Universidad de las Ciencias Informáticas

FACULTAD 6

Título: Herramienta de réplica de datos desde PostgreSQL hacia los gestores NoSQL, MongoDB y CouchDB.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Mariannys Santana Montero

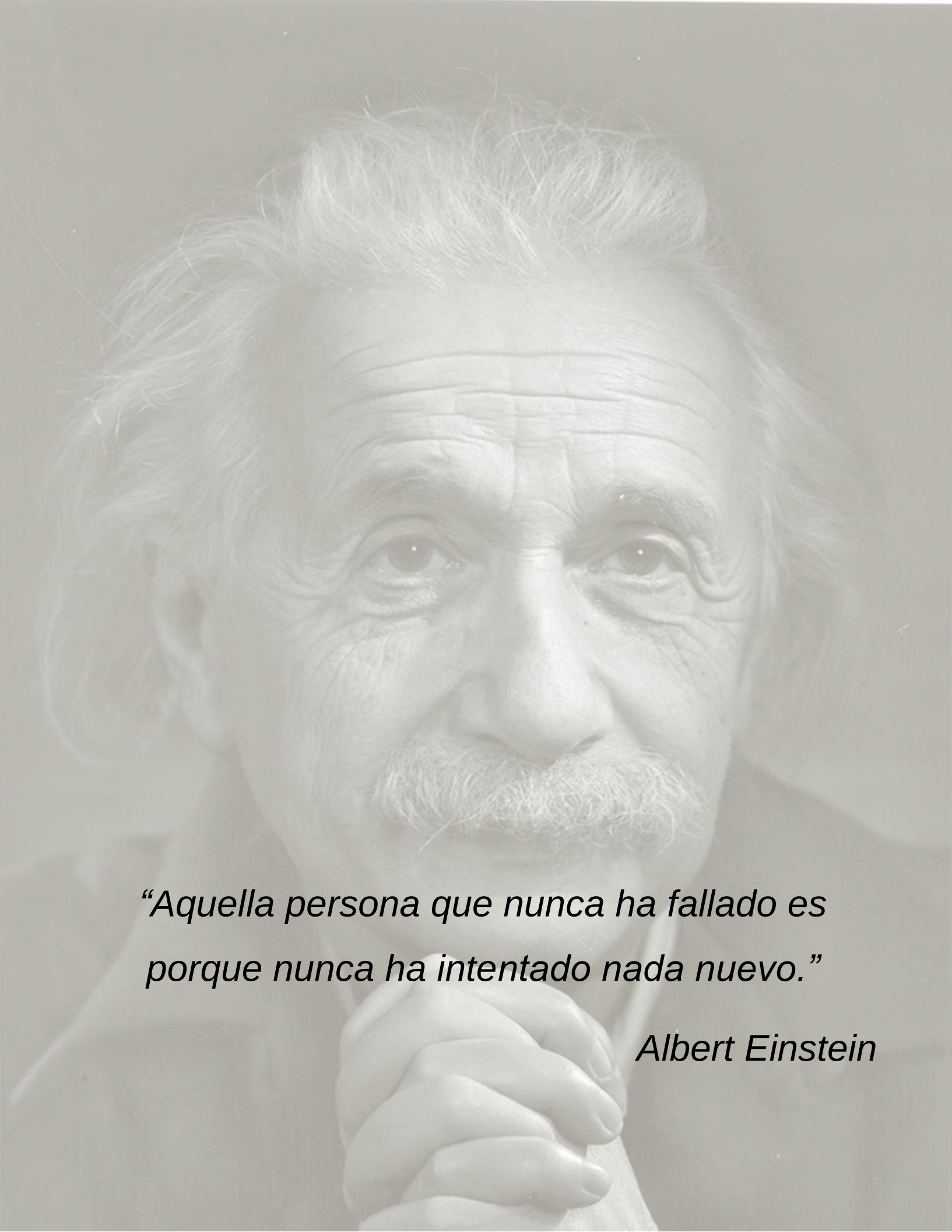
Oscar Eduardo Yañez Candebat

Tutores: MsC. Anthony Sotolongo León

Ing. Marcos Michel Martínez Pérez

La Habana, junio de 2014

"Año 56 de la Revolución"



“Aquella persona que nunca ha fallado es porque nunca ha intentado nada nuevo.”

Albert Einstein

Declaración de autoría

Declaramos ser autores de la presente tesis que tiene por título: **"Herramienta de réplica de datos desde PostgreSQL hacia los gestores NoSQL, MongoDB y CouchDB"** y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Mariannys Santana Montero

_____

Firma del autor

Oscar Eduardo Yañez Candebat

_____

Firma del autor

MsC. Anthony Sotolongo León

_____

Firma del tutor

Ing. Marcos Michel Martínez Pérez

_____

Firma del tutor

Datos de contacto

Tutor: MsC. Anthony Sotolongo León

Universidad de las Ciencias Informáticas

La Habana, Cuba

E-mail: asotolongo@uci.cu

Tutor: Ing. Marcos Michel Martínez Pérez

Universidad de las Ciencias Informáticas

La Habana, Cuba

E-mail: mmartinezp@uci.cu

Agradecimientos

Siempre supe que esta sería la redacción más difícil de todo el documento, pero no podía dejar de agradecer a todas las personas que contribuyeron de alguna forma a alcanzar esta meta, agradezco:

A mi mami por haberme traído al mundo, por ser mi guía, por cada beso y abrazo que siempre me reconforta, por las noches a mi lado sin dormir, por cada regaño que me hace mejor cada día, por ser mi compañerita de toda la vida, no existirá forma de agradecer todo el esfuerzo y dedicación que ha depositado en mí.

A mi papi por engendrarme, por ser mi ejemplo a seguir, por ser tan atento, porque cada momento que paso junto a él aprendo algo nuevo, gracias por cada cuento antes de dormir, por ser ese padre especial, recto y celoso pero muy sentimental, ese padre único que todos quisieran tener, gracias por ser el mejor de todos los padres del mundo.

A mi hermanita por ser mi segunda madre, por ser mi amiga incondicional, porque aunque está muy lejos de aquí nunca dejó de apoyarme y ayudarme, porque sé que en este momento se encuentra deseando todo lo mejor del mundo para mí. Te quiero mi hermana.

A mi sobrinita que ha sido como mi propia hija, por quererme, por tenerme siempre presente, por respetarme a pesar de sus malcriadeces, por tenerme como su ejemplo a seguir, cosa que me hace fuerte en cada paso que doy.

A mi novio, porque a pesar de nuestras diferencias de criterios, ha depositado toda su confianza en mí, por haberse convertido en una persona muy especial en mi vida, por enseñarme que todo se puede lograr solo hay que proponérselo y luchar, por quererme, por ayudarme, por cada crítica constructiva y por escogerme entre tantas como la mujer de su vida.

Agradecimientos

A mi compañero de tesis, que mostro un gran interés y disposición, apoyándome y dándome aliento en todo momento, y que además hizo que me convirtiera en su jefecita.

A Laura la loca por ayudarme y apoyarme en mis decisiones durante estos 5 años, por compartir buenos y malos momentos a mi lado, por apreciarme tanto, por permitirme formar parte de sus fiestas familiares y por ser una gran amiga y una persona de muy buenos sentimientos.

A Alfredo por soportar mi carácter durante 4 años y 3 meses, por estar a mi lado en los momentos más difíciles que he pasado durante estos años, por su buen corazón y su entrega incondicional con aquellos que llegan a formar parte de su vida, por compartir junto a mí, momentos inolvidables.

A mis tutores por su ayuda durante el proceso de tesis, y por cada comentario realizado.

Al grupo de fiestas conformados por: la flaquita de pelo largo enfermiza que viene conmigo desde 1er año siempre dándome buenos consejos, gracias por ellos Anet; Yadian compañero no solo de aula sino también de provincia y también de viajes, gracias por toda tu ayuda; Eduard el pelú, lleno de ocurrencias, gracias por ser capaz de sacarle una sonrisa a cualquiera; Ovi el bruto, el que no puede estar 5 min sin pelear, pero muy buen bailador y donde también entran Laura y Josué, gracias a todos ellos por pasar junto a mi momentos que quedarán para la historia.

Agradezco en general a todos mis compañeros de aula, de todos los años, porque cada uno de ellos aportó un granito de arena en este hermoso período de mi carrera universitaria, en especial a Meybis y a Jhony, por ser de las mejores amistades que se puede tener.

A todos ustedes Muchas gracias.

Mariannys

Agradecimientos

Como diría nuestro Apóstol Martí, toda la gloria del mundo, se resume en un grano de maíz, por eso el agradecimiento es la virtud más linda que hemos adquirido en esta carrera tan grande que es la vida.

Agradezco a mi familia por siempre confiar en mis convicciones y ser mi fuerza motora para lograr este resultado.

Agradezco a mi hermosa princesa Izmeydi Zambrana Ayala por ser la mujer que siempre quise y por brindarme su amor y cariño cuando me hizo falta.

Agradezco a mi compañera, amiga y jefa Mariannys Santana Montero ha sido la mejor compañera de trabajo que he tenido sin ella no lo hubiese logrado.

Agradezco a nuestra revolución cubana por darme el privilegio de haberme hecho un hombre de bien y a la Universidad de Ciencias Informáticas por haberme nutrido de los conocimientos pertinentes para ejercer mi profesión donde le haga falta al país.

Agradezco al excelente claustro de profesores que año tras año no has seguido correcta y eficazmente para no fallar en la obtención de este triunfo.

Agradezco al tribunal de los cortes por su carácter crítico y autocritico en todo momento y por nunca dudar de nosotros.

Agradezco a mis compañeros del grupo por acogerme desde que empecé a formar parte de él y por siempre estar ahí para brindarme su ayuda incondicional.

Agradezco a la vida por haberme puesto en el camino personas como Juan Carlos, Yoel, Yadian, Hector y Liosdanys y además por haberme dado un hermano como Alfredo Sabina Diez, nunca me olvidare de su entrañable amistad ni de lo que significaron para mí en este tiempo.

Agradecimientos

Agradezco a mis tutores Marcos Michel Martínez y Anthony Rafael Sotolongo por haber depositado en todo momento su confianza y habernos sembrado la capacidad de emprendernos por caminos difíciles porque como diría nuestro Apóstol a las estrellas no se sube por caminos llanos.

Oscar Eduardo

Dedicatoria

Le dedico este logro en mi vida a unas personitas que ocupan gran parte de mí ser:

A mi viejito que sé que hoy está conmigo aquí apoyándome como siempre.

A mi tía Deisy que aunque ya no se encuentra físicamente entre nosotros, se encuentra en pensamiento.

Mariannys

Le dedico este trabajo de diploma a mi Madre querida que ha sido mi mejor amiga en estos años y por ella es que soy hoy un joven integrado a la sociedad.

Le dedico esta tesis a mis compañeros de batalla Amadeo Yañez Couso y Gladys Guevara, al compañero Lazaro Martinez, Gustavo Rodriguez y Mercedes Mustelier que en el transcurso de este tiempo han sido personas imperecederas.

Le dedico esta Tesis en toda su totalidad a Mirtha Valenzuela Ruiz por haber sido mi guía y estandarte en esta contienda.

Le dedico este triunfo a mi hermano del alma Roger Isidro Yañez Candebat ,a mi hermano Yasel, a mi Padre Oscar Yañez Valenzuela, a Guillermo Yañez y a mis hermanas Lianne Yañez Rey y Sabrina Yañez.

Este triunfo es dedicado también a mi compañero de la vida Agustin Lazaro y a mi Amigo Alfredo Sabina.

Oscar Eduardo

Resumen

RESUMEN

En la actualidad los Sistemas Gestores de Bases de Datos juegan un papel fundamental en el almacenamiento de información de diversas organizaciones y empresas. Esto les ha permitido tener una constante evolución, de manera que surgieron los sistemas relacionales y los no relacionales.

En la Universidad de las Ciencias Informáticas específicamente en el Centro de Tecnología de Gestión de Datos se utiliza como gestor base PostgreSQL para la mayoría de sus proyectos. Estos manipulan grandes cantidades de información y debido a que PostgreSQL es un gestor relacional los reportes a realizar pudieran tornarse lentos según el volumen de información. Existe la necesidad de preparar bases de datos no relacionales con el objetivo de agilizar el proceso de respuesta.

En la presente investigación se propone una herramienta de réplica de datos del gestor relacional PostgreSQL hacia los no relacionales MongoDB y CouchDB. El objetivo de esta herramienta es mantener la sincronización entre los datos del gestor PostgreSQL con los gestores MongoDB y CouchDB. La herramienta desarrollada garantiza la réplica de información del gestor de base de datos PostgreSQL hacia los gestores MongoDB y CouchDB, la misma permite una sincronización de los datos. Esto posibilita contar con bases de datos no relacionales pobladas de grandes volúmenes de datos y listas para utilizarse en proyectos donde se necesita una rápida respuesta de consultas.

Palabras claves: Bases de Datos, PostgreSQL, MongoDB, CouchDB, Réplica de Datos.

Abstract

ABSTRACT

Currently Managers Database Systems play a key role in the storage of information from various organizations and companies. This has allowed them to have a constantly evolving, so relational and non-relational systems emerged.

At the University of Information Sciences specifically in the Technology Center Data Management PostgreSQL as database manager for most of their projects is used. These manipulate large amounts of information and because PostgreSQL is a relational operator to make reports may become slow depending on the volume of information. There is a need to develop non-relational data bases in order to streamline the response process.

In the present investigation tool data replication manager PostgreSQL relational to non -relational MongoDB and CouchDB is proposed. The purpose of this tool is to maintain synchronization between the data manager and PostgreSQL to CouchDB MongoDB managers. The developed tool ensures replication manager information PostgreSQL database MongoDB and CouchDB to the managers, it enables synchronization of data. This makes it possible to have non-relational database populated data from large volumes of data and ready for use in projects where fast query response is needed.

Keywords: Databases, PostgreSQL, MongoDB, CouchDB, Data Replication.

Índice de contenido

Índice de Contenido

Índice de contenido

Índice de Figuras

Índice de Tablas

Introducción

INTRODUCCIÓN

La constante evolución de la capacidad de almacenamiento de datos generó la necesidad de tener herramientas diseñadas específicamente para el procesamiento de grandes bancos de información. Por tal motivo surgieron los Sistemas Gestores de Bases de Datos, DBMS, (por sus siglas en inglés DataBase Management Systems). Un *"DBMS es una herramienta para crear y administrar grandes cantidades de datos de forma eficiente, así como para permitir que las informaciones persistan a través de largos períodos de tiempo de manera segura"* [1].

Debido al auge de los gestores de bases de datos, aparecen los DBMS relacionales, y con ellos se definió el lenguaje SQL. Este lenguaje se *"basa en el modelo de datos relacional donde todos los datos están organizados estrictamente en forma de tablas de valores y donde todas las operaciones de la base de datos son realizadas sobre estas tablas"* [2].

Con la gran utilización de las aplicaciones web a nivel mundial, ha surgido la necesidad de manipular la información almacenada en las bases de datos de manera más rápida y escalable, donde los DBMS relacionales no garantizan dicha optimización. Por tal motivo se han desarrollado los sistemas de bases de datos no relacionales, conocidos como NoSQL (por sus siglas en inglés Not only SQL*). "Estos sistemas no cumplen con el esquema entidad-relación y pueden manipular enormes cantidades de información de manera muy rápida"* [3].

En los últimos años, estos desarrollos tecnológicos han experimentado un acelerado avance, por lo que en la actualidad representan una herramienta clave en proyectos, empresas y organizaciones. Con el objetivo de impulsar el proceso de informatización de la sociedad cubana, surge la Universidad de las Ciencias Informáticas (UCI), creada sobre el concepto de universidad docente productiva. Esta incluye entre sus objetivos, la formación de ingenieros y la creación de productos, servicios y soluciones informáticas, para ser brindados al mundo. Dentro de la UCI se encuentra el Centro de Tecnologías de Gestión de Datos (DATEC), especializado en las tecnologías de bases de datos que tiene como objetivo proveer soluciones y consultorías relacionadas con las bases de datos.

Introducción

En la actualidad, los proyectos que gestionan grandes volúmenes de datos desarrollados en el centro DATEC utilizan PostgreSQL como gestor de base de datos. En ocasiones por causa del gran volumen de información que manejan las bases de datos, pudieran tornarse lentos e ineficientes los reportes a realizar en dichos proyectos.

La condición de bandera y el prestigio que tiene el centro DATEC a nivel nacional en cuanto a soluciones informáticas basadas en tecnologías de bases de datos, ha llevado al centro a valorar la necesidad de utilizar las bases de datos no relacionales para el desarrollo de futuros proyectos, principalmente las orientadas a documentos ya que brindan posibilidades de modelación avanzada, donde dos de los gestores más utilizados son MongoDB y CouchDB [4].

Con intención de eliminar las deficiencias que puedan presentarse ante la manipulación de grandes volúmenes de información, surge la necesidad de preparar las bases de datos en dicho centro para nuevos desarrollos; pensando en los sistemas no relacionales debido a que estos brindan mejor desempeño en cuanto a tiempo de respuesta. Para solucionar dicho problema es indispensable contar con un mecanismo para el paso de la información del gestor PostgreSQL a los gestores NoSQL, MongoDB y CouchDB.

Por los elementos antes expuestos surge el siguiente **problema de la investigación:** ¿Cómo garantizar la réplica de datos desde PostgreSQL hacia los gestores NoSQL, MongoDB y CouchDB?

La presente investigación tiene como **objeto de estudio:** el proceso de réplica de datos, enmarcado en el **campo de acción:** réplica de datos desde PostgreSQL hacia los gestores NoSQL, MongoDB y CouchDB.

Para dar solución al problema anteriormente planteado se define como **objetivo general** de la investigación: desarrollar una herramienta de réplica de datos desde PostgreSQL hacia los gestores NoSQL, MongoDB y CouchDB.

Para lograr el objetivo general anteriormente expuesto se trazan los siguientes **objetivos específicos:**

- Caracterizar técnicas y herramientas de réplica de datos.

Introducción

- Realizar el análisis y diseño de la herramienta de réplica de datos.
- Implementar la herramienta de réplica de datos.
- Realizar pruebas a la herramienta desarrollada.

Se planificaron las siguientes **tareas de la investigación** para dar cumplimiento a los objetivos específicos:

- Análisis de las características de la réplica de datos.
- Análisis de las herramientas de réplica para PostgreSQl.
- Análisis de los modelos de almacenamiento de datos del gestor PostgreSQL y de los gestores MongoDB y CouchDB.
- Selección de las tecnologías para el desarrollo de la herramienta.
- Selección de la metodología de desarrollo a utilizar.
- Desarrollo de los artefactos de la metodología.
- Modelación y diseño de la herramienta de réplica de datos.
- Implementación de la herramienta réplica de datos.
- Diseño de casos de pruebas para la herramienta de réplica de datos implementada.
- Aplicación de los casos de prueba para validar la herramienta de réplica de datos implementada.

Las tareas de la investigación se derivaron de las siguientes **preguntas científicas**:

- ¿Cuáles son las características de los modelos de bases de datos relacionales y no relacionales?
- ¿Qué características tiene una solución de réplica de datos?

- ¿Cómo se deben desarrollar las funcionalidades identificadas?
- ¿En qué medida la herramienta de réplica soluciona el proceso de réplica de datos desde PostgreSQL a los gestores NoSQL, MongoDB y CouchDB?

Para desarrollar la investigación se emplearon los siguientes métodos científicos de la investigación:

- *Método Histórico:* presuponen el estudio detallado de todos los antecedentes, causas y condiciones históricas en que surgió y se desarrolló un objeto o proceso determinado. Estos métodos analizan la trayectoria completa del fenómeno, su condicionamiento a las diferentes prioridades de la historia, revelan las etapas fundamentales de su desenvolvimiento y las conexiones históricas fundamentales [5].
- *Método lógico:* se basan en el estudio histórico del fenómeno, ponen de manifiesto lo que se repite en el desarrollo del objeto y hayan el conocimiento más profundo de su esencia [5].

  Se hace uso de estos métodos para precisar si en la actualidad existen sistemas informáticos que permitan la réplica de datos desde PostgreSQL hacia gestores NoSQL.
- *Método de análisis y síntesis:* el análisis permite la división mental del fenómeno en sus múltiples relaciones y componentes para facilitar su estudio y la síntesis establece mentalmente la unión de las partes previamente analizadas, posibilita descubrir sus características generales y las relaciones esenciales entre ellas [5].

  El análisis está presente en toda actividad científica técnica, principalmente en las primeras etapas de la investigación y la síntesis fundamentalmente en las etapas finales en la obtención de las conclusiones.

  Se hace uso de este método para el análisis de la información con el objetivo de extraer los elementos más importantes relacionados con la réplica de datos.
- *Método de modelación:* es el método mediante el cual se crean abstracciones para representar la realidad compleja. El modelo elaborado por el investigador es semejante al objeto de estudio.

Introducción

En algunas etapas del conocimiento el modelo está en condiciones de sustituir el objeto que se estudia. En el proceso de investigación ofrece información sobre el objeto que se estudia [5].

Se hace uso de este método para modelar mediante el modelo de dominio la realidad existente del negocio.

El presente documento se estructura en tres capítulos que abarcan todo lo relacionado con el trabajo investigativo, la modelación y la implementación del sistema.

Capitulo1: Fundamentación teórica.

En este capítulo se realiza una fundamentación teórica de la investigación para lo cual se definen un conjunto de conceptos y elementos asociados con la réplica de datos. Se caracterizan algunas de las herramientas de réplica de datos para PostgreSQL. Se identifican las herramientas y tecnologías necesarias para el desarrollo de la aplicación, así como la metodología que va a guiar el proceso de desarrollo.

Capítulo 2: Análisis y Diseño.

En este capítulo se realiza el diseño del marco de trabajo que permite comprender los conceptos asociados al dominio del sistema. Para un mayor entendimiento de la solución propuesta, se profundiza en los procesos del negocio. Además se diseña la arquitectura de la herramienta de réplica de datos.

Capítulo 3: Implementación y pruebas de la solución.

En este capítulo se realiza el proceso de implementación del sistema, que permite establecer las tareas asignadas a cada historia de usuario y el estándar de codificación utilizado. Así como también se describe el proceso de pruebas y se realiza un análisis de los resultados de este proceso.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA DE LA RÉPLICA DE DATOS.

Introducción

En este capítulo se define el marco teórico de la investigación, para ello se realiza un estudio sobre conceptos como bases de datos, sistemas gestores de bases de datos, sistemas SQL, NoSQL y réplica, los cuales constituyen elementos claves para el desarrollo del presente trabajo. Además se realiza un análisis de algunas herramientas de réplicas existentes y de la solución propuesta, así como también, se describen las herramientas y metodología a utilizar en el desarrollo de la aplicación en aras de obtener un excelente resultado.

1.1 Base de datos.

A lo largo de los años se ha elevado constantemente la necesidad de almacenar información, que posteriormente pudiera ser consultada, modificada o eliminada. Como respuesta a esta necesidad en 1963 en California se escucha por primera vez el término base de datos, que no es más que *"un almacén que permite guardar grandes cantidades de información de forma organizada"* [1].

Una base de datos puede ser definida como una colección de datos persistentes que son usados por sistemas de alguna empresa.

Un sistema de base de datos es básicamente un sistema de almacenamiento de datos; en otras palabras, es un sistema informático cuyo propósito general es almacenar información y permitir que los usuarios recuperen y actualicen esta información [6].

1.2 Bases de datos relacionales y no relacionales.

Bases de datos relacionales

Capítulo 1: Fundamentación Teórica

Una base de datos relacional, es aquella que cumple con el modelo relacional, esta consta de tres aspectos fundamentales:

- *Estructural:* donde el usuario percibe la información de la base de datos como tablas.
- *Integridad:* donde las tablas satisfacen ciertas restricciones de integridad.
- *Manipulación:* donde los operadores se encuentran disponibles para que el usuario manipule estas tablas [7].

Las bases de datos relacionales se estructuran en dos secciones [2]:

- *El esquema:* definición de la estructura de la base de datos, principalmente almacena el nombre de cada tabla, el nombre de cada columna, el tipo de dato de cada columna y la tabla a la que pertenece cada columna.
- *Los datos o instancia:* contenido de la base de datos en un momento dado.

Las bases de datos relacionales permiten establecer interconexiones (relaciones) entre los datos (que están guardados en tablas) y a través de dichas conexiones relacionar los datos de ambas tablas, de ahí proviene su nombre "Modelo Relacional". Este modelo contiene operaciones de inserción (INSERT), actualización (UPDATE) y borrado (DELETE), operaciones que se realizan sobre las mismas tablas. Utiliza el lenguaje de programación SQL mediante el cual se pueden realizar todas las operaciones mencionadas anteriormente y además se puede a través de este realizar Consultas, Joins y Transacciones [8].

Bases de datos no relacionales

Los sistemas de bases de datos no relacionales orientados a documentos, no almacenan la información en forma de tablas o columnas, sino en forma de registros o documentos. Esto implica que cada registro o documento, puede contener una información con diferente forma cada vez, pudiendo así almacenar sólo los atributos que interesen en cada uno de ellos y facilitando el polimorfismo de datos bajo una misma colección de información. También se pueden almacenar estructuras de datos complejas en un sólo documento, como por ejemplo, almacenar la información sobre una publicación

de un blog (título, cuerpo de texto, autor, etc.) junto a los comentarios y etiquetas vertidos sobre el mismo, todo en un único registro [3].

Como lenguaje de programación utilizan NoSQL, el cual puede trabajar con consultas de tipo Map Reduce, las cuales pueden ejecutarse en todos los nodos a la vez (cada uno operando sobre una porción de los datos) y reunir luego los resultados antes de devolverlos al cliente. Estos sistemas realizan operaciones directamente en memoria, y sólo vuelcan los datos a disco cada cierto tiempo. Esto permite que las operaciones de escritura sean realmente rápidas [9].

Clasificación de las bases de datos no relacionales:

- *Orientada a Columnas:* modelo tabular donde cada fila puede tener una configuración diferente de columnas. Cada clave está asociada con varios atributos (columnas). Ejemplos: HBase, Hypertable, Cassandra, Riak. Buenas en: gestión de tamaño, cargas de escrituras masivas y alta disponibilidad.
- *Orientada a Key-Value:* llave-valor es la forma más típica, donde cada elemento está identificado por una llave única, lo que permite la recuperación de la información de manera muy rápida. Muchas de ellas están basadas en la publicación de Google acerca de su BigTable y de Amazon. Dentro de estas bases de datos se pueden encontrar a BigTable de Google, Dynamo de Amazon, HBase, Riak, Voldemort, Tokio-Cabbinety Memcached entre otras.
- *Orientada a Grafos:* almacenan la información como grafos donde las relaciones entre los nodos son lo más importante. Son muy útiles para representar información de redes sociales como SourceForge, Amazon.
- *Orientada a Documentos:* almacenan la información como un documento (generalmente con una estructura simple como JSON o BSON) y con una llave única. Se puede encontrar a MongoDB y CouchDB entre las más importantes de este tipo [10].

1.3 Sistemas Gestores de Bases de Datos.

Con la evolución de la utilización de la información digital surge la necesidad de tener herramientas diseñadas específicamente para el procesamiento de grandes bancos de esta información. Por tal motivo aparecen los DBMS, que es un tipo de software muy específico, con el fin de servir de interfaz entre la base de datos, el usuario y las aplicaciones que lo utilizan. Tiene como propósito el manejo sencillo de los datos con el fin de llegar más fácilmente a la información contenida dentro de la base de datos [1].

Este software permite la utilización y actualización de los datos almacenados en una o varias bases de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez. El objetivo fundamental de este sistema consiste en suministrar al usuario las herramientas que le permitan manipular los datos, de forma que no le sea necesario conocer el modo de almacenamiento de los datos en la computadora, ni el método de acceso empleado.

Un DBMS es definido como *"una herramienta para crear y administrar grandes cantidades de datos de forma eficiente, así como para permitir que las informaciones persistan a través de largos períodos de tiempo de manera segura"* [6].

1.3.1 Relacionales

Los gestores de bases de datos relacionales se basan en el modelo relacional de forma que administran las bases de datos relacionales y utilizan como lenguaje de programación SQL. Entre los sistemas de gestión relacionales se pueden mencionar a MySQL, Oracle, SQLite, PostgreSQL entre otros [11].

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional de código abierto, 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. PostgreSQL es un gestor relacional y por tanto almacena los datos en forma de tablas (Fig. 1) y

relaciones entre ellas. Este gestor es caracterizado por la estabilidad, potencia, robustez, facilidad de administración e implementación de estándares [12].

	id_person [PK] serial	name character var	apellido character var	edad integer	sexo character var
1	1	rfh	bvjbhj	68	kgjk
2	2	mary	santana	22	femenino
3	3	lola	lima	12	femenino
4	4	oighs	kjgfeipg	45756	KUGFTF
*					

Fig. 1 Forma de almacenamiento en PostgreSQL.

Entre sus características más importantes se encuentran [12]:

- Posee una gran escalabilidad. Es capaz de ajustarse al número de CPUs (Central Processing Unit, por sus siglas en inglés) y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta.
- Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel, como puede ser Oracle.

1.3.2 No relacionales

Los gestores de bases de datos no relacionales son aquellos que no cumplen con el esquema entidad relación y que utilizan el lenguaje de programación NoSQL. Entre los gestores no relacionales se encuentran Cassandra, Memcached, MongoDB, CouchDB [11].

MongoDB es un sistema de base de datos multiplataforma orientado a documentos JSON (Java Script Object Notation, por sus siglas en inglés), de esquema libre. Esto significa que cada entrada o registro puede tener un esquema de datos diferentes, con atributos o columnas que no tienen por qué repetirse de un registro a otro. Funciona en sistemas operativos Windows, Linux, OS X y Solaris. Salvo que está diseñada para ser una verdadera base de datos de objetos, más que para un almacenamiento de

clave/valor puro. Es una base de datos no relacional, es decir, no utiliza SQL. Permite almacenar documentos sin un esquema predefinido, pero ofrece algunas diferencias en cuanto a la organización de la información [4].

A diferencia de otros gestores como por ejemplo PostgreSQL, aquí puede guardar documentos con distintos campos dentro de la misma colección, de modo que esta estructura tiene un sentido de categorización de los elementos que contiene, sin obligar a que estos tengan los mismos datos. MongoDB es imprescindible en aplicaciones que almacenan grandes cantidades de datos o datos complejos, por ejemplo, para aplicaciones con estructuras complejas como blogs (post, comentarios, rollbacks, etc.) o aplicaciones de analítica (Google analytics) [13].

Las características que más se destacan de MongoDB son su velocidad y su amplio pero sencillo sistema de consulta de los contenidos de la base de datos, así como [13].

- *Ausencia de transacciones:* esto le permite a MongoDB ser más rápido y escalable a nivel horizontal.
- *Almacenamiento Orientado a documentos BSON (Binary JSON):* MongoDB almacena todo un registro en un mismo documento (Fig. 2), por lo que no hay necesidad de especificar estructura alguna y sus documentos pueden cambiarse individualmente. Además con el uso de BSON MongoDB es muy rápido en la búsqueda, indexación, almacenamiento y recuperación de la información.
- *Soporte a Querys dinámicas:* como en las bases de datos tradicionales, MongoDB también acepta la ejecución de consultas dinámicas (a diferencia de otras como CouchDB).
- *Indexación de Documentos:* todos los documentos son automáticamente indexados con una clave llamada _id. Esta clave asegura que cada documento es único, MongoDB permite indexar documentos embebidos (se puede crear un índice en un código postal).
- *Análisis de Rendimiento de Queries:* MongoDB provee una herramienta para el análisis de consultas que permite determinar el rendimiento de las consultas o conocer posibles defectos en su estructura o simplemente mejorar el tiempo de respuesta de las mismas.

- *Réplica de Datos:* MongoDB provee un mecanismo llamado replicación maestro-esclavo, con lo que solo una base de datos está activa para escritura en un momento dado. Todas las peticiones de escritura se realizan en la base de datos maestra y esta las pasa a la réplica esclavo.

```
public.persona
{
        { "_id" : { "$oid" : "536bdd406c9cc145ac96a439"} , "id_person" : 2 , "name" :
        "mary" , "apellido" : "santana" , "edad" : 22 , "sexo" : "femenino"}
        { "_id" : { "$oid" : "536beb8c6c9cc145ad96a439"} , "id_person" : 3 , "name" :
        "lola", "apellido" : "lima" , "edad" : 12 , "sexo" : "femenino"}
}
```

Fig. 2 Forma de almacenamiento en MongoDB.

Apache CouchDB es un gestor de bases de datos de código abierto, cuyo propósito está puesto en la facilidad de su uso y en ser una base de datos que asume la web de manera completa. Se trata de una base de datos NoSQL que emplea JSON para almacenar los datos. Una de sus características más peculiares es la facilidad con la que permite hacer replicaciones [4].

CouchDB implementa una forma de Control de Concurrencia Multiversión (MVCC) a fin de evitar la necesidad de bloquear el archivo de base de datos durante las escrituras. Las características que más se destacan son [14]:

- *Almacenamiento de documentos:* CouchDB almacena los datos como documentos (Fig. 3), esto es, uno o más pares campo/valor expresados en JSON. Los valores de los campos pueden ser datos simples como cadenas de caracteres, números o fechas. Pero también se pueden usar listas ordenadas y vectores asociativos. Todos los documentos en una base de datos CouchDB tienen un identificador único y no requieren un esquema determinado.
- *Semántica ACID:* CouchDB provee una semántica de atomicidad, consistencia, aislamiento y durabilidad. Lo hace implementando la forma de MVCC, lo que significa que puede manejar un gran número de lectores y escritores en paralelo, sin que surjan conflictos.

- *Consistencia Eventual:* CouchDB garantiza consistencia eventual para poder ofrecer tanto disponibilidad como tolerancia a las particiones.

"_id"	"df97e46207b795c611d3047a3400138d"
"_rev":	"1-2f652e87e01f81a998ce20f56cf6b568"
"Nombre_de_la_Tabla":	"public.persona"
"apellido":	"Santana"
"Edad":	22
"id_person":	2
"Name":	"Mary"
"Sexo":	"Femenino"

Fig. 3 Forma de almacenamiento en CouchDB.

1.4 Réplica de datos

La réplica de datos "*es un mecanismo utilizado para propagar y diseminar datos en un ambiente distribuido, con el objetivo de tener mejor desempeño y confiabilidad, mediante la reducción de dependencia de un sistema de base de datos centralizado*" [15].

La replicación de la información en una base de datos apunta a garantizar la credibilidad y aumentar la disponibilidad de la información. Esta disponibilidad puede observarse desde dos perspectivas:

- Aumentar el paralelismo en las consultas, dado que la misma información residirá en más de una localidad de la red.
- Mejorar la disponibilidad de los datos ante eventuales caídas de nodos de la red.

1.4.1 Características de la réplica de datos

Captura de cambio

Capítulo 1: Fundamentación Teórica

Al ocurrir algún cambio en las tablas de la base de datos, es necesario capturar los mismos para realizar las actualizaciones en las réplicas, para esto existen dos formas fundamentales, por triggers o por minería de logs.

- *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. Dependiendo de la base de datos, los triggers pueden ser INSERT, UPDATE o DELETE. Capturan cambios hechos para la base de datos y los envían al publicador. Los trigger poseen una estructura básica conformada por: una llamada de activación, que no es más que la sentencia que permite disparar el código a ejecutar; una restricción, que es la condición necesaria para realizar el código y puede ser de tipo condicional o de tipo nulidad; y una acción a ejecutar, que es la secuencia de instrucciones a ejecutar una vez que se han cumplido las condiciones iníciales [16].
- *Minería de Logs:* se sostiene en la lectura de los logs de cambio. Proceso que permite extraer información que reside de manera implícita en los datos almacenados. Estos son referenciados a capturar cambios hechos en las bases de datos. Para la captura de transacciones basadas en log, los cambios de las bases de datos solo pueden ser distribuidos asíncronamente [16].

Dirección de transmisión de los datos

La dirección en que fluyen los datos de una Base de Datos a otra del sistema de replicación, está determinada por el esquema maestro-maestro (master-master) o el maestro-esclavo (master-slave).

Maestro: servidor primario que acepta modificaciones del usuario.

Esclavo: servidor que recibe modificaciones del maestro, solo lectura para el usuario.

- *Esquema maestro- maestro:* se configuran varios nodos como maestro, permitiendo la replicación bidireccional. Un esquema propietario maestro-maestro permite enviar consultas de lectura/escritura a múltiples servidores replicados. La replicación maestro-maestro posee grandes ventajas entre las que se encuentra que si un maestro falla, otros maestros continuarán

actualizando la base de datos y que los maestros pueden estar en diferentes lugares físicos [15].

- *Esquema maestro-esclavo:* las modificaciones sobre ese elemento se hacen primero sobre la copia maestra y luego se propagan a las copias esclavas. Permite que un sólo servidor maestro pueda recibir consultas de lectura/escritura, mientras que los servidores esclavos sólo pueden realizar consultas de lectura [15].

Forma de transmitir los cambios en el entorno de replicación.

Se puede obtener replicación de datos considerando diferentes criterios. Uno de ellos, consiste en clasificar la replicación por la forma de transmisión de los cambios que existe para lograr la consistencia de los datos a lo largo de todas las réplicas. De esta manera se puede hablar de dos tipos de replicación: sincrónica o asincrónica.

- *Replicación sincrónica:* funciona actualizando primero la BD destino, esta envía una confirmación de que los datos se han actualizado en cada una de sus instancias y entonces la BD origen ejecuta los cambios. Estos se ejecutarán de manera inmediata siempre y cuando se confirme [16].
- *Replicación asincrónica:* las réplicas son ejecutadas en su BD origen, de manera que queden registrados los cambios y en un intervalo de tiempo predeterminado las actualizaciones llegarán a su BD destino [16].

Fragmentación de los datos.

La fragmentación se determina por la forma en que las relaciones generales se dividen en fragmentos horizontales (filas), verticales (columnas) o mixtos. El problema de la fragmentación se refiere al particionamiento de la información para distribuir cada parte a los diferentes sitios de la red. Existen tres tipos de fragmentación:

- *Fragmentación horizontal: c*onsiste en el particionamiento en tuplas de una relación general en subconjuntos, donde cada subconjunto puede contener datos que tienen propiedades comunes

y se puede definir expresando cada fragmento como una operación de selección sobre la relación general. Para fragmentar horizontalmente los datos, se divide una tabla mediante operaciones de selección en subconjuntos, cada fragmento se ubica en un nodo o bases de datos, y se reconstruyen con operaciones de unión [17].

- *Fragmentación vertical: c*onsiste en la subdivisión de atributos en grupos. Los fragmentos se obtienen proyectando la relación general sobre cada grupo. Para fragmentar verticalmente, una tabla se divide en subconjuntos, cada fragmento debe incluir la llave primaria de la tabla. Su reconstrucción se realizará con una operación de unión de los fragmentos componentes. En este contexto, una fragmentación óptima es aquella que produce un esquema de fragmentación que minimiza el tiempo de ejecución de las consultas de usuario [18].
- *Fragmentación híbrida:* en varios casos, una fragmentación horizontal o vertical de un esquema de una base de datos no será suficiente para satisfacer los requerimientos de aplicaciones de usuario. Una fragmentación vertical puede ser seguida de una horizontal, o viceversa, produciendo un árbol de particionamiento estructurado. Ya que los dos tipos de particionamiento se aplican uno después del otro, esta alternativa se le conoce como fragmentación híbrida.

1.4.2 Componentes de la réplica de datos

- *Capturador de Cambios:* captura los cambios que se realizan sobre la base de datos y se los entrega al Distribuidor de Cambios.
- *Distribuidor de Cambios:* determina para dónde debe ser enviado cada cambio realizado en la BD, los envía y se responsabiliza de que cada cambio llegue a su destino.
- *Aplicador de Cambios:* ejecuta sobre la base de datos origen los cambios que sean realizados.
- *Monitoreo:* supervisa el proceso de réplica, es decir, ¿qué datos han sido capturados y aplicados?
- *Sincronización inicial:* se refiere al paso inicial de los datos de una base de datos a otra.

1.5 Herramientas de réplica para PostgreSQL

Reko es un replicador de escenario multi-maestro, con transmisión de los cambios asíncrona y que captura y almacena los mismos mediante triggers. Actualmente, Reko constituye una solución robusta ante un alto conjunto de escenarios de replicación para los cuales ha sido probado, y cuenta además con una comunidad de desarrollo en la Universidad de las Ciencias informáticas que lo respalda [19].

Entre las características y/o funciones que posee hasta el momento este software se pueden citar [19]:

- La replicación de acciones a través de triggers se integra con los gestores de bases de datos Oracle y PostgreSQL.
- Provee la facilidad de seleccionar el subconjunto de tablas y datos a ser replicados mediante la definición de filtros aplicables a las tablas y la selección de usuarios propios del gestor.
- 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.
- 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.
- 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.

Slony-I es un software que permite hacer replicaciones maestro/esclavo asíncrono, realizando actualizaciones en cascada. Sistema diseñado para su uso en centros de datos y sitios de respaldo, en el que el modo de funcionamiento normal es cuando todos los nodos están disponibles. Permite replicar grandes bases de datos [20].

El panorama para el desarrollo de Slony-I es que es un esclavo de replicación del sistema principal que incluye todas las características y capacidades necesarias para replicar bases de datos de gran tamaño

a un número razonablemente limitado de los sistemas esclavistas. Slony-I es un sistema diseñado para su uso en centros de datos y sitios de respaldo.

SymmetricDS es un software de replicación de datos asíncrona, multi-maestro, permite subscriptores múltiples y sincronización direccional o bidireccional. Utiliza tecnologías web y de bases de datos para replicar tablas entre bases de datos relacionales, casi en tiempo real. Fue diseñado para escalar a un gran número de bases de datos, trabajar con conexiones de bajo ancho de banda, y resistir a periodos de inoperatividad de la red.

SymmetricDS constituye una potente herramienta para la replicación de datos, ya sean homogéneos o no, independiente del gestor de base de datos que se utilice, siempre y cuando soporte la tecnología de trigger. El enrutamiento y filtrado de los datos se especifica mediante expresiones SQL en la configuración de los triggers. Las expresiones SQL se utilizan para crear el trigger que SymmetricDS instala para capturar los cambios de los datos. Usando esta técnica los datos pueden enviarse a un nodo específico o a un grupo de nodos [21].

Todas las herramientas descritas son compatibles con cualquier sistema operativo, además son asincrónicas y la mayoría permiten realizar réplicas maestro-maestro. Es posible realizar la fragmentación solo con SymmetricDS, Reko. Ninguna de estas herramientas realiza réplicas hacia sistemas NoSQL (Tabla 1).

Tabla 1 Resumen de herramientas de réplica.

	Reko	***Slony-I***	***SymmetricDS***
Dirección de transmisión de los datos.			
Maestro-Maestro	X		X
Maestro-Escavo		X	
Forma de transmisión de los datos.			
Sincrónica			
Asincrónica	X	X	X
Captura de cambios.			

Trigger	X	X	X
Logs			
Fragmentación de los datos.			
Horizontal	X		
Vertical			
Hibrida			X

1.6 Herramienta a desarrollar

Una vez estudiadas las herramientas de réplica existentes, se llega a la conclusión de que la herramienta a desarrollar estará realizando la réplica de datos, basándose en las principales características analizadas anteriormente de forma que, el proceso de captura de cambio se realizará mediante triggers. Estos pueden generar valores de columnas, prevenir errores de datos, sincronizar tablas y modificar valores de una vista, además de ser los más utilizados en herramientas de réplica externas al gestor de bases de datos, por ejemplo en SymmetricDS.

Para llevar a cabo la dirección de transmisión de los datos se utilizará el esquema maestro-esclavo, pues el objetivo que se quiere cumplir es la transmisión de datos solo desde el gestor PostgreSQL a gestores NoSQL, es decir, en un único sentido, ya que así lo amerita la necesidad del entorno de trabajo. El software a desarrollar será una herramienta de replicación de datos asincrónico, pues esta es de consistencia débil entre los datos almacenados, ya que existe una cierta latencia de la copia replicada respecto a la original, además se caracteriza porque permite un alto nivel de autonomía en los sitios. La herramienta no contará con una fragmentación de los datos para ser replicados, ya que se realizará la réplica solo a los cambios realizados, sin tener que fragmentar para distribuir cada parte a los diferentes sitios de la red.

1.7 Metodologías y herramientas propuestas

Metodologías

Capítulo 1: Fundamentación Teórica

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y soporte documental para el desarrollo de productos software. Indicando paso a paso las actividades a realizar para lograr el producto informático deseado, indicando qué personas deben participar en el desarrollo de las actividades y qué papel deben jugar. Además detallan la información necesaria para realizar una actividad y qué se debe producir como resultado de su ejecución [22].

- **Metodología XP**

La metodología XP (Extreme Programming, por sus siglas en inglés) es un conjunto de prácticas y reglas empleadas para desarrollar software. Esta constituye la metodología más utilizada dentro del grupo de las ágiles. Su objetivo principal es asegurar la producción de software con buena calidad y cubriendo las necesidades y requisitos del usuario [23].

En comparación con otras metodologías como RUP (Rational Unified Process, por sus siglas en inglés), XP es mucho más rápido, ya que conlleva menos protocolo, lo que evita que existan jerarquías dentro del grupo.

Esta metodología se basa principalmente en tres principios [23]:

- *Simplicidad:* consiste en desarrollar solo el sistema que realmente se necesita. Implica resolver solo las necesidades actuales.
- *Decisión:* implica saber tomar decisiones y reparar un error cuando se detecta.
- *Comunicación:* consiste en que se hace casi imposible la falta de comunicación pues XP pone en interacción directa a clientes y desarrolladores.

Se define XP como metodología base para el proyecto a desarrollar por los siguientes aspectos enunciados a continuación:

- *Da lugar a una programación organizada y* rápida o extrema: se centra en el desarrollo de las funcionalidades y no en la documentación del proceso de desarrollo.
- *Promueve el trabajo en equipos pequeños:* se cuenta con dos personas para el ciclo de desarrollo de software.

- *Se realizan pruebas continuas durante el proyecto:* durante el desarrollo del software se realizarán entregas parciales para ir evaluando el trabajo realizado.
- *Se basa en la comunicación fluida entre todos los participantes:* durante el desarrollo del software se mantendrá una constante comunicación entre el equipo de desarrollo y el cliente.

Herramienta CASE

Se define como Herramienta CASE (Computer Aided Software Engineering, por sus siglas en inglés) la aplicación de determinadas técnicas y métodos, mediante los cuales se puede modelar o diseñar sistemas por medio de programas y procedimientos con una respectiva documentación [24].

- **Visual Paradigm for UML 8.0**

Visual Paradigm for UML v8.0, es una herramienta CASE, multiplataforma de modelado UML, muy potente y fácil de utilizar. Soporta el ciclo de vida completo de desarrollo de un software, es una herramienta multiplataforma que permite importar y exportar diagramas en XML y como imágenes (ya sea con extensiones jpg o png). Permite crear todo tipo de diagramas UML, revertir código fuente a modelos UML y generar código fuente desde dichos diagramas. Brinda la posibilidad de realizar ingeniería inversa a diferentes lenguajes de programación, tales como: Java, C++, CORBA IDL, PHP, XML, entre otros [24].

Por los elementos enunciados anteriormente se selecciona como herramienta CASE a utilizar para el modelado, Visual Paradigm for UML 8.0, debido a que brinda facilidades en el diseño del producto de forma rápida y con calidad. Además brinda soporte para el lenguaje de modelado seleccionado.

Lenguaje de programación

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos.

- **Java 7.0**

Capítulo 1: Fundamentación Teórica

Java es un lenguaje de programación orientado a objetos. En su sintaxis este lenguaje toma muchos elementos de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros, o memoria[25][25] [25].

Características de Java [26]:

- Totalmente orientado a objetos (Encapsulación, Herencia y Polimorfismo).
- Independencia de la plataforma.
- Dispone de un amplio conjunto de librerías documentadas.
- Facilidad para adicionar nuevas funcionalidades.
- Amplia documentación sobre el lenguaje.
- Software libre.

Por los elementos expuestos anteriormente se selecciona este lenguaje para realizar la implementación de la aplicación, ya que el equipo de desarrollo posee experiencia acerca de este, además de las ventajas que proporciona el lenguaje Java al ser multiplataforma y de software libre.

Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado IDE (Integrated Development Enviroment, por sus siglas en inglés) es un programa compuesto por un conjunto de herramientas que utilizan los programadores para desarrollar código. Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: un editor de texto, un compilador, un intérprete, herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y opcionalmente un sistema de control de versiones [27].

- **NetBeans 7.2**

NetBeans es un entorno de desarrollo integrado de código abierto que permite la flexibilidad en el trabajo por sus posibilidades de modificación por parte de los desarrolladores, elaborado principalmente para el lenguaje de programación Java. Permite que las aplicaciones sean desarrolladas a partir de un

conjunto de componentes de software llamados módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. NetBeans es un producto que ofrece una serie de ventajas como son: la creación de aplicaciones multiplataforma, facilidades y garantías para la migración, facilidad de uso, cumplimiento de regulaciones, y flexibilidad entre plataformas [28].

Teniendo en cuenta los elementos antes planteados y que NetBeans posibilita la gestión de paquetes y tiene avanzadas detecciones de errores (incluso antes de compilar), se selecciona este IDE como base para realizar la implementación de la aplicación.

Lenguaje de Modelado

El lenguaje de modelado está formado por un conjunto de símbolos que estandarizan diseños específicos y además cuentan con modos de disponerlos para modelar parte de un diseño de software orientado a objetos [29].

- ***Lenguaje UML 2.0***

Como lenguaje de modelado se decidió utilizar Lenguaje Unificado de Modelado UML (Unified Modeling Language, por sus siglas en inglés). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un plano del sistema incluyendo aspectos conceptuales tales como: procesos de negocio y funciones del sistema. Además, presenta aspectos concretos como: expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables [30].

Este lenguaje cuenta con una notación estándar y semánticas, para el modelado de un sistema orientado a objetos. Presenta ventajas que permiten encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica, es útil para el desarrollo del modelaje visual de cualquier proyecto, mejora el soporte al control y planeación de los proyectos, alta reutilización y minimización de costos. Este lenguaje de modelado puede ser utilizado tanto por humanos como por máquinas [29].

Capítulo 1: Fundamentación Teórica

UML brinda un mayor rigor en la especificación permitiendo automatizar determinados procesos. Además, es bastante comprensible para aquellas personas que no poseen un conocimiento profundo sobre informática, lo cual permite la comunicación y un mejor entendimiento entre los clientes y el equipo de desarrollo de un proyecto.

Conclusiones del Capítulo 1

Conclusiones

En el capítulo se realizó un estudio y caracterización de los conceptos de bases de datos, sistemas gestores de bases de datos y réplica de datos, que consolidan un mejor entendimiento del trabajo. Se realizó un estudio de las herramientas de réplica Reko, SymmetricDS y Slony-I, lo cual logró constatar, que según la bibliografía estudiada no existe un sistema que sea capaz de realizar la réplica de datos desde PostgreSQL hacia los gestores NoSQL MongoDB y CouchDB. Se identificaron las principales características de réplica de datos a utilizar en la solución propuesta. Se seleccionó como metodología de desarrollo de software XP, como entorno de desarrollo NetBeans 7.2, como lenguaje de programación Java 7.0, como herramienta CASE Visual Paradigm 8.0 y como lenguaje de modelado UML 2.0.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA HERRAMIENTA DE RÉPLICA DE DATOS.

Introducción

Este capítulo contiene el análisis de la información vinculada con el objeto de estudio, procesos a automatizar y conceptos asociados al dominio del sistema. Para un mayor entendimiento de la solución propuesta, se realiza una breve explicación de los artefactos utilizados durante el desarrollo del trabajo, como las Historias de Usuario (HU), la Lista de Reserva de Producto (LRP) y las tarjetas de Clase, Responsabilidad y Colaboración (CRC). Además se analizan los patrones utilizados en el diseño de la solución.

2.1 Descripción de la Solución

Se propone desarrollar una herramienta que realice la transición de datos desde un gestor de bases de datos a otro. Dicha herramienta permitirá a los usuarios realizar la réplica de datos desde PostgreSQL hacia los gestores NoSQL, MongoDB y CouchDB, con el fin de lograr una sincronización entre las bases de datos y brindar un mejor desempeño en cuanto a tiempo de respuesta. Brindando al usuario la posibilidad de que al realizar alguna acción sobre la base de datos origen, se actualice en la base de datos destino de forma automática, de esta forma las bases de datos pudieran encontrarse disponibles en todo momento.

La herramienta se llamará SysTSN (Sistema de Transferencia de SQL a NoSQL) y contará con tres módulos fundamentales para el trabajo con la misma: Servidor Maestro, Réplicas y Mensajes.

Módulo Servidor Maestro: permitirá realizar la conexión con el servidor maestro, una vez establecida esta se creará en la base de datos origen un nuevo esquema llamado *uci_reply_01* que contendrá una función disparadora (trigger) y cuatro tablas:

- *uci_reply_change:* registra de cada cambio realizado en la base de datos*:* el identificador, la tabla donde se realizó el cambio, el esquema al que pertenece, la operación que se realizó (public o private), el usuario que lo realizó, la operación realizada (INSERT, UPDATE o DELETE), la fecha en que se realizó, la consulta que se ejecutó, los datos viejos y los datos nuevos. Estos datos son usados por la aplicación para cargar en el módulo de Servidor Maestro los cambios que han sido realizados en la base de datos.
- *uci_reply_configuration:* registra la configuración de la réplica realizada; para lo cual almacena el identificador de cada réplica, el host, la base de datos, el puerto, el usuario, el password, el tiempo de actualización y la fecha y hora de la última y la nueva actualización. Estos datos son cargados por la aplicación en el módulo de Réplicas, para dar información al usuario sobre cada réplica.
- *uci_reply_set_reply*: registra el set de réplica establecido para cada réplica. Para ello almacena el identificador de la réplica, el esquema, la tabla establecida para realizarle la réplica y el identificador del cambio.
- *register_change:* registra los cambios que han sido replicados. Esta tabla almacena el identificador que posee en la tabla *uci_reply_configuration* la réplica realizada y el identificador que posee en la tabla *uci_reply_change* cambio actualizado en dicha réplica. Con estos datos se determina cuáles son los cambios que aún no han sido replicados, ya que estos representan la diferencia de estos identificadores y los de la tabla *uci_reply_change.*

Módulo Réplicas: permite la configuración y el trabajo con cada una de las réplicas realizadas.

Módulo Mensajes: permite la visualización de los mensajes de las configuraciones y de la cantidad de réplicas realizadas.

Una vez que se realiza una operación de INSERT, UPDATE o DELETE el trigger (Fig.4) creado en el esquema *uci_reply_01* se encarga de capturar los cambios realizados y poblar la tabla *uci_reply_change,* para lo cual inserta en la misma los siguientes datos: la tabla donde se realizó la operación, el esquema al que pertenece, la operación realizada, el usuario que la realizo, la fecha en

que se realizó, la consulta ejecutada, los datos viejos y los nuevos datos, para que luego estos datos puedan ser capturados por la aplicación para efectuar la réplica.

```
CREATE OR REPLACE FUNCTION uci_reply_01.register_trigger()
  RETURNS trigger AS
$BODY$
begin
if TG_OP='INSERT' then
    INSERT INTO uci_reply_01.uci_reply_change(tabla, esquema, operacion, usuario, fecha, consulta_ejc,data_old, data_new)
    VALUES (TG_TABLE_NAME,TG_TABLE_SCHEMA,TG_OP,current_user,current_date,current_query(),null,row_to_json(NEW));
end if;
if TG_OP='DELETE' then
    INSERT INTO uci_reply_01.uci_reply_change(tabla, esquema, operacion, usuario, fecha, consulta_ejc,data_old, data_new)
    VALUES (TG_TABLE_NAME,TG_TABLE_SCHEMA,TG_OP,current_user,current_date,current_query(),row_to_json(OLD),null);
end if;
if TG_OP='UPDATE' then
    INSERT INTO uci_reply_01.uci_reply_change(tabla, esquema, operacion, usuario, fecha, consulta_ejc,data_old, data_new)
    VALUES (TG_TABLE_NAME,TG_TABLE_SCHEMA,TG_OP,current_user,current_date,current_query(),row_to_json(OLD),row_to_json(NEW));
end if;
return new;
end;$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION uci_reply_01.register_trigger()
  OWNER TO postgres;
```

Fig. 4 Código de la función disparadora.

SysTSN estará compuesta por los siguientes componentes:

- *Capturador de cambios:* captura los cambios que se realizan sobre el gestor PostgreSQL y se los envía a la tabla de configuración.
- *Tabla de configuración:* define para dónde debe ser enviado cada cambio realizado en la base de datos origen y los envía a su destino.
- *Aplicador de cambios:* aplica los cambios que sean replicados hacia gestores NoSQL orientadas a documentos.
- *Tabla de control de cambio:* posee el control de los cambios realizados y determina cuales han sido replicados y cuáles no, y en caso de que alguno no se haya replicado manda a realizar la réplica.

- *Limpieza:* realiza la limpieza una vez replicados los cambios realizados.

La herramienta no contará con el componente de Carga inicial, pues esta se realizará mediante MIGDAT, que es una herramienta informática, que se encarga de realizar la migración de los datos entre los gestores de base de datos PostgreSQL, MongoDB y CouchDB. Debido a que las réplicas se realizarán partiendo de las migraciones realizadas previamente, el almacenamiento de los datos para los gestores NoSQL se basan en la forma de almacenamiento utilizada por MIGDAT; para MongoDB se crean colecciones que son equivalentes a las tablas y documentos que representan las filas y para CouchDB solamente se crean documentos, los cuales contienen un atributo con el nombre de la tabla.

Modelo de Dominio

A pesar de que la metodología XP no precisa un procedimiento para definir el negocio, con el fin de proveer una mejor comprensión de los principales conceptos asociados al dominio del sistema, se decide realizar un modelo de dominio.

Un modelo de dominio presenta uno o más diagramas de clases y puede ser tomado como el punto de partida para el diseño del sistema. Este modelo captura los tipos más importantes de objetos que existen, o los eventos que suceden en el entorno donde estará el sistema, se identifican conceptos, se definen estos conceptos y se unen o relacionan en un diagrama de clases UML [31].

El objetivo principal de este diagrama es comprender y describir las clases más importantes del negocio. Este es una representación visual del entorno real del proyecto. Constituye un rol protagónico en el desarrollo del software.

La Fig. 5 muestra el modelo de dominio correspondiente a la herramienta de réplica de datos desde PostgreSQL hacia los gestores NoSQL, MongoDB y CouchDB.

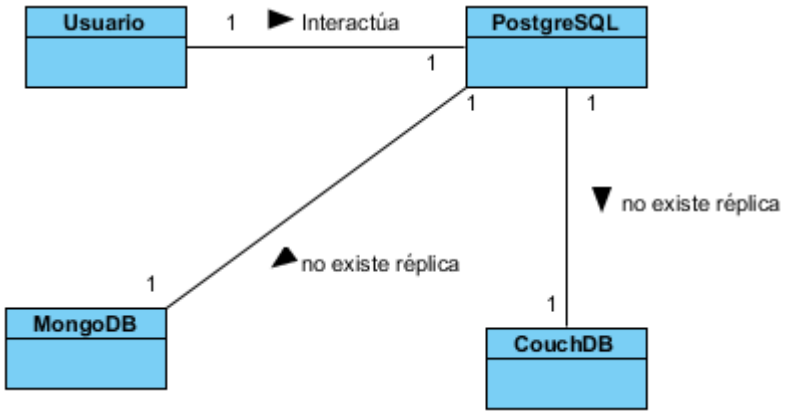


Fig. 5 Modelo de dominio.

Descripción de las clases del dominio

- *Usuario:* persona que interactúa con el sistema de gestión de bases de datos.
- *PostgreSQL:* Servidor de bases de datos maestro.
- *MongoDB:* Servidor de bases de datos esclavo.
- *CouchDB:* Servidor de bases de datos esclavo.

2.2 Diseño de la aplicación

Historias de Usuario

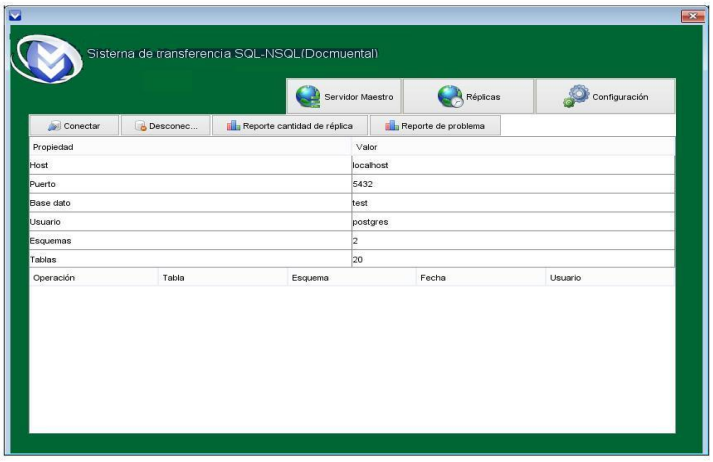
Las HU son la técnica utilizada por la metodología XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales (RF) o no funcionales (RNF). Cada HU es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en pocas semanas, a efectos de planificación, las historias pueden ser de una a tres semanas de tiempo de programación (para no superar el tamaño de una iteración). Las HU son descompuestas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración [32].

Capítulo 2: Análisis y diseño

Con el objetivo de determinar los RF y los RNF con que debe cumplir la herramienta de réplica, se determinaron 6 HU. La Tabla 2 muestra un ejemplo de una HU, las restantes pueden ser consultadas en la planilla llamada "*Historias de Usuario*", que se encuentra en el Expediente de Proyecto adjunto al Trabajo de Diploma.

Tabla 2 Historia de Usuario "Configurar el set de réplica".

Historia de Usuario	
Número: 1	**Nombre de la Historia de Usuario:** Configurar el set de réplica.
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Oscar Eduardo Yañez Candebat	**Iteración asignada:** 1
Prioridad en negocio: Muy Alta	**Puntos estimados:** 2.4 semanas
Riesgo en desarrollo: Alta	**Puntos reales:** 2.2 semanas
Descripción: Permite configurar todas las tablas que serán replicadas hacia NoSQL.	
Observaciones:	
Prototipo de interfaz:	



Lista de Reserva del Producto

La lista de reserva del producto recoge todos los RF y los RNF con los que debe contar la Herramienta de réplica desde PostgreSQL hacia NoSQL, estos pasan a esta lista después de haber sido descritos en las HU expuestas anteriormente. En esta lista los requisitos se muestran ordenados por su prioridad definida en cuatro categorías. Además de una estimación en semanas del tiempo necesario para la realización de la funcionalidad, así como la persona que realiza dicha estimación.

En la Tabla 3 se recogen los RF y los RNF con los que debe cumplir la aplicación.

Tabla 3 Lista de reserva del producto.

Ítem *	Descripción	Estimación	Estimado por
Requisitos Funcionales (RF)			
Prioridad: Muy Alta			
1	Configurar el set de réplica.	2.2 semanas	Analista
2	Captura de cambios	2.2 semanas	Analista
3	Aplicar cambio.	2.0 semanas	Analista

Prioridad: Alta			
4	Mostrar reportes de problemas.	1.2 semanas	Analista
Prioridad: Media			
5	Mostrar reportes de cantidad de datos replicados.	1.2 semanas	Analista
Prioridad: Baja			
6	Limpiar la tabla de control de cambio.	1.0 semanas	Analista
Requisitos no Funcionales (RNF)			
Software			
7	Debe existir un servidor de bases de datos PostgreSQL 9.2 o superior, CouchDB 1.0.1 y MongoDB 2.0; el equipo debe tener instalado la máquina virtual de Java.		Analista
Hardware			
8	El ordenador que se utilizará debe contener como mínimo 1 GB de memoria RAM, espacio en disco duro de 50 MB y un microprocesador a 1.5 GHz.		Analista
Portabilidad			
9	Esta aplicación deberá poder utilizarse en sistemas operativos Windows y GNU/Linux preferentemente Ubuntu GNU/Linux 12.4.		Analista
Disponibilidad			
10	Solo podrán disponer de los datos aquellos usuarios que tengan permiso de lectura sobre las bases de datos.		Analista

Plan de Iteraciones

Capítulo 2: Análisis y diseño

Un plan de iteraciones es una planificación donde se definen las HU que deben ser implementadas en cada versión del programa. Además es donde quedan plasmados los tiempos de implementación idóneos para las HU según la prioridad de estas, los cuales son establecidos por los desarrolladores conjuntamente con el cliente.

En la Tabla 4 se puede observar el Plan de Iteraciones acordado entre el cliente y el desarrollador.

Tabla 4 Plan de iteraciones.

No	Descripción de la iteración	Historias a implementar	Duración total

1	Se desarrollan 3 funcionalidades de prioridad en negocio: Muy Alta, relacionada con el proceso de transición de datos en sí, es decir, configurar el set de réplica, capturar y aplicar los datos a replicar.	1, 2, 3	6.4 semanas
2	Se desarrolla la funcionalidad de prioridad en negocio: Alta, que permite mostrar reporte de los problemas.	4	1.2 semanas
3	Se desarrolla la funcionalidad de prioridad en negocio: Media, que permiten mostrar reportes de la cantidad de réplicas realizadas.	5	1.2 semanas
4	Se desarrolla la funcionalidad de prioridad en negocio: Baja, que permite realizar la limpieza de la tabla de control de cambios.	6	1.0 semanas

Tarjetas CRC

Las tarjetas CRC (Class-Responsibility-Collaboration, por sus siglas en inglés) permiten al programador centrarse en el desarrollo orientado a objetos, rompiendo con la clásica programación procedural. El uso de estas tarjetas es el medio utilizado por la metodología XP para la descripción del diseño del sistema.

Una tarjeta CRC representa un objeto, la clase a la que pertenece cada objeto aparece escrita en la parte superior de la tarjeta, en la columna, a la izquierda, se muestran las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran [32].

Capítulo 2: Análisis y diseño

Con el objetivo de realizar un diseño de la aplicación, se identificaron 32 tarjetas CRC. En la Tabla 5 se muestra un ejemplo de la tarjeta CRC realizada para la clase IinstallContoller. Las restantes pueden ser consultadas en la planilla llamada "*Tarjetas CRC*", que se encuentra en el Expediente de Proyecto adjunto al Trabajo de Diploma.

Tabla 5 Tarjeta CRC "InstallController".

Tarjeta CRC	
Clase: IInstallController	
Responsabilidades	**Colaboraciones**
➢ Crear el esquema de las tablas, la configuración de esas tablas, el cambio realizado, el registro de funciones, el trigger en las tablas y la relación entre ellas. ➢ Leer todas las tablas. ➢ Devolver todas las tablas y los esquemas. ➢ Realizar la instalación.	➢ ITableChange ➢ ITableConfig ➢ IPostgresCatalogo ➢ CPostgresSQL ➢ SMaster

Diagrama de clases

A pesar de que la metodología XP no tiene definido entre sus artefactos de diseño el diagrama de clases, se decide realizar el mismo para una mejor comprensión del diseño y las clases, así como también de los patrones utilizados durante el desarrollo.

El diagrama de clases representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Este constituye un elemento fundamental en la concepción de la aplicación que se propone, ya que servirán de guía a los desarrolladores al constituir una aproximación del sistema que se desea implementar, contribuyendo de esta forma a la calidad del software.

Capítulo 2: Análisis y diseño

En el Expediente de Proyecto adjunto al Trabajo de Diploma se encuentra el diagrama de clases correspondiente al desarrollo de la aplicación. Este diagrama muestra las relaciones entre las 32 clases que en conjunto conforman el funcionamiento interno de la aplicación.

2.3 Arquitectura base de la aplicación

Una arquitectura de software define la estructura del sistema. Esta estructura se constituye de componentes, módulos o piezas de código que nacen de la noción de abstracción, cumpliendo funciones específicas, e interactuando entre sí con un comportamiento definido. De esta manera, la arquitectura de software puede ser vista como la estructura del sistema en función de la definición de los componentes y sus interacciones. La arquitectura constituye un artefacto de la actividad de diseño, que servirá de medio de comunicación entre los miembros del equipo de desarrollo, los clientes y usuarios finales [33].

Patrones de arquitectura

El patrón MVC (Modelo, Vista, Controlador), es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz que ve el usuario de la lógica del negocio en tres componentes distintos. Este patrón facilita e incrementa la reutilización y flexibilidad del código.

Descripción del patrón MVC [34]:

- *Vista (View):* Representa la interfaz de usuario y todas las herramientas con las cuales el usuario hace uso del programa.
- *Modelo (Model):* Es donde está toda la lógica del negocio, la representación de todo el sistema incluido la interacción con una base de datos, si es que el programa así lo requiere.
- *Controlador (Controller):* Este componente es el que responde a la interacción (eventos) que hace el usuario en la interfaz y realiza las peticiones al modelo para pasar estos a la vista.

En la Fig. 6 puede observarse la estructura de la aplicación que será desarrollada siguiendo el patrón MVC.

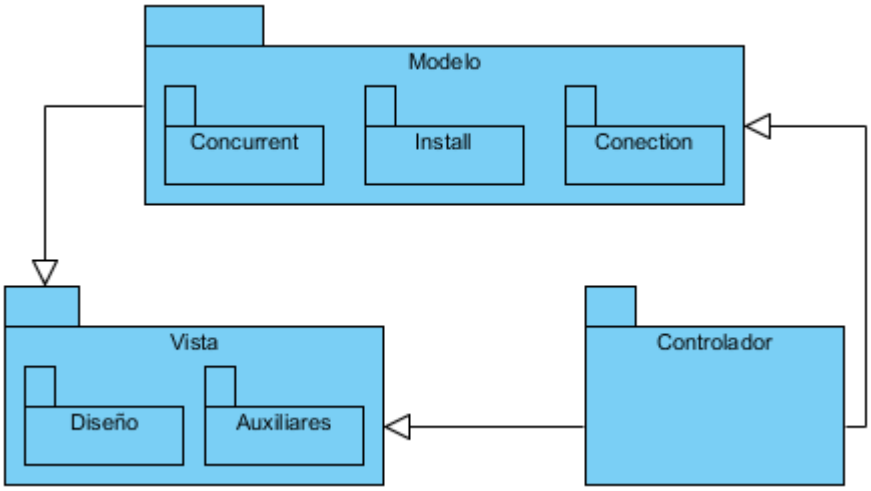


Fig. 6 Diagrama de clases "Patrón MVC".

Patrones de diseño

Los patrones de diseño constituyen el esqueleto de las soluciones a problemas comunes en el desarrollo del software. Son descripciones de clases, cuyas instancias colaboran entre sí. *"Cada patrón describe un problema que ocurre una y otra vez en el entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma"* [35].

Estos patrones son soluciones reutilizables de problemas recurrentes que aparecen durante el proceso de diseño de software orientado a objetos, con el objetivo de transmitir la experiencia a los demás desarrolladores. Con el conocimiento de estos patrones, los programadores son capaces de identificar las situaciones en las que éstos tienen aplicación, y utilizarlos sin tener que detenerse para analizar el problema y vislumbrar diferentes estrategias de resolución [35].

Patrones GRASP

Capítulo 2: Análisis y diseño

Los patrones GRASP (General Responsability Asigment Software Patterns, por sus siglas en inglés) *"describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones"* [36]. Los patrones GRASP son utilizados principalmente para la asignación de responsabilidades.

A continuación se describen los patrones GRASP utilizados en el Trabajo de Diploma [29]:

- *Experto*: consiste en asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Este patrón se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objeto. Con la aplicación de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide.

  En la Fig. 7 se evidencia el patrón experto en la clase CConnection ya que esta es la experta en establecer las conexiones.

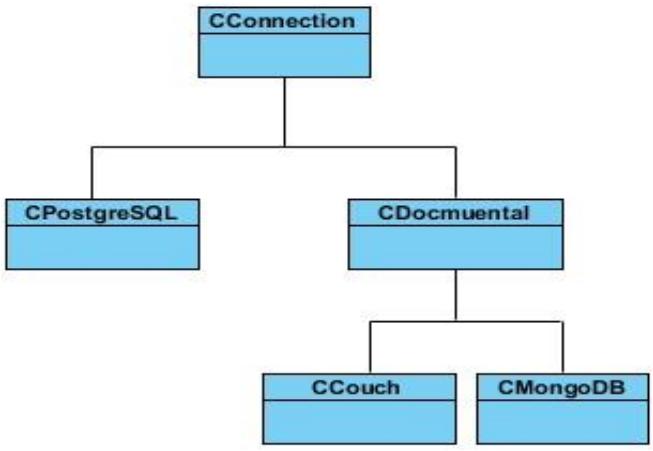


Fig. 7 Diagrama de clases "Patrón experto".

- *Creador:* consiste en asignarle a una clase determinada la responsabilidad de crear una instancia de otra clase, es decir, guía la asignación de responsabilidades relacionadas con la

creación de objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento.

En la Fig. 8 se evidencia el patrón creador ya que la clase IInstallController es la que crea las instancias de las clases ITableChange y la ITableConfig.

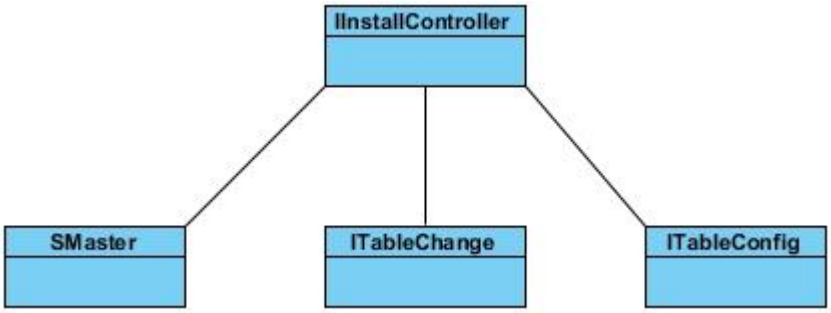


Fig. 8 Diagrama de clases "Patrón creador".

- *Bajo acoplamiento:* consiste en asignar una responsabilidad para mantener el bajo acoplamiento. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. El bajo acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios y también más reutilizables que acrecientan la oportunidad de una mayor productividad.

  En la Fig. 9 se evidencia el patrón bajo acoplamiento ya que la clase IInstallController es la que cuenta con lo necesario para que las clases IPostgresSQL y CPostgresSQL puedan realizar su trabajo.

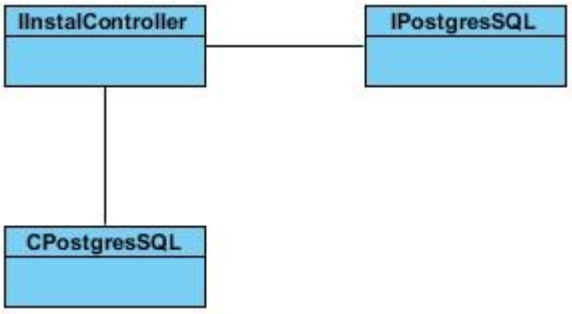


Fig. 9 Diagrama de clases "Patrón bajo acoplamiento".

- *Alta cohesión:* consiste en asignar una responsabilidad de modo que la cohesión siga siendo alta. Es un patrón evaluativo que el diseñador aplica al valorar sus decisiones de diseño. La utilización de este patrón mejora la calidad y facilidad con que se entiende el diseño.

  En la Fig. 10 se evidencia el patrón alta cohesión al existir una estrecha relación entre las clases las clases ITableChange, ITable y la ITableConfig.

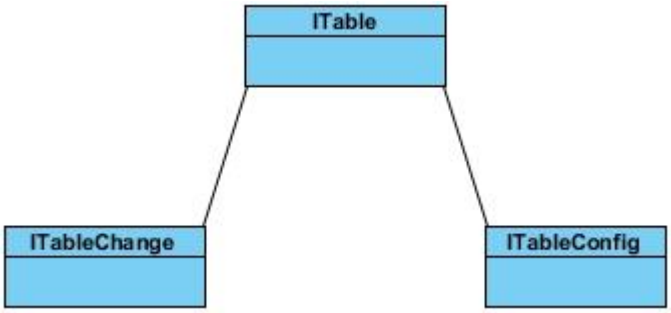


Fig. 10 Diagrama de clases "Patrón alta cohesión".

- *Controlador:* consiste en asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase determinada. Este patrón garantiza un mayor potencial de los componentes reutilizables.

En la Fig. 11 se evidencia el patrón controlador al ser la clase TController quien tiene todo el proceso de control sobre las clases SMaster y MainFrame.

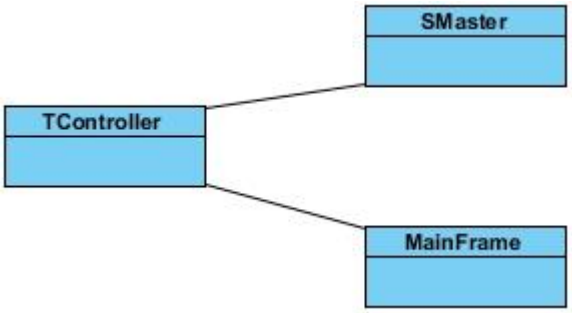


Fig. 11 Diagrama de clases "Patrón controlador".

Patrones GOF

Los patrones GOF (Gang of Four, por sus siglas en inglés) permiten reutilizar soluciones que han sido útiles en el pasado. Constituyen guías, no reglas rigurosas en el proceso de desarrollo del software. Estos patrones capturan el conocimiento que tienen los expertos a la hora de diseñar y ayudan a generar software que soportan y facilitan el cambio y la mejora [35].

A continuación se describe el patrón GOF utilizado en el Trabajo de Diploma:

- *Fachada:* consiste en simplificar la interfaz entre dos sistemas de software ocultando un sistema complejo detrás de una clase que hace de pantalla o fachada. La idea principal es la de ocultar todo lo posible la complejidad de un sistema, el conjunto de clases o componentes que lo forman, de forma que solo se ofrezca un punto de entrada al sistema tapado por la fachada. Su uso aísla los posibles cambios que se puedan producir en alguna de las partes [35].

  En la Fig. 12 se evidencia el patrón fachada al ser la clase ModelEdit una pantalla de las clases ModelTableChange y MdelTableReply.

Fig. 12 Diagrama de clases "Patrón fachada".

Conclusiones del Capítulo 2

Conclusiones

En el capítulo se describieron las tablas que guardan la información relacionada con la réplica y el trigger encargado de la captura de los datos. Se identificaron 6 HU. Para la implementación de las funcionalidades se planificaron 4 iteraciones para un tiempo total de 9.8 semanas. Se definieron los RNF de portabilidad, software, hardware, y de disponibilidad con los que debe contar la aplicación. Se confeccionaron 31 tarjetas CRC, donde se evidencia las relaciones entre las clases, así como también la responsabilidad de cada una de ellas. Se definió para la arquitectura el patrón MVC y para el diseño patrones GRASP, como experto, creador, bajo acoplamiento, alta cohesión y controlador y patrones GOF como fachada.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Introducción

En este capítulo se presentan los artefactos de las actividades de implementación y pruebas necesarias para cumplir los objetivos de la Herramienta de réplica de datos desde PostgreSQL hacia los gestores NoSQL, MongoDB y CouchDB. Se explican además los estándares de codificación utilizados en las clases principales. Así como también se muestran los resultados de las pruebas desarrolladas.

3.1 Tareas de ingeniería

Las Tareas de Ingeniería son un artefacto definido por la metodología XP que se realiza durante la etapa de implementación, con el objetivo de facilitar el trabajo a la hora de analizar, evaluar e implementar cada una de las funcionalidades elaboradas. Estas tareas son definidas a partir de las HU y tributan al cumplimiento de las mismas.

Con el objetivo de dar cumplimiento a las HU definidas, se determinaron 9 Tareas de Ingeniería. En la Tabla 6 y Tabla 7, se muestra un ejemplo de las tareas que corresponden a la HU "*Configurar el set de réplica*", las restantes pueden ser consultadas en la planilla llamada "*Tareas de Ingeniería*", que se encuentra en el Expediente de Proyecto adjunto a este Trabajo de Diploma.

Tabla 6 Tarea de Ingeniería "Diseñar la interfaz gráfica para la configuración del set de réplica".

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Diseñar la interfaz gráfica para la configuración del set de réplica.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1.2 semanas
Fecha Inicio: 12/02/2014	**Fecha Fin:** 19/02/2014
Programador Responsable: Oscar Eduardo Yañez Candebat	

Descripción: Se realiza un diseño de la interfaz para la configuración del set de réplica.

Tabla 7 Tarea de Ingeniería "Implementar configuración del set de réplica".

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 1
Nombre Tarea: Implementar configuración del set de réplica.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1.2 semanas
Fecha Inicio:20/02/2014	**Fecha Fin:** 27/02/2014
Programador Responsable: Oscar Eduardo Yañez Candebat	
Descripción: El usuario tendrá la posibilidad de configurar el origen y el destino de la réplica que se va a realizar.	

3.2 Estándares de codificación

Un estándar de codificación comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, que dé la impresión de que fue escrito por un solo programador, esta importante determinación se debe tomar al iniciar un proyecto haciendo que los desarrolladores trabajen de forma coordinada. La legibilidad del código fuente repercute directamente en el entendimiento que pueda tener otro programador del mismo, aspecto crucial, ya que todo software tiene que someterse constantemente a mantenimiento y mejora de sus funcionalidades.

La nomenclatura que se especifica a continuación es la que se deberá llevar a cabo en la implementación de la solución propuesta.

Nomenclatura de las clases y variables

Los nombres de las clases y de las variables deben comenzar con la primera letra en mayúscula y el resto se pondrá con minúscula, en caso de que el nombre sea compuesto la primera letra de cada palabra comenzará con mayúscula. Ejemplo:

- ITableChange
- IPostgresCatalogo

Nomenclatura de las funcionalidades

En el caso de las funciones de la aplicación, se seguirá la norma de codificación de java que establece que la primera letra de la funcionalidad siempre irá en minúscula y si el nombre es compuesto la palabra que le sigue irá en mayúscula. Ejemplo:

- calculateNextRefresh(String format, GregorianCalendar last)
- notifyObservers()

Normas de los comentarios

Se debe comentariar lo que se haga dentro del desarrollo, establecer las pautas que conlleven a lograr un código más legible y reutilizable y así se pueda aumentar su mantenibilidad a lo largo del tiempo.

Nomenclatura de los comentarios

Los comentarios deben ser claros y precisos de forma tal que se entienda el propósito de los que se está desarrollando, estos siempre van a estar seguidos de: // en la línea donde se quiera especificar dicho comentario.

3.3 Interfaces principales de la aplicación

La herramienta de réplica SysTSN cuenta con tres interfaces, que posibilitan el trabajo con la misma. La Fig. 13 representa la interfaz para la conexión del servidor maestro, la Fig. 14 para la administración y el trabajo con las réplicas y la Fig. 15 para visualizar los mensajes generados al realizar configuraciones y replicaciones.

Capítulo 3: Implementación y prueba

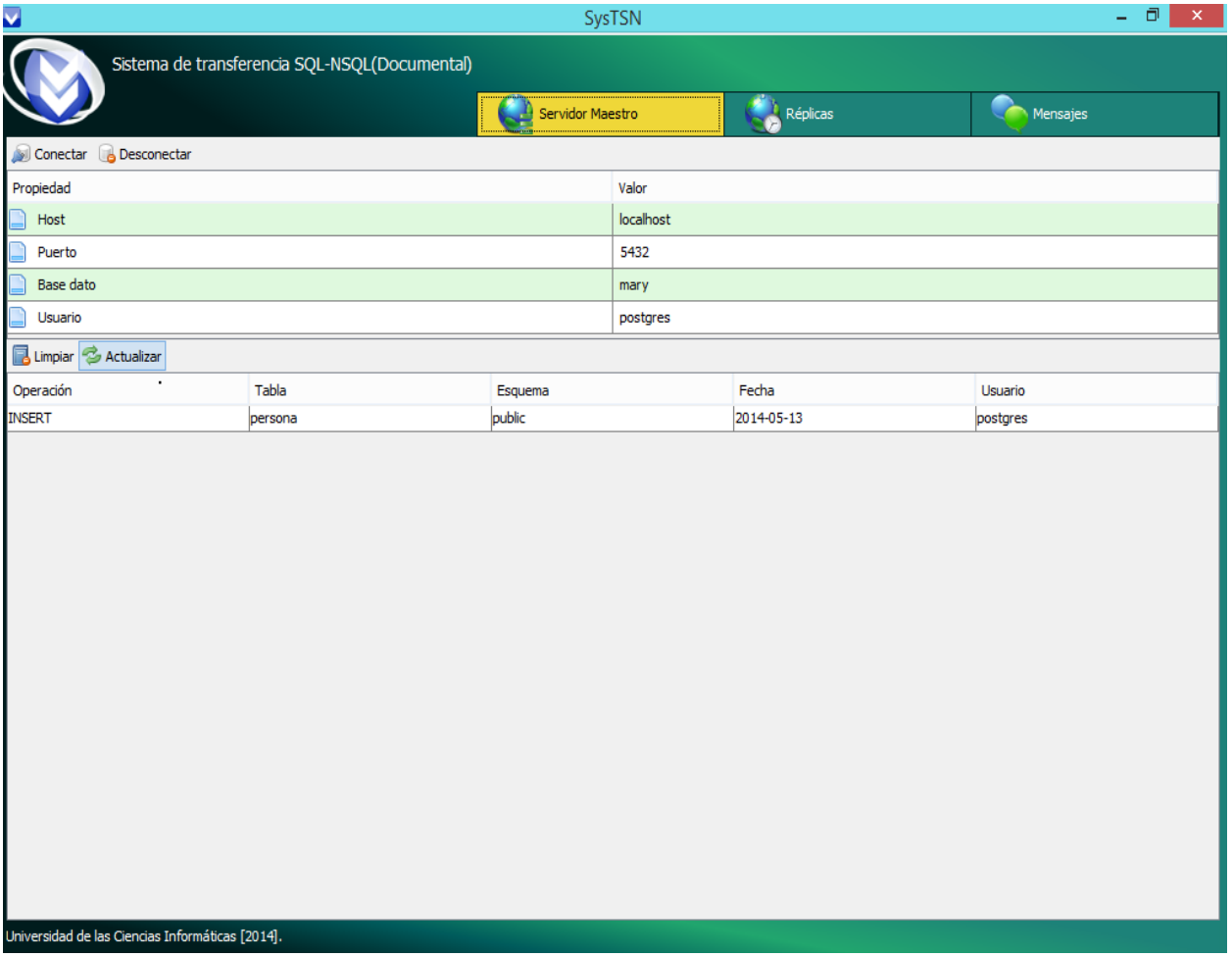


Fig. 13 Conexión del servidor maestro.

La Fig. 13 muestra la interfaz que permite realizar la conexión del servidor maestro, así como también la limpieza y actualización de los cambios realizados. Esta interfaz cuenta con una barra superior con dos botones, uno para realizar la conexión al servidor y otro para desconectar el servidor; un panel inferior a esta barra con los datos de la conexión establecida, donde se encuentra el host, el puerto, la base de datos y el usuario; otra barra inferior que permite limpiar y actualizar por medio de botones los cambios que se reflejan en el área inferior.

Capítulo 3: Implementación y prueba

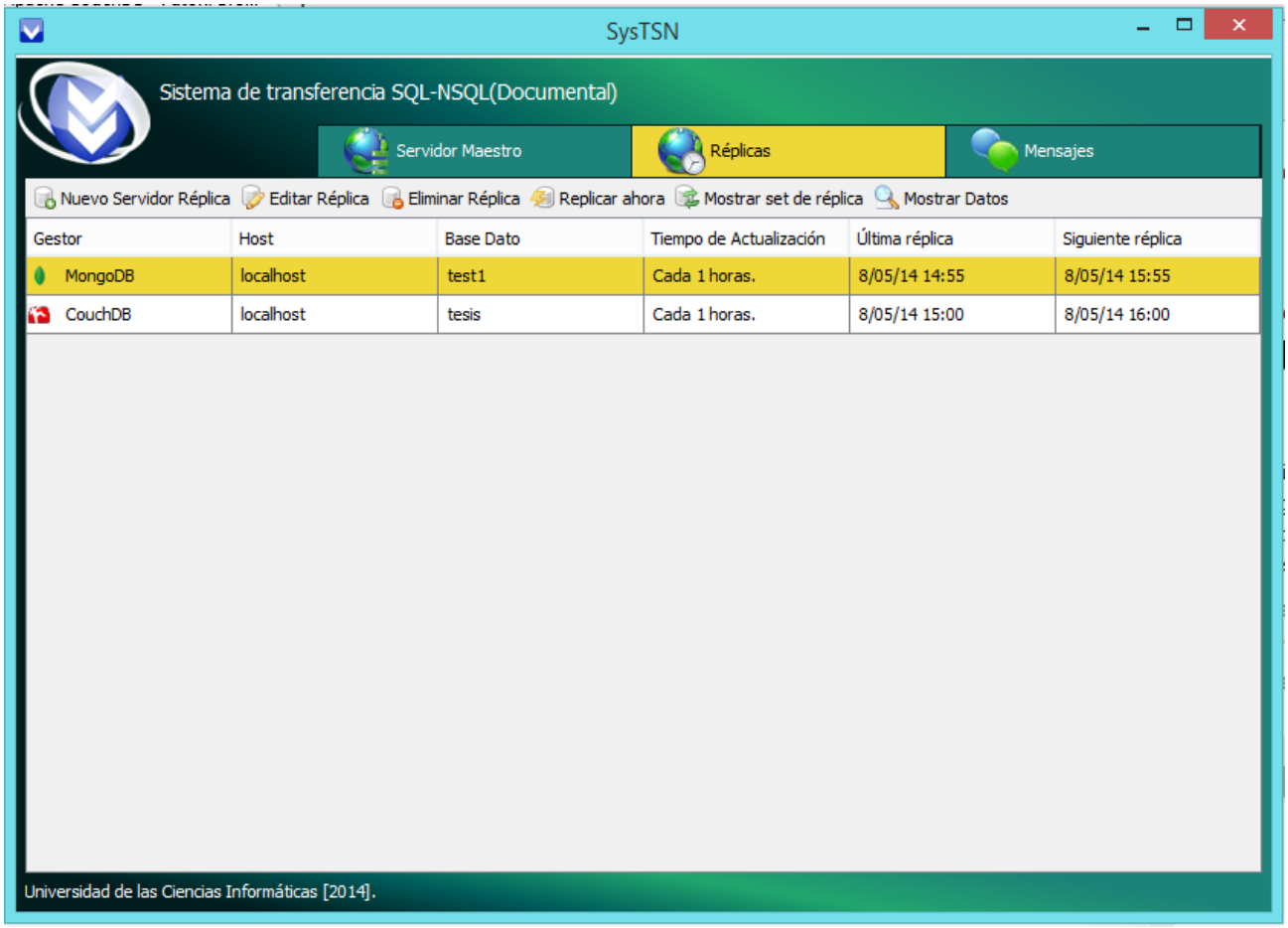


Fig. 14 Administración de la réplica.

La Fig. 14 muestra la interfaz que permite realizar la administración de las réplicas. Esta interfaz tiene una barra superior que brinda opciones de configuración de nuevo servidor de réplica, que establece la configuración para una nueva réplica MongoDB o CouchDB; de editar réplica, que edita la réplica seleccionada en el área de trabajo; de eliminar réplica, que elimina la réplica deseada; de replicar ahora, que realiza la réplica en caso de que se quiera replicar en el instante; de mostrar el set de réplica, que muestra el set de réplica establecido para una réplica seleccionada y de mostrar datos, que muestra los datos replicados hacia MongoDB.

La interfaz contiene además un área de trabajo con la réplica, que permite la selección de cada una de ellas para el trabajo con la misma, en esta área se encuentran los datos de las réplicas realizadas como el gestor, el host, la base de datos, el tiempo de actualización, la última réplica y la siguiente

réplica.

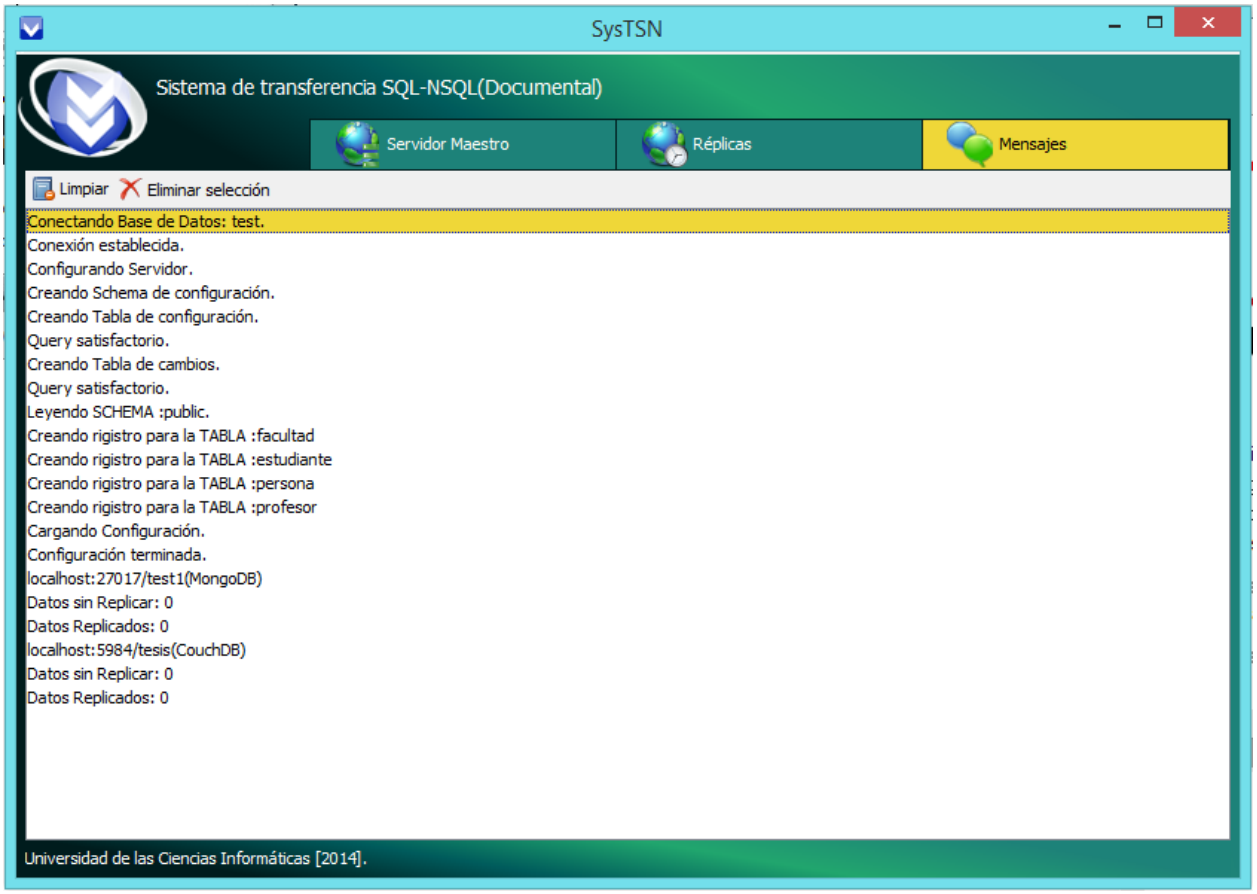


Fig. 15 Visualización de mensajes.

La Fig. 15 representa la interfaz que muestra todos los mensajes pertenecientes a la conexión del servidor y al proceso de replicación de datos. Cuenta con dos botones en una barra superior que brindan la opción de limpiar toda el área de mensajes o eliminar solo el mensaje seleccionado.

3.4 Pruebas

Las pruebas son un conjunto de actividades dentro del desarrollo del software que tienen el objetivo de verificar y revelar la calidad del producto. Este proceso permite el aumento de la calidad final de software. Con el fin de lograr este aumento, se realizan las pruebas para detectar defectos en el software y comprobar que todos los requisitos se hayan implementado correctamente. Existen

diferentes niveles de prueba, tales como: pruebas de desarrollador, unidad, integración. XP divide el proceso de pruebas en dos grupos, 1) Pruebas Unitarias y 2) Pruebas de aceptación [22].

3.4.1 Tipos de pruebas

Pruebas unitarias

Las pruebas unitarias son una de las piedras angulares de XP. Estas se le hacen a pequeñas porciones de código por separados, para garantizar que determinado módulo satisfaga un comportamiento esperado antes de integrarse al sistema. Se establecen antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema. Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados [22].

El objetivo fundamental de estas pruebas es asegurar el correcto funcionamiento de las interfaces o flujos de datos entre componentes. Las mismas son empleadas cuando la interfaz de un método no es clara y la implementación es complicada.

Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las HU, en cada ciclo de la iteración del desarrollo. Estas pruebas tienen como objetivo verificar las funcionalidades del sistema. Las pruebas de aceptación son consideradas como “pruebas de caja negra”. Estas pruebas resultan de gran importancia dado que las mismas verifican y controlan el grado de aceptación del cliente con el producto. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos [22].

Con el objetivo de validar el sistema se seleccionaron las pruebas de aceptación, ya que estas responden a las necesidades del cliente, garantizando que la herramienta cumpla con sus especificaciones funcionales. Se decide no realizar pruebas unitarias, ya que las funcionalidades de la solución operan de forma integrada y no cumple ningún objetivo probarlas individualmente.

3.4.2 Técnicas de pruebas

Hay dos maneras de probar cualquier producto construido, 1) si se conoce la función específica para la que se diseñó el producto, donde se aplican pruebas que demuestren que cada función es plenamente operacional, mientras se buscan los errores de cada función, 2) si se conoce el funcionamiento interno del producto, se aplican pruebas para asegurarse de que todas las piezas encajan. Es decir, que las operaciones internas se realizan de acuerdo con las especificaciones, y que se han probado todos los componentes internos de manera adecuada. Al primer enfoque de prueba se le denomina “prueba de caja negra” y al segundo “prueba de caja blanca”.

Pruebas de caja blanca

Las pruebas de caja blanca, en ocasiones llamadas pruebas de caja de cristal, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Este método analiza la estructura lógica del programa y, para cada alternativa que puede presentarse, los datos de prueba ideados conducirán a ella [22].

Pruebas de caja negra

Las pruebas de caja negra describen las pruebas que se aplican sobre la interfaz del software utilizando los casos de prueba. Con estos, se pretende demostrar que las funciones del software son operativas, se definen las entradas al sistema y los resultados esperados de estas. Las pruebas de caja negra tratan de encontrar errores de funciones incorrectas o faltantes, de interfaz, de estructuras de datos o en acceso a bases de datos externas, de comportamiento o desempeño y de inicialización y término [22].

Existen diversos métodos para implementar esta prueba, entre las que se encuentran [22]:

- *Partición de Equivalencia:* encargada de dividir el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- *Análisis de Valores Límites:* prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

- *Grafos de Causa-Efecto:* permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Una vez analizadas ambas técnicas de pruebas, se decide utilizar las pruebas de caja negra y dentro de estas el método de la Partición de Equivalencia. Debido a que permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de clases de prueba que hay que desarrollar.

3.4.3 Descripción del proceso de pruebas.

Para validar el correcto funcionamiento de la solución, se realizará un proceso para desarrollar las pruebas que comprende un caso de prueba para cada uno de los escenarios generales que hacen referencia a cada una de las HU identificadas. Este proceso se realiza con el objetivo de verificar cada escenario por los que está compuesto dicha solución.

Con el fin de probar la solución fueron identificados 6 casos de pruebas. En la Tabla 8 se puede observar otro ejemplo de caso de prueba correspondiente a la HU “*Configurar el set de réplica”*.

Tabla 8 Caso de prueba "Configurar el set de réplica".

Escenario	Descripción	Host	Puerto	Base de Datos	Usuario	Contraseña	Servidor	Respuesta del sistema	Flujo central
EC 1.1 Configurar el set de réplica.	En este escenario se realiza la configuración del set de réplica.	V	V	V	V	V	V	El sistema configura las bases de datos del set de réplica.	1. El usuario selecciona en la barra superior la opción Servidor Maestro. 2. El sistema muestra la interfaz para la inserción de los datos. 3. El usuario introduce los datos y oprime el botón Aceptar. 4. El sistema configura las bases de datos.
		local host	5432	test	postgres	postgres	MongoDB		
EC 1.2 Configurar el set de réplica con campos incorrectos	En este escenario se realiza la configuración del set de réplica con datos incorrectos.	I	V	V	V	V	N/A	El sistema muestra un mensaje.	1. El usuario selecciona en la barra superior la opción Servidor Maestro. 2. El sistema muestra la interfaz para la inserción de los datos. 3. El usuario introduce incorrectamente los datos y oprime el botón Aceptar. 4. El sistema muestra un mensaje.
		local	5432	test	postgres	postgres			
		V	I	V	V	V	N/A	El sistema muestra un mensaje.	
		local host	5620	test	postgres	postgres			
		V	V	I	V	V	N/A	El sistema muestra un mensaje.	
		local host	5432	ANA	postgres	postgres			
		V	V	V	I	V	N/A	El sistema muestra un mensaje.	
		local host	5432	test	raul	postgres			
		V	V	V	V	I	N/A	El sistema muestra un mensaje.	
		local host	5432	test	postgres	sql			

Capítulo 3: Implementación y prueba

Para realizar cada caso de prueba se identifican variables que permiten validar el funcionamiento de cada uno de estos casos. En la Tabla 9 se muestra un ejemplo de variables para el caso de prueba "*Configurar el set de réplica*".

Tabla 9 Variables del caso de prueba "Configurar el set de réplica".

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Host	Campo de texto	No	Campo alfa que permite entrar el host.
2	Puerto	Campo de texto	No	Campo numérico que permite entrar el puerto.
3	Base de datos	Campo de texto	No	Campo alfa que permite entrar la base de datos.
4	Usuario	Campo de texto	No	Campo alfa que permite entrar el nombre del usuario.
5	Contraseña	Campo de texto	No	Campo alfa que permite entrar la contraseña.
6	Servidor	Campo de selección	No	Campo de selección del servidor.

En la Tabla 10 se puede observar otro ejemplo de caso de prueba correspondiente a la HU "Aplicar cambio". Las restantes pueden ser consultadas en la planilla llamada "*Caso de prueba*", que se encuentra en el Expediente de Proyecto adjunto al Trabajo de Diploma.

Tabla 10 Caso de prueba "Aplicar cambio".

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Aplicar cambios.	En este escenario se aplican los cambios realizados.	El sistema aplica todos los cambios realizados en la bases de datos origen.	1. El usuario selecciona en la barra superior la opción Nuevo Servidor Réplica. 2. El usuario oprime el botón Replicar ahora. 3. El sistema aplica los cambios replicando hacia la base de datos destino.

Las siguientes figuras muestran como se lleva a cabo el escenario Aplicar cambio. En la Fig. 16 se evidencian los datos en PostgreSQL antes de aplicar los cambios mediante la réplica; en la Fig. 17 se evidencian los datos en CouchDB una vez aplicados los cambios.

	id_person [PK] serial	name character var	apellido character var	edad integer	sexo character var
1	1	Mariannys	Santana	22	Femenino
2	2	Oscar	Yañez	24	Masculino
3	3	Claudia	Montero	52	Femenino
4	4	Josue	Mojena	24	Masculino
5	5	Jennifer	Tamayo	12	Femenino
6	6	Osmirio	Santana	54	Masculino

Fig. 16 Datos en PostgreSQL.

Source


```
{
   "_id": "e774eaa58f980d5ab8e593310800aae4",
   "_rev": "1-355a5c470023fe9abe9eeeb766b09667",
   "id_person": 1,
   "apellido": "Santana",
   "name": "Mariannys",
   "sexo": "Femenino",
   "edad": 22,
   "Nombre_de_la_Tabla": "public.persona"
}
{
   "_id": "e774eaa58f980d5ab8e593310800cc4d",
   "_rev": "2-b4adf18161ae7135ee49858cd7276c7b",
   "id_person": 5,
   "apellido": "Tamayo",
   "name": "Jennifer",
   "sexo": "Femenino",
   "edad": 12,
   "Nombre_de_la_Tabla": "public.persona"
}
```


Showing revision 2 of 2

Fig. 17 Datos en CouchDB.

3.4.4 Análisis de los resultados de las pruebas

Con el objetivo de validar el correcto funcionamiento e implementación de cada una de las HU se aplicó el proceso de desarrollo de pruebas. En la Tabla 11 se muestra el proceso de prueba donde se evidencian ejemplos de no conformidades detectadas y al mismo tiempo las que fueron corregidas por el equipo de desarrollo, las restantes no conformidades pueden ser consultadas en la planilla llamada "*No Conformidades*", que se encuentra en el Expediente de Proyecto adjunto al Trabajo de Diploma. En la Fig. 18 se puede evidenciar que en la primera iteración se identificaron un total de cuatro no conformidades, tres de ellas significativa y una no significativa; en la segunda se identificó una significativa y una no significativa, evidenciándose en la última iteración la no existencia de no conformidades, terminando de esta manera la fase de pruebas con un total de seis no conformidades detectadas, las cuales fueron resueltas por el equipo de desarrollo de forma satisfactoria.

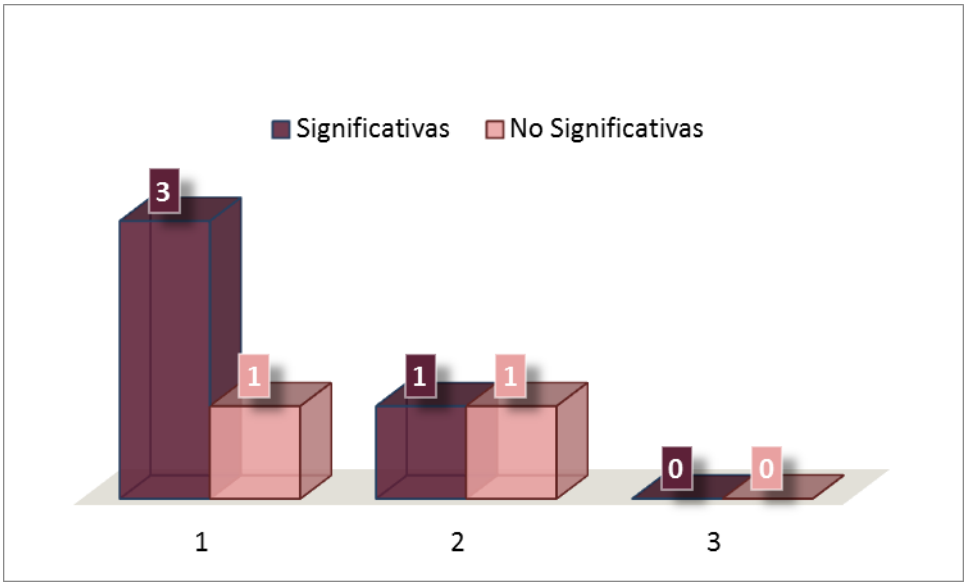


Fig. 18 Resultado de las pruebas por iteración.

Tabla 11 No conformidades detectadas

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del Equip Desarrollo
Aplicación	1	El sistema no muestra los cambios realizados en la base de datos del servidor maestro.	Servidor Maestro	Prueba	NS	06/05/2014 PD 07/05/2014 RA	Se corrigió el error mostrando los cambios realizados.
Aplicación	2	El sistema no permite realizar una selección de las tablas a replicar.	Ambiente general	Prueba	S	08/05/2014 PD 13/05/2014 RA	Se corrigió el error permitiendo poder seleccionar las tablas a replicar.
Aplicación	3	El sistema no carga los datos del servidor maestro al que se realiza la conexión.	Servidor Maestro	Prueba	S	08/05/2014 PD 13/06/2014 RA	Se corrigió el error logrando mostrar los datos del servidor maestro al que se está conectado.

Conclusiones del Capítulo 3

Conclusiones

En el capítulo se identificaron 9 tareas de ingeniería para llevar a cabo la implementación de las 6 HU definidas en la fase de análisis. Para los estándares de codificación se utilizaron las nomenclaturas definidas por el lenguaje de programación Java. Con el fin de validar la aplicación se utilizó la técnica de caja negra, con el método de partición de equivalencia y las pruebas de aceptación a partir del diseño de caso de prueba. Se aplicaron las pruebas seleccionadas permitiendo contar luego de 3 iteraciones con una versión final de la aplicación.

Conclusiones Generales

CONCLUSIONES

Después de terminada la investigación y cumplirse los objetivos definidos, se arribaron a las siguientes conclusiones:

- Se caracterizaron algunas de las herramientas de réplicas de datos como Reko, Slony-I y SymmetricDS, lo cual permitió identificar las características de la solución desarrollada.
- Se identificaron 6 HU que guiaron la implementación de las funcionalidades deseadas por el cliente.
- Se implementó la herramienta de réplica de datos, la cual garantiza una sincronización entre las bases de datos de PostgreSQL y las bases de datos de MongoDB y CouchDB.
- Se realizaron las pruebas de aceptación aplicando la técnica de caja negra y el método de partición de equivalencia, lo cual permitió validar la aplicación en 3 iteraciones, garantizando una alta calidad de la misma.

RECOMENDACIONES

Con el objetivo de mejorar y extender el alcance de la Herramienta de réplica de datos desde PostgreSQL hacia los gestores NoSQL, MongoDB y CouchDB, se recomienda:

- Agregar filtros a los datos a replicar.

REFERENCIAS BIBLIOGRÁFICAS

1. **Camellea**, *Gestión de Base de Datos con ADO.NET.* 2004.
2. **Cattell, R.**, *Scalable SQL and NoSQL Data Stores.* 2010. **39**.
3. **Lungu Ion , T.B.G.**, *The Development of a Benchmark Tool for NoSQL Databases* 2013. **IV**.
4. **Vazquez Ortiz, Y.S.L.**, Anthony Rafael, *Miradas a bases de datos NoSQL de código abierto orientadas a documentos.* 2013. **6**.
5. **Francisco Martín, D.W.L.B., Dr. Eduardo; Castellanos Álvarez, Dr. Juan A. ; Gil Fundadora, Dr. Silvia,** *Metodología de la Investgación*, ed. C.d.E.d.E.y.M. Ambiente. 2006, Cienfuegos.
6. **Valenzuela Ruz, V.**, *Diseño de Bases de Datos.* 2003.
7. **J., D.C.**, *Introducción a los Sistemas de Bases de Datos.* 2001.
8. **Cood Menlo, P.**, *The relational model of database managment: version 2*. Wesley Logman Publishing Co. ed, ed. I. 10-201-14192-2. 1990.
9. **Requena, C.**, *¿Que es NoSQL?* 2010.
10. *Your Ultimate Guide to the Non Relational Universe*. 2009-2011; Available from: http://www.nosql_database.org.
11. **Pérez, C.S.**, *Bases de Datos: RDBMS vs No-SQL, una R-Evolución*. 2011.
12. Martinez, R. *Sobre PostgreSQL*. 2009-2013; Available from: http://www.postgresql.org.es/sobre_postgresql.
13. **Luis, L.M.G.**, ed. *Un poco de MongoDB ¿Qué es?¿Qué ofrece?* 2013.
14. *CouchDB*. 2013; Available from: http://www.couchdb.org.
15. **Tardia, M.A.M.**, *Replicación.* 2011.
16. **Buretta, M.**, *Data Replication Tools and Techniques for managing distributed information*. Wiley ed. 1997.
17. **Fraire Huacuja, H.J.**, *Automatización del Diseño de la Fragmentación Horizontal en Bases de Datos Distribuidas.* 2011.
18. **Rodriguez, L.**, *Segmentacion Vertical dinamica de Bases de Datos multimedia usando reglas activas.* 2012.
19. **Amat Reyes, A.N.R., Madielenis; Campanioni Sardiña, Yusmary; Perez Matos, Leiser,** *Reko: Replicador de datos para sistemas de bases de datos relacionales distribuidas.* 2011.
20. **Calderón Peraza, E.A.**, *Replicación de bases de datos PostgreSQL Slony I en Window.* 2011.
21. **Long, E.H.**, Chris *SymetricDS User Guide Version 1.0.* 2007-2010.
22. Pressman, *Ingenieria del Software.*
23. **Javier Clemente Mendez, E.R.C.**, *Metodologia agil¨Extreme Programming¨ (XP).* 2010.
24. **Reyna, R.** *HERRAMIENTAS CASE*. 2012; Available from: http://es.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.
25. *¿Qué es JavaScript ?* ; Available from: http://www.lcc.uma.es/~eat/services/html-js/manual14.html.
26. **Khawar Zaman, A.U., Cary E,** *Developing Enterprise Java Aplications with J2EE and UML.* 2011.
27. *Que es un IDE de programacion*. 2010; Available from: http://www.buenastareas.com/ensayos/Que-Es-Un-Ide-De-Programacion/163537.html.
28. **Marcotte, L.**, *Database Replication with Slony-I.* 2010.

Referencias Bibliográficas

29. **Craig, L.**, *UML y Patrones*. 2da Edición ed. 2003.
30. **Aragón, D.F.**, ed. *UML y Patrones Introduccion al analisis y diseño orientado a objetos.* 2003: Mexico.
31. **Hector Ortiz, K.** *Representación del Modelo de Objetos de Dominio*. PLATAFORMA PARA EL CONTROL DEL USO DE SOFTWARES EDUCATIVOS 2009; Available from: http://www.eumed.net/libros/2009c/583/Representacion%20del%20Modelo%20de%20Objetos%20de%20Dominio.
32. **Joskowicz, I.J.**, *Reglas y Prácticas en Extreme Programming.* 2008.
33. **CAMACHO, E.C., FABIO; NUÑEZ, GABRIEL**, *ARQUITECTURAS DE SOFTWARE GUÍA DE ESTUDIO.* 2004.
34. **Mestras, J.P.**, *Estructura de las Aplicaciones Orientadas a Objetos, El patrón Modelo-Vista-Controlador (MVC), Programación Orientada a Objetos.* 2004.
35. **Holzner, S.** (2006) *Design Patterns For Dummies*.
36. **Mora, R.C.**, *Patrones de GRASP*. 2003-2004.

Bibliografía

BIBLIOGRAFÍA

1. **Camellea**, *Gestión de Base de Datos con ADO.NET.* 2004.
2. **Cattell, R.**, *Scalable SQL and NoSQL Data Stores.* 2010. **39**.
3. **Lungu Ion , T.B.G.**, *The Development of a Benchmark Tool for NoSQL Databases* 2013. **IV**.
4. **Vazquez Ortiz, Y.S.L.**, Anthony Rafael, *Miradas a bases de datos NoSQL de código abierto orientadas a documentos.* 2013. **6**.
5. **Francisco Martín, D.W.L.B., Dr. Eduardo; Castellanos Álvarez, Dr. Juan A. ; Gil Fundadora, Dr. Silvia,** *Metodología de la Investgación*, ed. C.d.E.d.E.y.M. Ambiente. 2006, Cienfuegos.
6. **Valenzuela Ruz, V.**, *Diseño de Bases de Datos.* 2003.
7. **J., D.C.**, *Introducción a los Sistemas de Bases de Datos.* 2001.
8. **Cood Menlo, P.**, *The relational model of database managment: version 2*. Wesley Logman Publishing Co. ed, ed. I. 10-201-14192-2. 1990.
9. **Requena, C.**, *¿Que es NoSQL?* 2010.
10. *Your Ultimate Guide to the Non Relational Universe*. 2009-2011; Available from: http://www.nosql_database.org.
11. **Pérez, C.S.**, *Bases de Datos: RDBMS vs No-SQL, una R-Evolución*. 2011.
12. Martinez, R. *Sobre PostgreSQL*. 2009-2013; Available from: http://www.postgresql.org.es/sobre_postgresql.
13. **Luis, L.M.G.**, ed. *Un poco de MongoDB ¿Qué es?¿Qué ofrece?* 2013.
14. *CouchDB*. 2013; Available from: http://www.couchdb.org.
15. **Tardia, M.A.M.**, *Replicación.* 2011.
16. **Buretta, M.**, *Data Replication Tools and Techniques for managing distributed information*. Wiley ed. 1997.
17. **Fraire Huacuja, H.J.**, *Automatización del Diseño de la Fragmentación Horizontal en Bases de Datos Distribuidas.* 2011.
18. **Rodriguez, L.**, *Segmentacion Vertical dinamica de Bases de Datos multimedia usando reglas activas.* 2012.
19. **Amat Reyes, A.N.R., Madielenis; Campanioni Sardiña, Yusmary; Perez Matos, Leiser,** *Reko: Replicador de datos para sistemas de bases de datos relacionales distribuidas.* 2011.
20. **Calderón Peraza, E.A.**, *Replicación de bases de datos PostgreSQL Slony I en Window.* 2011.
21. **Long, E.H.**, Chris *SymetricDS User Guide Version 1.0.* 2007-2010.
22. Pressman, *Ingenieria del Software.*
23. **Javier Clemente Mendez, E.R.C.**, *Metodologia agil¨Extreme Programming¨ (XP).* 2010.
24. **Reyna, R.** *HERRAMIENTAS CASE*. 2012; Available from: http://es.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.
25. *¿Qué es JavaScript ?* ; Available from: http://www.lcc.uma.es/~eat/services/html-js/manual14.html.
26. **Khawar Zaman, A.U., Cary E,** *Developing Enterprise Java Aplications with J2EE and UML.* 2011.

Bibliografía

27. *Que es un IDE de programacion*. 2010; Available from: http://www.buenastareas.com/ensayos/Que-Es-Un-Ide-De-Programacion/163537.html.
28. **Marcotte, L.**, *Database Replication with Slony-I.* 2010.
29. **Craig, L.**, *UML y Patrones*. 2da Edición ed. 2003.
30. **Aragón, D.F.**, ed. *UML y Patrones Introduccion al analisis y diseño orientado a objetos.* 2003: Mexico.
31. **Hector Ortiz, K.** *Representación del Modelo de Objetos de Dominio*. PLATAFORMA PARA EL CONTROL DEL USO DE SOFTWARES EDUCATIVOS 2009; Available from: http://www.eumed.net/libros/2009c/583/Representacion%20del%20Modelo%20de%20Objetos%20de%20Dominio.
32. **Joskowicz, I.J.**, *Reglas y Prácticas en Extreme Programming.* 2008.
33. **CAMACHO, E.C., FABIO; NUÑEZ, GABRIEL**, *ARQUITECTURAS DE SOFTWARE GUÍA DE ESTUDIO.* 2004.
34. **Mestras, J.P.**, *Estructura de las Aplicaciones Orientadas a Objetos, El patrón Modelo-Vista-Controlador (MVC), Programación Orientada a Objetos.* 2004.
35. **Holzner, S.** (2006) *Design Patterns For Dummies*.
36. **Mora, R.C.**, *Patrones de GRASP*. 2003-2004.