Universidad de las Ciencias Informáticas

Facultad 6

Título: Herramienta de Migración de Datos entre los Gestores de Base de Datos PostgreSQL, MongoDB y CouchDB.

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

Autores:
Raydel Wilson Moré
Marcos Raúl Cordero Vázquez

Tutores:
Msc. Anthony Rafael Sotolongo León
Ing. Anyer Gámez Guedes

Junio 2013

¨La gloria del mundo es transitoria, y no es ella la que nos da la dimensión de nuestra vida, sino la elección que hacemos de seguir nuestra Leyenda Personal, tener fe en nuestras utopías y luchar por nuestros sueños.¨

Paulo Coelho

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmamos la presente a los _____ del año 2013.

Marcos Raúl Cordero Vázquez

_____

Firma del Autor

Raydel Wilson Moré

_____

Firma del Autor

Msc. Anthony Rafael Sotolongo León

_____

Firma del tutor

Ing. Anyer Gámez Guedes

_____

Firma del tutor

Tutores:

Ing. Anyer Gámez Guedes
Especialidad de graduación: Ingeniería en Ciencias Informáticas
Cargo: Especialista
Área: Centro de Tecnologías de Gestión de Datos (DATEC)
Correo Electrónico: aguedes@uci.cu
Universidad de las Ciencias Informáticas, Habana, Cuba

Msc. Anthony Rafael Sotolongo León
Especialidad de graduación: Ingeniería en Ciencias Informáticas
Cargo: Líder de Proyecto.
Área: Centro de Tecnologías de Gestión de Datos (DATEC)
Correo Electrónico: asotolongo@uci.cu
Universidad de las Ciencias Informáticas, Habana, Cuba

Agradecimientos de Raydel Wilson Moré

Agradezco a cada una de las personas que de una forma me ayudaron durante estos cinco años de estudio con mucho sacrificio, a todos los profesores que me impartieron clases y que me guiaron en mi preparación para forjarme como un futuro Ingeniero en Ciencias Informáticas, en especial a los profes Lacoste, Vladimir, Yanet Parra, Oscar, Yuraisy, Kiel, Pedro Puig, Yudelkis, Vilma Orozco, Glennis Tamayo, Aliosmi, a la tia Migdalia y a mis tutores por tanta paciencia y dedicación para conmigo en el desarrollo de la tesis Anyer Gámez y Anthony R Sotolongo.

A mis compañeros de aula que siempre tuve su ayuda y amistad incondicional en especial a Sureny, Yuned, Yailema, Migue, Lieny, Raúl, Poti, Osman, Javier, José Carlos, Tony Dize, Maikel, Danaisy, Rayner, Suly, Yanet, Tahimy, Dasley, Osvaldo, Anita Espinosa, Adrian Cruz, Darlon, mi ahijado Leandry, Jova.

A mi dúo de tesis Marquito, por tener tanta paciencia conmigo y por ofrecerme su amistad.

A mis amigos de los años que siempre han confiado en mí en especial a Doris, Laura, Olivia, Vilma, Tony, Susana, Dayana, Annia, Cecilia, Osmaro, Carlos Lavado, Carlos Ortiz, Alejandro, Yaima, mi comadre, Diana, Elaine y Hermes.

A mis amistades y vecinos que de toda la vida han cuidado de mí procurando hacerme un hombre de bien en especial a Ñaña, Pilar, Clarita, Nidia, Gladis, Niurka, mi ahijada Aymara e Isauri, Nancy, Yolandita, Yarini, Varo, Maritza, mima Hilda, mi padrino Osmarito, mis tíos Yanet y Oscarito, mis madrinas y mi padrino Barbarita, Dulce y José Antonio, Sergito.

A mi familia, por siempre haber contado con su ayuda y apoyo incondicional, a mis hermanos Raulito e Ivan, mi cuñi Lizaeth, mis sobrinos Rachel, Raulitin e Ivansito, mis tios Elba, Aisar, Maba y a mi familia por parte de madre mis tías Berta, Flor, mis primas Jenny, Dainelis, Lianet, Odalis, Bebo, mis ahijados mi cucu Daniela y Cristian, Deyanira, a mi prima Milagros, gracias por ser como eres conmigo.

A mis padres, primero, por haberme dado la vida y haber podido llegar hoy, hasta donde estoy, les doy mil gracias por existir y por permitirme vivir este momento inolvidable en mi vida y por confiar en mí. Gracias mami y papi.

A la Santísima Caridad del Cobre por sus bendiciones.

Agradecimientos de Marcos Raúl Cordero Vázquez

Agradezco a todos mis compañeros de aula, en especial a Raúl Alejandro, Lieny, Suly la pesa, Yasmari, Poti, Ingeniero Raydel Wilson, Audrey, Javier, Osman, Maikel, TonyDize, a Kain.

Agradezco a mis profesores Miguel Mon, Eliomar, David Conde, Vladimir Martell el pipi y Jeovany.

Agradezco a todos los que me han acompañado en los años de estudio, a la Angada, Lex Karel, Dayron, Luisito, Yonnys, Ronald, Dayaikel, el piquete del tizazo, a La Gometa, José Mestre alias Julito, a Carly la Yutong, a Dejamon, Riqui, Luis Ángel, a Lele, Pepo, Elio, Yunier la gente de la JAI, a la Panchanga, Cristian, María, José, Mabí, Jigue, Jiguito, Yeyo, Ivan, Carin y Pupy.

Agradezco a mi suegra Rosita, Eyaine, Amelia, Alexander.

A mi familia, mis hermanos Eduardito y Bética, a mis abuelos por parte de padre, Bety, mi papá Marco Antonio, mi tío Raulito, mi primo Guirindango, mi prima Carla, Mary, a mi Chipojita, mi mamá Yadira.

Le agradezco a Dios por mantener mi fe y darme fuerza para lograr mis sueños.

Dedicatoria de Raydel Wilson Moré

Quisiera dedicarles la tesis a toda mi familia y amigos que han estado conmigo en las buenas y en las malas y darles las gracias por existir y por haber confiado en mí. En especial quiero dedicarles mi trabajo de diploma a cuatro personas que han sido muy importante en mi vida, la primera persona es mi hermano Raulito por ser ese amigo, padre y hermano que es él, por darme buenos consejos y apoyarme y ayudarme en todo lo que me hizo falta, la segunda persona es mi prima Mily, por ser mi Ada Madrina como le digo yo, eres un ejemplo para mí en la vida de como una persona lucha y se sacrifica por lo que quiere venciendo todo tipo de obstáculos al igual que mi hermano.

A mi padre, por ser siempre el ejemplo de sabiduría, perseverancia, dignidad, disciplina, abnegación, sacrificio y lucha por todos los sueños que uno como persona se propone en la vida y por demostrarme que en la vida hay que confiar en uno mismo para poder lograr cosas buenas y bonitas. Y la última persona a la que le dedico mi trabajo de diploma es a mi madre, si no fuera por ella nada de esto hubiera pasado si no me hubieses traido al mundo, gracias por existir, gracias por darme la vida, gracias por educarme de la forma que lo hiciste, gracias por estar siempre a mi lado aconsejándome y preparándome para la vida, gracias por ser amiga, hermana, mujer y madre, gracias por ser como eres, te quiero exactamente cómo eres hoy, mañana y siempre. Te dedico el día de hoy por ser la mejor madre del mundo. Gracias mami.

Dedicatoria de Marcos Raúl Cordero Vázquez

Quisiera dedicarles mi trabajo de diploma más que a nadie a mis abuelos Ofelia por criarme como el hombre de la casa, darme fuerza para enfrentar todo tipo de situaciones, por regañarme cuando me comportaba mal, por ayudarme a realizar todos mis sueños y a mi abuelo Raúl Vázquez por dármelo todo, por ser mi papá y amigo, por su compañía y amor, por pensar siempre en mí y tenerme presente toda su vida. Papi donde quiera que estés mi triunfo es tuyo, tu nietecito ya es ingeniero.

Resumen

El presente trabajo surge de la necesidad de hacer portables los datos almacenados entre los gestores de bases de datos PostgreSQL, CouchDB y MongoDB, manteniendo la información a pesar de poseer diferentes modelos de datos. Por esta razón se desarrolló una aplicación que permite el proceso de migración de datos entre estos gestores de base de datos antes mencionados. Debido a esto, se realizó un estudio profundo del funcionamiento de cada una de las herramientas que están relacionadas con este tema, así como las técnicas utilizadas en este tipo de herramientas. La realización de la aplicación denominada MIGDAT constituye una alternativa de gran ayuda para el proceso de migración entre estos gestores, tanto en los ambientes productivos, como no productivos.

Palabras Claves: migración de datos, PostgreSQL , CouchDB y MongoDB.

Índice

Índice de figuras

Índice de tablas

Introducción

El desarrollo tecnológico y el aumento de grandes flujos de datos en el mundo han hecho necesaria la utilización de sistemas informáticos donde centralizar los datos, con el objetivo de conservar la información y permitir futuras búsquedas, las que pueden resultar complejas. Debido a esto, con el surgimiento de la Informática se desarrollan los Sistemas Gestores de Base de Datos (SGBD), los que han evolucionado permitiendo almacenar, organizar y realizar consultas de datos en breves períodos de tiempo y de forma eficiente.

El lenguaje más común definido para ejecutar consultas a bases de datos relacionales es SQL (Lenguaje de Consulta Estructurado). Este lenguaje se convirtió en un estándar de facto, por ser descriptivo y extensible además de ser muy utilizado por la mayoría de los motores de bases de datos más destacados que se basan en el modelo relacional, como es el caso de Firebird, MySQL, SQLite, Apache Derby y PostgreSQL.

Con la creciente tendencia de consultar grandes cúmulos de datos con mayor rapidez, muchas bases de datos y almacenes de información están expandiéndose de forma exponencial, lo que hace que las entidades tengan que ampliar su capacidad de almacenamiento con frecuencia. La llegada de estas nuevas necesidades y requisitos, han hecho emerger nuevos paradigmas centrados en las llamadas bases de datos NoSQL (Not Only SQL), que manejan los datos de forma estructurada. Priorizados en la simplicidad y el rendimiento. (1)

Los sistemas NoSQL proponen una estructura de almacenamiento más versátil. Algunas implementaciones bien conocidas que se pueden mencionar son RavenDB, Neo4j, Cassandra, BigTable, Riak, Hadoop, MongoDB y CouchDB, las cuales garantizan el manejo de enormes cantidades de datos, facilidad de distribución, alta escalabilidad y excelentes tiempos de respuestas. Ejemplo de instituciones en el mundo que utilizan las bases de datos NoSQL, son: Source Forge, OpenSky, SecondMarket, Disney, Santosoft, entre muchas otras. (1)

Las bases de datos NoSQL se clasifican en orientadas a columnas, Key-Value (llave-valor), grafos y a documentos en las cuales se encuentran como las mas usadas MongoDB y CouchDB, destacándose en

la integración con sistemas de compleja información, la utilización de componentes que pueden ser usados como bloques de construcción, flexibilidad, el rendimiento y la fiabilidad, incluso cuando se requiere de una escalabilidad cambiante y rápida, su rico, pero sencillo método de consulta hacia contenidos de la Base de Datos, además presentan un buen balance entre rendimiento y funcionalidad.

A pesar de sus puestas en práctica en grandes firmas no son idóneas para todo, de hecho en la mayoría de los casos, las bases de datos relacionales deberían seguir siendo la primera opción. La capacidad de hacer modelos relacionales y las garantías de atomicidad, consistencia, aislamiento y durabilidad son muy importantes para muchas aplicaciones.

Mejorar el desempeño de la base de datos, cumplir con nuevos requerimientos de usuario; así como la compatibilidad con otras aplicaciones, la actualización de versiones, la reducción de costos, el aumento en el volumen de información, nuevos procesos de negocio, entre otros escenarios posibles; son muchas de las causas por lo que se hace necesario la migración de datos y comprende de una forma implícita la importancia que repercute en el mundo. (2)

Existen muchas empresas que son sometidas a procesos de migración de los datos, ya sea entre bases de datos relacionales a NoSQL o viceversa, tal así es el caso de la Organización Europea de Investigaciones Nucleares que utilizan CouchDB (32) y el periódico The Guardian, en el cual se pone de manifiesto la migración de base de datos relacional a no relacional como es el caso de MongoDB, obteniendo ganancia en rendimiento, todo esto con el objetivo de ganar en escalabilidad, velocidad y en clientes, los cuales son cada vez más exigentes. (2)

Las bases de datos NoSQL debido a la importante carencia que presentan las tradicionales bases de datos relacionales en cuanto a la escalabilidad, velocidad, distribución y manejo de datos estructurados, hacen que este nuevo modelo sea muy empleado en la actualidad (24). Debido a esto, en el Departamento de PostgreSQL, perteneciente al Centro de Tecnología de Gestión de Datos (DATEC), ubicado en la Universidad de Ciencias Informáticas (UCI), existe un grupo llamado Programación y Migración, el mismo brinda el servicio de migración de datos a los clientes utilizando los mecanismos y procesos no automatizados, dicho departamento ha identificado las bases de datos NoSQL orientadas a documentos como un área clave para la adopción de este nuevo modelo de almacenamiento, de hecho se han desarrollado sistemas que las incluyen como es el caso de NAIRE y el mismo no cuenta con la

experiencia suficiente ni las herramientas para enfrentar un proceso de migración de datos.

El proceso de migración de datos de un gestor a otro puede resultar complejo y engorroso, pues los datos son incompatibles y se necesitan grandes conocimientos de estos gestores para poder llevar a cabo este proceso. La migración se puede realizar manualmente si la persona encargada de hacerlo cuenta con estos conocimientos, pero esto resultaría muy arduo y lento si la base de datos a migrar contiene grandes volúmenes de datos.

Por lo anteriormente expuesto, se plantea como **problema de la investigación:** ¿Cómo realizar el proceso de migración de los datos entre los gestores de base de datos PostgreSQL, MongoDB y CouchDB?

Definiendo como **objeto de estudio**: el proceso de migración de datos, enmarcado en el **campo de acción**, proceso de migración de datos en los gestores de base de datos PostgreSQL, MongoDB y CouchDB.

Para solucionar el problema planteado, se identifica como **objetivo general** de la investigación: desarrollar una herramienta informática que permita realizar la migración de los datos entre los gestores de base de datos PostgreSQL, MongoDB y CouchDB.

En correspondencia con el mismo, se definen los siguientes **objetivos específicos**:

- Realizar el análisis de técnicas y tecnologías para la migración de datos entre los gestores de Base de Datos PostgreSQL, MongoDB y CouchDB.
- Realizar la implementación de la herramienta para la migración de datos entre los gestores de Base de Datos PostgreSQL, MongoDB y CouchDB.
- Realizar la validación de la herramienta para la migración de datos entre los gestores de Base de Datos PostgreSQL, MongoDB y CouchDB.

Para darle cumplimiento a los objetivos específicos planteados, se propone la realización de las siguientes **tareas de la investigación:**

- Caracterización de las metodologías, herramientas y tecnologías a utilizar en el desarrollo de la herramienta, para la migración de datos entre los gestores de Base de Datos PostgreSQL, MongoDB y CouchDB.
- Identificación de las funcionalidades de la herramienta para la migración de datos entre los gestores de Base de Datos PostgreSQL, MongoDB y CouchDB.
- Diseño de la herramienta a partir de las funcionalidades identificadas.
- Implementación de las funcionalidades identificadas.
- Diseño de los casos de pruebas para la herramienta de migración de datos entre los gestores de Base de Datos PostgreSQL, MongoDB y CouchDB.
- Aplicación de los casos de pruebas para validar que la herramienta responde a las necesidades del usuario final.

El presente trabajo de diploma está estructurado en 3 capítulos:

Capítulo I: “Fundamentos teóricos de la investigación”: El capítulo comprende el estudio de los aspectos teóricos que soportan la investigación a desarrollar. Se realiza además, un análisis de las metodologías, herramientas y tecnologías a utilizar para el desarrollo de la herramienta de migración de datos entre los gestores de base de datos PostgreSQL, MongoDB y CouchDB.

Capítulo II: “Características y Diseño del sistema”: En el presente capítulo se analizan cada uno de los artefactos y elementos para el diseño de la aplicación, lo que permitirá un mejor entendimiento de las necesidades de la herramienta a desarrollar. Además se expondrán las principales características de la herramienta, su diseño, su arquitectura y los requisitos a tener en cuenta en su desarrollo.

Capítulo III: “Implementación y Pruebas”: En el capítulo se describen los principales artefactos generados en la fase de implementación y prueba de la herramienta de migración de datos entre los gestores de base datos PostgreSQL, MongoDB y CouchDB. Se desarrolla la descripción de la implementación del sistema y se especifican las pruebas a las que fue sometida la herramienta de Migración de Datos en cada una de las iteraciones. Proceso que guía la identificación y corrección de fallos cometidos en las HU, así como su verificación y materialización. Contribuye a elevar la calidad del producto desarrollado y la seguridad en los programadores para efectuar modificaciones.

Capítulo I: Fundamentos teóricos de la investigación

Introducción al Capítulo I

El capítulo comprende el estudio de los aspectos teóricos que soportan la investigación a desarrollar. Se realiza además un análisis de las metodologías, herramientas y tecnologías a utilizar para el desarrollo de la herramienta de migración de datos entre los gestores de base de datos PostgreSQL, MongoDB y CouchDB.

1.1 Conceptos asociados a la Investigación

1.1.1 Migración de datos

Una migración de BD es un proceso que se realiza para mover o trasladar los datos almacenados de un formato de datos a otro, para lo cual es indispensable que antes de empezar cualquier proceso de esta naturaleza, se tenga clara y documentada la razón por la cual se está migrando, además de elaborarse la planeación detallada de las actividades contempladas. (3)

Existen diversos motivos para hacer una migración, tales como: mejorar el desempeño de la BD, cumplir con nuevos requerimientos de usuario, de la aplicación o políticas de seguridad; así como la compatibilidad con otras aplicaciones, la actualización de versiones, la estandarización de la tecnología de información, la reducción de costos que se puede tener al cambiar por software libre y nuevos procesos de negocio. (3)

Según Susana Laura Corona (Dirección General de Servicios Cómputos, Universidad Nacional Autónoma de México), el proceso de migración cuenta con una series de fases o etapas que no se pueden violar, los cuales son conocidos como factores críticos de éxitos. (3)

Factores Críticos de Éxitos de una Migración de Datos:

Son los elementos o aspectos que resultan esenciales para que se alcancen los mejores resultados de la herramienta. Cuyos resultados satisfactorios aseguran un proyecto exitoso de migración de datos.

- Planeación de la fuente de datos: en esta etapa se deben establecer los objetivos, alcance, estrategias y fases a seguir.

- Mapeo de la Información: donde se tienen en cuenta las características de cada uno de los elementos y tablas con las cuales se va a trabajar, así como su relación e integridad origen.
- Selección de la Herramienta: es importante hacer un análisis de las diferentes alternativas existentes buscando la mejor opción considerando la relación costo-beneficio de cada una de ellas.
- Pruebas: pruebas de las aplicaciones que usarán la base de datos migrada y pruebas del proceso de migración.
- Migración: procesos de extracción, transformación y carga, los cuales permiten obtener los datos desde su origen, modificarlos para cumplir con la integridad y la consistencia e insertarlos finalmente en la BD destino.
- Resultados: medición y análisis de los resultados.

1.1.2 Base de Datos (BD)

Una Base de Datos (BD) es un conjunto de datos interrelacionados entre sí, almacenados con carácter relativamente permanente en la computadora. O sea, que una BD puede considerarse una colección de datos variables en el tiempo. (4)

1.1.3 Modelos de BD.

Un modelo de datos es básicamente una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos y conceptos matemáticos. (5)
Algunos modelos con frecuencia utilizados en las bases de datos son:

- Bases de datos jerárquicas.
- Base de datos de red.
- Bases de datos orientadas a objetos.
- Bases de datos deductivas.
- Bases de datos distribuidas.

Además de los modelos anteriormente expuestos también se encuentran los modelos relacionales y no relacionales, con los cuales se trabajará para la realización de la herramienta de migración de datos.

Base de datos relacional

Una base de datos relacional es una base de datos basada en un modelo relacional. El término se refiere a una colección específica de datos, pero a menudo es usado como sinónimo del software utilizado para gestionar esa colección de datos. Ese software se conoce como sistema gestor de base de datos relacional o RDBMS (Relational Database Management System). (5)

Este es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que esta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla). (5)

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia. Esto tiene la considerable ventaja que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL (Structured Query Language o Lenguaje Estructurado de Consultas), un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos, el cual elimina posibles errores de diseño y posibilita que no existan problemas tales como la redundancia de datos y otros elementos importantes a la hora de diseñar e implementar una base de datos relacional.

Bases de Datos no Relacional

Las bases de datos NoSQL son sistemas de almacenamiento de información que no cumplen con el esquema entidad-relación. Mientras que las bases de datos relacionales basan su funcionamiento en tablas, joins y transacciones ACID, las bases de datos NoSQL no imponen una estructura de datos en forma de tablas y relaciones entre ellas, ya que suelen permitir almacenar información en otros formatos como clave-valor, Mapeo de Columnas, Documentos o Grafos.

La principal característica de las bases de datos NoSQL es que están pensadas para manipular enormes cantidades de información de manera muy rápida. Están preparadas para escalar horizontalmente sin perder rendimiento. (6)

Clasificación de las Bases de Datos no Relacional

BD No Relacional 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-Cabbinet y MemcacheDB entre otras. (7)

BD No Relacional Orientada a Documentos:

Estas almacenan la información como un documento (generalmente con una estructura simple como JSON o BSON) y con una llave única. Podemos encontrar a MongoDB y CouchDB entre las más importantes de este tipo. (7)

BD No Relacional Orientada a Grafos:

En el caso de estas bases de datos 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. Encontramos a Neo4j, Infinite Graph entre otras. (7)

BD No Relacional Orientada a Columnas:

Esto es un 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. (7)

Las bases de datos relacionales tradicionales permiten definir la estructura de un esquema que demanda reglas rígidas y garantizan atomicidad, consistencia, aislamiento y durabilidad (ACID). Las aplicaciones web modernas presentan desafíos muy distintos a las que presentan los sistemas empresariales tradicionales: sistemas bancarios, datos a escala web, alta frecuencia de lecturas y escrituras, cambios de esquema de datos frecuentes. Las aplicaciones sociales (no bancarias) no necesitan el mismo nivel de ACID. En la actualidad se muestra un mayor uso en los modelos relacionales y no relacionales debido a las demandas que emergen en el mundo actual. La típica base de datos relacional cuenta hoy día con PostgreSQL, MySQL, Oracle entre otras aplicaciones y algunas de las opciones de NoSQL actualmente disponibles son: Cassandra, MongoDB, CouchDB, BigTable y Dynamo.

1.1.4 Sistemas Gestores de Base de Datos (SGBD)

Los Sistemas Gestores de Base de Datos (SGBD) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. (8)

Los SGBD comerciales surgen por la necesidad de sustituir los métodos de almacenamiento basados en Sistemas de Archivos los cuales se limitaban a la estructuración física de los datos. Los SGBD pueden verse como una sola estructura de almacenamiento mediante restricciones de integridad permitiendo manipular la información a través de esquemas que establecen métodos de acceso. (8)

Los SGBD libres surgen por la necesidad imperiosa de ahorrarle a pequeñas y medianas empresas el dinero correspondiente al pago de las licencias así como por las numerosas ventajas que implica utilizar SGBD cuyos principios estén basados en el Software Libre.

Dentro de los SGBD libres se encuentran un conjunto de implementaciones, como son PostgreSQL, MongoDB y CouchDB con los cuales se estará trabajando para el desarrollo de la herramienta.

PostgreSQL 9.1

La implementación del SGBD PostgreSQL comenzó en 1986, los conceptos iníciales para el sistema fueron presentados en “The Design of Postgres”. En 1996, el proyecto cambia su concepto al mundo del código abierto e inicia su versión 6.0. En el año 2000 se comienza la implementación del soporte. Corre el año 2004, y ya PostgreSQL es reconocido como uno de los mejores motores de bases de datos del mundo. En el año 2005, PostgreSQL pasa la prueba de Coverty Inspected, en la cual encontraron sólo 20 errores en 775,000 líneas de código, lo cual constituyó un orgullo y un compromiso para el proyecto. (9)

La versión de PostgreSQL 9.1 proporciona un gran número de características que normalmente solo se encontraban en las bases de datos comerciales tales como Oracle o SQL Server. A continuación se presentan un conjunto de características.

Lenguajes Procedurales

PostgreSQL 9.1 tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL 9.1 es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido. (10)

SGBD Objeto-Relacional

PostgreSQL 9.1 aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arreglos. (10)

Arquitectura Cliente-Servidor

Usa una arquitectura proceso-por-usuario cliente/servidor, hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL 9.1. (10)

Alta concurrencia

Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés), el cual permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Esta estrategia es superior al uso de bloqueos por tabla o por filas comunes en otras bases, eliminando la necesidad del uso de bloqueos explícitos. (11)

Amplia variedad de tipos nativos.

PostgreSQL provee nativamente soporte para:

Números de precisión arbitraria.

Texto de largo ilimitado.

Figuras geométricas (con una variedad de funciones asociadas)

Direcciones IP (IPv4 e IPv6).

Arreglos.

Licencia de PostgreSQL

Está bajo la licencia de PostgreSQL basada en BSD (Berkeley Software Distribution). (12)

Altamente Extensible

Soporta operadores funcionales, métodos de acceso y tipos de datos definidos por el usuario. (10)

Integridad Referencial

Soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la BD. (10)

Multiplataforma

PostgreSQL 9.1 es un SGBD multiplataforma. (9)

CouchDB 1.0.1

El proyecto de software libre nace en Abril de 2005 de la idea de Damien Katz de crear un nuevo motor de bases de datos orientado a documentos para entornos web.

En un primer momento CouchDB estaba escrito en C++ y ya incluía algunas de las características esenciales del proyecto, tales como: la falta de esquema relacional, almacenamiento y actualizaciones atómicas. En estos inicios, el proyecto estaba muy influenciado por Lotus Notes (sistema cliente/servidor) y su idea de base de datos orientada a documentos. (13)

Couch (sofá, diván) DB (Database) es un nuevo concepto de bases de datos, de la mano de la comunidad Apache. El nombre no ha sido elegido al azar, ni de manera casuística. Su nombre evoca relax, y ésa es su filosofía. Evoca tranquilidad, debido a que todo ahora es más sencillo y produce menos problemas,

incluso para personas no técnicas. Lo primero que llama la atención de CouchDB es que no sigue el concepto de base de datos tradicional entidad-relación (modelo relacional), sino un modelo de almacenamiento documental, gestionado por un potente motor de consultas sencillísimo de utilizar. (13)

Las BD CouchDB guardan documentos nombrados de modo unívoco y se guardan en formato JSON o BSON que permite a las aplicaciones leer y modificar estos documentos.

Cada documento puede tener campos no definidos en otros documentos. Los mismos no están asociados a un esquema de bases de datos estricto. Cada documento contiene metadatos (datos sobre datos) como el identificador unívoco del documento (id) y su número de revisión (rev). Los campos de un documento pueden ser de varios tipos como strings, números, booleanos, colecciones entre otros. Cuando se hacen cambios sobre un documento CouchDB se crea una nueva versión del documento, denominado revisión. Se mantiene un historial de modificaciones gestionado automáticamente por la BD. CouchDB no dispone de mecanismos de bloqueo (locking) ante escrituras. (13)

La primera versión publicada, únicamente disponible para Windows, la versión 0.2 fue publicada en Agosto del 2006. Otro de los puntos de inflexión y que generó un gran interés fue el anuncio de lo siguiente:

1. Se desecha XML por JSON (JavaScript Notation Object) para la transferencia de datos.
2. Se desecha el lenguaje de consulta y validación y se adopta Javascript con la intención de integrar Mozilla. (13)

Tras estos hechos, lo más reseñable es que en Noviembre de 2008 CouchDB pasa a ser un proyecto Apache de primer nivel, al lado de otros como Apache HTTP Server, Tomcat, Ant, entre otros. (13)

Características principales de CouchDB 1.0.1:

- Plataforma de bases de datos simplificada
- Centrada en documentos
- No sigue el modelo relacional
- Facilita la distribución, alta escalabilidad y la tolerancia a fallos
- Preparada para funcionar offline
- Replicación bidireccional

- Orientada a Internet

Arquitectura de CouchDB 1.0.1

En su motor late un corazón tolerante a fallos, los cuales se producen en un entorno controlado, y son tratados de forma que no afectan al sistema servidor, si no en peticiones aisladas. No se esperan comportamientos aleatorios ni errores en las operaciones. Asimismo, gestiona perfectamente un entorno de tráfico variable, absorbiendo peticiones concurrentes sin errores, tomando para ello, más tiempo para cada petición hasta su finalización. Por otra parte, CouchDB está diseñado para escalabilidad extrema, permitiendo el crecimiento o la disminución del hardware, y la replicación incremental. (13)

CouchDB toma como referente el teorema de CAP:

Consistency: Consistencia o la capacidad de que todos los clientes vean los mismos datos, incluso con actualizaciones concurrentes.

Availability: Disponibilidad o la capacidad de que todos los clientes accedan a la misma versión de los datos.

Partition tolerance: Partición tolerante o la capacidad de que la base de datos pueda estar dividida en múltiples servidores.

Concurrencia de CouchDB 1.0.1:

CouchDB utiliza un sistema de Control de Concurrencia Multi Versión (MVCC en inglés), para gestionar el acceso concurrente a la base de datos, ejecutando en paralelo las peticiones, incluso en un sistema de carga extrema. Los documentos son versionados con un sistema de control de versiones (de forma similar a Subversión). Cada actualización genera una nueva versión del documento y lo guarda por encima del antiguo. Esto optimiza el rendimiento concurrente, ya que una lectura puede leer la última versión mientras aún se está actualizando, sin tener que esperar la actualización. (13)

Licencia de CouchDB 1.0.1:

CouchDB es adoptado por IBM y posteriormente por Apache Foundation con lo que la licencia pasa de ser GNU GPL (General Public License) a ser Apache License. (13)

MongoDB 2.0.4

MongoDB es el sistema de base de datos desarrollada por Geir Magnusson y Dwight Merriman. Es una base de datos orientada a documentos JSON, salvo que está diseñada para ser una verdadera base de

datos de objetos, más que para un almacenamiento de clave/valor puro. (14)

Es una base de datos no relacional, es decir, no utiliza SQL. El nombre viene del término inglés “humongous” (colosal) y puede ser definida como una BD documental sin esquema, escalable y de alto rendimiento. Algunos especialistas la han catalogado como la “MySQL de las bases de datos NoSQL”, al ver que es una BD rapidísima, sencilla en la funcionalidad ofrecida y que, como hizo MySQL hace años, se está ganando la atención de mucha gente. (14)

Es un sistema de base de datos multiplataforma orientado a documentos, de esquema libre. Esto significa que con cada entrada o registro se puede tener un esquema de datos diferente, con atributos o “columnas” que no se tienen porque repetir de un registro a otro. Está escrito en C++, con lo que es bastante rápido a la hora de ejecución de tareas. (14)

Para almacenar los documentos, utiliza una serialización binaria de JSON, llamada BSON, que es una lista ordenada de elementos simples. El núcleo de la base de datos es capaz de interpretar su contenido, de modo que lo que a simple vista parece un contenido binario, realmente es un documento que contiene varios elementos.(14)

Licencia de MongoDB 2.0.4:

Está licenciado como GNU AGPL 3.0, de modo que se trata de un software de licencia libre.

Almacenamiento orientado a documentos:
Documentos estilo JSON con esquemas dinámicos ofrecen simplicidad y poder.

Consultas:
Ricas y basadas en documentos, rápidas para consultas de lectura y escritura básica.

Sistemas Operativos Compatibles:
Windows, Linux, Mac OS X y Solaris.

Alto rendimiento (14)

- La falta de juntes y poder incrustar hacen rápidas las lecturas y escrituras.
- Índices que incluyen indizar claves de documentos empotrados y vectores.

Soporte comercial:

Soporte comercial, capacitación y consultoría disponibles.

Teniendo en cuenta el estudio analizado de estos gestores de BD, se tiene una fuerte y clara definición sobre la importancia que repercute cada una de sus características en el desarrollo de la herramienta, garantizándose de esta manera la correcta selección de los mismos.

1.2 Metodología de Desarrollo de Software (XP)

1.2.1 XP (Extreme Programming)

En todo proceso de desarrollo de software se hace necesario el uso de una guía que posibilite el progreso eficiente en la construcción de aplicaciones. Como procedimientos y técnicas para toda la documentación y generación de artefactos del ciclo de vida del desarrollo del software, se requiere el uso de una metodología.

Una metodología para el desarrollo de un proceso de software es un conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentos y aspectos de formación para los desarrolladores de software.

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (15)

Se define esta metodología de desarrollo por brindarle la oportunidad al equipo de trabajo la comodidad y flexibilidad a los cambios que surgan en cualquier etapa de desarrollo del ciclo de vida de la aplicación. Además de ser una metodología de desarrollo orientada al progamador y brinda una serie de características que a continuación se presenta:

El desarrollo bajo XP tiene enumeradas características como son: (16)

- Desarrollo iterativo e incremental.

- Pruebas unitarias continuas.
- Integración del equipo de programación con el usuario.
- Corrección de todos los errores.
- Refactorización del código.
- Propiedad del código compartida.

Entre las características más significativas de XP se encuentran: (16)

- Orientado a la persona que produce y usa el software.
- Reduce el costo del cambio en las etapas de vida del sistema.

La metodología XP se basa en cuatro valores imprescindibles para el desarrollo de software: (16)

- **Simplicidad:** enfocado en un diseño sencillo del código generado.
- **Comunicación:** potenciada por el desarrollo en pares, la presencia del cliente y la simplicidad en cuanto al código.
- **Retroalimentación:** propiciada por el protagonismo del cliente que participa activamente y por el trabajo en ciclos cortos.
- **Coraje:** enfrentando decisiones, en ocasiones complejas, que pudieran afectar el tiempo de desarrollo y la calidad del producto.

El ciclo de vida ideal consta de 4 fases:

- **Planificación del proyecto:** en esta primera fase se realiza la recopilación de todos los requerimientos del proyecto, también debe haber una interacción con el usuario, y se debe planificar bien entre los desarrolladores del proyecto que es lo que se quiere para el proyecto para así lograr los objetivos finales.
- **Diseño:** en esta fase se logrará crear parte del proyecto, la parte física (lo bonito) la interfaz que tendrá el usuario o cliente con el proyecto.
- **Codificación:** en esta fase de la codificación los clientes y los desarrolladores del proyecto deben estar en comunicación para que los desarrolladores puedan codificar todo los necesario para el proyecto que se requiere, en esta fase está incluido todo lo de codificación o programación por parte de los

desarrolladores del proyecto.

- **Pruebas:** para esta fase lo que se implementa es el uso de test que son pruebas que se le hacen al proyecto o a los códigos que se vallan implementando.

1.3 Tecnologías y herramientas asociadas al desarrollo de la herramienta de migración de datos

1.3.1 Lenguaje de Modelado

El lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar un diseño de software orientado a objetos. Algunas organizaciones los usan extensivamente en combinación con una metodología de desarrollo de software para avanzar de una especificación inicial a un plan de implementación y para comunicar dicho plan a todo un equipo de desarrolladores. Algunos metodólogos del software orientado a objetos distinguen tres grandes "generaciones" cronológicas de técnicas de modelado de objetos. (17)

- En la primera generación, se incluye a autores y técnicas como Rumbaugh, Jacobson, Booch, los métodos formales, Shlaer-Mellor y Yourdon-Coad.
- En la segunda generación se realizaron múltiples intentos para integrar dichas técnicas en marcos coherentes tales como FUSIÓN. Se empezaba a reconocer los beneficios que la estandarización de las técnicas conllevaría: hacer las cosas de una manera adecuada, que permitiría un lenguaje y unas prácticas comunes entre los diferentes desarrolladores.
- La tercera generación consiste en intentos creíbles de crear dicho lenguaje unificado por la industria, cuyo mejor ejemplo es UML.

Lenguaje Unificado de Modelación (UML)

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema, incluye aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Uno de los objetivos principales de la creación de UML era posibilitar el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos. (17)

Es por esta razón que se seleccionó como lenguaje de modelado UML debido a todas las características que brinda el mismo, para el desarrollo de la herramienta las cuales se presentan a continuación:

De forma general las principales características son:

1. Lenguaje unificado para la modelación de sistemas.

2. Tecnología orientada a objetos.

3. El cliente participa en todas las etapas del proyecto.

4. Corrección de errores viables en todas las etapas.

5. Aplicable para tratar asuntos de escala inherentes a sistemas complejos de misión crítica, tiempo real y cliente/servidor.

Se utiliza el lenguaje UML por que indica qué es lo que supuestamente hará el sistema, no precisamente cómo lo hará. Pero constituye una guía y una representación de las especificaciones del sistema, ayudando a tomar decisiones para lograr un sistema que cumpla con todas las expectativas del cliente y los usuarios.

1.3.2 Herramienta Case

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. CASE es también definido como el Conjunto de métodos, utilidades y técnicas que facilitan el mejoramiento del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases. Realmente son las herramientas CASE el mejor método para el análisis y soluciones de software, ya que han venido a mejorar los aspectos claves en el desarrollo de los sistemas de información. (18)

Visual Paradigm 8.0

Visual Paradigm es una herramienta UML que soporta el ciclo de vida completo de desarrollo de software: análisis y diseño orientados a objetos, pruebas y despliegue. Esta aplicación ayuda a lograr mayor rapidez en la construcción de aplicaciones informáticas, reduciendo el costo a utilizar durante el desarrollo de las mismas y es un producto de alta calidad. Una de sus características fundamentales es la interoperabilidad entre diagramas, ya que es capaz de exportar los diagramas de un modelo con facilidad. (19)

Es una herramienta CASE que apoya todo el proceso de desarrollo del sistema, facil de usar y con entorno flexible de guía para los desarrolladores, respaldando cada una de las demandas que impone la herramienta para asi darle cumplimiento a cada uno de sus objetivos, esta entre otras razones es por la cual se definio como herramienta CASE para el desarrollo de la aplicación.

Características: (18)

- Diagramas de Procesos de Negocio - Proceso, Diagrama de Clases.
- Ingeniería de ida y vuelta.
- Ingeniería inversa - Código a modelo, código a diagrama.
- Ingeniería inversa Java, C++, Esquemas XML, XML, Python.
- Generación de código - Modelo a código, diagrama a código.
- Diagramas de flujo de datos.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
- Permite exportar diagramas:
  - Como imágenes: diagrama actual, todos los diagramas
  - Como PDF.

1.3.3 Lenguaje de Programación

Python

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”. Es un lenguaje que favorece un código legible. Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos. (20)

Un lenguaje interpretado o de script es aquel que se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora (lenguajes compilados). La ventaja de los lenguajes compilados es que su ejecución es más rápida. Sin embargo los lenguajes interpretados son más flexibles y más portables. Permite trabajar con mayor rapidez e integrar sus sistemas con mayor eficacia, obteniendo beneficios casi

inmediatos en la productividad y reducción de costos de mantenimiento. (20)

Python tiene, no obstante, muchas de las características de los lenguajes compilados, por lo que se podría decir que es semi interpretado. En Python, como en Java y muchos otros lenguajes, el código fuente se traduce a un pesado código máquina intermedio llamado bytecode la primera vez que se ejecuta, generando archivos .pyc o .pyo (bytecode optimizado), que son los que se ejecutarán en sucesivas ocasiones. (20)

Multiplataforma

El intérprete de Python está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que si no utilizamos bibliotecas específicas de cada plataforma nuestro programa no podrá correr en todos estos sistemas sin grandes cambios. Python es un lenguaje de scripts, sencillo, pero potente. Un script es un conjunto de instrucciones que se ejecutan paso a paso, instrucción a instrucción. Esto significa que Python no genera ejecutables, sino que es Python el encargado de ejecutar nuestro código. Es por tanto un lenguaje interpretado, no compilado. (20)

Principales Características:

- Programación estructurada y clara (el tabulador es parte del propio lenguaje).
- Alta productividad: gran velocidad de desarrollo.
- Soporta múltiples paradigmas de programación: orientada a objetos, estructurada, funcional.
- Interpretado, dinámico, fuertemente tipado, gestión de memoria automática.
- Lenguaje sencillo de aprender.

¿Por qué Python?

Los usuarios de Python consideran a este mucho más limpio y elegante para programar. Este permite dividir el programa en módulos reutilizables desde otros programas. Viene con una gran colección de módulos estándar que se pueden utilizar como base de los programas (o como ejemplos para empezar a aprender Python). También hay módulos incluidos que proporcionan E/S de ficheros, llamadas al sistema, sockets y hasta interfaces a GUI (interfaz gráfica con el usuario) como Tk, GTK y Qt. (20)

Python se utiliza como lenguaje de programación interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. El intérprete se puede utilizar de modo

interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del programa. (20)

Es un lenguaje de programación fácil de aprender y potente. Tiene eficaces estructuras de datos de alto nivel y una solución de programación orientada a objetos simple, pero eficaz. La elegante sintaxis de Python y su naturaleza interpretada hacen de él el lenguaje ideal para guiones (scripts) y desarrollo rápido de aplicaciones; en muchas áreas y en la mayoría de las plataformas, o sea, está expresamente diseñado para ser usado como una extensión para las aplicaciones que necesiten una interfaz programable. Proporciona un equilibrio muy bueno entre lo práctico y lo conceptual puesto que es un lenguaje interpretado, o sea, suele ocurrir que se vuelve algo lento en la ejecución de los programas, ya que en lugar de compilar nuestros programas escritos en este lenguaje para obtener ficheros binarios y librerías, los ejecutamos sobre la marcha. Esta característica nos trae ventajas como por ejemplo: (20)

- No hay que hacer pasos intermedios entre que tocamos el código y lo ejecutamos.
- Es más fácil encontrar errores y probar el programa, ya que el programa corre hasta que encuentra uno, en lugar de ser un compilador el que se detiene, sin haber ejecutado ninguna instrucción. (Esto puede ser también una desventaja, porque tales errores podrían ser críticos y un compilador los filtraría, pero el hecho de que sea de alto nivel hace que no sea fácil provocarlos).

Contiene una gran biblioteca de módulos que se pueden usar para hacer toda clase de tareas que abarcan desde programación para Web hasta gráficos.

Además Python cuenta con bibliotecas que respaldan y ayudan al desarrollo de la herramienta vinculadas con MongoDB y CouchDB como son Couchdbkit, Pymongo y Psycopg2.

1.3.4 Entorno de Desarrollo Integrado (IDE)

Un IDE (*Integrated Development Environment*) en español Entorno Integrado de Desarrollo es un programa informático compuesto por un conjunto de herramientas de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los entornos de desarrollo pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

Aptana 3.0

Es un potente IDE de desarrollo web que nos facilita el trabajo de desarrollar páginas o emprender proyectos que requieren trabajar con mucho código, como es el caso de páginas web o emprendimientos online que usan mucho código para las hojas de estilos (CSS), Javascript y HTML, en sus distintas versiones. (21)

Este IDE de desarrollo permite una transparente sincronización entre nuestro sistema operativo y el entorno de trabajo, mediante cargas de archivos usando distintos protocolos, como FTP y sFTP. (21)

Aptana Studio 3.0 se encuentra disponible de forma gratuita y entre sus mejores características destacan su integración con Eclipse, que permite visualizar en tiempo real el resultado de un proyecto cuando sea visto y ejecutado desde un navegador que definamos (Safari, Mozilla Firefox) (21). Además se pueden realizar aplicaciones de escritorios en el lenguaje Python. Se definió Aptana Studio como IDE ya que es una plataforma para la programación en diferentes lenguajes. Además cumple con los requisitos que demanda el desarrollo del software, como bibliotecas, plugin y extensiones, todas son comodidades que a criterio de los desarrolladores pone en manos de los mismos, con el objetivo de darle cumplimiento a los objetivos que define la aplicación. Brinda la posibilidad de programar en Python, haciendo uso del complemento Pydev y Extensiones Pydev.

Pydev 1.6.2 es un plugin de terceros para Aptana Studio, utilizado para programar en Python basado en el apoyo de refactorización de código y depuración gráfica. Ciertas características avanzadas tales como análisis de código, arreglos rápidos, se reservan para la versión libre Pydev 1.6.2 la cual se empleará en el desarrollo de la herramienta.

Características de Aptana Studio 3.0:

- Permite desarrollar páginas web y aplicaciones de escritorio de forma ordenada.
- Incluye plantillas de estilos para códigos determinados y personalizados.
- Incluye soporte para integración como plugin al IDE multiplataforma Eclipse.
- Incluye propia base de datos de bibliotecas.
- Soporta versiones de 64 bits.

QTDesigner 4.8.1

Para el diseño de la interfaz visual de la herramienta de migración de datos se empleara QTDesigner en su versión 4.8.1 ya que es un IDE de desarrollo que utiliza un documento XML para describir y almacenar la interfaz. Este documento sigue un formato propio denominado UI (User Interface), cuyo esquema establece una serie de elementos XML para los propios componentes, propiedades, funciones, entre otras características de la interfaz. (23)

Qt Designer es potente y a la vez fácil de utilizar además está orientado al desarrollo de aplicaciones a través del framework Qt. Está provisto de herramientas que permiten la creación de formas, etiquetas, botones y otros elementos propios de las interfaces. Permite además la modificación de las propiedades de cada elemento utilizado y una pre-visualización del diseño creado. (22)

Conclusiones del Capítulo I

En el presente capítulo se realizó un análisis de los principales aspectos y conceptos relacionados con la migración de datos, se analizaron temas de importancia dentro de las bases de datos a utilizar, dejando sentadas las bases teóricas del trabajo a desarrollar. A partir de la investigación realizada acerca de las tecnologías y herramientas se hará uso de la metodología XP para guiar el proceso de desarrollo de la herramienta de migración de datos y como herramienta para el modelado Visual Paradigm, empleando el lenguaje de modelado UML. Durante la implementación de la herramienta se hará uso del IDE de desarrollo Aptana 3.0, utilizando como lenguaje de programación Python y para el diseño de la interfaz de la aplicación se hará uso de QTDesigner 4.8.1, conjuntamente con el plugin Pydev 1.6.2.

Capítulo II: “Características y Diseño del sistema”

Introducción del Capítulo 2

En el presente capítulo se analiza cada uno de los artefactos y elementos para el diseño de la aplicación, lo que permitirá un mejor entendimiento de las necesidades de la herramienta a desarrollar. Además se expondrán las principales características de la herramienta, su diseño, su arquitectura y los requisitos a tener en cuenta en su desarrollo.

2.1 Modelo de Dominio

La metodología XP se inspira en la simplicidad, basada en desarrollar sólo el sistema que realmente se necesita, sin embargo su gran adaptabilidad permite el empleo de diagramas UML, siempre y cuando influyan en el mejoramiento de la comunicación. Debido a ello se decide realizar un modelo de dominio pues permite a los usuarios, clientes, desarrolladores e interesados, a utilizar un vocabulario común para lograr un efectivo entendimiento del contexto en que se encuentra el sistema proporcionando una perspectiva conceptual de los objetos implícitos en él. Se realiza su descripción a través de un diagrama de clases UML.

Según Ivar Jacobson, Grady Booch, y James Rumbaugh, “un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las cosas que existen o los eventos que suceden en el entorno en el que trabaja el sistema”. (25) Es una representación visual de clases conceptuales o de objetos reales en un dominio de interés.

Diagrama de clases del Modelo de Dominio.

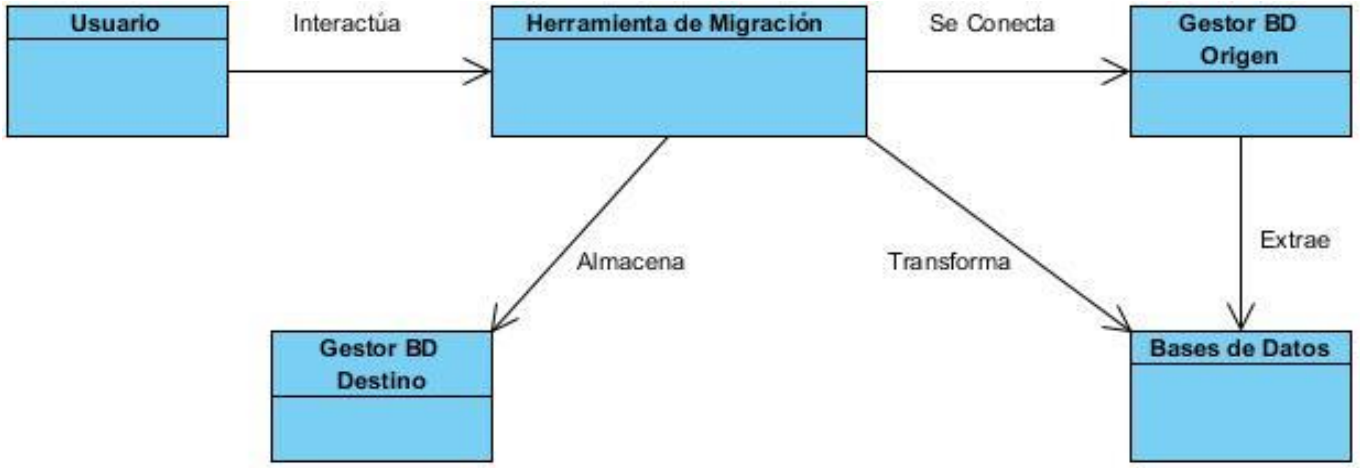

Figura 1. Modelo de Dominio.

Definición de las clases del modelo de dominio:

Usuario: Persona que interactúa con la herramienta.

Herramienta de Migración: Aplicación encargada de realizar todo el proceso de migración de los datos de un gestor a otro.

Gestor de BD Origen: Gestor donde se encuentran las bases de datos a migrar.

Gestor de BD Destino: Gestor donde se almacenarán las bases de datos migradas.

Base de Datos: Conjunto de datos.

2.2 Descripción del sistema propuesto

Para solucionar el problema planteado se decide desarrollar una herramienta de migración de datos entre los gestores de base de datos PostgreSQL, MongoDB y CouchDB, la cual va a tener como función automatizar el proceso de migración de datos. Permitirá que la pérdida de información debido al gran flujo de datos a migrar sea mínima de una manera rápida en correspondencia con el procedimiento que utiliza. Su diseño pretende colaborar con la persistencia de los datos y evitar la interferencia en la confidencialidad y persistencia de la información almacenada en las base de datos. La implementación de la herramienta estará condicionada por el siguiente procedimiento.

1. Selección de los Gestores de Base de Datos.
    - El usuario selecciona el Gestor de BD origen y destino con los cuales desea realizar la migración de datos.
2. Conexión a las Base de Datos
    - El usuario introduce todos los datos requeridos para poder realizar la conexión a las base de datos (Base de datos, Usuario, Contraseña, Host y Puerto) y luego pulsa el botón Conectar.
3. Tipos de Migración de datos.
    - Una vez seleccionado los gestores de base de datos y haber realizado la conexión correcta, se procede a seleccionar el tipo de migración (migrar formato JSON, migrar solo vistas y completa) de acuerdo con la selección de los gestores de base de datos seleccionados.
4. Finalmente el usuario pulsa el botón Migrar Base de Datos, para así obtener respuesta de la migración de datos con éxito.

Para llevar a cabo la implementación de la herramienta es necesario realizar la especificación de requisitos, los detalles del tiempo que requiere la implementación y la estimación del riesgo. Para ello en la fase de Exploración de la metodología *XP* se desarrolla un período de tiempo en el que se realiza un conjunto de funcionalidades determinadas. Se refiere a la realización de las Historias de Usuarios (HU) uno de los artefactos elementales que genera la metodología *XP*.

2.3 Historia de Usuarios

Las historias de usuarios son las técnicas utilizada en XP para especificar y administrar los requisitos del software de una forma eficaz. Son descripciones cortas y escritas en el lenguaje del usuario sin terminología técnica, debido a ello son lo suficientemente comprensibles y delimitadas para que los programadores puedan implementarlas en unas semanas. Permiten responder rápidamente a los requisitos cambiantes pues su tratamiento es dinámico y flexible.

Para el presente trabajo de diploma se obtienen un total de 6 HU que son implementadas en una iteración. A continuación en la Tabla 1 se muestra una de las HU desarrollada: “Realizar la migración de datos de PostgreSQL hacia MongoDB.” y el resto se encuentran en el Expediente de Proyecto.

Historia de Usuario	
Número: 1	**Nombre de la Historia de Usuario:** Realizar la migración de datos de PostgreSQL hacia MongoDB.
Usuario: Marcos R Cordero Vázquez	**Iteración asignada:** 1
Prioridad en negocio: Muy Alta	**Puntos estimados:** 1 semana
Riesgo en desarrollo: Alto	**Puntos reales:** 2 semanas
Descripción: El usuario accede a la interfaz principal, la cual permite seleccionar los gestores de base de datos origen y destino con los cuales se realizará la migración de datos, luego de entrar los parámetros de conexión, realiza la conexión a las base de datos correspondientes mediante el botón conectar. Por último ejecuta el botón de migrar base de datos para que este dé respuesta de éxito en la migración.	
Observaciones: Si la conexión a la base de datos origen no tuvo éxito, se muestra un mensaje de error “Error, No se pudo conectar”**.** Si la conexión a la base de datos destino no tuvo éxito, se muestra un mensaje de error “Error, No se pudo conectar”**.** Si la base de datos origen seleccionada ya existe en el gestor de base de datos destino se muestra un mensaje de error “Error, La base de datos ya existe”**.** Si el usuario al introducir datos deja campos vacíos se muestra un mensaje de error "Error: debe llenar todos los campos".	

Prototipo de interfaces:

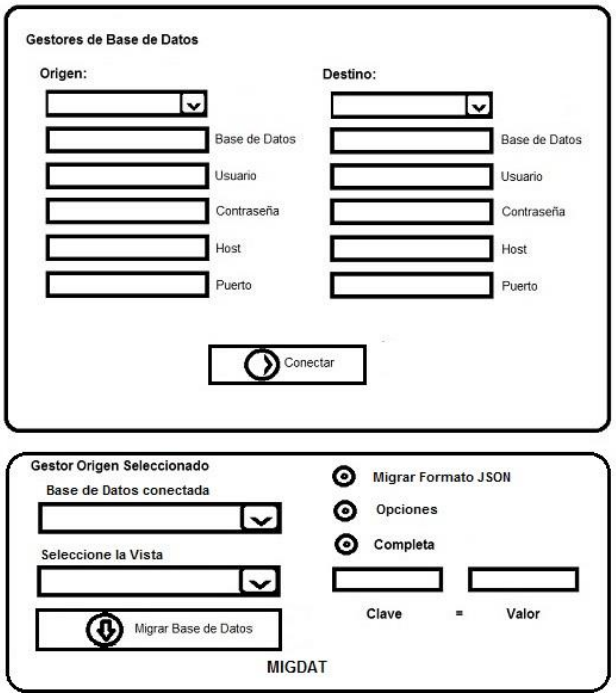

Tabla 1. Historia de Usuario "Realizar la migración de datos de PostgreSQL hacia MongoDB."

Las HU describen las funcionalidades que debe realizar la herramienta de migración de datos. Proveen información acerca de la prioridad de la funcionalidad a implementar así como el tiempo estimado de duración de dicha implementación. A continuación las HU son relacionadas en el artefacto Lista de Reserva del Producto en la cual se referencia información relevante acerca de las mismas.

2.4 Lista de reserva del producto

La lista de reserva del producto es una tabla que contiene los requisitos funcionales que debe cumplir la herramienta que se desea realizar, ordenados según la prioridad en el negocio (Muy Alta, Alta, Media y Baja). Se indica de cada uno de ellos la estimación, su implementación por semanas y el rol que lo estimó. Contiene por último los requisitos no funcionales que requiere el sistema a desarrollar.

Ítem	Descripción	Estimación	Estimado por
Prioridad: Muy Alta			
1.	Realizar la migración de datos de PostgreSQL hacia MongoDB.	2 semanas	Programador
2.	Realizar la migración de datos de PostgreSQL hacia CouchDB.	2 semanas	Programador
3.	Realizar la migración de datos de MongoDB hacia PostgreSQL.	2 semanas	Analista
4.	Realizar la migración de datos de CouchDB hacia PostgreSQL.	2,4 semanas	Analista
5.	Realizar la migración de datos de CouchDB hacia MongoDB.	2 semanas	Programador
6.	Realizar la migración de datos de MongoDB hacia CouchDB.	2 semanas	Programador
Prioridad: Alta			
Prioridad: Media			
Prioridad: Baja			

Requisitos No Funcionales		
1.	**Software:** Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 12.4 o superior. Bibliotecas: Couchdbkit, Pymongo y Psycopg2. Bases de datos: SGBD PostgreSQL9.1, MongoDB2.0.4, CouchDB1.0.1, PostgreSQL 9.2.	Analista
2.	**Hardware:** Se necesita 512 MB de memoria RAM mínimo, 70 MB de espacio libre en el disco duro como mínimo para su instalación y el micro a 500 MHz.	Analista
3.	**Usabilidad:** La aplicación deberá presentar facilidades para ser utilizada por usuarios con conocimientos básicos de la herramienta de migración de base de datos y de los gestores en cuestión.	Analista
4.	**Confidencialidad:** Solo tendrá acceso a la visualización de los datos sensibles el usuario que posea los privilegios necesarios para ello.	Analista
5.	**Integridad:** La herramienta mientras esté migrando garantizará que no habrá pérdida de ninguno de los datos y otorgará un reporte en caso de error.	Analista
6.	**Disponibilidad**: Se podrá hacer uso de la aplicación siempre que esté instalada y existan clientes que deseen realizar la migración de datos entre algunos de estos tres gestores.	Analista

Tabla 2. Lista de reserva del producto.

2.5 Tareas de Ingeniería

Después de haberse realizado la definición de las Historias de Usuarios, el equipo de desarrollo divide por cada una de ellas una serie de tareas de ingenierías que contribuyen al desarrollo de las historias de usuarios. A continuación se presentan algunas de las tareas de ingenierías desarrolladas durante la realización de la herramienta y el resto se encuentran en el Expediente de Proyecto.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Realizar Conexión a PostgreSQL.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.4 semanas
Fecha Inicio: 8/01/2013	**Fecha Fin:** 10/01/2013
Programador Responsable: Raydel Wilson Moré	
Descripción: Se llama al método connect () de la biblioteca psycopg2 pasándole los parámetros que introdujo el usuario.	

Tabla 3. Tarea de Ingeniería 1.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 1
Nombre Tarea: Realizar Conexión a MongoDB.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.4 semanas
Fecha Inicio: 11/01/2013	**Fecha Fin:** 13/01/2013
Programador Responsable: Raydel Wilson Moré	
Descripción: Se llama al método Connection () de la biblioteca Pymongo pasándole los parámetros que introdujo el usuario.	

Tabla 4. Tarea de Ingeniería 2.

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 1
Nombre Tarea: Realizar migración de PostgreSQL hacia MongoDB.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.8 semanas
Fecha Inicio: 17/01/2013	**Fecha Fin:** 21/01/2013
Programador Responsable: Marcos R Cordero Vázquez	
Descripción: Mediante el método Migrando_desde_PostgreSQL_a_MongoDB (), en la cual se extraen cada una de las tablas de PostgreSQL que conformarían una colección en MongoDB y cada una de las filas de estas tablas con sus campos conformarían un documento en su colección correspondiente y por último son salvados los campos. Los datos que produzcan error al ser insertados van hacer almacenados en un archivo para que el usuario los inserte manualmente.	

Tabla 5. Tarea de Ingeniería 4

Tarea de Ingeniería	
Número Tarea: 9	**Número Historia de Usuario:** 3
Nombre Tarea: Realizar migración de MongoDB a PostgreSQL (condicional).	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.8 semanas
Fecha Inicio: 14/02/2013	**Fecha Fin:** 18/02/2013
Programador Responsable: Marcos Raúl Cordero Vázquez	

Descripción: Mediante el método Migrando_desde_MongoDB_a_PostgreSQL (). Si el usuario escoge migrar los documentos en formato Json, si la versión de PostgreSQL que se está usando es superior a la 9.2, mediante el método Migrando_desde_MongoDB_a_PostgreSQL_JSON () se extraen los documentos de MongoDB de cada una de las colecciones, se crea una tabla por cada colección y se le van insertando los documentos correspondientes, sino si escoge migración condicional el usuario debe pasar la condición y mediante el método Migrando_desde_MongoDB_a_PostgreSQL_Condicional () se extraen solo por cada colección los documentos que cumplan esa condición, luego para cada documento se extraen la claves, se concatenan creando una cadena y se agrupan los documentos que tengan esta cadena igual. Estos forman grupos de documentos semejantes, luego por cada grupo se crea una tabla y los documentos transformados en filas son insertados, si no escoge ninguna de estas, la migración se hará completa de igual forma que la condicional pero serian todos los documentos. Los datos que produzcan error al ser insertados van hacer almacenados en un archivo para que el usuario los inserte manualmente.

Tabla 6. Tarea de Ingeniería 9

Tarea de Ingeniería	
Número Tarea: 13	**Número Historia de Usuario:** 4
Nombre Tarea: Realizar migración de CouchDB a PostgreSQL.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.8 semanas
Fecha Inicio: 3/03/2013	**Fecha Fin:** 6/03/2013
Programador Responsable: Marcos Raúl Cordero Vázquez.	
Descripción: Mediante el método Migrando_desde_CouchDB_a_PostgreSQL (). Si el usuario escoge migrar en formato Json, se crea una tabla en PostgreSQL y se inserta cada uno de esos documentos haciendo uso del método Migrando_desde_CouchDB_a_PostgreSQL_JSON (). Si el usuario escoge migrar solo vistas haciendo uso del método Migrando_desde_CouchDB_a_PostgreSQL_Solo_Vistas (), que recibe solo el nombre de la vista que desea migrar, se conecta a la vista e iterando sobre ella se extraen sus documentos, luego para cada documento se extraen la claves, se concatenan creando una cadena y se agrupan los documentos que tengan esta cadena igual, por cada grupo se crea una tabla y se inserta cada documento del grupo transformado en las filas de esa tabla, si no escoge ninguna de esa opción la migración se realiza completa con las misma idea de las vistas pero esta vez haciendo esta operación para todos los documentos que son insertados en sus tablas correspondientes una vez que sean transformados en filas. Los datos que produzcan error al ser insertados van hacer almacenados en un archivo para que el usuario los inserte manualmente.	

Tabla 7. Tarea de Ingeniería 13

2.6 Plan de Iteraciones

Las HU después de ser descritas, identificadas y estimado el esfuerzo propuesto para la realización de cada una de ellas, se especifica cuáles serán implementadas en cada iteración del sistema por lo que se procede a la realización de un plan de iteraciones.

La metodología XP aporta mayor valor a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. El plan de iteraciones es empleado para la planificación, donde el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas.

Ralease	Descripción de la iteración	Orden de la HU a implementar	Duración total
1.	Tiene como objetivo desarrollar las HU con prioridad muy alta las cuales forman la base de la herramienta a desarrollar. Son las encargadas de realizar el proceso de migración de datos, la conexión, listar las bases de datos, realizar las migraciones completas o condicionales y finalmente realizar la migración correctamente.	HU 1 HU 2 HU 3 HU 4 HU 5 HU 6	12,4 semanas

Tabla 8. Plan de iteraciones.

2.7 Modelo de diseño

En la fase de diseño, XP propone mejorar la comunicación entre todos los integrantes del equipo, al crear una visión global y común de lo que se quiere desarrollar para lograr un diseño sencillo pues *"Kent Beck dice que en cualquier momento el diseño adecuado para el software es aquel que: supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos",*(26) además Pressman asegura: *"el diseño es el lugar donde se fomentará la calidad del software"*. (27)

Es preciso describir qué clases existen y cómo interactúan, para ello la metodología XP utiliza ciertas técnicas, llamadas tarjetas Clase, Responsabilidad y Colaboración (CRC). Sin embargo se puede hacer uso de diagramas UML, siempre y cuando influyan en el mejoramiento de la comunicación y se enfoquen en la información elemental. Por dicha razón se realiza un diagrama de las clases a implementar para mostrar sus relaciones y dependencias utilizando notación UML.

2.7.1 Diagrama de clases

El diagrama de clases facilita una abstracción de la implementación del sistema y admite una mejor compresión de los atributos, relaciones y métodos que contienen las clases, obteniéndose así una

representación significativa de lo que se va a construir de forma tal que satisfaga todos los requisitos funcionales. A continuación en la Figura 2 se muestra el diagrama de clases del diseño para la herramienta de migración de datos entre los Gestores de Base de Datos PostgreSQL, MongoDB y CouchDB. (25)

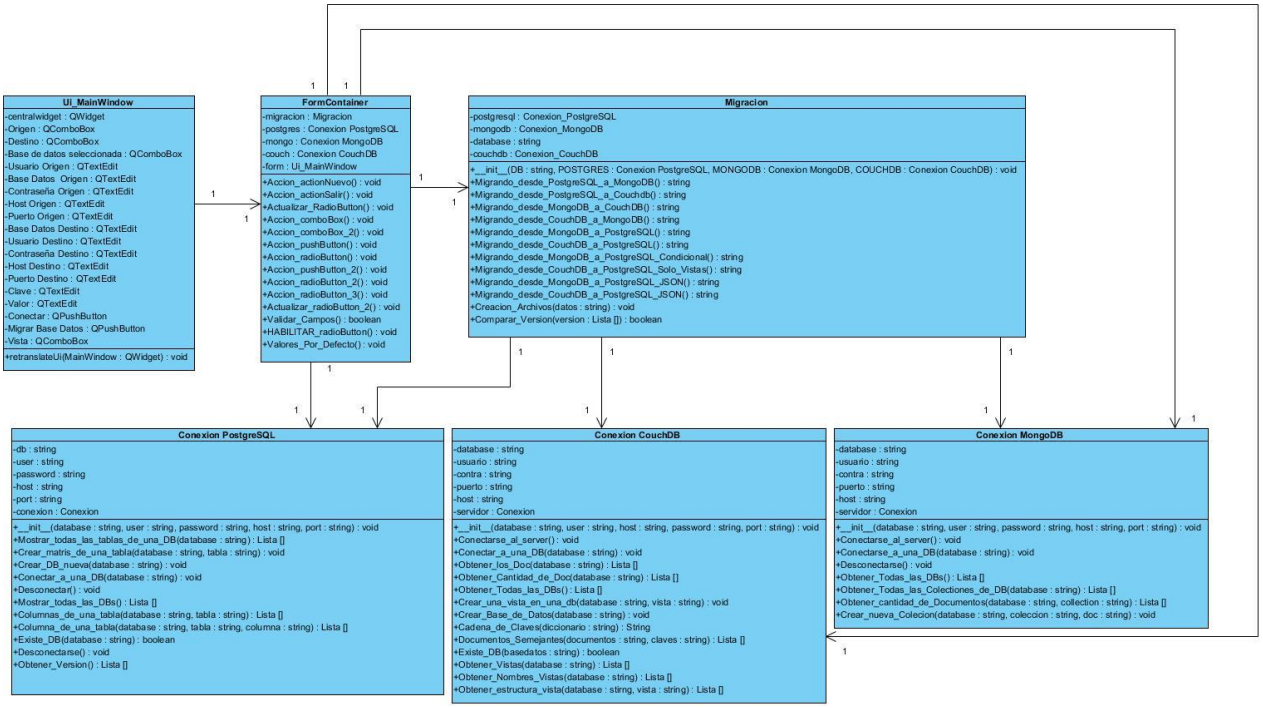

Figura 2. Diagrama de Clases

2.7.2 Tarjetas Clase, Responsabilidad y Colaboración

Las tarjetas CRC son una técnica que admiten diseñar el sistema en conjunto, para ello se debe cumplir con tres principios: Cargo o Clase, Responsabilidad y Colaboración (CRC). Permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Las tarjetas CRC permiten que el equipo completo contribuya en la tarea del diseño representando un objeto o clase de agrupamiento. La clase a la que pertenece el objeto se puede escribir en la parte superior de la tarjeta, en una columna a la izquierda se escriben las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad.

A continuación se muestran las tarjetas CRC generadas para el diseño de la herramienta de migración de datos.

1. Tarjetas CRC “Migración”

Tarjeta CRC	
Clase: Migración	
Responsabilidades	**Colaboraciones**
Migrar base de datos de PostgreSQL a MongoDB. Migrar base de datos de PostgreSQL a CouchDB. Migrar base de datos de MongoDB a PostgreSQL. Migrar base de datos de MongoDB a CouchDB. Migrar base de datos de CouchDB a PostgreSQL. Migrar base de datos de CouchDB a MongoDB. Migrar base de datos de MongoDB a PostgreSQL condicional. Migrar base de datos de CouchDB a PostgreSQL solo vistas. Migrar base de datos de MongoDB a PostgreSQL en JSON. Migrar base de datos de CouchDB a PostgreSQL en JSON. Creación Archivos. Comparar Versión.	Conexión PostgreSQL Conexión MongoDB Conexión CouchDB FormContainer

Tabla 9. Tarjetas CRC “Migración’

2.8 Patrones de arquitectura

La arquitectura MVC fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; lo que hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas.

Se decide utilizar Modelo-Vista-Controlador pues entre sus ventajas se encuentran permitir que la conexión entre el Modelo y sus Vistas sea dinámica, es decir, se produce en tiempo de ejecución y no en tiempo de compilación. Detalla las responsabilidades exactas de cada capa y la forma que tienen de relacionarse entre sí y permite un acoplamiento entre la Vista y el Controlador.

El hacer uso del patrón provee las ventajas de crear independencia de funcionamiento, facilitar el mantenimiento en caso de errores y ofrece maneras más sencillas de probar el correcto funcionamiento del sistema. Sus tres componentes son:

Modelo: Contiene todo el código relacionado con el acceso a datos. Es primordial que el código sea lo más genérico posible y se pueda reutilizar en otras situaciones y proyectos. Nunca se incluirá lógica en el modelo, solamente consultas a la base de datos y validaciones de entrada de datos. A continuación se muestra un ejemplo donde se evidencia el patrón Modelo, para este caso se pone de ejemplo a estas tres clases (Conexión PostgreSQL, Conexión CouchDB, Conexión MongoDB) en las cuales se manifiesta en cada una de ellas el acceso a datos y consultas a la base de datos.

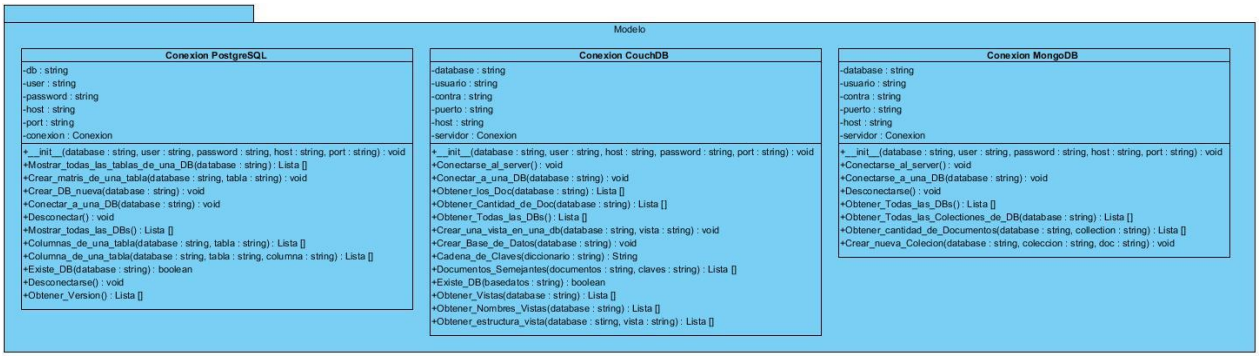

Figura 3. Capa Modelo

Vista: Contiene el código que representa la parte que será visualizada en pantalla por el usuario. A continuación un ejemplo de la clase interfaz (Ui_ MainWindow) donde se pone de manifiesto el patrón vista.

Figura 4. Capa Vista

Controlador: Es el punto de entrada de la aplicación, se mantiene a la escucha de todas las peticiones, ejecuta la lógica de la aplicación y muestra la vista apropiada para cada caso. A continuación un ejemplo de las clases (Conector, Migración) donde se evidencia el patrón de controlador, las mismas tienen las responsabilidades de ejecutar cada una de las peticiones lógicas que exige la herramienta.

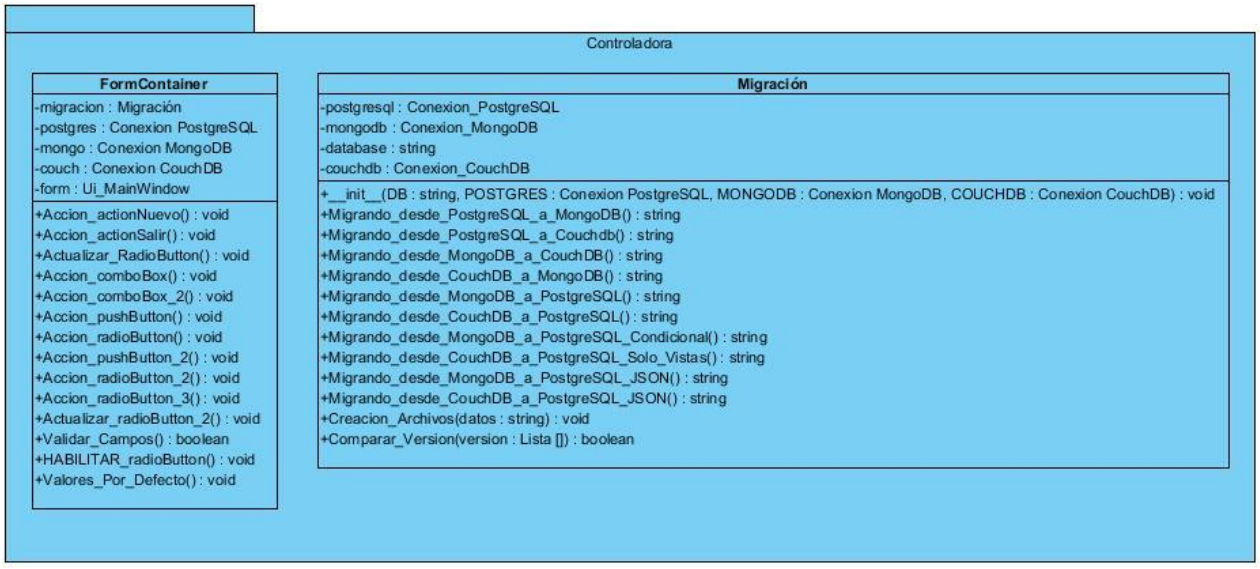

Figura 5. Capa Controlador

2.9 Patrones de Diseño

Los patrones de diseño son soluciones probadas y documentadas a problemas comunes, se emplean como estructura y soporte para el desarrollo de un sistema. En el desarrollo de múltiples aplicaciones hay problemas de diseños que se repiten o que son análogos, es decir, que responden a un cierto patrón. Con el uso de patrones los diseños serán mucho más flexibles, modulares y reutilizables. Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos.

2.9.1 Patrones GRASP

GRASP acrónimo que significa General Responsability Assignment Software Patterns (Patrones de Software para la Asignación General de Responsabilidad). Las patrones GRASP constituyen la base del cómo se diseñará el sistema. “Describen los principios fundamentales de diseño de objetos y la asignación de responsabilidades, expresados como patrones” (28). De los diferentes patrones que ofrece GRASP se ha tenido en cuenta para la modelación de la herramienta de migración de datos los siguientes:

Patrón Experto: El patrón garantiza que la responsabilidad de la creación de un objeto o la

implementación de un método, recaiga sobre la clase que conoce toda la información necesaria para crearlo lo que contribuye a un adecuado encapsulamiento, favoreciendo la robustez y fácil mantenimiento del sistema. (29) A continuación un ejemplo de la clase de Migración la cual contiene todos los métodos de migraciones de datos por lo que es la clase especialicidad en realizar todos los procesos de esta naturaleza.

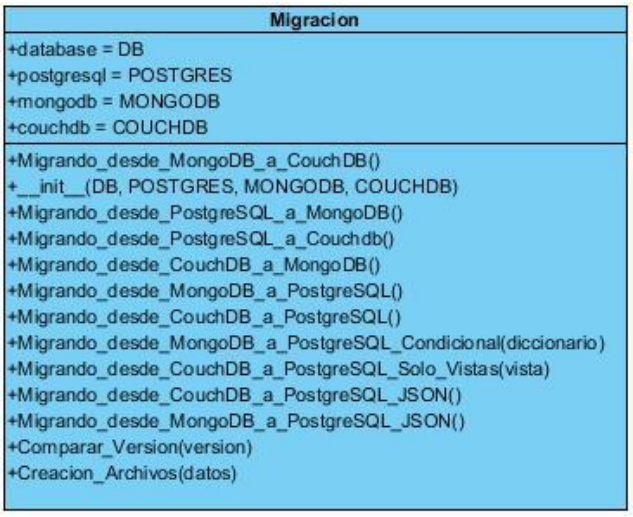

Figura 6. Ejemplo del uso del patrón experto.

Patrón Creador: Se asigna la responsabilidad a una clase de crear cuando contiene, agrega, compone, almacena o usa otra clase, lo que brinda una alta posibilidad de reutilizar la clase creadora. *(29)* A continuación se presenta un ejemplo evidenciando el patrón creador donde la clase Migración crea instancias de los objetos ConexiónPostgreSQL, ConexiónMongoDB y ConexiónCouchDB para implementar sus funcionalidades.

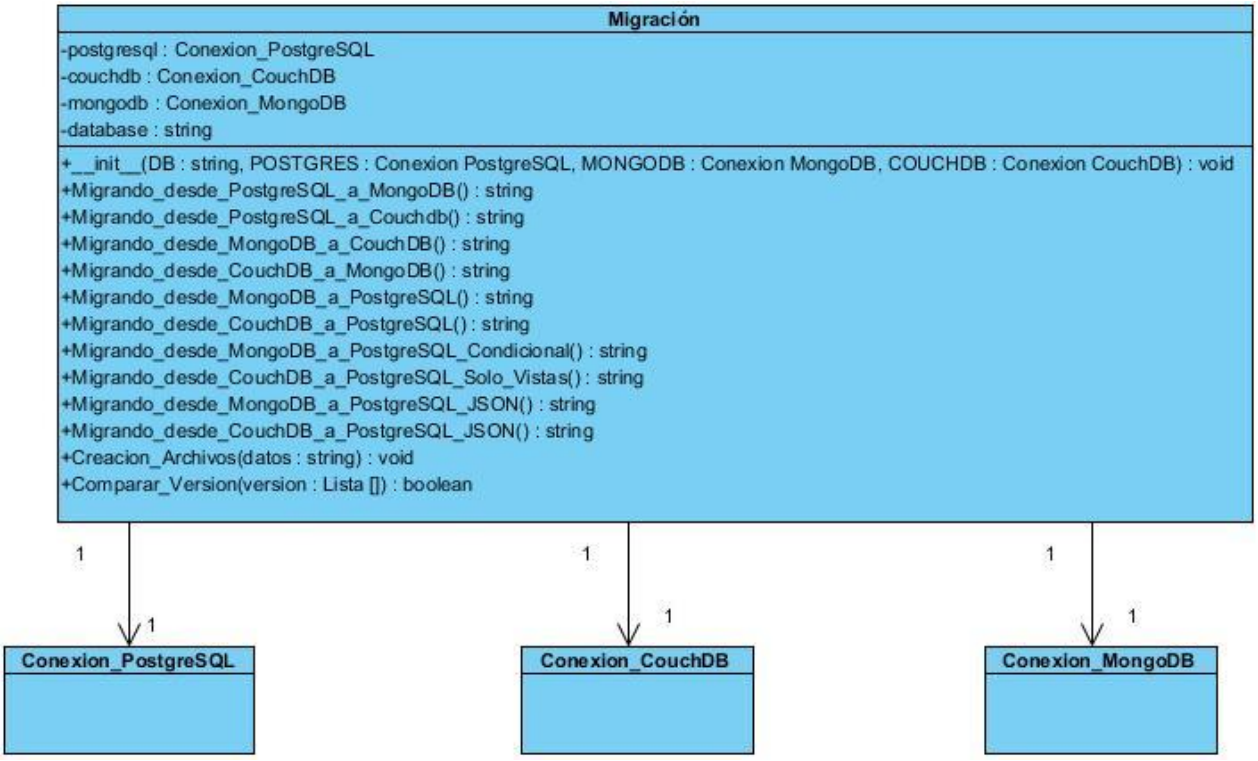

Figura 7. Ejemplo del uso del patrón creador.

Patrón Controlador: Asignar la responsabilidad de administrar un mensaje de evento del sistema a una clase que representa el sistema global, dispositivo, subsistema o representa un escenario de caso de uso en el que tiene lugar el evento del sistema. *(29)* A continuación se presenta un ejemplo donde se pone de manifiesto el patrón controlador en las clases Migración y Conector dado que contiene las responsabilidades para raelizar las migraciones de datos y permitir mediante la interfaz darle cumplimiento a las funcionalidades.

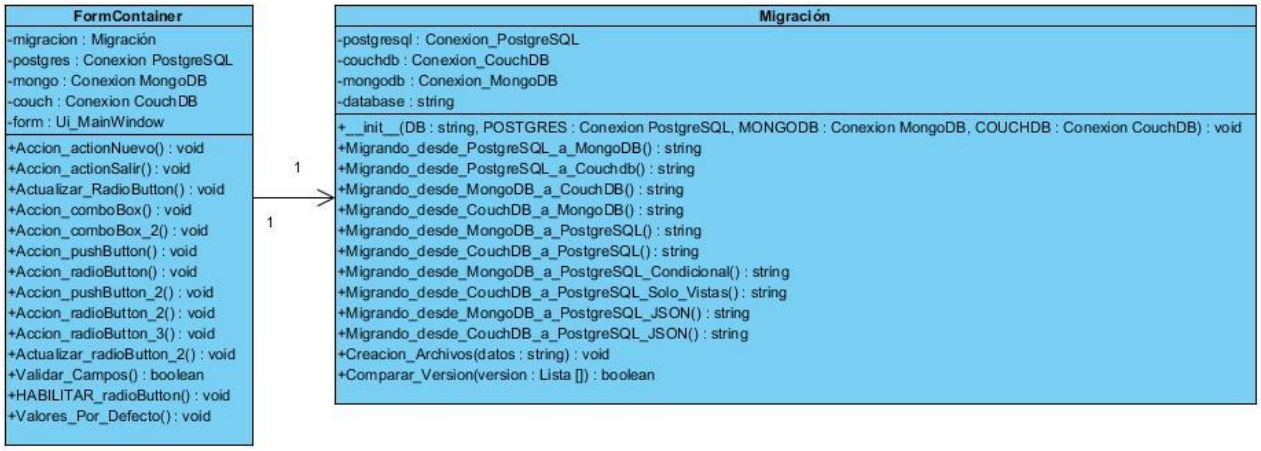

Figura 8. Ejemplo del uso del patrón controlador.

Alta Cohesión: Asignar responsabilidades de manera que una clase no tenga muchas funcionalidades no relacionadas o no realice un trabajo excesivo. Este patrón incrementa la claridad y facilita la comprensión del diseño. *(29)* A continuación se presenta un ejemplo donde se manifiesta el patrón alta cohesión donde la clase ConexiónCouchDB contiene toda la información sobre la conexión a la base de datos CouchDB y sus consultas.

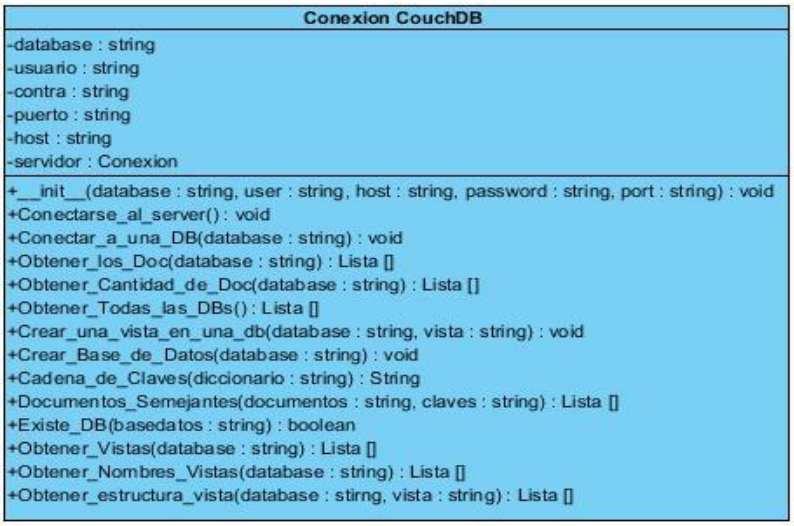

Figura 9. Ejemplo del uso del patrón alta cohesión.

Conclusiones del Capítulo II

Se identificaron 6 HU las cuales responden a los requisitos funcionales de la aplicación así como se definió la iteración en la que será implementada cada HU. Se especificaron los requisitos no funcionales a tener en cuenta a la hora de utilizar la aplicación. Para el diseño de la aplicación se identificaron 6 tarjetas CRC, donde se utilizó para la modelación del sistema el estilo arquitectónico MVC y los patrones de diseño que ofrece GRASP, facilitando el trabajo del diseño de la aplicación para la migración de datos.

Capítulo III: “Implementación y Pruebas”

Introducción del Capítulo 3

En el presente capítulo se describen los principales artefactos generados en la fase de implementación y prueba de la herramienta de migración de datos entre los gestores de base datos PostgreSQL, MongoDB y CouchDB. Se desarrolla la descripción de la implementación del sistema y se especifican las pruebas a las que fue sometida la herramienta de Migración de Datos en cada una de las iteraciones. Proceso que guía la identificación y corrección de fallos cometidos en las HU, así como su verificación y materialización. Contribuye a elevar la calidad del producto desarrollado y la seguridad en los programadores para efectuar modificaciones.

3.1 Estándares de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. “*XP enfatiza la comunicación de los programadores a través del código*“ (31) y plantea que estos pueden realizar cambios en cualquier parte del código en cualquier momento, por lo cual es indispensable que se sigan ciertos estándares de programación manteniendo el código legible para los miembros del equipo, de forma tal que se faciliten los cambios.

Emplear técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es suma relevancia para lograr la calidad del software pues lo convierte en un sistema fácil de comprender y mantener.

El estándar asegura que todos los programadores del proyecto trabajen de forma coordinada y comprensible para obtener un código fuente legible, pues repercute directamente en lo bien que se comprende un sistema de software, aspecto que es indispensable para lograr estabilidad del código, ya que con la misma el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento.

Python tiene un estándar propio definido, pero se define este por la experiencia de desarrolladores en otros lenguajes donde se tiene un estándar definido y se decidió hacer un híbrido entre el pep8 (estándar de Python) y el que proponemos.

3.1.1 Comentarios

Es conveniente dejar información que pueda ser leída tiempo después por personas (posiblemente el mismo programador) que necesitan entender que fue lo que se hizo en el fragmento de código. Los comentarios deben ser escritos correctamente y claros.

Generalmente deben usarse comentarios de una sola línea. Reserve los comentarios de bloques para la documentación formal o para comentar porciones de código.

A continuación un ejemplo del código de fuente donde se evidencia los comentarios de una funcionalidad.

```
"""Aquí migro los datos desde CouchDB para PostgreSQL pero solo las vistas""
def Migrando_desde_CouchDB_a_PostgreSQL_Solo_Vistas(self,vista):
    documentos = self.couchdb.obtener_estructura_vista(self.database,vista)
    if(len(documentos)>0):
        #aquí realizo la conexión con postgres
        #aquí creo la Base de Datos en postgres y me conecto
        self.postgresql.Crear_DB_nueva(self.database)
        conect_postgres = self.postgresql.Conectar_a_una_DB(self.database)
        posicion=0
        tachados=[]
        problema=[]
        while posicion < len(documentos):
            dic=documentos[posicion]
            if(not(dic in tachados)):
                #preparo el nombre de la tabla
                nombre_tabla = ""
                nombre_tabla = "Tabla_"+str(posicion)
```

Figura 10. Ejemplo de Comentarios.

3.1.2 Declaración de variables

Cada variable debe de ser declarada en una línea y comentada.

El nombre de las variables debe de comenzar con letras minúsculas y cada palabra relevante por la que esté compuesta debe ser con letra mayúscula.

```
#preparo el nombre de la tabla
nombre_tabla = ""
nombre_tabla = "Tabla_"+str(posicion)
```

Figura 11. Ejemplo de declaración de variables

3.1.3 Identificadores

Los identificadores pueden estar formados por cualesquiera de las 26 letras minúsculas o mayúsculas (A... Z, a... z), los 10 dígitos (0... 9) y el carácter subrayado "_". Debe evitarse el uso de caracteres internacionales (ej: ñ, ü) porque no siempre pueden ser leídos o entendidos correctamente en todos los lugares. No se debe usar el símbolo dólar "$" o la barra invertida "\" en los identificadores.

3.1.4 Sentencias

5. Sentencias Simples

Cada línea debe contener como máximo una sentencia. Se debe tener en cuenta que una sentencia de asignación puede resultar en la asignación de una función o de un objeto como literal.

```
#me conecto a la Base de Datos que deseo migrar
db = self.mongodb.Conectarse_a_una_DB(self.database)
#recojo los nombres de todas las colecciones de esa Base de Datos
lista_Collections = db.collection_names()
#y elimino esta colección dado que no importa su contenido para realizar la migración
if 'system.indexes' in lista_Collections:
    lista_Collections.remove("system.indexes")
#aquí van a estar todos los diccionarios de mongo
```

Figura 12. Ejemplo de sentencias simples

6. Sentencias Compuestas

Las sentencias compuestas son aquellas sentencias que contienen una lista de sentencias. Contienen estructura de control como if o for. Ello facilita agregar nuevas sentencias sin la introducción accidental de errores.

```
#en estos dos ciclos anidados voy llenando la lista documentos[]
for colec in lista_Collections:
    coll = db[colec]
    for documento in coll.find(diccionario):
        documentos.insert(len(documentos),documento)
```

Figura 13. Ejemplo de sentencias compuestas

3.2 Interfaz principal de la aplicación

La interfaces de aplicación es el medio con que el usuario puede comunicarse con un software, y comprenden todos los puntos de contacto entre el usuario y éste. Teniendo en cuenta las diferentes características que definen la metodología XP, en cuanto a la apariencia e interfaz externa, el sistema deberá ser fácil de usar y poseer un entorno agradable al usuario final.

La Herramienta de Migración de Datos (MIGDAT) consta de una Interfaz Principal que le permite al usuario realizar diferentes tipos de migraciones de datos. La interfaz de MIGDAT está compuesta por componentes que permiten una correcta navegación en la misma. En la parte superior de la aplicación se le brinda al usuario la selección de los gestores origen y destinos respectivamente entre los que se va a realizar la migración y la introducción de todos los datos para realizar la migración de una base de datos dados y la conexión a estos gestores.

En la parte inferior del software se le brinda al usuario la posibilidad de realizar o no la migración de datos completa, conjuntamente con las vistas, llave –valor o en formato Json.

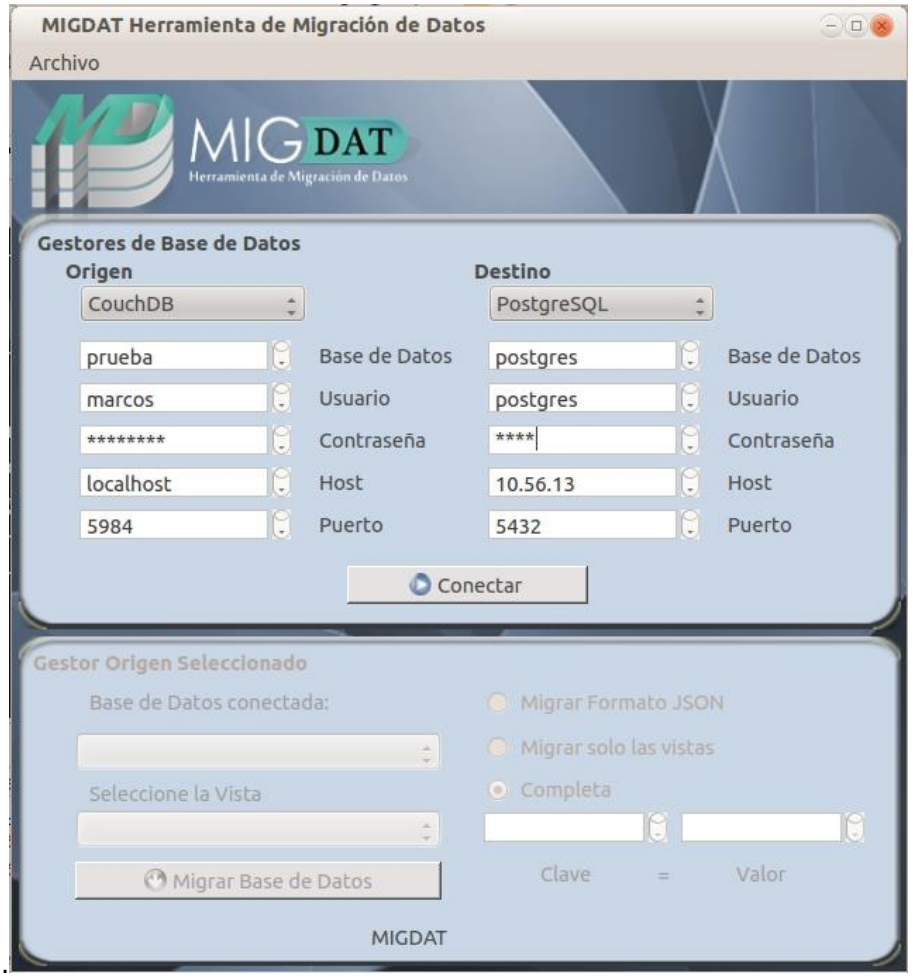

Figura 14. 1Imagen de la aplicación.

3.3 Pruebas

La fase de pruebas es una de las fases fundamentales del desarrollo de una aplicación. El objetivo de cada una de las pruebas no es el de prevenir errores sino de detectarlo basándose en técnicas y estrategias empleadas en cada una de las pruebas. Además son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un programa.

3.3.1 Niveles de pruebas

A la hora de evaluar dinámicamente un sistema software se debe comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto.

Las pruebas se aplican durante todo el ciclo de desarrollo del software para diferentes objetivos y en distintos niveles de trabajo, dentro de estos se distinguen: (27)

- Pruebas de Desarrollador
- Pruebas Independientes
- Pruebas de Unidad
- Pruebas de Integración
- Pruebas de Sistema
- Pruebas de Aceptación

En el desarrollo de la fase de pruebas de la herramienta de migración de datos se aplicarán las pruebas de desarrollador y de aceptación para probar que el sistema cumpla con las historias de usuarios previamente definidas en la fase de análisis y diseño.

3.3.2 Técnicas de Pruebas

Funcionales

- Prueba funcional

Asegurar el trabajo apropiado de las historias de usuarios, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados. Además se prueba una funcionalidad completa, donde pueden estar implicadas una o varias clases y la propia interfaz de usuario.

3.3.3 Pruebas de Aceptación

Las pruebas de aceptación son creadas sobre la base de las HU. El cliente debe especificar uno o diversos escenarios para comprobar que las HU han sido correctamente implementadas. Las mismas son consideradas como “pruebas de caja negra”. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. En caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una HU no se puede considerarse terminada hasta que pase todas las pruebas de aceptación. Para asegurarse que la aplicación desarrollada cumple sus requisitos, a continuación se representan algunas de las pruebas de aceptación realizadas a las HU, el resto se encuentran en el expediente de proyecto en la planilla: Casos de pruebas de aceptación.

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de Usuario: 1**
Nombre: Realizar la migración de datos de PostgreSQL hacia MongoDB.	
Descripción: Evaluar que se realice la migración de datos correspondiente.	
Condiciones de ejecución: Deben estar seleccionados los gestores de BD correspondientes.	
Entrada/Pasos de ejecución: Se introduce todos los datos necesarios para realizar la conexión a los gestores de BD correspondiente, luego se presiona el botón conectar. Después presiona el botón Migrar Base de Datos.	
Resultados Esperados: La herramienta muestra como resultado Datos migrados exitosamente.	
Evaluación de la prueba: Satisfactoria.	

Figura 15. Caso de prueba de aceptación_1

Caso de prueba de aceptación	
Código: HU4_P1	**Historia de Usuario: 4**
Nombre: Realizar la migración de datos de CouchDB hacia PostgreSQL.	
Descripción: Evaluar que se realice la migración de datos correspondiente.	
Condiciones de ejecución: Deben estar seleccionados los gestores de BD correspondientes.	
Entrada/Pasos de ejecución: Se introduce todos los datos necesarios para realizar la conexión a los gestores de BD correspondiente, luego se presiona el botón conectar. Después presiona el botón Migrar Base de Datos.	
Resultados Esperados: La herramienta muestra como resultado Datos migrados exitosamente.	
Evaluación de la prueba: Satisfactoria.	

Figura 16. Caso de prueba de aceptación_2

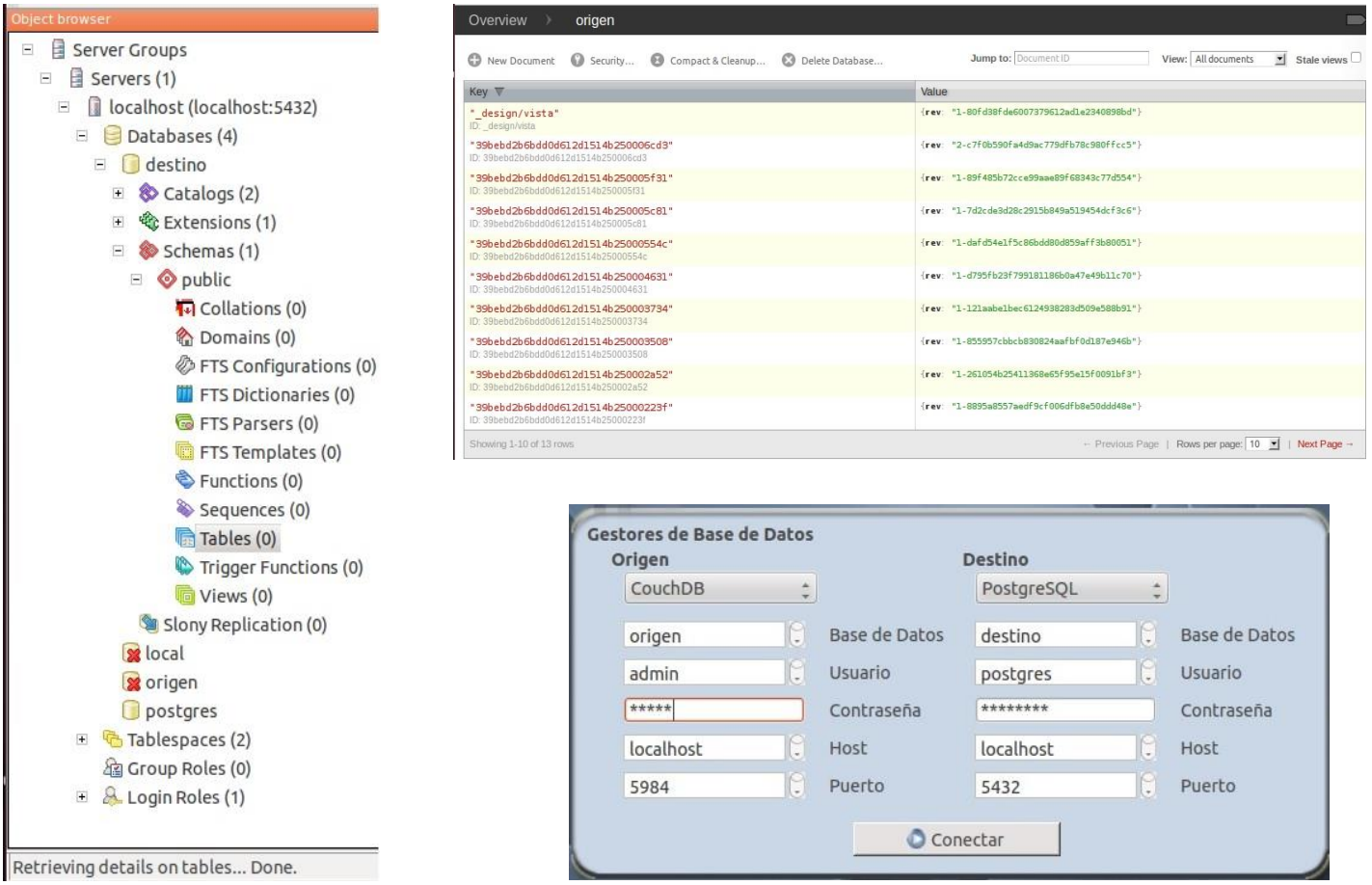

Figura 17. Ejemplo de prueba de aceptación_1

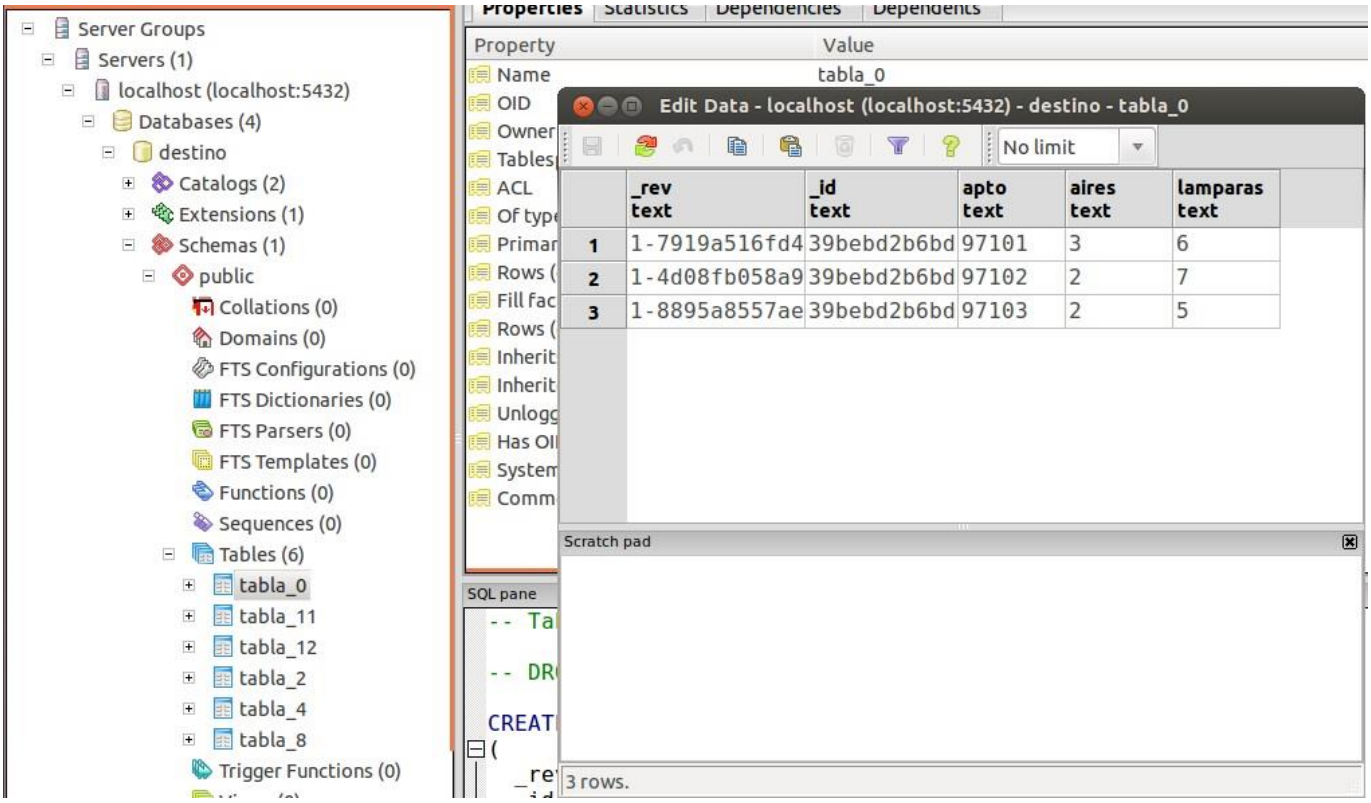

Figura 18. Ejemplo de prueba de aceptación_2

En el caso de que existan datos sin migrar serán guardados en un archivo (error.log). A continuación un ejemplo

Figura 19. Ejemplo de error de datos sin migrar

3.3.4 Métodos de Pruebas

- Caja negra

También conocidas como Pruebas de Comportamiento, estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente se ve como una “Caja Negra” cuyo comportamiento sólo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas. (27)

Las pruebas de caja negra pretenden demostrar que las funciones del software son operativas y que funcionan correctamente aceptando de forma adecuada la entrada de datos y produciendo una salida correcta. Este tipo de prueba nos permite demostrarle al cliente que la aplicación puede satisfacer las necesidades del mismo. Se decide aplicar la técnica de caja negra debido a que es más importante presentarle al cliente que las funcionalidades de la aplicación estén correctamente funcionando y así dejarlo satisfecho y conforme con el producto. (27)

Técnica utilizada para el método caja negra

- Partición de Equivalencia

La partición de equivalencia es una técnica del método de prueba de Caja Negra que divide el campo de entrada de un programa en variables con juegos de datos de entrada y salida. En esencia, esta técnica intenta dividir el dominio de entrada de un programa en un número finito de variables de equivalencia. De tal modo que se pueda asumir razonablemente que una prueba realizada con un valor representativo de cada variable es equivalente a una prueba realizada con cualquier otro valor de dicha variable. (30)

Las variables de equivalencia representan un conjunto de estados válidos y no válidos para las condiciones de entrada de un programa. Éstas se identifican examinando cada condición de entrada (normalmente una frase en la especificación) y dividiéndola en dos o más grupos. Se definen dos tipos de variables de equivalencia, las válidas, que representan entradas válidas al programa, y las no válidas, que representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato. (30)

3.3.5 Casos de pruebas basados en historias de usuarios

En la actividad Diseño de los Casos de Prueba, usando técnicas de caja negra se desarrollan los casos de prueba. Esta actividad incluye diseñar las pruebas e identificar los datos de prueba. Para cada funcionalidad a probar, se crea una matriz que muestra su correspondencia con los casos de prueba, conociéndose de esta forma que ítem cubren los casos de pruebas. A través de los casos de pruebas se verifica si la aplicación realmente funciona, en ellos se describen los diferentes escenarios y se indica la respuesta correcta que el sistema debe mostrar en cada escenario. Se debe dejar bien claro en cada caso de prueba el flujo central de la aplicación que no es más que los pasos a seguir para trabajar en la aplicación a la hora de probar cada historia de usuario.

En la siguiente tabla se detallan las variables que se encuentran asociadas a la Historia de Usuario: Realizar la migración de datos de PostgreSQL hacia MongoDB.

No	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Base de Datos	Campo texto		Nombre de la BD origen
2	Usuario	Campo texto		Nombre del usuario
3	Contraseña	Campo texto		Contraseña del usuario
4	Host	Campo texto		Host de la BD origen
5	Puerto	Campo texto		Puerto de la BD origen
6	Base de Datos	Campo texto		Nombre de la BD destino
7	Usuario	Campo texto		Nombre del usuario
8	Contraseña	Campo texto		Contraseña del usuario
9	Host	Campo texto		Host de la BD destino
10	Puerto	Campo texto		Puerto de la BD destino

Tabla 10. Variables asociadas a la HU

Esta descripción posibilitó que se realizara una matriz de datos, donde se evaluó y probó la validez de cada uno de los datos introducidos en el sistema, específicamente en la sección que se estuvo probando. Utilizando un juego de datos válidos e inválidos se identificó el empleo de la técnica de partición de equivalencia.

Nombre de la HU: Realizar la migración de datos de PostgreSQL hacia MongoDB.

Escenario	Descripción	V 1	V 2	V 3	V 4	V 5	V 6	V 7	V 8	V 9	V 10	Respuesta del sistema	Flujo central
EC 1.1 Introducir datos correctos.	Se introduce datos correctamente para conectarse a las base de datos origen y destino respectivamente.	V	V	V	V	V	V	V	V	V	V	El sistema muestra un mensaje que se conectó exitosamente.	El usuario introduce los datos necesarios para conectarse a la base de datos y luego presiona el botón conectar.
		(postgres)	*(postgres)*	*(postgres)*	*(localhost)*	*(5432)*	*(test)*	*(admin)*	*(admin)*	*(localhost)*	*(27017)*		
EC 1.2 Introducir datos incorrectos	*Se introduce datos incorrectos.*	I	V	V	V	V	V	V	V	I	V	El sistema muestra un mensaje de error que no se conectó por datos incorrectos.	El usuario introduce los datos necesarios para conectarse a la base de datos y luego presiona el botón conectar.
		(pepe)	*(postgres)*	*(postgres)*	*(localhost)*	*(5432)*	*(test)*	*(admin)*	*(admin)*	*(56)*	*(27017)*		
EC 1.3 Campos vacíos.	Se tratan de insertar los datos, pero se dejan campos vacíos.	V	V	V	I	V	V	V	I	I	V	El sistema muestra un mensaje de error que no se conectó.	
		(postgres)	*(postgres)*	*(postgres)*	*()*	*(5432)*	*(test)*	*(admin)*	*()*	*()*	*(27017)*		

Tabla 11. Ejemplo de caso de prueba de caja negra_1

Nombre de la HU: Realizar la migración de datos de PostgreSQL hacia CouchDB.

Escenario	Descripción	V 1	V 2	V 3	V 4	V 5	V 6	V 7	V 8	V 9	V 10	Respuesta del sistema	Flujo central
EC 1.1 Introducir datos correctos.	Se introduce datos correctamente para conectarse a las base de datos origen y destino respectivamente.	V	V	V	V	V	V	V	V	V	V	El sistema muestra un mensaje que se conectó exitosamente.	El usuario introduce los datos necesarios para conectarse a la base de datos y luego presiona el botón conectar.
		(postgres)	*(postgres)*	*(postgres)*	*(localhost)*	*(5432)*	*(test)*	*(admin)*	*(admin)*	*(localhost)*	*(5984)*		
EC 1.2 Introducir datos incorrectos	*Se introduce datos incorrectos.*	I	V	V	V	V	V	V	V	I	V	El sistema muestra un mensaje de error que no se conectó por datos incorrectos.	El usuario introduce los datos necesarios para conectarse a la base de datos y luego presiona el botón conectar.
		(pepe)	*(postgres)*	*(postgres)*	*(localhost)*	*(5432)*	*(test)*	*(admin)*	*(admin)*	*(56)*	*(5984)*		
EC 1.3 Campos vacíos.	Se tratan de insertar los datos, pero se dejan campos vacíos.	V	V	V	I	V	V	V	I	I	V	El sistema muestra un mensaje de error que no se conectó.	
		(postgres)	*(postgres)*	*(postgres)*	*()*	*(5432)*	*(test)*	*(admin)*	*()*	*()*	*(5984)*		

Tabla 12. Ejemplo de caso de prueba de caja negra_2

3.3.6 Presentación de los resultados de las pruebas funcionales

Caja Negra

Después de realizar las pruebas de Caja Negra mediante los casos de prueba asociados a cada historia de usuario, se comprobó el correcto funcionamiento de la herramienta y la correcta validación de los campos, verificando que solo se acepten los caracteres válidos para los mismos.

Conclusiones del capítulo III

Las pruebas de software, permiten la verificación de la calidad de un producto. Son utilizadas para identificar posibles fallos de una aplicación, básicamente es una fase en el desarrollo de software consistente en probar la aplicación construida. En este capítulo se realizó un estudio de las técnicas, tipos y métodos de pruebas aplicadas al sistema, en la cual se decidió aplicar a nivel de desarrollador los tipos de pruebas de aceptación y funcionales basado en el método de caja negra. Fueron probadas todas las funcionalidades. Se logró presentar los resultados logrando obtener una aplicación que responde a todos las historias de usuarios identificadas, la cual cuenta con una alta calidad. Se definió el estándar de codificación que propone la metodología XP e imágenes de la aplicación.

Conclusiones Generales

Con la realización de este trabajo de diploma se obtuvo una propuesta que da cumplimiento al objetivo general planteado, al lograr desarrollar una herramienta informática que permita realizar la migración de los datos entre los gestores de base de datos PostgreSQL, MongoDB y CouchDB. Para llegar a este resultado se puede concluir lo siguiente:

- Se realizó un estudio del marco conceptual referencial sobre el proceso de migración de datos, las bases de datos y sus modelos respectivamente así como sus implementaciones.
- Se seleccionaron las herramientas y tecnologías a utilizar para el desarrollo de la herramienta reconociendo que la mayoría se caracterizan por ser software libre y multiplataforma.
- Se definieron las características y se realizó el diseño de la Herramienta de Migración de Datos MIGDAT obteniéndose como resultado los artefactos necesarios para el desarrollo del mismo.
- Se realizó la implementación de la herramienta cumpliendo con las historias de usuarios identificadas en las fases de análisis y diseño.
- El diseño y ejecución de pruebas del sistema permitieron comprobar el correcto funcionamiento de la Herramienta de Migración de Datos MIGDAT, luego de quedar resueltas todas las no conformidades detectadas.

Recomendaciones

Luego de haber analizado los resultados del presente trabajo de diploma, se puede llegar a la siguiente recomendación:

Se recomienda analizar el diseño de las clases para facilitar la extensibilidad de la herramienta para su uso con otros gestores que se identifiquen en el departamento, y aplicarla en los proyectos productivos del mismo.

Referencias Bibliográficas

1. **Paramio, Carlos.** genbetadev.com. *genbetadev.com.* [En línea] [Citado el: 23 de Noviembre de 2012.]Abril 2011 http://www.genbetadev.com/bases-de-datos/el-concepto-nosql-o-como-almacenar-tus-datos-en-una-base-de-datos-no-relacional..

2. **Pereira, Manuel.** [En línea] [Citado el: 12 de Octubre de 2012.]Mayo 2013 http://manuelpereiragonzalez.blogspot.com/..

3. **Corona, Laura Susana.** *Factores Criticos de Exitos en la migracion de datos.* [En línea] [Citado el: 14 de Octubre de 2012.]2000.

4. **García, Lic Rosa María Mato.** *Diseño de Base de Datos.* . [En línea] [Citado el: 17 de Octubre de 2012.]Junio 2009

5.**Andrés Aizaga.** Modelos de Base de Datos. *Modelos de Base de Datos.* [En línea] [Citado el: 8 de Diciembre de 2012.] 2006 http://www.monografias.com/trabajos55/base-de-datos/base-de-datos2.shtml#modelos...

6. **Murillo, Yohan Sebastian Aristizabal.** *MONGODB & PHP. Caldas* [En línea] [Citado el: 18 de Octubre de 2012.]2012.

7.**NoSQL Databases**. Administrador de Base de Datos. [En línea] [Citado el: 21 de Noviembre de 2012.]2009 http://www.base de datos no relacional/Clasificación.html y http://nosql-database.org/..

8. **Bello, Andrés.** *Diseño de Base de Datos Problemas Resueltos.* [En línea] [Citado el: 20 de Octubre de 2012.]Agosto 2012

9. **Español, Comunidad de PostgreSQL en.** Acerca de PostgreSQL. *Acerca de PostgreSQL.* [En línea] [Citado el: 17 de Noviembre de 2012.] Octubre 2010 http://www.postgresql-es.org/sobre_postgresql. .

10. **Base de Datos PostgreSQL**. *Base de Datos PostgreSQL.* [En línea] [Citado el: 17 de Noviembre de 2012.] 2011 http://eaprende.com/gestor-de-basededatos-mysql-postresql-sqlite.html. .

11. **PostgreSQL**. *PostgreSQL.* [En línea] [Citado el: 20 de Noviembre de 2012.] Octubre 2010 http://postgres.org/PostgreSQL...

12. **Licencia.** PostgreSQL. *PostgreSQL.* [En línea] [Citado el: 21 de Noviembre de 2012.] Octubre 2010

http://www.postgre.org. .

13. **Perez, Ing. Rafael Hernandez.** CouchDB. *CouchDB.* [En línea] [Citado el: 9 de Diciembre de 2012.] 2012 http://couchdb.apache.org/..

14. **Horowitz, Eliot.** MongoDB. *MongoDb.* [En línea] [Citado el: 9 de Diciembre de 2012.] 2012 http://www.mongodb.org/.

15. **Barbones.** XP. *XP.* [En línea] [Citado el: 5 de Noviembre de 2012.] 2008

16. **José H. Canós, Patricio Letelier y Mª Carmen Penadés.** *Metodologia Agiles de Desarrollo de Software.* [En línea] [Citado el: 10 de Enero de 2013.] 2003

17. **García, Joaquín.** Lenguaje de Modelado. *Lenguaje de Modelado.* [En línea] [Citado el: 10 de Diciembre de 2012.] Mayo 2005 http://www.ingenierosoftware.com/analisisydiseno/uml.php..

18. **Fuster, José Gonzalo Genova.** *Evaluación comparativa de herramientas CASE para UML desde el punto de vista notacional.* .[En línea] [Citado el: 10 de Enero de 2013.] 2006

19. **System, Inc. Popkin Software and.** Modelado. *Modelado.* [En línea] [Citado el: 29 de Noviembre de 2012.] Febrero 2005 http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf ..

20. **Duque, Ing. Raúl Gonsález.** *Python para todos.* [En línea] [Citado el: 5 de Enero de 2013.] Julio 2008

21. **Mantellini, Héctor A.** IDE. *IDE.* [En línea] [Citado el: 11 de Enero de 2013.] 2012 http://www.awven.com/q121-aptana-studio-3-herramienta-profesional-desarrollo-codigo-abierto-web/ .

22. **Andrés, José.** *Editor de Materiales.* [En línea] [Citado el: 11 de Enero de 2013.] 2001

23. **Tony Navarrete, Sergio Sayago.** *Herramienta para la creacion de Interfaces.* [En línea] [Citado el: 11 de Enero de 2013.] 2005

24. **Weth, Christian von der.** *Multiterm Keyword Search in NoSQL Systems.* Nanyang Technological University : Published by the IEEE Computer Society, Digital Enterprise Research Institute, 2012. 1089-7801/12/$31.00 © 2012 IEEE. [En línea] [Citado el: 21 de Mayo de 2013.]

25. **Ivar Jacobson, Grady Booch y Rumbaugh, James.** *Proceso Unificadado de Desarrollo de Software.* s.l. : Primera Edición en Español. [En línea] [Citado el: 21 de Enero de 2013.] Noviembre 2012

26. **Letelier, Patricio.** *Métodologías ágiles para el desarrollo de software:eXtreme Programming (XP).* Valencia : s.n. [En línea] [Citado el: 21 de Diciembre de 2012.]2006

27. **Pressman, Roger S.** *Ingeniería del Software: Un enfoque práctico. Séptima Edición.* [En línea] [Citado el: 21 de Diciembre de 2012.]2005

28. **Larman, Craig.** *Introducción al análisis y deseño orientado a objetos.* México : s.n. [En línea] [Citado el: 22 de Diciembre de 2012.]2002

29. **Marcello Astudillo, Visconti y Hernán.** *Fundamentos de Ingeniería de Software.* Universidad Técnica Federico Santa María. : s.n. . [En línea] [Citado el: 20 de Diciembre de 2012.] Agosto 2011

30. **Somerville, Ian.** *Ingenieria de Software.*[En línea] [Citado el: 20 de Diciembre de 2012.] 2005

31. **Letelier, Patricio.** *Métodologías ágiles para el desarrollo de software.* Valencia : s.n. [En línea] [Citado el: 21 de Diciembre de 2012.] 2006

32. **Klint Finley** "Why Large Hadron Collider Scientists are Using CouchDB". *"Why Large Hadron Collider Scientists are Using CouchDB".* [Online] CouchDB-1, 2012. [Cited: 5 27, 2013.] Agosto 2010 http://www.readwriteweb.com/enterprise/2010/08/lhc-couchdb.php..

Bibliografía

1. **Acronis.sa** *El Costo Oculto de la Migración de Datos* [Citado el: 13 de Marzo de 2013.] 2008 http://www.acronis.com.uy/documentos/pdf/Data_Migration_wp.es.pdf. [En línea]
2. **Andrés Aizaga.** Modelos de Base de Datos. *Modelos de Base de Datos.* [En línea] [Citado el: 8 de Diciembre de 2012.] 2006 http://www.monografias.com/trabajos55/base-de-datos/base-de-datos2.shtml#modelos...
3. **Andrés, José.** *Editor de Materiales.* [En línea] [Citado el: 11 de Enero de 2013.] 2001
4. **Arthur Zaragosa** *Migración de Datos* [Citado el: 3 de Abril de 2013.] 2013 http://es.scribd.com/doc/49435158/Migrar-base
5. **Awven** [Citado el: 5 de Marzo de 2013.] 2012 http://www.awven.com/q121-aptana-studio-3-herramienta-profesional-desarrollo-codigo-abierto-web/. [En línea]
6. **Barbones.** XP. *XP.* [En línea] [Citado el: 5 de Noviembre de 2012.] 2008
7. **Base de Datos PostgreSQL**. *Base de Datos PostgreSQL.* [En línea] [Citado el: 17 de Noviembre de 2012.] 2011 http://eaprende.com/gestor-de-basededatos-mysql-postresql-sqlite.html. .
8. **Bello, Andrés.** *Diseño de Base de Datos Problemas Resueltos.* [En línea] [Citado el: 20 de Octubre de 2012.]Agosto 2012
9. **Corona, Laura Susana.** *Factores Criticos de Exitos en la migracion de datos.* [En línea] [Citado el: 14 de Octubre de 2012]
10. **Duque, Ing. Raúl Gonsález.** *Python para todos.* [En línea] [Citado el: 5 de Enero de 2013.] Julio 2008
11. **Español, Comunidad de PostgreSQL en.** Acerca de PostgreSQL. *Acerca de PostgreSQL.* [En línea] [Citado el: 17 de Noviembre de 2012.] Octubre 2010 http://www.postgresql-es.org/sobre_postgresql. .
12. **Fuster, José Gonzalo Genova.** *Evaluación comparativa de herramientas CASE para UML desde el punto de vista notacional.* .[En línea] [Citado el: 10 de Enero de 2013.] 2006
13. **García, Joaquin.** Lenguaje de Modelado. *Lenguaje de Modelado.* [En línea] [Citado el: 10 de Diciembre de 2012.] Mayo 2005 http://www.ingenierosoftware.com/analisisydiseno/uml.php..
14. **García, Lic Rosa María Mato.** *Diseño de Base de Datos.* . [En línea] [Citado el: 17 de Octubre de

2012.]Junio 2009

15. **Gerardo Fernández Escribano** [En línea] [Citado el: 1 de Marzo de 2013.] Diciembre 2002 http://aalbertovargasc.files.wordpress.com/2011/07/presentacion-xp.pdf.

16. **Horowitz, Eliot.** MongoDB. *MongoDb.* [En línea] [Citado el: 9 de Diciembre de 2012.] 2012 http://www.mongodb.org

17. **Ivar Jacobson, Grady Booch y Rumbaugh, James.** *Proceso Unificadado de Desarrollo de Software.* s.l. : Primera Edición en Español. [En línea] [Citado el: 21 de Enero de 2013.] Noviembre 2012

18. **Jedutún Guerrero** [En línea] [Citado el: 21 de Febrero de 2013.] Junio 2008 http://boards5.melodysoft.com/UBV_INGS/metodologias-agiles-de-desarrollo-43.html.

19. **José H. Canós, Patricio Letelier y Mª Carmen Penadés.** *Metodologia Agiles de Desarrollo de Software.* [En línea] [Citado el: 10 de Enero de 2013.] 2003

20. **Kent Beck** [En línea] [Citado el: 20 de Febrero de 2013.] 2004 http://wiki.monagas.udo.edu.ve/index.php/Metodolog%C3%ADas_SCRUM_y_XP#METODOLOG.C3.8DA_XP_.28EXTREME_PROGRAMMING.29.

21. **Klint Finley** "Why Large Hadron Collider Scientists are Using CouchDB". *"Why Large Hadron Collider Scientists are Using CouchDB".* [Online] CouchDB-1, 2012. [Cited: 5 27, 2013.] Agosto 2010 http://www.readwriteweb.com/enterprise/2010/08/lhc-couchdb.php

22. **Larman, Craig.** *Introducción al análisis y deseño orientado a objetos.* México : s.n. [En línea] [Citado el: 22 de Diciembre

23. **Letelier, Patricio.** *Métodologías ágiles para el desarrollo de software.* Valencia : s.n. [En línea] [Citado el: 21 de Diciembre

24. **Letelier, Patricio.** *Métodologías ágiles para el desarrollo de software:eXtreme Programming (XP).* Valencia : s.n. [En línea] [Citado el: 21 de Diciembre de 2012.]2006

25. **Licencia.** PostgreSQL. *PostgreSQL.* [En línea] [Citado el: 21 de Noviembre de 2012.] Octubre 2010 http://www.postgre.org. .

26. **Mantellini, Héctor A.** IDE. *IDE.* [En línea] [Citado el: 11 de Enero de 2013.] 2012 http://www.awven.com/q121-aptana-studio-3-herramienta-profesional-desarrollo-codigo-abierto-web/ .

27. **Marcello Astudillo, Visconti y Hernán.** *Fundamentos de Ingeniería de Software.* Universidad Técnica

Federico Santa María. : s.n. . [En línea] [Citado el: 20 de Diciembre de 2012.] Agosto 2011

28. **Murillo, Yohan Sebastian Aristizabal.** *MONGODB & PHP. Caldas* [En línea] [Citado el: 18 de Octubre de 2012.]2012.

29. **NoSQL Databases**. Administrador de Base de Datos. [En línea] [Citado el: 21 de Noviembre de 2012.]2009 http://www.base de datos no relacional/Clasificación.html y http://nosql-database.org/..

30. **Paramio, Carlos.** genbetadev.com. *genbetadev.com.* [En línea] [Citado el: 23 de Noviembre de 2012.]Abril 2011 http://www.genbetadev.com/bases-de-datos/el-concepto-nosql-o-como-almacenar-tus-datos-en-una-base-de-datos-no-relacional..

31. **Pereira, Manuel.** [En línea] [Citado el: 12 de Octubre de 2012.]Mayo 2013 http://manuelpereiragonzalez.blogspot.com/..

32. **Pérez, Ing. Rafael Hernández.** CouchDB. *CouchDB.* [En línea] [Citado el: 9 de Diciembre de 2012.] 2012 http://couchdb

33. **PostgreSQL**. *PostgreSQL.* [En línea] [Citado el: 20 de Noviembre de 2012.] Octubre 2010 http://postgres.org/PostgreSQL...

34. **Pressman, Roger S.** *Ingeniería del Software: Un enfoque práctico. Séptima Edición.* [En línea] [Citado el: 21 de Diciembre

35. **Somerville, Ian.** *Ingenieria de Software.*[En línea] [Citado el: 20 de Diciembre de 2012.] 2005

36. **System, Inc. Popkin Software and.** Modelado. *Modelado.* [En línea] [Citado el: 29 de Noviembre de 2012.] Febrero 2005 http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf ..

37. **Tony Navarrete, Sergio Sayago.** *Herramienta para la creacion de Interfaces.* [En línea] [Citado el: 11 de Enero de 2013.] 2005

38. **Ubuntu-Guía** [Citado el: 3 de Marzo de 2013.] Julio 2011 http://www.genbetadev.com/bases-de-datos/el-concepto-nosql-o-como-almacenar-tus-datos-en-una-base-de-datos-no-relacional. [En línea]

39. **Weth, Christian von der.** *Multiterm Keyword Search in NoSQL Systems.* Nanyang Technological University : Published by the IEEE Computer Society, Digital Enterprise Research Institute, 2012. 1089-7801/12/$31.00 © 2012 IEEE. [En línea] [Citado el: 21 de Mayo

Glosario de Términos

BD: Base de Datos.

SGBD: Sistema Gestor de Base de Datos.

DATEC: Centro de Tecnología de Gestión de Datos.

Plugin: Software que se relaciona con otro para aportarle una función nueva, generalmente específica que se ejecuta en la aplicación principal

XP: Extreme Programming.

HU: Historia de usuario.

TI: Tarea de ingeniería.

Tarjeta CRC: Tarjeta Clase Responsabilidad y Colaboración.

UML: Unified Modeling Language (Lenguaje Unificado de Construcción de Modelos).

IDE: Integrated Development Environment (Entorno Integrado de Desarrollo).

GRASP: General Responsability Assignment Software Patterns, (Patrones de Software para la asignación General de Responsabilidad).

SQL: Lenguaje de Consulta Estructurado.

NoSQL: Not Only SQL (No solamente SQL).

RDBMS: Relational Database Management System.

ACID: atomicidad, consistencia, aislamiento y durabilidad

IBM: International Business Machines.

Anexos

Anexo 1: Migración de datos (46)

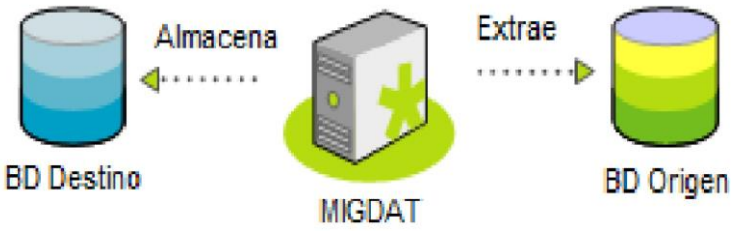

Figura 20. Migración de datos

Anexo 2: Gestores de Bases de Datos (45)

Figura 21. Gestores de bases de datos