarrollo de software en cada una de sus fases y disciplinas. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo.

UML no es un lenguaje de programación sino un lenguaje de propósito general para el modelado orientado a objetos y también puede considerarse como un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes (15).

1.8 Conclusiones

En este capítulo se profundizó en el conocimiento de algunos conceptos necesarios para la compresión de este trabajo. En la elaboración del capítulo se esboza el uso de las tecnologías y herramientas usadas en la confección de la funcionalidad .Para el desarrollo de la misma se utilizará como metodología de desarrollo Open UP por ser aplicable a un amplio conjunto de plataformas y aplicaciones de desarrollo y además es ágil e incremental. Se utilizará UML como lenguaje de modelado ya que permite especificar, visualizar, construir y documentar todos los elementos o artefactos durante todo el proceso de desarrollo. Se utilizará Visual Paradigm como herramienta CASE para realizar el modelado, así como materializar otras funcionalidades que este ofrece. El lenguaje de programación será Java. Además de la utilización del Eclipse por su configuración amigable y las facilidades que brinda. También se hará uso del framework Spring.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA.

2.1 Introducción

En el presente capítulo se realiza un análisis de la propuesta del sistema. Para ello se describen los conceptos más importantes del dominio, se especifican los requisitos funcionales y no funcionales, se identifican mediante el diagrama de casos de uso del sistema las relaciones entre los actores y casos de uso del sistema; así como las descripciones textuales de los casos de uso.

2.2 Breve descripción del sistema

El diseño de la funcionalidad: Programación de la captura y envío de los datos del Replicador Reko, permitirá al usuario definir los horarios y fechas para realizar las réplicas. Aumentando las funcionalidades del sistema para un mejor funcionamiento.

2.3 Modelo del Dominio

El modelo del dominio es una representación de los conceptos u objetos del mundo real, significativos para un problema. Tiene como objetivo fundamental la descripción de las clases más importantes en el sistema, no de los componentes de software.

2.3.1 Descripción de las Clases del Modelo de Dominio

- *Núcleo:* Maneja las configuraciones fundamentales del software y agrupa las principales funcionalidades de procesamiento información.
- *Capturador de Cambios:* Captura los cambios que se realizan sobre la Base Datos (BD) y se los entrega al Distribuidor de Cambios.
- *Aplicador de Cambios*: Ejecuta sobre la BD los cambios que sean replicados desde otro Nodo.
- *Distribuidor de Cambios:* Determina el destino del cambio realizado en la BD, los envía y se responsabiliza de su llegada.

- *BD Local de Réplica:* Es utilizada para guardar las configuraciones propias de la réplica, las acciones sobre la BD que han dado conflicto al aplicarse y las acciones o transacciones que no han podido llegar a su destino.
- *Consola Web de Administración:* Representa la interfaz del software. Permite realizar las configuraciones principales como el registro de nodos, configuración de las tablas a replicar, datos a replicar y el monitoreo del funcionamiento del software.
- *Servidor JMS:* Servidor de Mensajería bajo la especificación Java Message Service, es utilizado como punto intermedio en la distribución de la información enviada bajo JMS.
- *Base de datos:* Es la base de datos que se está replicando, el software de réplica envía los cambios que se realizan sobre ella y aplica los cambios que provienen de otros nodos de réplica.

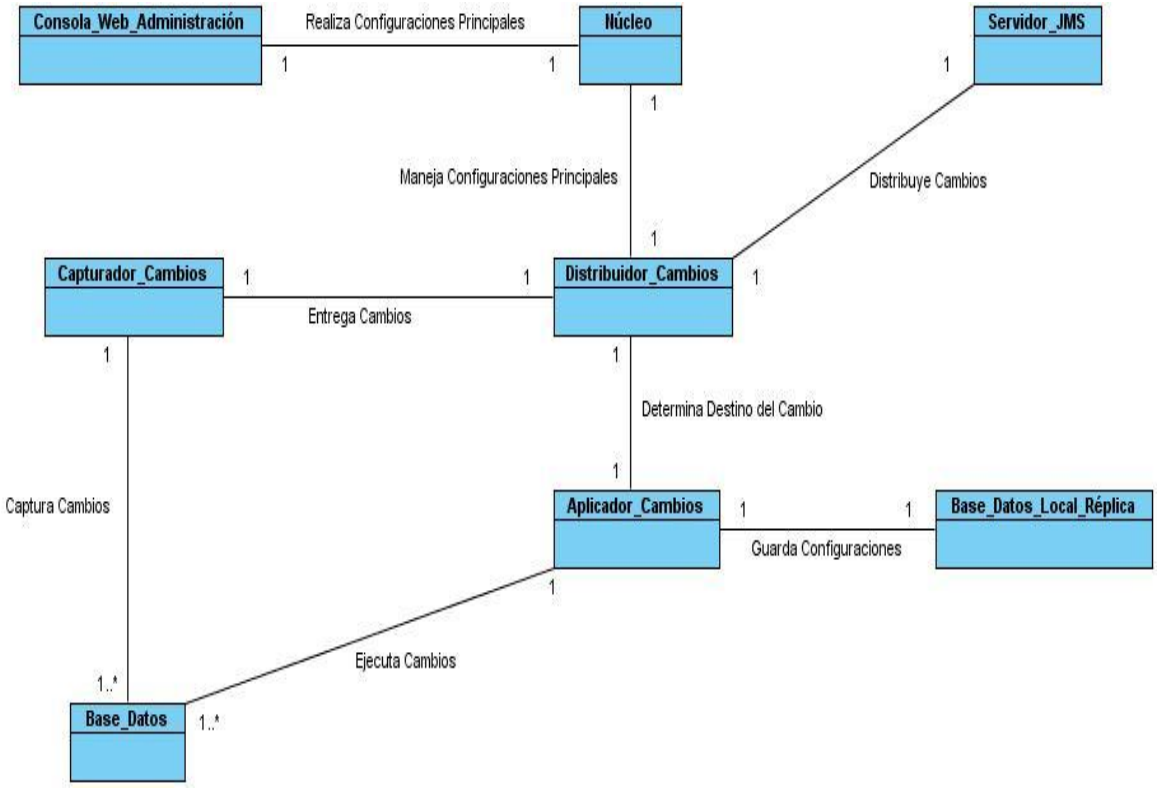


Figura 5. Modelo del Dominio.

2.4 Especificación de los Requisitos del Sistema

Los requisitos son condiciones o capacidades que necesita el usuario para resolver un problema o conseguir un objetivo determinado. Esta definición se extiende y se aplica a las condiciones que debe cumplir o poseer un sistema -o uno de sus componentes- , para satisfacer un contrato, una norma o una especificación.

2.4.1 Requisitos Funcionales

Los requisitos funcionales de un sistema describen su funcionalidad, los servicios que de él se esperan, o los que proveerá, entre ellos: sus entradas, salidas y excepciones.

RF1. Configurar la captura y envío de la información de forma diaria.
RF2. Configurar la captura y envío de la información de forma semanal.
RF3. Configurar la captura y envío de la información de forma mensual.
RF4. Ordenar la realización de la captura y envío de la información.

2.4.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Constituyen las características que hacen al producto atractivo, usable, rápido o confiable al usuario.

Requisitos no funcionales

1. *Apariencia o interfaz externa:* La administración y configuración de la réplica se realiza a través de una interfaz Web, por lo que es administrable vía remota usando un navegador.
2. *Usabilidad:* La aplicación informática debe garantizar un acceso fácil para el usuario contando con un interfaz que satisfaga sus necesidades.
3. *Soporte*: Se debe asegurar el soporte para los usuarios de manera que se puedan satisfacer sus necesidades a partir de mejoras, una vez puesta en marcha la aplicación. Para ello se crearán una serie de manuales de usuarios y videos tutoriales, y se mantendrá la asistencia a los usuarios.

4. *Seguridad:* Los nodos de replicación manejan credenciales entre ellos para verificar la autenticidad de los datos transferidos. El envío de datos se realiza utilizando protocolos de comunicación seguros como son HTTPS y SSL.
5. *Software:* Se requiere para el funcionamiento del sistema disponer de un servidor JMS, un servidor de base de datos PostgreSQL y navegador Mozilla 1.4 o superior. Soporta cualquier sistema operativo con la máquina virtual de Java funcionando en una versión 1.5 o superior y no necesita de un ambiente gráfico en el servidor para su funcionamiento.
6. *Hardware:* Se requiere para Windows XP Professional (SP1) una memoria de 64MB y 98MB de espacio en el disco duro. En la versión de Linux una memoria de 64MB y espacio en el disco de 58 MB.
7. *Disponibilidad:* Se debe garantizar el funcionamiento de la aplicación durante las 24 horas del día y los siete días de la semana, con el menor tiempo posible de recuperación de fallos. Se deben crear copias de respaldo periódicas que puedan restaurar el sistema en caso de fallo crítico o pérdida total de la información.
8. *Requisitos Legales:* El software desarrollado en la plataforma puede ser utilizado por otros proyectos de la UCI siempre que se mantengan las declaraciones de autor contenidas en el software. Se informe al líder del equipo de desarrollo el nombre del producto en el que se incorpora el software. Estas contribuciones se someten a lo dispuesto en la sección Aceptación de contribuciones. De esta obligación se exceptúan aquellos proyectos en los cuales las modificaciones realizadas tienen carácter confidencial.

2.5 Definición de los Casos de Uso del Sistema

Los casos de uso son descripciones de las funcionalidades del sistema y se utilizan para obtener información de cómo debe trabajar este.

2.5.1 Diagrama de Casos de Uso del Sistema

El diagrama de casos de uso del sistema sirve para especificar la comunicación y el comportamiento de un sistema, mediante su interacción con los usuarios y/u otros sistemas.

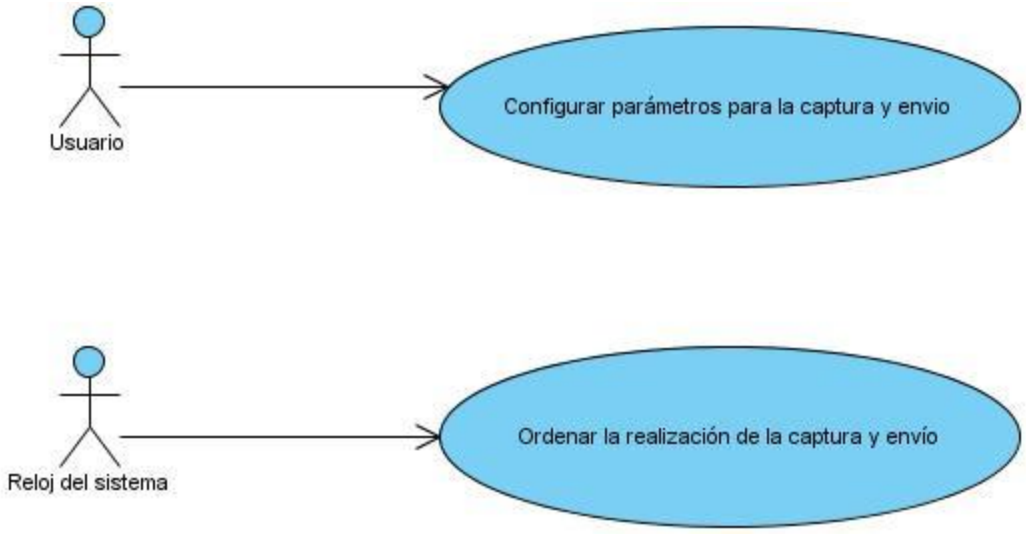


Figura 6. Diagrama de Caso de Uso del Sistema.

2.5.2 Descripción de los Casos de Uso del Sistema

Los requisitos funcionales definidos se agruparon en 2 casos de uso.

- Configurar parámetros para la captura y envío. (Ver Anexo 1)
- Ordenar la realización de la captura y envío. (Ver Anexo 2)

2.6 Conclusiones

Siguiendo los pasos que plantea UML y debido a que la metodología empleada no realiza el proceso de negocio, se definieron en este capítulo conceptos, que fueron relacionados mediante un diagrama de modelo del dominio. Se definieron 4 requisitos funcionales agrupados en 2 casos de uso. A demás de los requisitos no funcionales del sistema. También se sentaron las bases para las restantes fases del proceso de diseño e implementación, a través de la descripción de los casos de uso.

CAPÍTULO 3. DISEÑO DEL SISTEMA.

3.1 Introducción

En este capítulo se presenta el diagrama de clases del diseño que dará solución a los casos de uso identificados y descritos en el capítulo anterior y se realizarán los diagramas de interacción para los diferentes escenarios. Finalmente se presentará el diagrama de despliegue, con el objetivo de tener una idea más clara de cómo quedará distribuida la infraestructura una vez desplegada.

3.2 Descripción de la arquitectura del software de réplica Reko.

La arquitectura de software está definida por la IEEE 1471-2000 como: *"La organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución"* (16) .

Partiendo de esta definición, se concluye que cada sistema de software contará con su propia arquitectura, cuya estructura dependerá de las características específicas de cada proyecto. Inicialmente esta se define como candidata en la Fase de Inicio, pasando luego por un refinamiento en la Fase de Elaboración donde se define como Línea Base de la Arquitectura.

La arquitectura está dada por seis categorías diferentes de estilos arquitectónicos, los cuales versan en dependencia de los requerimientos del software. Una de estas categorías está conformada por los Estilos Arquitectónicos de Llamada y Retorno:

- Modelo Vista Controlador (MVC).
- Arquitecturas en Capas.
- Arquitecturas Basadas en Componentes.

La arquitectura que tiene definida el Replicador Reko es Basada en Componentes, pues sus partes encapsulan un conjunto de comportamientos que pueden ser reemplazados por otros.

El desarrollo basado en componentes permite alcanzar un mayor nivel de reutilización de software, permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados y simplifica el mantenimiento del sistema, entre otras cuestiones. Un componente típicamente es una "*unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio*" (17).

Los principales componentes presentes en el software son:

Capturador de cambios: Encargado de capturar los cambios que se realizan en la base de datos y entregarlos al distribuidor.

Distribuidor: Determina el destino de cada cambio realizado en la base de datos, los envía y se responsabiliza de su llegada.

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

Administración: Permite realizar las configuraciones principales del software como el registro de nodos, configuración de las tablas a replicar y el monitoreo del funcionamiento.

Independientemente de lo antes expuesto, el componente **Administración** responde a un modelo multicapas, donde cada capa tiene funcionalidades y objetivos precisos, así su implementación se encuentra desacoplada de la programación de cualquier otra y la comunicación con una capa inferior ocurre a través de interfaces. Además de estar separadas lógica y estructuralmente, las capas se encuentran separadas de manera física.

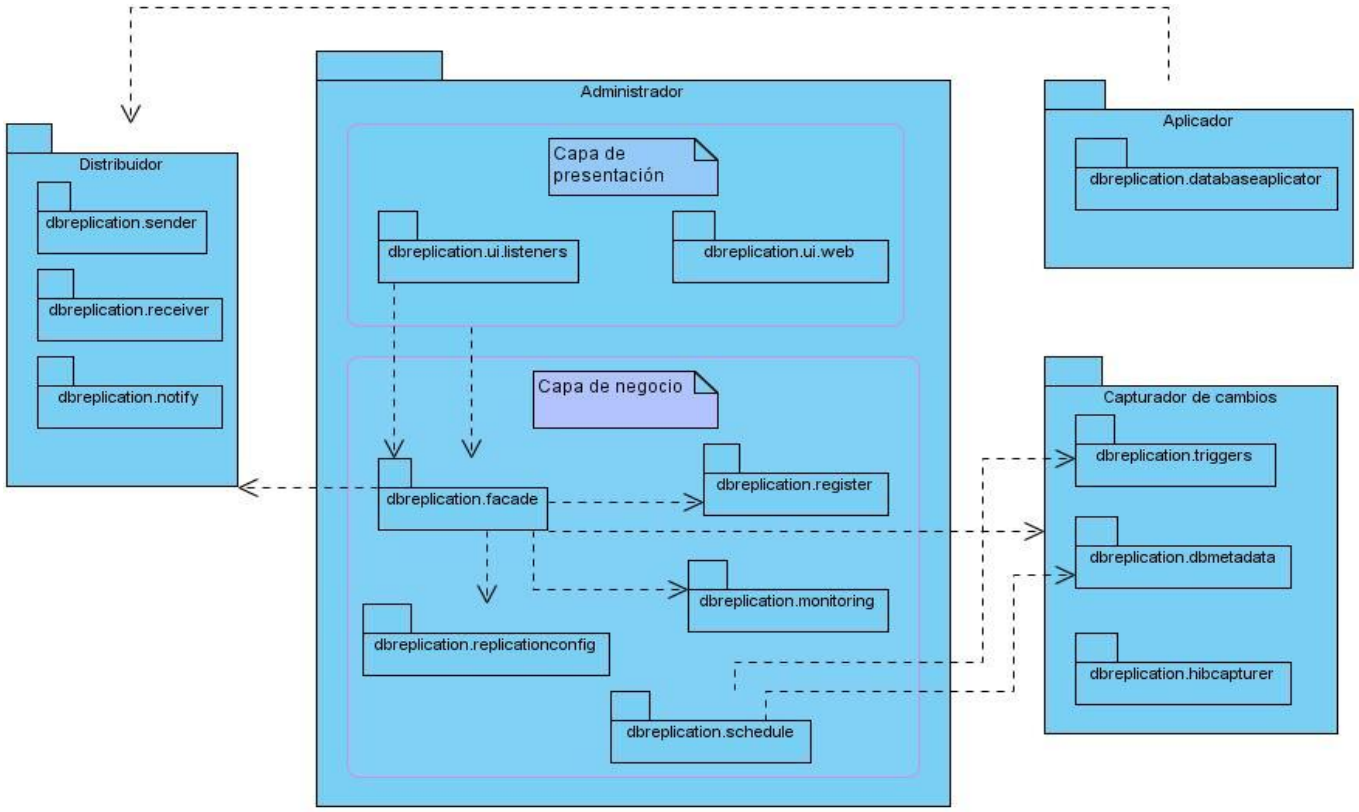


Figura 7. Arquitectura de Reko Basada en Componentes.

3.3 Patrones de Diseño Utilizados

3.3.1 Patrón Creador

Este patrón es el encargado de que una clase B cree una instancia de una clase A, siempre que:

- La clase B contenga a la clase A.
- B sea una agregación (o composición) de A.
- B almacene a A.
- B tenga los datos de inicialización de A (datos que requiere su constructor).
- B use a A.

Este patrón se utilizó en la implementación de las clases de la aplicación, ejemplo: en la clase Gestionar Tarea.

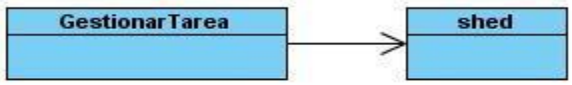


Figura 8. Ejemplo de aplicación del patrón Creador.

3.3.2 Patrón Controlador

Es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. Se utiliza en la clase SaveMainConfigController.

Figura 9. Ejemplo de aplicación del patrón Controlador.

3.4 Modelo de Diseño

Es una abstracción del Modelo de Implementación y su código fuente, el cual se emplea, fundamentalmente, para representar y documentar su diseño. Es usado como entrada esencial en las actividades relacionadas con la implementación. Representa a los casos de uso en el dominio de la solución.

3.4.1 Diagramas de Clases del Diseño

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema. Muestra sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, cuando se crea el diseño conceptual de la información que se manejará en el sistema, conjuntamente con los componentes que se encargarán del funcionamiento y la relaciones entre unos y otros. A continuación se muestran los diagramas referentes al diseño de la funcionalidad.

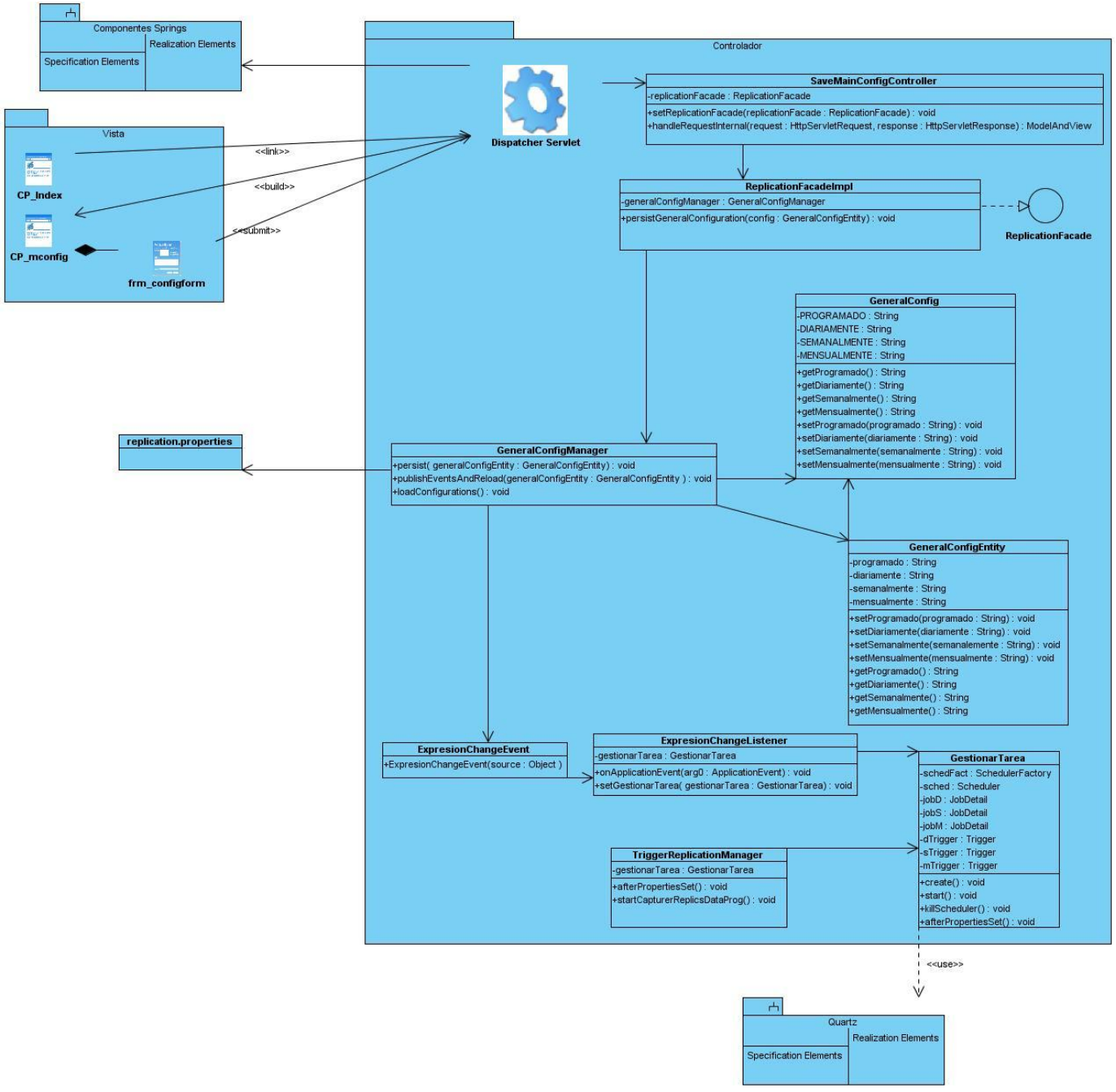


Figura 10. Diagrama de Clase del Diseño para el CU Configurar parámetros para la captura y envío.

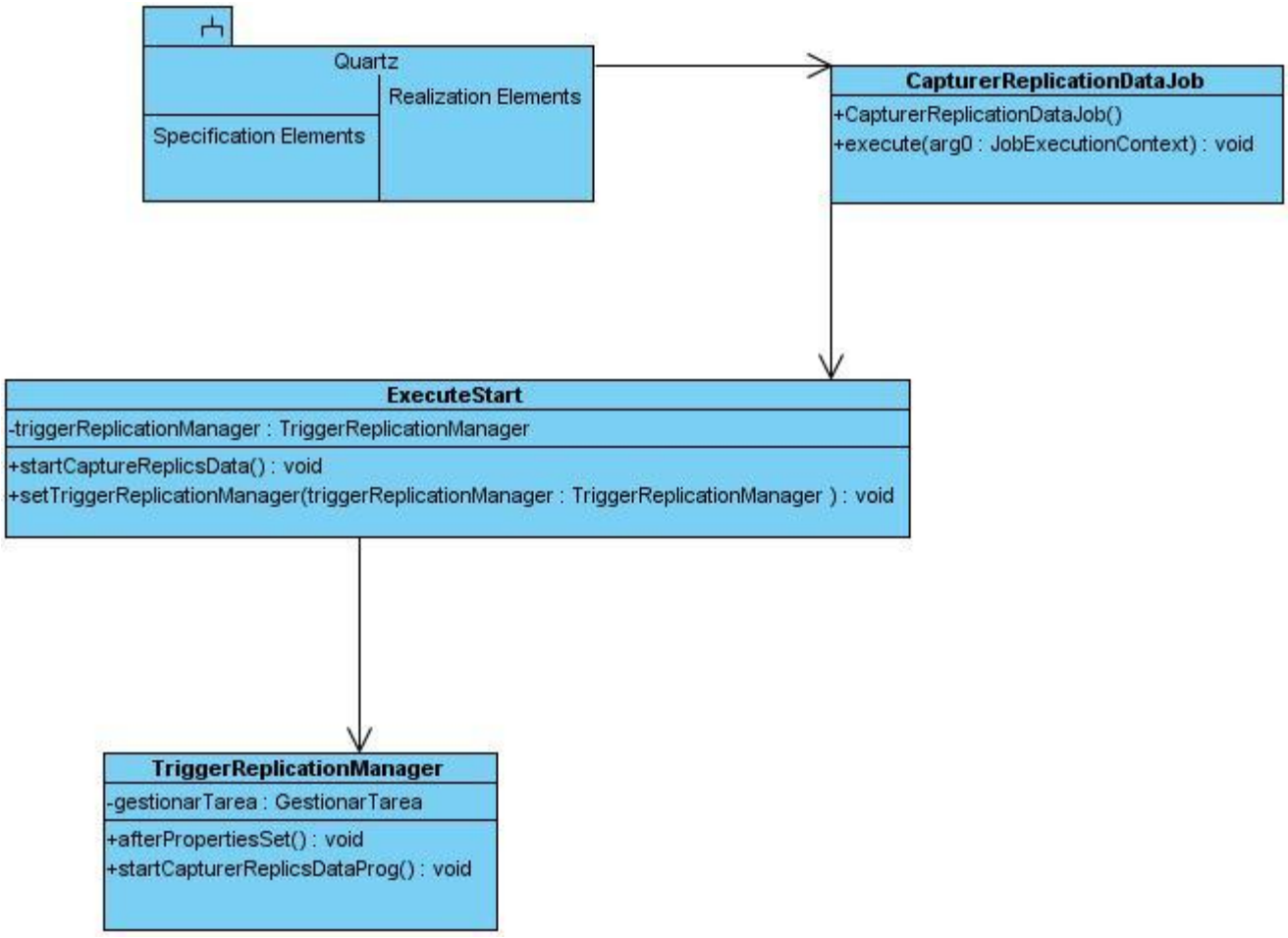


Figura 11. Diagrama de Clase del Diseño para el Ordenar la realización de la captura y envío.

3.4.2 Diagramas de Secuencia

Un diagrama de secuencia muestra las interacciones entre objetos ordenadas en secuencia temporal. Muestra los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los objetos para llevar a cabo la funcionalidad descrita por el escenario.

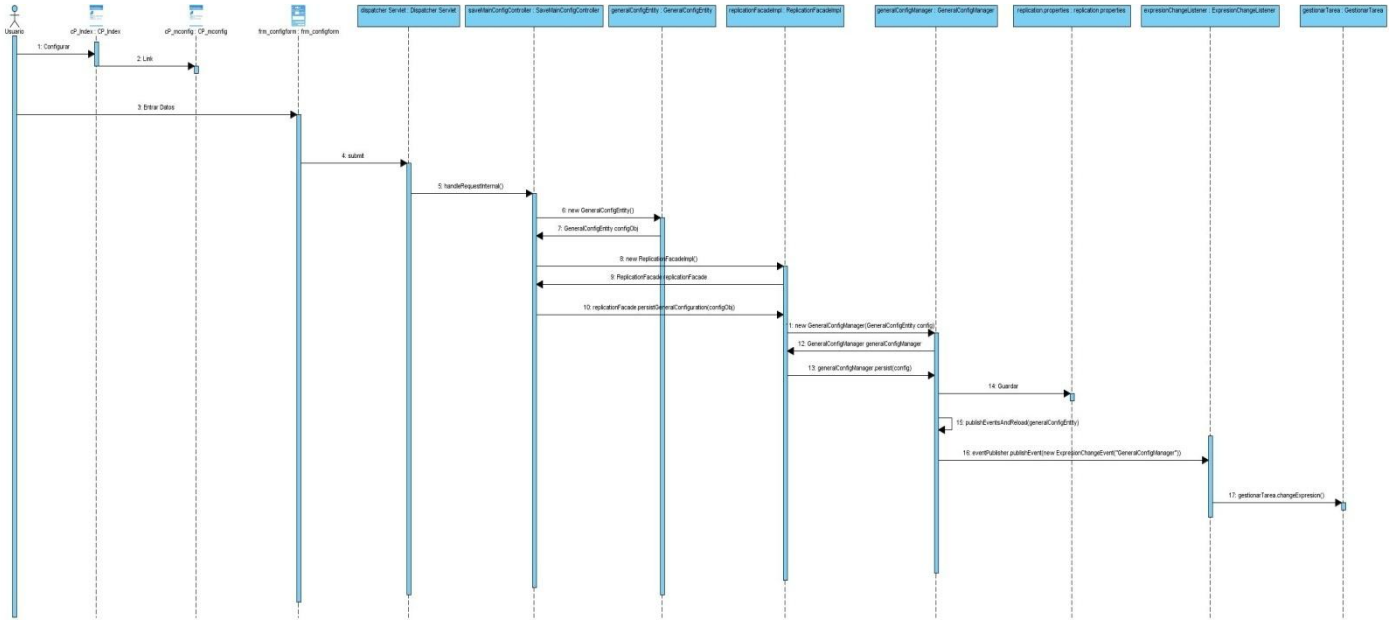


Figura 12. Diagrama de Secuencia para el CU Configurar parámetros para la captura y envío.

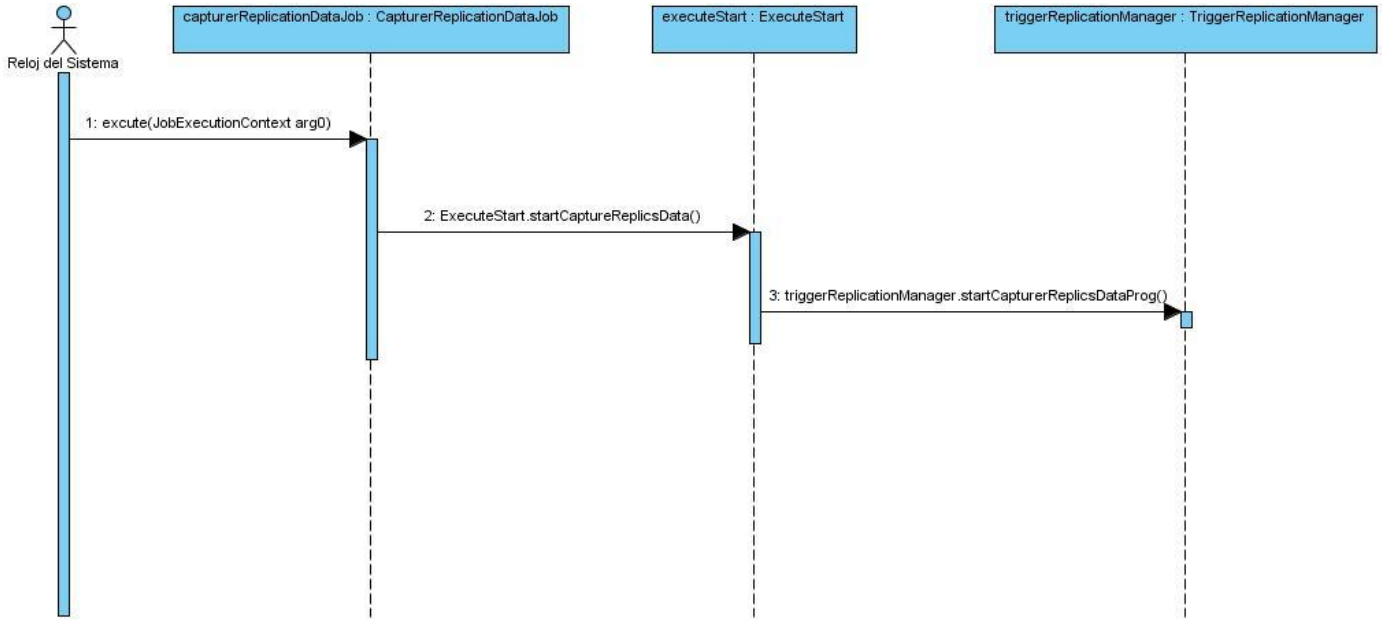


Figura 13. Diagrama de Secuencia del el CU Ordenar la realización de la captura y envío.

3.4.3 Modelo de Despliegue

El modelo de despliegue es utilizado para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos.

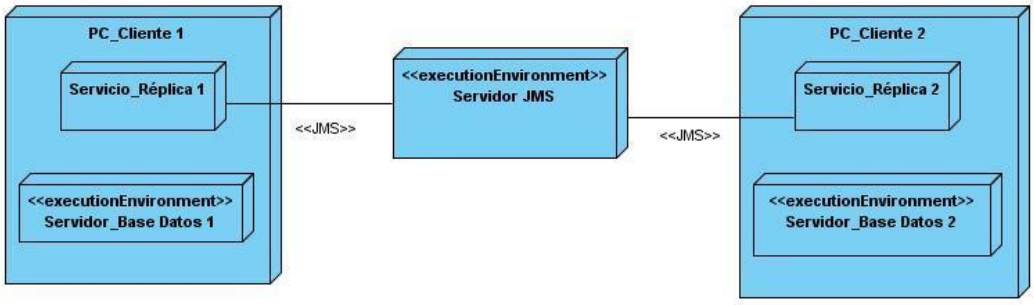


Figura 14. Diagrama de Despliegue.

3.5 Conclusiones

Se presentó la arquitectura del Replicador Reko, Arquitectura basada en Componentes. Para diseñar la funcionalidad se utilizaron diferentes patrones de diseño como el Creador, Alta cohesión, Bajo Acoplamiento y Controlador. Se mostraron los Diagramas de Clases del diseño, Secuencia y Despliegue.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA.

4.1 Introducción

En este capítulo se describe la implementación del sistema, se muestran fragmentos de código de los principales métodos implementados, además de los diagramas de componentes así como las pruebas realizadas a la funcionalidad.

4.2 Implementación

Diagrama de Componentes

Los diagramas de componentes muestran las organizaciones y dependencias lógicas entre componentes de software, sean estos de código fuente, binarios, archivos, bibliotecas cargadas dinámicamente o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. A continuación se muestra el diagrama de componente de la aplicación desarrollada:

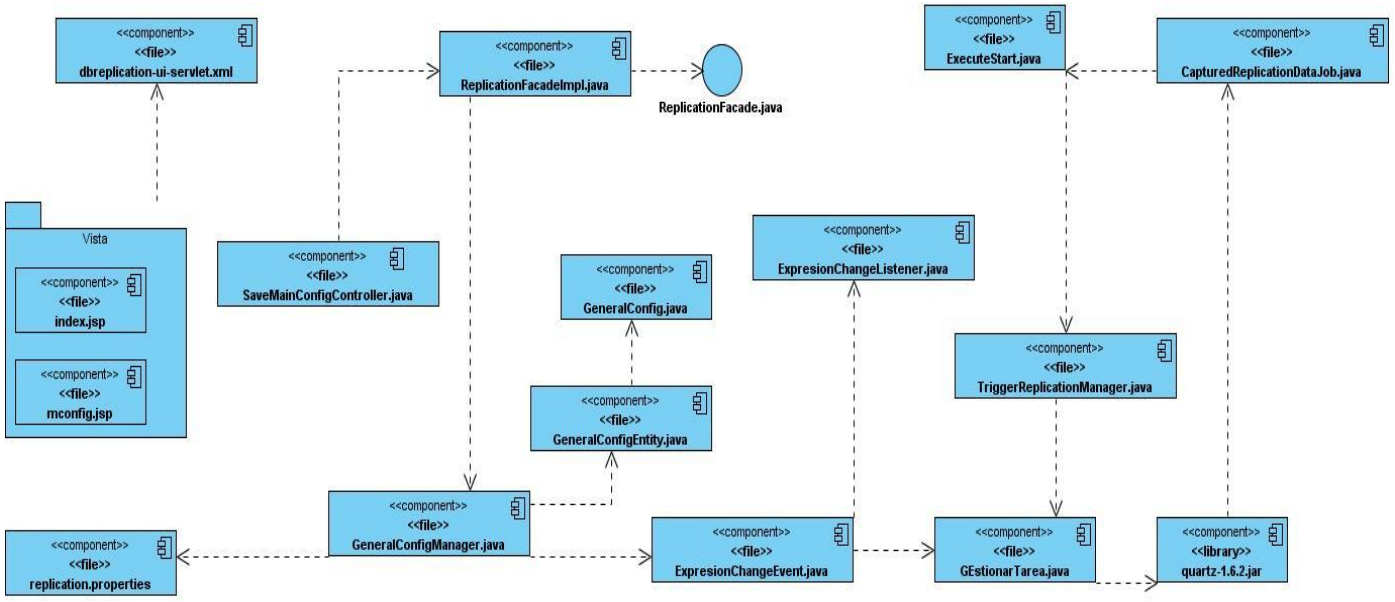


Figura 15. Diagrama de Componentes de la funcionalidad.

4.3 Interfaces de la Aplicación

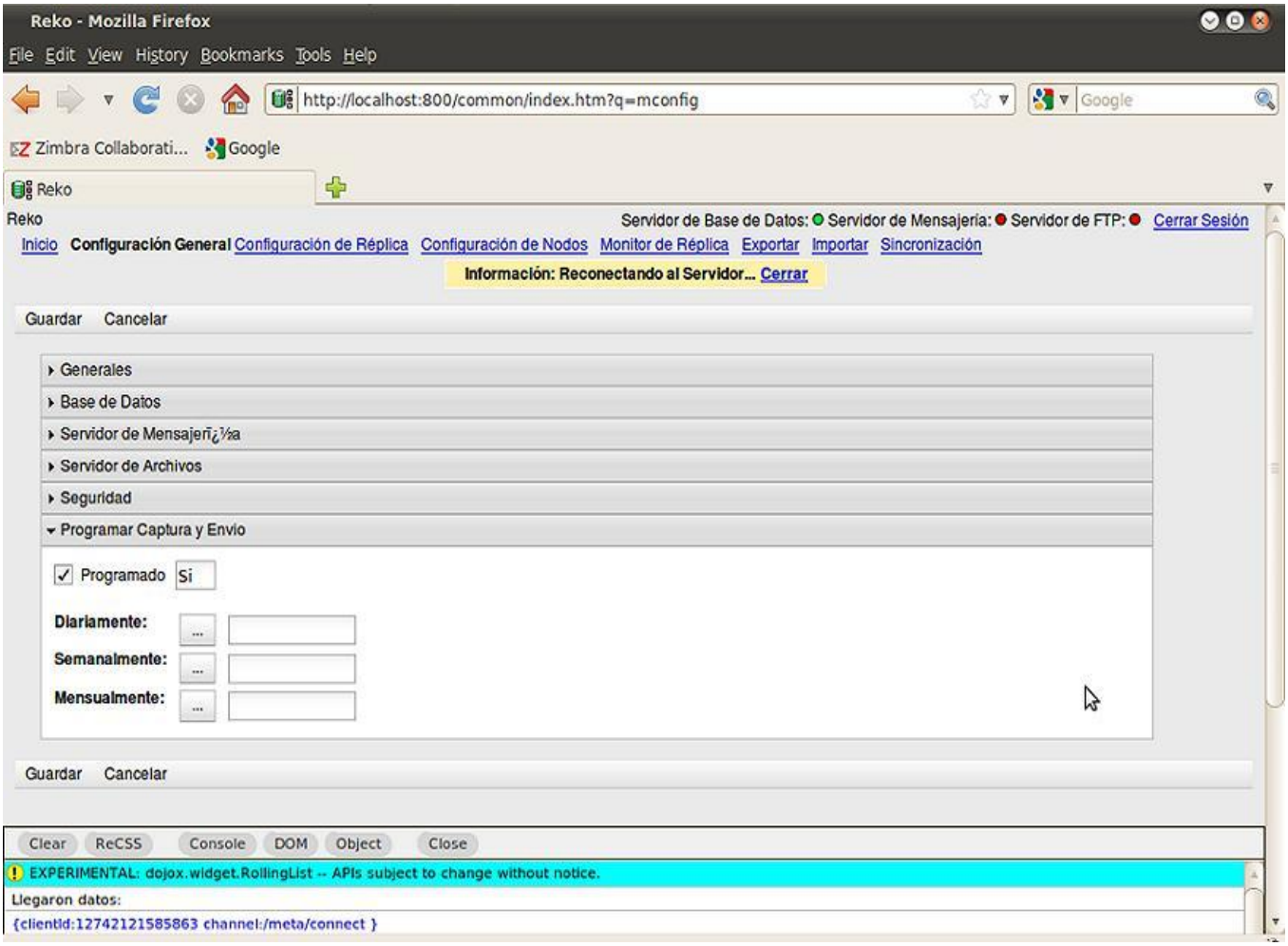


Figura 16. Programar Captura y Envío.

Interfaz para la programación de forma Diaria. (Ver Anexo 3)

Interfaz para la programación de forma Semanal. (Ver Anexo 4)

Interfaz para la programación de forma Mensual. (Ver Anexo 5)

Interfaz de la configuración de réplica. (Ver Anexo 6)

4.4 Código de la principal clase (Gestionar Tarea) para la implementación de la funcionalidad.

Este fragmento de código pertenece a la clase Gestionar Tarea. El método create () crea el objeto Scheduler el cual es el encargado de registrar las tareas a programar. El método start () inicia la programación de las tareas. El método changeExpresion () reprograma las tareas en caso de que haya existido un cambio en la programación. El método killSchedule () borra las tareas que fueron programadas. El método afterPropertiesSet () el cual está definido en la interfaz org.springframework.beans.factory.InitializingBean y se ejecuta una vez que las propiedades del bean se hayan establecido.

```
package cu.uci.dbreplication.schedule;

import java.text.ParseException;
import org.quartz.CronTrigger;
import org.quartz.JobDetail;
import org.quartz.SchedulerException;
import org.quartz.SchedulerFactory;
import org.quartz.Trigger;
import org.quartz.impl.StdSchedulerFactory;
import org.springframework.beans.factory.InitializingBean;
import cu.uci.dbreplication.triggers.TriggerReplicationManager;
import cu.uci.dbreplication.utils.config.GeneralConfig;

public class GestionarTarea implements InitializingBean {
    private TriggerReplicationManager triggerReplicationManager;
    SchedulerFactory schedFact;
    org.quartz.Scheduler sched;
    JobDetail jobD;
    JobDetail jobS;
    JobDetail jobM;
    Trigger dTrigger;
    Trigger sTrigger;
    Trigger mTrigger;

    public void create() {
        try {
            schedFact = new StdSchedulerFactory();
            sched = schedFact.getScheduler();
            jobD = new JobDetail("myjobD", null,
                    CapturerReplicationDataJob.class);
            jobS = new JobDetail("myjobS", null,
                    CapturerReplicationDataJob.class);
            jobM = new JobDetail("myjobM", null,
                    CapturerReplicationDataJob.class);
```

```
            if (Expresion.isDiariamenteValid()) {
                dTrigger = new CronTrigger("dTrigger", null, Expresion
                        .getExprecionDiariamente());
                sched.scheduleJob(jobD, dTrigger);
            }
            if (Expresion.isSemanalmenteValid()) {
                sTrigger = new CronTrigger("sTrigger", null, Expresion
                        .getExpresionSemanalmente());
                sched.scheduleJob(jobS, sTrigger);
            }
            if (Expresion.isMensualmenteValid()) {
                mTrigger = new CronTrigger("mTrigger", null, Expresion
                        .getExpresionMensualmente());
                sched.scheduleJob(jobM, mTrigger);
            }
        } catch (Exception e) {
        }
    }

    public void start() throws SchedulerException {
        sched.start();
    }

    public void changeExpresion() throws SchedulerException, ParseException {
        if (GeneralConfig.getProgramado().equalsIgnoreCase("si")) {
            triggerReplicationManager.killTimer();
            if (Expresion.isDiariamenteValid()) {
                dTrigger = new CronTrigger("dTrigger", null, Expresion
                        .getExprecionDiariamente());
                sched.deleteJob("myjobD", null);
                jobD = new JobDetail("myjobD", null,
                        CapturerReplicationDataJob.class);
                sched.scheduleJob(jobD, dTrigger);
```

```
            }
            if (Expresion.isSemanalmenteValid()) {
                sTrigger = new CronTrigger("sTrigger", null, Expresion
                        .getExpresionSemanalmente());
                sched.deleteJob("myjobS", null);
                jobS = new JobDetail("myjobS", null,
                        CapturerReplicationDataJob.class);
                sched.scheduleJob(jobS, sTrigger);
            }
            if (Expresion.isMensualmenteValid()) {
                mTrigger = new CronTrigger("mTrigger", null, Expresion
                        .getExpresionMensualmente());
                sched.deleteJob("myjobM", null);
                jobM = new JobDetail("myjobM", null,
                        CapturerReplicationDataJob.class);
                sched.scheduleJob(jobM, mTrigger);
            }
            start();
        }else {
            killSchedule();

          triggerReplicationManager.startTimer();
    }
}

public void killSchedule() throws SchedulerException
{
    sched.deleteJob("myjobM", null);
    sched.deleteJob("myjobS", null);
    sched.deleteJob("myjobD", null);
}


public void afterPropertiesSet() throws Exception {
    create();
}
```

4.5 Pruebas

Durante la implementación, se le realizaron a la funcionalidad diferentes comprobaciones con el objetivo de probar su correcto funcionamiento; se intentó encontrar fallos, corregirlos y así lograr que las funciones sean operativas, que las entradas se acepten de forma adecuada, se produzca una salida correcta y que la integridad de la información externa se mantenga.

Las comprobaciones realizadas se centraron en lo que se espera del sistema. Se chequeó de forma periódica su comportamiento durante la ejecución. Además se verificó sistemáticamente que cada una de las funcionalidades necesarias para el cumplimiento de los requerimientos del sistema realizaran lo esperado, para lograr de ese modo satisfacer las necesidades del cliente, comprobándose además el funcionamiento correcto de las validaciones ante entradas de datos erróneos.

Se trataron las validaciones exhaustivamente debido a que el sistema informático fue desarrollado con el propósito fundamental de llevar a cabo actualizaciones reales, por lo que era necesario lograr la entrada correcta de datos para su posterior uso. Para lograr lo antes mencionado se hizo uso de las diferentes técnicas de validación, imponiéndose el no dejar campos vacíos o sin seleccionar, así como introducir datos correctos donde se requerían.

4.5.1 Configuración del entorno de Prueba

La configuración del entorno donde se vayan a ejecutar las diferentes pruebas que se realizan a un software es un aspecto muy importante dentro del proceso de pruebas, pues si no se analizan bien los recursos de software y hardware que necesita el producto que se está construyendo, a la hora de probarlo se prescindirá de los elementos necesarios para la ejecución de un proceso de pruebas exitoso.
Por ello se tuvo en cuenta algunos requerimientos de hardware y de software que hicieron posible un mejor desarrollo de las pruebas, en aras de que se lograran minimizar los errores de la aplicación en desarrollo. Los requerimientos que se consideraron necesarios para las pruebas se referencian a continuación:

Requerimientos de software:

· PC con Windows XP o superior.

· PC con Linux (Ubuntu).

· Máquina Virtual de Java 1.5.

- Servidor JMS.

- Servidor de base de datos PostgreSQL.

Requerimientos de hardware:

· PC con Microprocesador Pentium IV o superior.

· Se requiere para Windows XP Professional (SP1) una memoria de 64MB y 98MB.

· Linux una memoria de 64MB y espacio en el disco de 58 MB

4.5.2 Tipo de prueba aplicada

Prueba de Sistema

Son las pruebas que se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados. En un ciclo iterativo estas pruebas ocurren más temprano, tan pronto como subconjuntos bien formados de comportamiento de caso de uso son implementados.

- Prueba de Rendimiento**:** Está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado.

4.5.3 Método de Prueba

Pruebas de Caja Negra

Las pruebas de caja negra son aquellas 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.

Diseño de las pruebas del Replicador Reko

Caso de Prueba: Configurar parámetros para la captura y envío.

Nombre de la Sección	**Escenarios de la Sección**	**Descripción de la Funcionalidad**	**Flujo Central**
SC: 1 Configurar parámetros para la captura y envío.	ES1.1: Seleccionar la Opción Programar.	El usuario decide programar la captura y envío de la información y selecciona la Opción Programar captura y envío en el menú de Configuración general. El sistema muestra la interfaz con los parámetros a seleccionar.	Selecciona el Combo box de la opción Programar.
SC: 1 Configurar parámetros para la captura y envío.	ES: 1.2 Seleccionar la Opción Programar Diariamente.	El usuario decide programar la captura y envío de la información de forma Diaria .El sistema muestra 3 formas para seleccionar una. Selecciona del menú la opción Diariamente. Se muestra una interfaz con los datos de la hora y los minutos que desea insertar. El usuario lo inserta y Presiona el botón Guardar.	Selecciona del menú la opción Diariamente.
SC: 1 Configurar parámetros para la captura y envío.	ES: 1.3 Seleccionar la Opción Programar Semanalmente.	El usuario decide programar la captura y envío de la información de forma Semanal. El sistema muestra 3 formas para seleccionar una. Selecciona del	Selecciona del menú la opción Semanalmente.

		menú la opción Semanalmente. Se muestra una interfaz con los días, la hora y los minutos .El usuario selecciona el/los día(s) así como la hora y los minutos que desea insertar. Presiona el botón Guardar.	
SC: 1 Configurar parámetros para la captura y envío.	ES: 1.4 Seleccionar la Opción Programar Mensualmente.	El usuario decide programar la captura y envío de la información de forma Mensual. El sistema muestra 3 formas para seleccionar una. Selecciona del menú la opción Mensualmente. Se muestra una interfaz con meses, los días, la hora y los minutos .El usuario selecciona el /los meses, el /los día(s) así como la hora y los minutos que desea insertar. Presiona el botón Guardar.	Selecciona del menú la opción Mensualmente.

El diseño de las pruebas por cada sección definida anteriormente se encuentra (Ver Anexo 7).

No conformidades detectadas.

Elemento	Nº	No conformidad	Aspecto Correspondiente	Etapa de Detección	Signif	No Signif
Interfaz 1	1	No se quedaba marcado la opción	Cuando se trataba de programar la captura y el envío	Etapa de diseño y Aplicación de Pruebas.	X	

		Programar captura y envío				
Interfaz 2	2	No se lograba guardar la configuración de la hora y los minutos de forma Diaria	Cuando se insertaban las horas y los minutos.	Etapa de diseño y Aplicación de Pruebas.	X	
Interfaz 3	3	No se podía escoger 2 o más meses al mismo tiempo en la configuración de forma Mensual	Cuando se intentaba seleccionar varios meses a la vez	Etapa de diseño y Aplicación de Pruebas.	X	
Interfaz 4	4	No se enviaban los datos del formulario en la parte referente a los campos de captura y envío.	Cuando se oprimía el botón Guardar	Implementación	X	
Interfaz 5	5	No se actualizaba el fichero cuando se recogían los datos de los horarios.	Cuando se oprimía el botón Guardar.	Implementación	X	
Interfaz 6	6	Se continuaba	Se realizaba la	Implementación	X	

		replicando los datos cada 20 segundos a pesar de tener programado un horario.	réplica en el horario establecido la primera vez y luego continuaba replicando cada 20 segundos.	y Prueba		

4.6 Conclusiones

Se logró implementar la funcionalidad con éxito. Se le realizaron pruebas de caja negra al sistema detectándose 6 no conformidades en los 3 casos de prueba que se le aplicaron .Las no conformidades encontradas fueron registradas y solucionadas para un mejor funcionamiento del software.

CONCLUSIONES

- Se diseñó la funcionalidad que permite definir horarios de forma Diaria, Semanal y Mensual para la captura y envío de la información en el Replicador Reko.
- Se implementó la funcionalidad que permite definir horarios de forma Diaria, Semanal y Mensual para la captura y envío de la información en el Replicador Reko.
- Se validó la funcionalidad realizándose pruebas de caja negra, detectándose solo 5 NO conformidades que fueron solucionadas.

RECOMENDACIONES

- Integrar la funcionalidad a la versión actual del software y que sea liberado por calidad.
- Continuar con el desarrollo del Replicador Reko, adicionando nuevas funcionalidades como el envío no solo de datos, sino de archivos de mayor tamaño (ficheros, imágenes).

REFERENCIAS BIBLIOGRÁFICAS

1. **Hernández Monroy, René Amilcar.** Lenguaje XML como solución a las bases de datos y su replicación. *Biblioteca Central de la Universidad de San Carlos de Guatemala.* [Online] noviembre 2005. [Cited: marzo 2, 2010.] http://biblioteca.usac.edu.gt/tesis/08/08_6343.pdf..

2. **García Pérez, Carlos.** Adictos al trabajo. [En línea] [Citado el: 2 de marzo de 2010.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=quartz.

3. Quartz Enterprise Job Scheduler. [En línea] [Citado el: 2 de marzo de 2010.] http://www.quartz-scheduler.org/.

4. versiontracker. [En línea] [Citado el: 3 de marzo de 2010.] http://www.versiontracker.com/dyn/moreinfo/win/56443&vid=10788682&mode=info..

5. Fillgap. [En línea] [Citado el: 3 de marzo de 2010.] http://www.fillgap.com.mx/fillgap2004/ls.htm#int.

6. Karen Ware.com. [En línea] [Citado el: 3 de marzo de 2010.] http://www.karenware.com/powertools/ptreplicator.asp.

7. **Pimentel, Luis Alberto.** Artículo-Documento-Reko. *forge.* [En línea] [Citado el: 4 de marzo de 2010.] http://forge.uci.cu.

8. **Weitzenfeld, Alfredo.** Ingeniería de software orientada a objetos. [En línea] octubre de 2002. [Citado el: 4 de marzo de 2010.] http://www.deeplunar.org/fusm/ingenieria%20de%20software/.

9. CrisHK. *Asesorías y consultorías.* [En línea] [Citado el: 4 de marzo de 2010.] http://www.cris.hk/?p=services/home/ECLIPSE-Java-ECLIPSE-C-ECLIPSE-PHP&type=eclipse.

10. **Saavedra López, Esteban.** Scribd. *Grails: Framework para el desarrollo de aplicaciones Web.* [En línea] [Citado el: 5 de marzo de 2010.] http://www.scribd.com/doc/15020671/Grails-Framework-para-el-desarrollo-de-aplicaciones-Web.

11. **Lago Bagüés, Ramiro.** Proactiva Calidad. [En línea] enero de 2008. [Citado el: 5 de marzo de 2010.] http://www.proactiva-calidad.com/java/spring/introduccionSpring.html.

12. **Aguilar, Vicente y Suau, Pablo.** xtec.ca. *MySqL VS Postgres.* [En línea] 18 de agosto de 2000. http://www.xtec.cat/formaciotic/dvdformacio/materials/td116/calaix/m3/MySQL%20vs_%20PostgreSQL.htm.

13. altacracks. [Online] [Cited: marzo 10, 2010.] http://www.altacracks.com/programas-gratis/visual-paradigm-for-uml-professional-edition..

14. **Ricardo, Balduino.** Introduction to OpenUP (Open Unified Process). [Online] agosto 2007. [Cited: marzo 12, 2010.] http://www.eclipse.org/epf/general/OpenUP.pdf.

15. Grupo de Usuarios Microsoft. [Online] [Cited: marzo 12, 2010.] http://www.mug.org.ar/.

16. **Brey, Gustavo Andrés y Passerini, Nicolas.** Arquitectura de Software. [En línea] [Citado el: 2 de abril de 2010.] http://apit.wdfiles.com/local--files/start/02_apit_arquitectura_de_software.pdf.

17. **Moreno Navarro, Juan José.** Introducción al Software basado en Componentes. [En línea] [Citado el: 12 de abril de 2010.]

18. Diccionario de computación e informática. . [Online] [Cited: abril 13, 2010.] http://www.sitiosespana.com/paginas/diccionario_informatica/c.htm..

BIBLIOGRAFÍA

1. **Hernández Monroy, René Amilcar.** Lenguaje XML como solución a las bases de datos y su replicación. *Biblioteca Central de la Universidad de San Carlos de Guatemala.* [Online] noviembre 2005. [Cited: marzo 2, 2010.] http://biblioteca.usac.edu.gt/tesis/08/08_6343.pdf..

2. **García Pérez, Carlos.** Adictos al trabajo. [En línea] [Citado el: 2 de marzo de 2010.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=quartz.

3. Quartz Enterprise Job Scheduler. [En línea] [Citado el: 2 de marzo de 2010.] http://www.quartz-scheduler.org/.

4. versiontracker. [En línea] [Citado el: 3 de marzo de 2010.] http://www.versiontracker.com/dyn/moreinfo/win/56443&vid=10788682&mode=info..

5. Fillgap. [En línea] [Citado el: 3 de marzo de 2010.] http://www.fillgap.com.mx/fillgap2004/ls.htm#int.

6. Karen Ware.com. [En línea] [Citado el: 3 de marzo de 2010.] http://www.karenware.com/powertools/ptreplicator.asp.

7. **Pimentel, Luis Alberto.** Artículo-Documento-Reko. *forge.* [En línea] [Citado el: 4 de marzo de 2010.] http://forge.uci.cu.

8. **Weitzenfeld, Alfredo.** Ingeniería de software orientada a objetos. [En línea] octubre de 2002. [Citado el: 4 de marzo de 2010.] http://www.deeplunar.org/fusm/ingenieria%20de%20software/.

9. CrisHK. *Asesorías y consultorías.* [En línea] [Citado el: 4 de marzo de 2010.] http://www.cris.hk/?p=services/home/ECLIPSE-Java-ECLIPSE-C-ECLIPSE-PHP&type=eclipse.

10. **Saavedra López, Esteban.** Scribd. *Grails: Framework para el desarrollo de aplicaciones Web.* [En línea] [Citado el: 5 de marzo de 2010.] http://www.scribd.com/doc/15020671/Grails-Framework-para-el-desarrollo-de-aplicaciones-Web.

11. **Lago Bagüés, Ramiro.** Proactiva Calidad. [En línea] enero de 2008. [Citado el: 5 de marzo de 2010.] http://www.proactiva-calidad.com/java/spring/introduccionSpring.html.

12. **Aguilar, Vicente y Suau, Pablo.** xtec.ca. *MySqL VS Postgres.* [En línea] 18 de agosto de 2000. http://www.xtec.cat/formaciotic/dvdformacio/materials/td116/calaix/m3/MySQL%20vs_%20PostgreSQL.htm.

13. altacracks. [Online] [Cited: marzo 10, 2010.] http://www.altacracks.com/programas-gratis/visual-paradigm-for-uml-professional-edition..

14. **Ricardo, Balduino.** Introduction to OpenUP (Open Unified Process). [Online] agosto 2007. [Cited: marzo 12, 2010.] http://www.eclipse.org/epf/general/OpenUP.pdf.

15. Grupo de Usuarios Microsoft. [Online] [Cited: marzo 12, 2010.] http://www.mug.org.ar/.

16. **Brey, Gustavo Andrés y Passerini, Nicolas.** Arquitectura de Software. [En línea] [Citado el: 2 de abril de 2010.] http://apit.wdfiles.com/local--files/start/02_apit_arquitectura_de_software.pdf.

17. **Moreno Navarro, Juan José.** Introducción al Software basado en Componentes. [En línea] [Citado el: 12 de abril de 2010.]

18. Diccionario de computación e informática. . [Online] [Cited: abril 13, 2010.] http://www.sitiosespana.com/paginas/diccionario_informatica/c.htm.

ANEXOS

Anexo 1: Descripción del Caso de Uso Configurar parámetros para la captura y envío.

Caso de Uso:	• Configurar parámetros para la captura y envío.
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario selecciona Configurar captura y envío en el menú Configuraciones Generales. El sistema muestra en la interfaz las opciones para la programación de los horarios de captura y envío de la información. El caso de uso finaliza cuando el usuario termina de insertar los parámetros**.**
Precondiciones:	
Referencias	RF1. RF2. RF3
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario decide Configurar la Captura y Envío	2. El sistema muestra la interfaz con las opciones para la configuración.
3. El usuario puede seleccionar una de las siguientes opciones. • Diariamente • Semanalmente • Mensualmente	4. El sistema en dependencia de la opción que solicita realizar el Usuario, ejecuta las siguientes acciones: a) Si decide realizar la configuración Diariamente. ¨Ver Sección Diariamente¨. b) Si decide realizar la configuración Semanalmente. ¨Ver Sección Semanalmente.¨ c) Si decide realizar la configuración

	Mensualmente .¨Ver Sección Mensualmente¨.
Sección "Diariamente"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario decide realizar la captura y envío de forma diaria.	2. El sistema muestra una interfaz para introducir la hora y los minutos de la captura y envío de los datos.
3. El usuario introduce los horarios de captura y envío.	4. El sistema valida si los datos introducidos son correctos.
5. Se Acepta la información.	
	6. El sistema guarda la configuración. Finaliza el caso de uso. .
Flujo alterno de la sección "Diariamente"	
	4.1 Si los datos introducidos nos son correctos el sistema muestra un mensaje.
Prototipo de Interfaz	
Poscondiciones	La configuración queda guardada.

Sección "Semanalmente"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario decide realizar la captura y envío de forma semanal.	2. El sistema muestra una interfaz para seleccionar El/los días de la semana e introducir la hora y los minutos de la captura y envío.

3. El usuario selecciona El/los días de la semana e introduce los horarios en el que desea se realice la captura y envío.	4. El sistema valida si los datos introducidos son correctos.
5. Se Acepta la información.	
	6. Guarda la configuración. Finaliza el caso de uso.
Flujo alterno de la sección "Semanalmente"	
	4.1- Si los datos introducidos no son correctos el sistema muestra un mensaje.
Prototipo de Interfaz	
Poscondiciones	La configuración queda guardada.

Sección "Mensualmente"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario decide realizar la captura y envío de forma mensual.	2. El sistema muestra una interfaz con los días y meses para seleccionar e introducir la hora y los minutos para la captura y envío.
3. El usuario selecciona el día, el mes e introduce los horarios en que va a capturar y enviar la información.	4. El sistema valida si los datos introducidos son correctos.
5. Se Acepta la información.	
	6. Guarda la configuración. Finaliza el caso de uso.
Flujo alterno de la sección "Mensualmente"	
	4.1- Si los datos introducidos no son correctos el sistema muestra un mensaje.
Prototipo de Interfaz	
Poscondiciones	La configuración queda guardada.

Anexo 2: Descripción del Caso de Uso Ordenar la realización de la captura y envío.

Caso de Uso:	Ordenar la realización de la captura y envío.
Actores:	Reloj del Sistema
Resumen:	El Reloj del Sistema busca las tareas que estén programadas para esa fecha y manda a ejecutar la captura y envío de la información.
Precondiciones:	Previamente se deben haber configurado y guardado los horarios para la realización de la captura y envío de los datos.
Referencias	RF4
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El reloj activa el proceso de captura y envío de la información con los datos configurados previamente.	2. El sistema realiza el proceso para la captura y envío de la información. Finaliza el Caso de Uso.
Flujo alterno	
Prototipo de Interfaz	
Poscondiciones	.

Anexo 3: Interfaz para la programación Diaria.

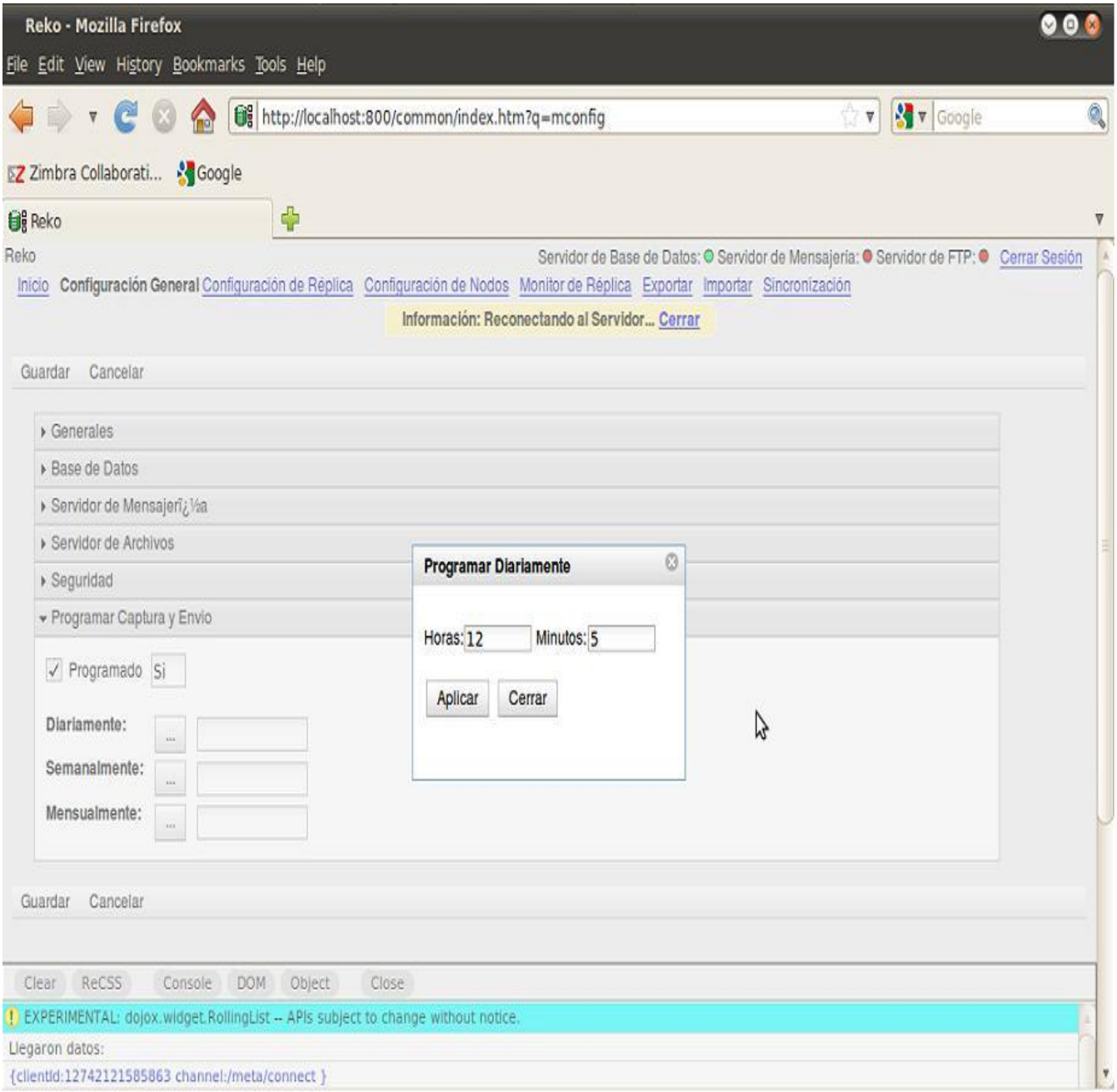


Figura 17. Programar Captura y Envío. Programar Diariamente.

Anexo 4: Interfaz para la programación Semanal.

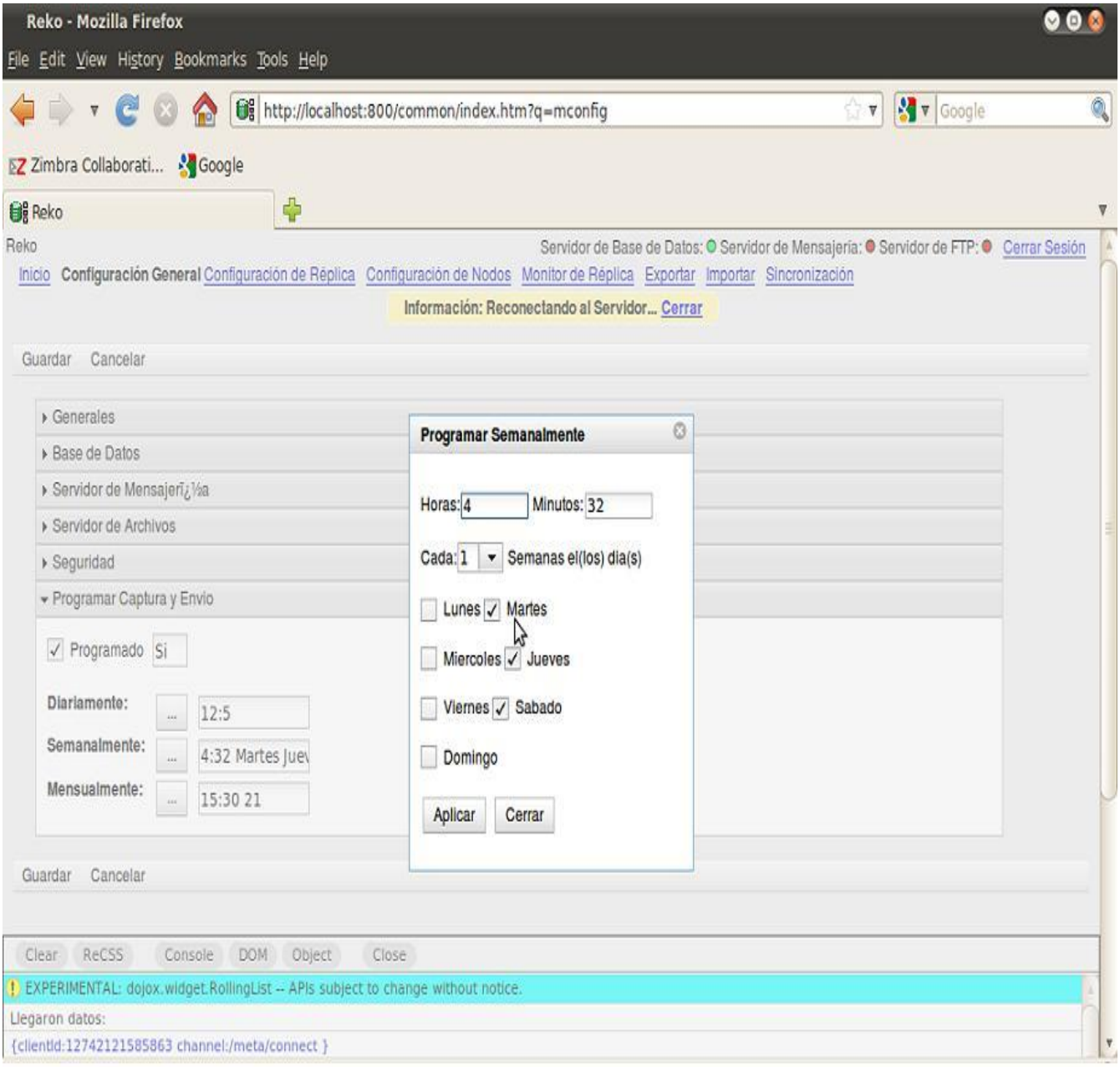


Figura 18. Programar Captura y Envío. Programar Semanalmente.

Anexo 5: Interfaz para la programación Mansual.

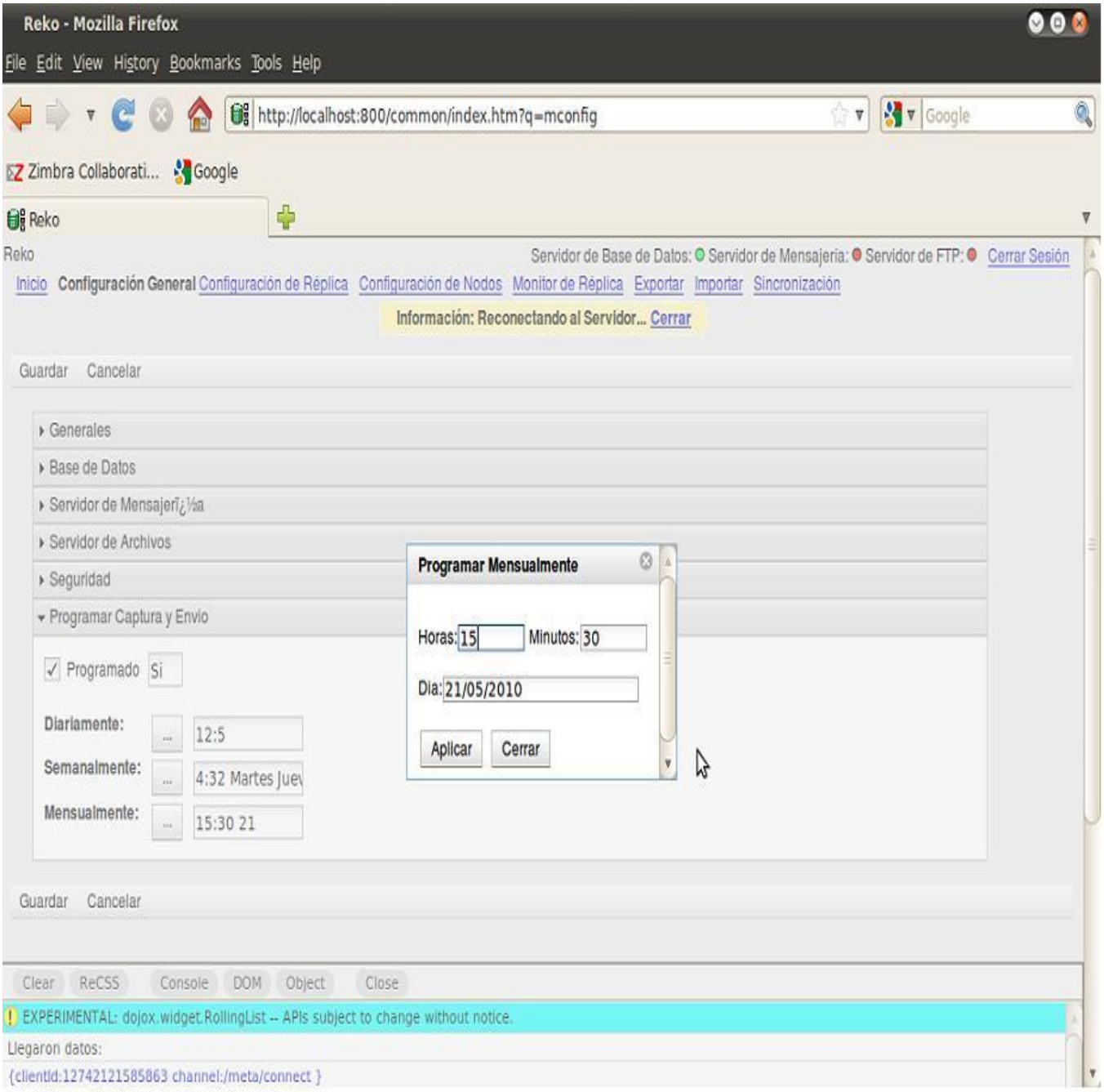


Figura 19. Programar Captura y Envío. Programar Mensualmente.

Anexo 6: Interfaz para la configuración de réplica.

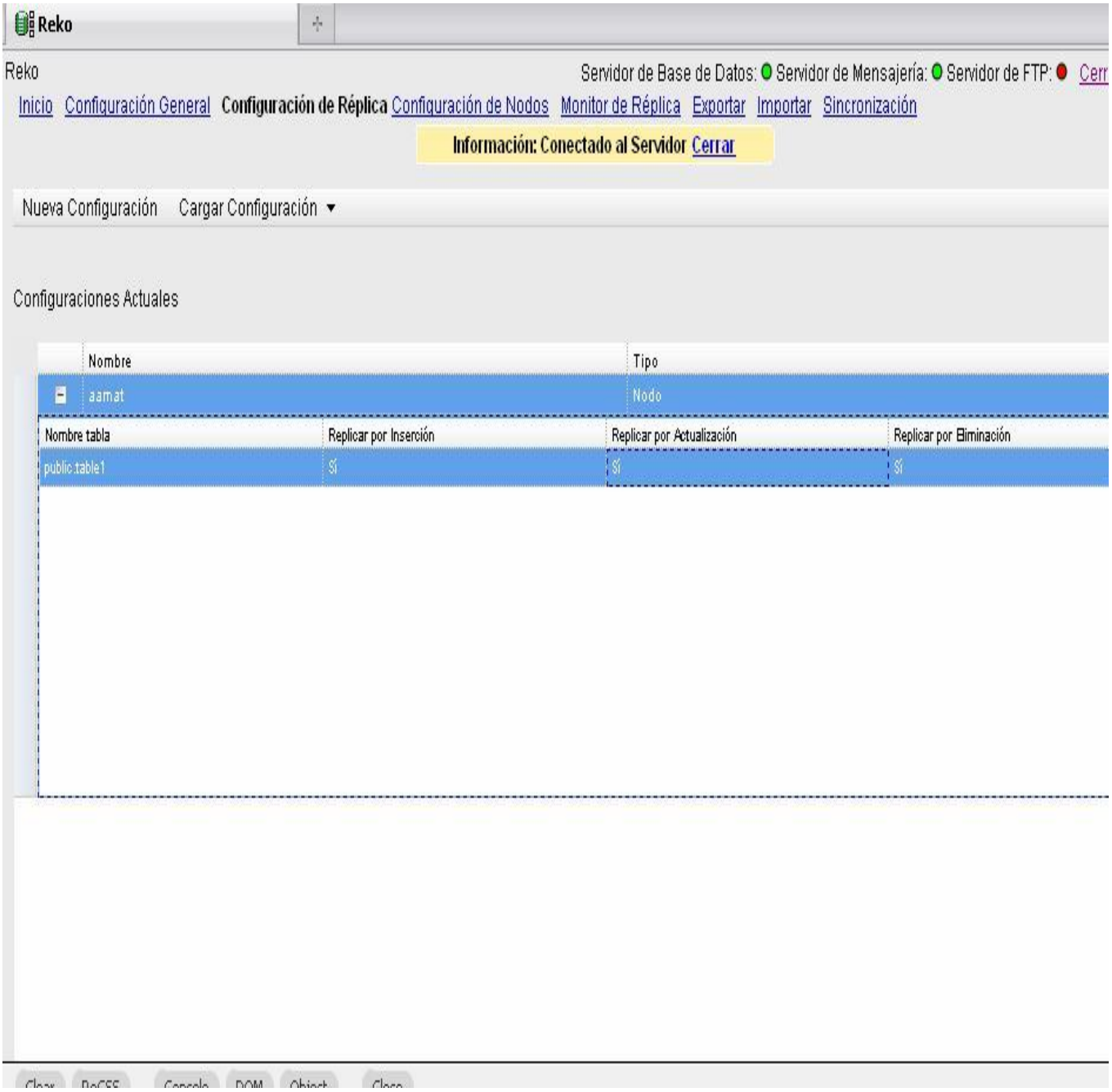


Figura 20. Configuración de Réplica.

Anexo 7: Diseño de las pruebas por cada sección del Caso de Uso Configurar parámetros para a captura y envío de la información.

SC: 1.1 Configurar parámetros para la captura y envío.

Variables.

1. Horas.
2. Minutos.

ID del Escenario	Escenario	V1	V2	Resultado de la prueba
EC:1.2	1.2 Seleccionar la Opción Programar Diariamente.	V	V	Se guardó la configuración de la hora y los minutos seleccionados para realizar la captura y envío de la información.

Variables.

1. Días
2. Horas
3. Minutos.

ID del Escenario	Escenario	V1	V2	V3	Resultado de la prueba
EC:1.3	1.2 Seleccionar la Opción Programar Semanalmente.	V	V	V	Se guardó la configuración del día o los días escogidos la hora y los minutos seleccionados para realizar la captura y envío de la información.

Variables.

1. Días
2. Horas
3. Minutos.
4. Meses.

ID del Escenario	**Escenario**	V1	V2	V3	V4	**Resultado de la prueba**
EC:1.4	1.2 Seleccionar la Opción Programar Mensualmente.	V	V	V	V	Se guardó la configuración del mes o los meses escogidos, el día o los días escogidos, la hora y los minutos seleccionados para realizar la captura y envío de la información.

GLOSARIO DE TÉRMINOS

En el glosario aparecen un grupo de conceptos básicos relacionados con la elaboración del sistema informático.

Siglas: SOA: Arquitectura orientada a servicios.

JEE: Comunidad Java.

HTTP: (Hypertext Transfer Protocol): Protocolo de transmisión del hipertexto.

J2EE: Java 2 Enterprise Edition.

J2SE: Java Standar Edition.

EJB: Enterprise Java Beans.

OO: Orientado a Objeto.

JSP: (Java Server Page): Página servidora de java.

RUP: (Rational Unified Process): Proceso unificado del Rational.

UML: (Unified Modeling Language): Lenguaje de modelado unificado.

MVC: (Model-View-Controller): Modelo Vista Controlador.

XML: (Extensible Markup Language): Lenguaje de marcas extensible.

(TCP/IP, FTP, HTTP): Protocolos de transferencia de archivos.

HTTPS:Protocolo de transferencia seguro.

Oracle: Gestor de bases de Datos.

MySQL: Gestor de bases de Datos.

JDK: Java Development Kit.

Específicos:

Replicador: Programa encargado de replicar datos e información de una estación de trabajo a otra.

Multiplataforma:(Cross-platform) Programa o dispositivo que puede utilizarse sin inconvenientes en distintas plataformas de hardware y sistemas operativos. Un programa en lenguaje Java posee esta característica (18).

Clúster: Unidad de Almacenamiento.