Universidad de las Ciencias Informáticas

Facultad 6

Centro de Tecnología de Gestión de Datos

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

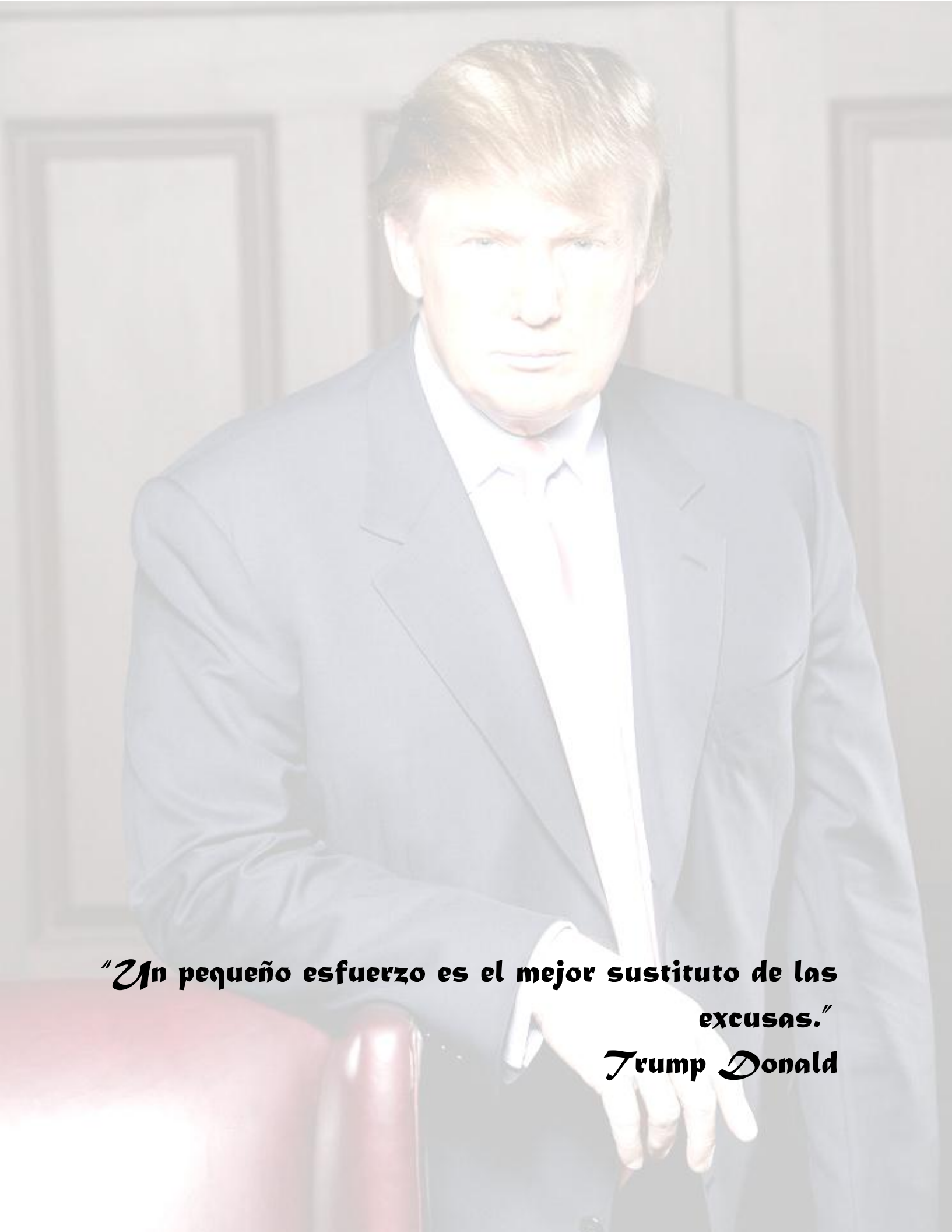
Título: Conectores Adicionales para Multicorn

Autor: Angel Manuel López Acosta

Tutor: Ing.Yunior Mesa Reyes

La Habana 2013

"Año 55 de la Revolución"



"Un pequeño esfuerzo es el mejor sustituto de las excusas."

Trump Donald

Declaración de Autoría

Declaración de Autoría:

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma con carácter exclusivo. Y para que así conste firmo la presente a los ____ días del mes de: _____ del año: _____.

_____	_____
Angel Manuel López Acosta	Ing. Yunior Mesa Reyes
Tesista	Tutor

Datos de Contacto:

Autor: Angel Manuel López Acosta
Universidad de las Ciencias Informáticas
e-mail: alopez@estudiantes.uci.cu

Tutor: Ing. Yunior Mesa Reyes
Título: Ingeniero en Ciencias Informáticas
Categoría docente: Instructor
Años de experiencia: 3
Año de graduado: 2009
Correo electrónico: ymreyes@uci.cu

Agradecimientos:

Antes que nada, quiero agradecer a los que ya no están conmigo, fueron las personas que más confiaron en mí y las que dieron todo de sí para que pudiese seguir adelante, a mis abuelos Angélica y Víctor Manuel (mama y papa) por ser como mis padres, a mi tía Isabel (tía Isa); gracias a todos por educarme y por estar ahí siempre para mí cuando más los necesité, por complacerme en todo lo que estuvo a su alcance. A mis abuelos Lucía y Adalberto que tampoco se encuentran, pero que también son parte importante de mi vida.

A mis padres por ayudarme y apoyarme en todo momento. Mami gracias por ser más que una amiga todo este tiempo, por no perder la confianza en mí aunque todo parecía estar perdido. Papi, gracias por estar ahí todo este tiempo.

A mi hermana Adanerys por ser mi confidente de toda la vida, por enseñarme las cosas de la vida tal y como son. Gracias por poder contar contigo siempre. A mis dos pequeños gigantes Alvarito y Oscar, gracias por alegrarme y por enseñarme que una sonrisa te puede cambiar la vida.

A todos mis familiares, en especial a Che, Tata, el Niño y Chichi. Gracias a todos por demostrar ser una familia en los momentos que más se necesitaban, por estar ahí sin necesidad de que fueran llamados.

A Sandy, Liutmila, Yaniris, Ailid, Dained, Claudia, Adrian, Retureta, Brayan y Asiel aunque la vida nos hizo tomar caminos diferentes siempre están presentes, gracias por compartir conmigo parte de sus vidas y por hacerme sentir parte de sus familias.

A Adonis, gracias por ayudarme estos últimos años, por estar junto a mí, por hacerme salir del bache en el que tantas veces caía. Gracias por ser tú, tan incondicional conmigo.

A mis amigos:

Dalied, por ser mi hermana todo este tiempo, por regañarme y no por no tener nunca una excusa a la hora de tenderme la mano. Ana Rosa gracias por todo lo que hiciste por mi todos estos años. Lili por ser tan solidaria y darme todos los consejos para ser mejor persona. Nelly por compartir su vida conmigo y brindarme su amistad. Wilber por ser el mejor hermano que una persona pueda tener. Janet y Lisandra por ayudarme tanto en tan corto tiempo y por dejarme entrar en sus vidas. A Katerine, Arianna, Marla, Claudia, Clara, Margarita y Juliet.

A mi tutor Yunior, gracias por ayudarme y ser tan insistente conmigo y ser más que un tutor, un amigo. A todos los profesores que ayudaron en mi formación durante estos años como un profesional.

Dedicatoria

A mis Abuelos:

Angélica Díaz Armas, Víctor Manuel Acosta Estévez y María Isabel Díaz Armas

A mis padres:

María de los Ángeles Acosta Días y Adalberto López Rodríguez

A mi hermana:

Adanerys López Acosta

A mis sobrinos:

Oscar Manuel Pérez López y Álvaro Pérez López

Resumen

En la actualidad, se hace necesaria la migración hacia el software libre por parte de las pequeñas y medianas empresas, principalmente aquellas pertenecientes a los países menos desarrollados, entre ellos Cuba. PostgreSQL es uno de los Sistemas Gestores de Bases de Datos de software libre y código abierto más utilizado en la actualidad, altamente personalizable y extensible. En la presente investigación se muestra una herramienta de extensibilidad de PostgreSQL, nombrada Multicorn, extensión que permite acceder a datos externos a las bases de datos de PostgreSQL. La investigación se centra en el desarrollo de nuevos conectores de datos para esta extensión que permitan ampliar las funcionalidades de extensibilidad de PostgreSQL, aumentando las potencialidades del gestor.

Palabras clave: FDW, Extensión, Multicorn, PostgreSQL

Índice

Índice de contenidos

Índice de contenidos

Índice de figuras

Índice de tablas

Índice de tablas

Introducción

Con la indetenible evolución de las Tecnologías de la Información y las Comunicaciones (TIC) se ha acelerado el crecimiento económico y social en el mundo actual, permitiendo que los cúmulos de datos sean cada vez mayores. El manejo de estos datos generados es difícil de manejar por el hombre de forma manual, debido a la complejidad y al gran volumen que pueden llegar a alcanzar, surgiendo de esta forma lo que hoy se conoce como base de datos.

Las bases de datos mejoran la recopilación de grandes volúmenes de datos, aumentando la rapidez de la manipulación de los mismos. Su funcionamiento y manejo se han perfeccionado gracias a la creación de los diferentes Sistemas Gestores de Base de Datos (SGBD) que facilitan a los usuarios el trabajo con los datos almacenados. Múltiples son las ventajas que permiten la utilización de estos, pero existe el inconveniente de que el uso de muchos de ellos se encuentra restringido por problemas de licencia, impidiendo acceder a los servicios que prestan los mismos.

Por esta situación Cuba se encuentra entre los países que más impulsa el uso de software no privativo sin mantenerse aislada de los avances tecnológicos, alcanzando resultados sobresalientes en el área de América Latina. En el país, la industria del software es relativamente joven, fortaleciéndose en la medida de sus posibilidades; es por ello que el gobierno cubano tiene como una de sus tareas principales lograr total independencia y soberanía dentro de este campo con el objetivo de informatizar la sociedad e insertarse en el mercado mundial cada vez más competitivo.

En la actualidad, se hace necesaria la migración hacia el software libre por parte de las pequeñas y medianas empresas principalmente aquellas pertenecientes a los países menos desarrollados, entre ellos Cuba, donde existen políticas que defienden la migración al software libre, potenciando su utilización basado en sus grandes ventajas sobre todo las relacionadas con su costo. Entre las instituciones de avanzada en el proceso de desarrollo de software en nuestro país está la Universidad de la Ciencias Informáticas (UCI), la misma cuenta con numerosos proyectos encargados de desarrollar productos de software para varios clientes tanto nacionales como extranjeros. Esta institución, como muchas otras en el país, ha estado transitando por un proceso de migración, garantizando que sus centros productivos se conviertan en un motor impulsor en el desarrollo de todos sus productos con herramientas no privativas.

Uno de los centros inmerso en este proceso es el Centro de Tecnologías de Gestión de Datos (DATEC) y dentro de este el departamento PostgreSQL, que tiene entre sus misiones contribuir al desarrollo de

tecnologías de bases de datos, tomando como base el gestor PostgreSQL. Este departamento actualmente es de referencia en la universidad por desarrollar tecnologías de bases de datos en torno a este gestor, además de brindar servicios de instalación, transferencia tecnológica, formación y capacitación a otros centros de la universidad y de Cuba, los cuales han estado haciendo uso de SGBD propietarios como SQL Server y MySQL teniendo un costo millonario por los mismos. Dichas empresas poseían un desconocimiento de la existencia de otros Gestores de Bases de Datos, como PostgreSQL, por lo que el gestor casi no era utilizado, teniendo como consecuencia el uso de diferentes SGBD.

Según encuestas realizadas a especialistas, usuarios y miembros de la Comunidad Técnica Cubana de PostgreSQL en el marco de eventos, capacitaciones, reuniones y encuentros realizados en la UCI entre los años 2009 y 2011 arrojaron que en un inicio solamente un 17% de los usuarios encuestados hacían uso de PostgreSQL para administrar sus bases de datos; ya en el 2011 la situación es otra, aunque aún persiste un porciento representativo utilizando software propietario. Hoy en día solo el 2% de las entidades empresariales no hacen uso de ningún SGBD y el uso de PostgreSQL se ha elevado hasta el 46% junto al uso de otros SGBD libres, aunque aún algunas persisten en la utilización de sistemas propietarios, esto evidencia que el software libre se ha consolidado como alternativa, técnicamente viable y económicamente sostenible al software comercial. (1)

La diversidad de gestores que se utilizan en el país conlleva a un grupo de inconvenientes, los principales son:

- Los gestores propietarios son utilizados de manera ilegal amparados en el bloqueo económico y comercial impuesto por Estados Unidos, de pagarlos, los montos ascenderían a miles de dólares.
- Independientemente del dinero a pagar para utilizar los gestores propietarios, hay que contar con la alta posibilidad de que en estas soluciones existan puertas traseras por las que se escaparía información valiosa.
- Las empresas de desarrollo de software están imposibilitadas de comercializar productos que tengan como base gestores propietarios, de los cuales no se posea licencia.

A raíz de esta situación Cuba se propone minimizar el uso de los SGBD propietarios y potenciar el uso de PostgreSQL. Bajo esta premisa y teniendo en cuenta esta situación, se comenzó a promover la utilización de PostgreSQL en las aplicaciones empresariales del país; alineados a que Cuba se encuentra hoy

inmersa en un proceso de migración a software libre con el propósito de alcanzar una independencia tecnológica muy necesaria, sobre todo cuando de SGBD se trata, teniendo en cuenta que la información siempre es sensible y más en un país bloqueado y asediado por el mayor gigante de las comunicaciones a nivel mundial.

PostgreSQL es un SGBD objeto-relacional, distribuido bajo licencia BSD[1]; goza de una excelente popularidad en el mercado, ya posee una gran comunidad que le brinda soporte y en sus últimas versiones se encuentra a la par de los demás SGDB disponibles en el mercado mundial. (2)

La extensibilidad de PostgreSQL es una característica visible desde la versión 3 del gestor y actualmente permite a los usuarios añadir nuevas funcionalidades en las bases de datos existentes en los entornos de producción cubanos y usar estas para tareas que no puede realizar ningún otro sistema de base de datos utilizado actualmente en las empresas cubanas. En la versión 8.4 de PostgreSQL se incluye la gestión de datos externos mediante SQL/MED ("Administración SQL para datos externos"), así como las Envolturas de Datos Externas, del inglés, (*Foreign Data Wrappers*) FDW, estas características se han fusionados en la versión 9.1, lo que ha permitido la creación de herramientas de extensibilidad como:

- Conectores de Datos Foráneos (CDF): añaden y consultan fuentes de datos externas desde PostgreSQL.
- Extensiones: crean, cargan, y administran fácilmente nuevas características de la base de datos.

Multicorn es una extensión añadida al gestor en esta versión; es desarrollada en el lenguaje de programación Python y permite acceder a fuentes de datos externas a PostgreSQL mediante CDF. El usuario puede añadir nuevos conectores con el fin acceder a otras fuentes de datos, haciendo más fácil y sencillo el acceso a la información contenida en estas en los entornos de producción, aumentando así la extensibilidad de Multicorn y del gestor.

Esta extensión ha sido útil en entornos de producción donde se utilizaban SGBD privativos como MySQL y SQL Server, facilitando el manejo de los datos contenidos en dichas bases de datos hacia PostgreSQL. Actualmente los conectores que posee la extensión Multicorn resultan insuficientes en los entornos de producción cubanos donde se requiere acceder a información de otras fuentes de datos; al mismo tiempo

[1] *Berkeley Software Distribution*, licencia que respeta las cuatro libertades del Software Libre (usar el programa con cualquier propósito, estudiar cómo funciona y modificarlo, distribuir copias del programa, mejorar el programa y hacer públicas las mejoras), pero además da posibilidad a los desarrolladores de construir soluciones comerciales basadas en el software inicial.

en DATEC se utilizan otros SGBD para almacenar un conjunto de información que en un momento determinado se necesitan monitorear, encontrándose como obstáculo que aún Multicorn no posee conectores que permitan acceder a esas fuentes, constituyendo un freno tecnológico y una de las limitantes para alcanzar la soberanía tecnológica cubana.

Dada la anterior situación se identifica como **problema de investigación:** ¿Cómo contribuir a que Multicorn sea útil en la industria del software cubano y en otros entornos de producción para los cuales no fue creado inicialmente?

Se plantea como **objeto de estudio** de la presente investigación: La extensibilidad del SGBD PostgreSQL enmarcado en el **campo de acción**: Proceso de desarrollo de la extensión Multicorn en PostgreSQL.

Para dar solución al problema de la investigación se ha definido como **objetivo general**: Desarrollar conectores de datos adicionales que permitan ampliar las funcionalidades de extensibilidad del SGBD PostgreSQL.

Teniendo como premisa lograr el cumplimiento de dicho objetivo general se plantearon los siguientes **objetivos específicos**:

- Fundamentar la selección de los conectores, así como las tecnologías, metodología y herramientas a utilizar en su desarrollo para Multicorn.
- Realizar el análisis y diseño de los conectores a implementar para Multicorn.
- Realizar la implementación y validación de los conectores implementados para Multicorn.

Para darle cumplimento a los objetivos propuestos se plantea el siguiente conjunto de **tareas de la investigación**:

1. Elaboración del marco teórico de la investigación.
2. Identificación de las fuentes de datos a importar.
3. Identificación de los requisitos así como la metodología, herramientas y tecnologías a utilizar.
4. Realización del análisis de la solución de los conectores.
5. Realización del diseño de la solución de los conectores.

6. Implementación del diseño realizado.
7. Realización de pruebas que validen la solución propuesta.

El presente trabajo de diploma está estructurado de la siguiente forma:

Capítulo 1: Fundamentación teórica

La investigación realizada para este capítulo contiene los fundamentos teóricos para entender el problema existente, así como conceptos fundamentales que permitieron realizar un análisis de la extensión Multicorn. Se realiza un estudio para definir las herramientas, metodología y tecnologías a utilizar en el desarrollo de los conectores para la extensión de PostgreSQL.

Capítulo 2: Análisis y Diseño.

Se realiza el análisis y el diseño de los conectores a desarrollar para Multicorn mediante los artefactos generados por la metodología definida. Se describen sus principales características y los requerimientos.

Capítulo 3: Implementación y Pruebas

Se realiza un estudio para verificar si el sistema cumple con los requisitos descritos por el cliente, definiendo pruebas y técnicas que son aplicadas a la extensión Multicorn. Se muestran los resultados luego de haber realizado los casos de pruebas por cada una de las funcionalidades de Multicorn.

Capítulo 1: Fundamentación Teórica

Introducción:

En este capítulo se presentan un grupo de conceptos teóricos a los que se hará referencia en el resto del trabajo. Comprende un estudio de definiciones fundamentales para desarrollar la investigación, tales como: base de datos, SGBD y extensión. Se muestran las características de las herramientas identificadas, así como las tecnologías y la metodología a utilizar durante el desarrollo de la investigación.

1.1 Base de Datos

Una base de batos es un conjunto de información estructurada en registros y almacenada en un soporte electrónico legible desde un ordenador. Cada registro constituye una unidad autónoma de información que puede estar a su vez estructurada en diferentes campos o tipos de datos que se recogen en dicha base de datos. Por ejemplo, en un directorio de miembros de una asociación, un registro será la ficha completa de cada uno de los socios. En cada registro se recogerán determinados datos, como el nombre, la profesión, la dirección o el teléfono, cada uno de los cuales constituye un campo. (3)

Un concepto más generalizado para definir una base de datos es definirla como un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. Entre las características de las bases de datos se encuentra: independencia lógica y física de los datos, redundancia mínima, acceso concurrente por parte de múltiples usuarios, integridad de los datos, consultas complejas optimizadas y seguridad de acceso y auditoría. (3)

En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital, solucionando los problemas que existían en el almacenamiento de los datos, para ello existen programas denominados SGBD que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

1.2 Sistema Gestor de Base de Datos

Con el excesivo cúmulo de información, la centralización de la mayoría de las aplicaciones que contienen las base de datos y la cantidad de personas que concurren en dichos sistemas surge la necesidad de emplear un sistema de administración que controle tanto a los datos como a sus usuarios.

Un SGBD es una colección de programas que permiten a los usuarios crear y mantener una base de datos, son un sistema de software de propósito general que facilita los procesos de definición, construcción y manipulación de la base de datos para distintas aplicaciones. (3)

Los SGBD son: “Un software o conjunto de programas que permiten crear y mantener una base de datos que actúa como interfaz entre los programas de aplicación y el sistema operativo. El objetivo principal es proporcionar un entorno eficiente a la hora de almacenar y recuperar la información de las base de datos. Este software facilita el proceso de definir, construir y manipular bases de datos para diversas aplicaciones.” (4) Se trata de un conjunto de programas no visibles al usuario final que se encargan de la privacidad, integridad, seguridad de los datos e interacción con el sistema operativo. Proporciona una interfaz entre los datos, los programas que los manejan y los usuarios finales. Cualquier operación que el usuario hace con la base de datos está controlada por el gestor.

El propósito general de los SGBD es el manejo de forma clara, sencilla y ordenada de un conjunto de datos, que posteriormente se convertirán en información relevante para una organización.

Proveen facilidades para la manipulación de grandes volúmenes de datos, entre las que se encuentran:

- Simplifican la programación de equipos de consistencia.
- Garantizan que los cambios de la base de datos sean siempre consistentes sin importar si hay errores mediante el manejo de las políticas de respaldo adecuadas.
- Organizan los datos con un impacto mínimo en el código de los programas.
- Disminuyen drásticamente los tiempos de desarrollo y aumentan la calidad del sistema desarrollado, si son bien explotados por los desarrolladores.

Estos sistemas de administración proporcionan a los usuarios la gestión del acceso a la base de datos de manera simultánea y garantizan que no sucedan inconvenientes con la integridad de los mismos. Existen

SGBD muy potentes que proporcionan muchas de las funciones estándar que el programador necesita para lograr grandes avances, entre los que se encuentran: Oracle, DB2, PostgreSQL, SQL-Server, Fox Pro, Access y SQLite. (3)

Algunos SGBD actualmente almacenan sus datos en ficheros DBF, que no es más que el formato de archivo de datos utilizado originalmente por el producto dBase para computadoras personales. Actualmente es utilizado por dBase, Clipper, FoxPro y sus equivalentes en Windows como *Visual* dBase Objects, Visual FoxPro, Delphi, 1C, etc. El archivo DBF, contiene una tabla que está formada por el encabezado del registro y datos de los mismos. El encabezado del registro, define la estructura del archivo y contiene cualquier información relacionada con la tabla. (5)(6)

1.2.1 Microsoft Access

Microsoft Access es un sistema interactivo de administración de bases de datos para Windows que tiene la capacidad de organizar, buscar y presentar la información resultante del manejo de sus bases de datos. Entre sus principales características se encuentran:

- Es gráfico, por lo que aprovecha al máximo la potencia gráfica de Windows, ofreciendo métodos usuales de acceso a los datos y proporcionando métodos simples y directos de trabajar con la información.
- Facilita la administración de datos, ya que sus posibilidades de consulta y conexión ayudan a encontrar rápidamente la información deseada, cualquiera que sea su formato o lugar de almacenamiento.
- Produce formularios e informes sofisticados y efectivos, así como gráficos y combinaciones de informes en un solo documento.
- Permite lograr un considerable aumento en la productividad mediante el uso de los asistentes y las macros. Estos permiten automatizar fácilmente muchas tareas sin necesidad de programar. (7)

Entre sus mayores inconvenientes figuran que no es multiplataforma, pues sólo está disponible para sistemas operativos de Microsoft, su uso es inadecuado para grandes proyectos de software que requieren tiempos de respuesta más breves. Es un software privativo por cuanto se debe pagar un precio por su uso, no es recomendable para bases de datos en cuanto a volumen de datos o de usuarios.

1.2.2 Microsoft Excel

No es más que una hoja de cálculo con la capacidad de almacenar datos, se ha convertido en un pilar fundamental de la informática financiera, Excel es además, ampliamente considerado como uno de los programas de hojas de cálculo de más fácil acceso, con diseño intuitivo, funcionalidad simple y asistentes de ayuda para guiar a los nuevos usuarios a través de los procesos más complejos, por esta razón ha sido muy útil para empresas cubanas, sin embargo posee el inconveniente de que en las hojas electrónicas de cálculo se puede perder información cuando la aplicación se divide en demasiados archivos. Microsoft cambió por completo la interfaz del usuario para todos sus programas de oficina sin excluir a Excel, además de grandes cambios extensivos a todo el sistema, con el objetivo de hacer el trabajo más eficiente; pero provocó que los usuarios no entendieran bien al programa, trayendo consigo un malgasto de tiempo. También se modificaron los archivos de Excel a partir de la versión 2007 con la extensión ". xlsx", lo que significó un gran problema para los usuarios que hacían uso de versiones anteriores, pues solo a partir de esa versión se podían leer ese tipo de archivos. (8)

Otros inconvenientes de importancia que posee Microsoft Excel es el gran tamaño que pueden llegar a alcanzar sus archivos, solo es compatible con los sistemas operativos Windows y Mac, además de poseer un alto costo por ser software privativo.

1.2.3 MongoDB

MongoDB es un sistema de base de datos multiplataforma orientado a documentos, de esquema libre, escrito en C++, lo que le confiere cierta cercanía al *bare metal* o recursos de hardware de la máquina, de modo que es bastante rápido a la hora de ejecutar tareas. Además está licenciado como GNU AGPL 3.0, por lo que se trata de un software de licencia libre. Funciona en sistemas operativos Windows, Linux, OS X y Solaris. Se caracteriza por su velocidad y su rico pero sencillo sistema de consulta de los contenidos de la base de datos. Los distintos documentos se almacenan en formato BSON, o *Binary* JSON. (9) A pesar de sus puestas en práctica en algunas grandes empresas, se enfrenta a un problema de credibilidad importante con muchas empresas. No hay una gran cantidad de desarrolladores y administradores que conocen la tecnología, lo que hace difícil a las empresas encontrar personas con los conocimientos técnicos apropiados. Posee pocas normas en común con demás bases de datos, lo que significa que es difícil cambiar simplemente de un proveedor a otro.

1.2.4 CouchDB

La filosofía con la cual está diseñado este motor, está orientada al relax (relajación) y lo que busca es que las personas entiendan fácilmente los conceptos y funcionamiento de la herramienta, esto incluye diseñadores, los cuales en muchas ocasiones requieren de funcionalidades avanzadas en su desarrollo, por lo general son complejas de implementar con los sistemas convencionales. CouchDB está diseñado para manejar errores en ambientes de producción donde se pueden presentar fallas, de tal forma, que el usuario no se deba preocupar mucho por estas, soportado de esta manera la escalabilidad. Aunque posee muchas ventajas el uso de CouchDB, presenta serios problemas con la seguridad, por defecto, la base de datos permite el acceso a cualquier persona, aún si se crean administradores que vigilen por su seguridad además se tiene acceso al administrador web donde se pueden ver que bases de datos existen y otras informaciones. No es integrable con algunos *frameworks* que requieren bases de datos relacionales para su funcionamiento. (10) (11)

1.2.5 PostgreSQL

PostgreSQL es uno de los SGBD de código abierto más utilizado del mundo, distribuido bajo licencia BSD, lleva más de 15 años desarrollándose y su arquitectura posee de una excelente reputación por su fiabilidad, integridad de datos y estabilidad; (1) dispone de versiones para prácticamente todos los sistemas operativos posee soporte para llaves foráneas, *joins*, vistas y disparadores e incluye la mayoría de los tipos de datos de SQL92 y SQL99, de igual forma, soporta el almacenamiento de grandes objetos binarios como imágenes, sonidos y vídeos. Tiene interfaces de programación nativas para C/C++, Java, .Net, Perl, Php, Python, Ruby, Tcl y ODBC, entre otros, y una excepcional documentación. (12)

Este gestor ofrece sofisticadas características tales como *tablespaces*, replicación asíncrona, transacciones anidadas (*savepoints*), copias de seguridad en caliente (en línea), un sofisticado planificador (optimizador) de consultas y escritura anticipada del registro (*write ahead logging*) para ser tolerante a fallos de hardware además soporta juegos de caracteres internacionales, codificaciones de caracteres *multibyte*, *unicode* y ordena dependiendo de la configuración del idioma local, diferencia mayúsculas y minúsculas así como el formato, es altamente escalable tanto en la cantidad bruta de datos que puede manejar así como en el número de usuarios concurrentes que puede atender.

Por las características que presenta este SGBD, la UCI hace uso del mismo. PostgreSQL presenta un grupo de funcionalidades que no cubren totalmente las necesidades de las empresas cubanas

imposibilitando su adaptación al mismo; gracias a la flexibilidad que presenta el gestor, se le pueden agregar funcionalidades extras. Para ello se utilizan extensiones las cuales hacen optimizaciones para lograr una mejor manipulación de los datos contenidos en la base de datos, aumentando así las potencialidades del gestor, siendo estas insuficientes para lograr adaptar el mismo a las empresas cubanas.

Extensión PostgreSQL

Una extensión PostgreSQL incluye múltiples objetos de SQL, característica que permite recoger dichos objetos en un solo paquete para simplificar la gestión de base de datos PostgreSQL, a este paquete se le nombra extensión. Para definirla, se necesita al menos un script que contiene los comandos SQL para crear los objetos de la extensión y un archivo de control que especifica algunas propiedades básicas de la propia extensión. Algunas extensiones incluyen tablas de configuración, que contienen datos que pueden ser añadidos o modificados por el usuario después de la instalación de la extensión. Entre las ventajas del mecanismo de extensión es que proporciona formas convenientes para administrar las actualizaciones de los comandos SQL que definen los objetos de una extensión. El mecanismo de actualización se puede utilizar para resolver un importante caso especial: convertir una colección de los objetos "sueltos" en una extensión. (13)

Entre las extensiones que cuenta PostgreSQL se encuentra PostGIS, extensión que añade soporte de objetos geográficos a PostgreSQL y permite realizar análisis mediante consultas SQL espaciales o mediante conexión a aplicaciones GIS (Sistema de Información Geográfica). (14) Otra de las extensiones más recientes añadidas al gestor es Multicorn la cual permite acceder a datos externos al gestor. (15)

Multicorn

Multicorn es una extensión de PostgreSQL la cual fue incorporada al gestor a partir de su versión 9.1 desarrollada en el lenguaje de programación Python, permite acceder a cualquier fuente de datos en la base de datos PostgreSQL mediante CDF. (16) Esta extensión tiene como antecedente la funcionalidad llamada SQL/MED, la cual fue introducida en el estándar SQL y no es más que una forma estandarizada de manejar el acceso a objetos remotos en bases de datos SQL. (16) Tanto en SQL/MED como en Multicorn existen los llamados FDW que es una especie de "driver" para acceder a tipos de datos externos.

Multicorn posee un conjunto de conectores que permiten acceder a datos externos de PostgreSQL; a continuación se muestran algunos de los conectores que actualmente se encuentran disponibles en su sitio web oficial, entre los que se encuentran:

- CsvFdw: permite acceder a datos en ficheros CSV.
- GitFdw: permite acceder a los datos de Git.
- GoogleFdw: permite acceder a búsquedas de Google.
- ImapFdw: permite el acceso a mensajes electrónicos almacenados en un servidor mediante el protocolo IMAP.
- LdapFdw: permite el acceso a datos almacenados en un directorio LDAP.
- ProcessFdw: permite hacer consultas del estado del sistema.
- RssFdw: permite acceder a datos almacenados en fuentes RSS.
- SqlAlchemyFdw: permite acceder a los datos almacenados en un Sistema de Gestión de Bases de Datos Relacionales RDBMS remoto.

1.3 Metodología de desarrollo de Software

Una metodología es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo. (17)

Las metodologías están encaminadas a estructurar, planear y controlar el proceso de desarrollo en sistemas de información. Toda metodología debe estar adecuada a las necesidades del software y cuánto pueda abarcar el mismo. Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un software. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo software, pero los requisitos de un software a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del mismo.

Estas metodologías se pueden catalogar en dos grupos, las tradicionales (pesadas) y las ágiles (ligeras):

- Las tradicionales se enfocan en el control del proceso, estableciendo estrictamente las actividades involucradas, los artefactos que se deben producir así como las herramientas y notaciones que se usarán.
- Las metodologías ágiles dan protagonismo al individuo, colaboración con el cliente y desarrollo incremental del software con iteraciones cortas. Esta se usa en los proyectos con requisitos cambiantes y de poca complejidad en su desarrollo, pero manteniendo una alta calidad.

1.3.1 Metodología Extreme Programming (XP)

Es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizada para proyectos cuyo plazo de entrega es corto y el equipo de desarrollo es pequeño. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. Metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, la comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como una metodología especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. El cliente tiene el derecho de añadir, cambiar o quitar requerimientos en cualquier momento y permite al desarrollador crear el sistema con la mejor calidad posible, pedir al cliente en cualquier momento aclaraciones de los requerimientos y cambiar los requerimientos en base a nuevos descubrimientos. (19) Por estas características se decide utilizar XP, además de que constituye la línea base de la arquitectura definida para los proyectos del departamento PostgreSQL. (19)

1.4 Herramientas y Tecnologías

Las herramientas y tecnologías de información apoyan al desarrollo de aplicaciones y desempeñan un papel fundamental en su evolución, constituyen el soporte en la administración del proyecto, adecuándose a las características y objetivos propios de la organización. Con el fin de realizar un software con la calidad requerida se realizó un análisis de las tecnologías y herramientas necesarias para el desarrollo de la aplicación.

1.4.1 Lenguaje de Modelado

El lenguaje de modelado se utiliza en combinación con una metodología de desarrollo de software, proporcionando terminologías para permitir el correcto modelado de cada uno de los objetos. En este caso se centra en la representación gráfica de un sistema. Provee características para modelar un sistema de software a nivel arquitectónico. Mediante este lenguaje la arquitectura de un sistema se define como un conjunto de componentes, conectores y las conexiones entre ellos.

Lenguaje Unificado de Modelado (*Unified Modeling Language*, por sus siglas en inglés UML)

El Lenguaje Unificado de Modelado (UML) es un lenguaje estándar de modelado para software, un lenguaje para la visualización, especificación, construcción y documentación de los artefactos de sistemas en los que el software juega un papel importante. Básicamente UML permite a los desarrolladores visualizar los resultados de su trabajo en esquemas o diagramas estandarizados. (20)

UML es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Se utiliza para entender, diseñar, examinar, configurar, mantener, y controlar la información sobre los sistemas. Permite modelar, construir y documentar los elementos que forman un sistema de software orientado a objetos estandarizando a 9 tipos de diagramas. Se ha convertido en poco tiempo en una notación estándar no sólo para la comunidad de investigadores en ingeniería del software, sino también para la industria. Se decide utilizar UML por las características antes mencionadas además es un lenguaje consolidado en el mundo y también en la UCI.

1.4.2 Herramienta de Modelado: Visual Paradigm for UML 8.0

Visual Paradigm es una herramienta que permite realizar modelado UML siguiendo el estándar UML 2.1. Esta herramienta tiene unas características gráficas muy cómodas que facilitan la realización de los diagramas de modelado que sigue el estándar de UML, que son: Diagramas de clase, Casos de Uso, Comunicación, Secuencia, Estado, Actividad y Componentes.

Visual Paradigm ofrece:

- Entorno de creación de diagramas para UML 2.1.
- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.

- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDE.
- Disponibilidad en múltiples plataformas. (21)

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Por estas características se decide utilizar esta herramienta de modelado además de ser gratuita, fácil de instalar, utilizar y actualizar.

1.4.3 Lenguaje de Programación

Un lenguaje de programación es un conjunto de sintaxis y reglas semánticas diseñadas para describir el conjunto de acciones consecutivas que un equipo debe ejecutar, es decir es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo. Permite especificar de una manera más precisa sobre qué datos debe operar una computadora, sobre cómo estos datos deben ser almacenados o transmitidos y qué acciones debe tomar bajo una determinada circunstancia.

Python

Python es un lenguaje de programación con una sintaxis muy limpia que favorece un código legible. Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos. (22)

- **Lenguaje Interpretado o de Script:** 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. Python tiene, 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 pseudo-código máquina intermedio llamado *bytecode* la primera vez que se ejecuta, generando archivos .pyc o .pyo (*bytecode* optimizado) además que son los que se ejecutarán en sucesivas ocasiones. (22)

- **Fuertemente Tipado:** No se permite tratar a una variable como si fuera de un tipo distinto al que se tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente. Por ejemplo, si tenemos una variable que contiene un texto (variable de tipo cadena o string) no podremos tratarla como un número (sumar la cadena “9” y el número 8). En otros lenguajes el tipo de la variable cambiaría para adaptarse al comportamiento esperado, aunque esto es más propenso a errores. (22)

- **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 librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios. (22)

- **Orientada a Objetos:** La orientación a objetos es un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se trasladan a clases y objetos en nuestro programa. La ejecución del programa consiste en una serie de interacciones entre los objetos. Python también permite la programación imperativa, programación funcional y programación orientada a aspectos. (22)

Este lenguaje es con el cual se realizó Multicorn, pues posee una Interfaz de Programación de Aplicaciones API (del inglés *Application Programming Interface*) fuerte y robusta, es fácil añadir "optimizaciones" mediante la implementación de operaciones como la traducción a SQL. Los conectores a implementar para la extensión se desarrollan en este lenguaje de programación pues la misma está desarrollada en Python.

1.4.4 Entorno de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado (por sus siglas en inglés IDE), es un programa informático conformado por un conjunto de herramientas de programación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDE proveen un marco de trabajo amigable para la

mayoría de los lenguajes de programación tales como C++, java, C#, Delphi, Visual Basic y Python. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. Los IDE pueden ser aplicaciones por sí solos o pueden ser parte de aplicaciones existentes, además de dedicarse en exclusiva a un solo lenguaje de programación.

JetBrains PyCharm 2.7

PyCharm es un entorno de desarrollo integrado (IDE) utilizado para programar en Python. Proporciona análisis de código y un depurador gráfico. Este IDE es desarrollado por JetBrains, compañía checa de desarrollo del software y tecnología muy reconocida mundialmente.

Características

- Asistencia y análisis de código con completamiento, resaltando los errores de sintaxis y propiciando soluciones rápidas.
- Navegación por el código y proyecto: Facilita las vistas especializadas de proyectos, además de vistas de estructura de archivos y salto rápido entre los mismos, así como entre clases y métodos.
- Python *Refactoring*: permite renombrar, extraer método, introducir variables e introducir constantes.
- Django herramientas de desarrollo web.
- Posee un depurador integrado de Python.
- Unidad de Pruebas Integrado.
- Integración a controladores de versiones: interfaz de usuario unificada para Mercurial, Git, Subversion, Perforce y CVS con listas de cambios y fusionar.

PyCharm posee varias opciones de licencia, que cuentan con la funcionalidad del software mismo y difieren en su precio y las condiciones de uso, es gratuito para las instituciones educativas y los proyectos de código abierto. IDE multi-plataforma del cual existen versiones para Windows, Mac OS X y Linux. (23) Por tener estas características es seleccionado este IDE para el desarrollo de los conectores para Multicorn.

1.4.5 PgAdmin III

PgAdmin III es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia OpenSource. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets[2], lo que permite que se pueda usar en Linux, FreeBSD, Solaris, Mac OS X y Windows. (24)

PgAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados. La conexión al servidor puede hacerse mediante conexión TCP/IP. Por todas las facilidades que ofrece esta herramienta, fue seleccionada para realizar las consultas de forma gráfica a las fuentes de datos.

Conclusiones del Capítulo

En este Capítulo se estudió detalladamente los conceptos de base de datos, gestor de base de datos; se explica además conceptos asociados al gestor PostgreSQL, entre ellos: extensión, Multicorn y FDW. Fue seleccionada la metodología Extreme Programing como guía de desarrollo de software. Se seleccionaron las herramientas y tecnologías para la implementación de los conectores, determinando utilizar Visual Paradigm 8.0 como herramienta CASE, que emplea como lenguaje de modelado UML y el IDE JetBrains PyCharm 2.7, como lenguaje de programación Python.

[2] wxWidgets: Las wxWidgets son unas bibliotecas multiplataforma y libres, para el desarrollo de interfaces gráficas programadas en lenguaje C++. (24)

Capitulo 2: Análisis y Diseño

Introducción:

En el siguiente capítulo se presenta la propuesta de conectores para Multicorn. Se hacen especificaciones sobre sus características, además sobre sus requisitos funcionales y no funcionales. Se definen las historias de usuarios y se realiza un diagrama de clases del diseño que se utiliza como complemento de la metodología definida. Para el negocio se establece un modelo de dominio a través del cual se puede entender mejor el problema. Asimismo se exponen las tareas de ingeniería y el plan de iteraciones en las cuales serán implementadas cada historia de usuario y se realiza una estimación de la duración de las tareas de la ingeniería garantizando una correcta implementación.

2.1 Propuesta de conectores a implementar

En marcos de eventos realizados por la Comunidad Técnica Cubana de PostgreSQL se aplicaron encuestas a especialistas de diferentes organismos. Los resultados de estas encuestas se tomaron como base para determinar cuáles conectores implementar, ya que al analizarlas detenidamente se descubrió que numerosas instituciones hacen uso de esas fuentes y Multicorn no posee conectores para acceder a los datos contenidos en ellos (Tablas 1 y 2).

Tabla 1. Encuestas realizadas

Encuesta	Año	Lugar	Cantidad de Encuestados
Utilización y Desarrollo de PostgreSQL.	2009	UCIENCIA-1er PGDay Cubano	85
Necesidad de los Sistemas de Gestión de Bases de Datos.	2010	2do PGDay Cubano	90
Utilización y Desarrollo de PostgreSQL.	2010	2do PGDay Cubano	95
Utilización y Desarrollo de PostgreSQL.	2010	Escuela de Verano 2010	160
Utilización y Desarrollo de PostgreSQL.	2010	Escuela de Verano 2010	160

Necesidad de los Sistemas de Gestión de Bases de Datos.	2011	3er PGDay Latinoamericano	120
Utilización y Desarrollo de PostgreSQL.	2011	3er PGDay Latinoamericano	120
Encuesta a participantes en Diplomado en Tecnologías de Bases de Datos PostgreSQL. Primera Edición.	2011	Escuela de Verano 2011	97
Encuesta sobre impacto del uso de PostgreSQL en empresas cubanas.	2011	Investigación para Tesina de Diplomado.	134

En la tabla anterior se recoge el nombre de las encuestas realizadas, el año en el que fueron realizadas así como en el evento que fueron realizadas y la cantidad de encuestados en cada una de ellas.

Tabla 2. Organismos encuestados

Organismo	Encuestados (%)
Universidad de las Ciencias Informáticas (UCI)	49
Empresa de Telecomunicaciones de Cuba S.A (ETECSA)	21
Centro Coordinador para la Formación y Desarrollo del Capital Humano del MIC (FORDES)	11
Ministerio de la Informática y las Comunicaciones (MIC)	4
Grupo Empresarial GEOCUBA	2
Empresa Nacional de Software S.A (DESOFT)	3
Oficina Nacional de Estadísticas (ONE)	2
Ministerio del Azúcar (MINAZ)	1
Empresa cubana especializada en electrónica. (COPEXTEL)	2
Ministerio de Educación Superior (MES)	1
Joven Club de Computación y Electrónica.	3

Esta tabla demuestra el porciento de organismos que participaron en las encuestas recogidas en la tabla anterior (Tabla 1). En las encuestas también se recogieron datos relacionados con los gestores que utilizan estos organismos quedando graficado en la siguiente figura (Figura 2).

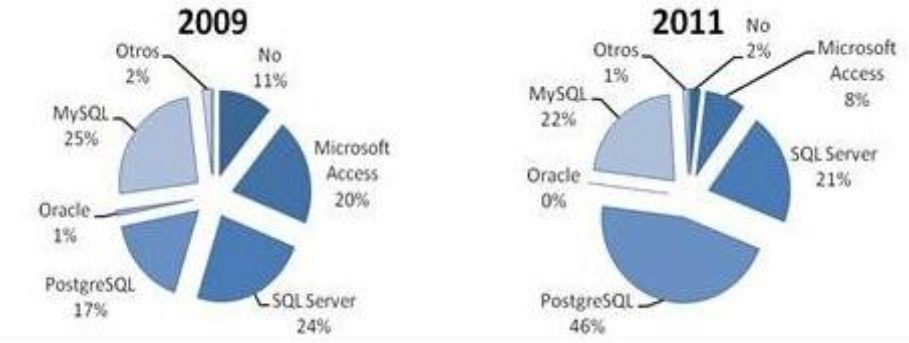


Figura 1. Uso de los SGBD en Cuba

Conectores de datos adicionales para Multicorn

Como se explicó en el capítulo anterior estos conectores no son más que FDW que permiten acceder a la información contenida en tablas de base de datos externas a PostgreSQL, las cuales serán implementadas para luego ser adicionadas a la extensión Multicorn, facilitando la migración a PostgreSQL. Las FDW añadidas serán FdwAccess, FdwCouchDB, FdwDBF, FdwExcel y FdwMongoDB.

2.2 Modelo de Dominio

Se llama modelo del dominio a la representación visual de los conceptos u objetos del mundo real en un dominio de interés. Este modelo agrupa los conceptos de un dominio siendo el mecanismo fundamental para comprender el dominio del problema y para establecer conceptos comunes. (25)

La metodología XP no establece una técnica específica para definir el negocio, por lo que es necesario realizar este proceso de manera entendible para llevar a cabo el desarrollo de los conectores. El modelo de dominio presenta como objetivo principal ayudar a comprender los conceptos con los que trabajan y utilizan los usuarios y con los que deberá trabajar la aplicación. El modelo de dominio se describe mediante diagramas de UML específicamente mediante diagramas de clases.

El modelo de dominio es un diagrama con los objetos que existen (reales), relacionados con el proyecto que se va a realizar y las relaciones que hay entre ellos. En la Figura 2 se muestra como el usuario interactúa con el gestor PostgreSQL que puede poseer la extensión Multicorn mediante la cual se accede

a los datos externos al PostgreSQL mediante los conectores de Multicorn, ellos heredan el método *execute* de la interfaz *ForeignDataWrapper*.

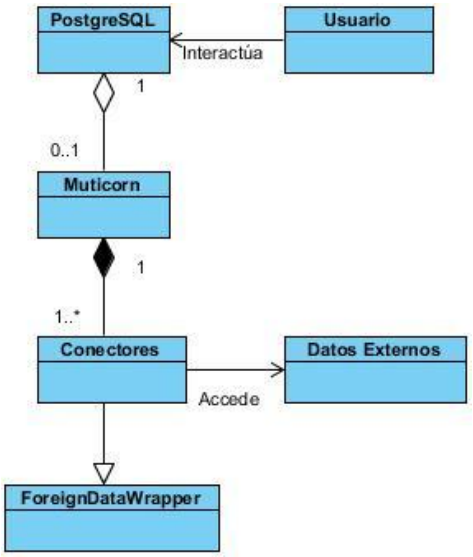


Figura 2. Diagrama Modelo de Dominio

Conceptos del Diagrama Modelo de Dominio:

Usuario: Persona que interactúa con el SGBD PostgreSQL.

PostgreSQL: SGBD PostgreSQL.

Multicorn: Extensión Multicorn.

Conectores: Conectores de datos de Multicorn.

Datos Externos: Fuentes de Datos Externos a PostgreSQL.

ForeignDataWrapper: Interfaz definida en Python que contiene las FDW de las cuales necesitan los conectores a implementar.

2.3 Historias de Usuario (HU)

Uno de los artefactos más importantes que genera la metodología XP son las Historias de Usuario. Estas tienen similar propósito que los casos de uso y son confeccionadas por el cliente. Las mismas expresan el punto de vista del cliente en cuanto a las necesidades del sistema. Son descripciones cortas y escritas en

el lenguaje del usuario sin terminología técnica que proporcionan los detalles sobre la estimación del riesgo y cuánto tiempo conllevará su implementación. (26)

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. (27)

También se utilizan en la fase de pruebas, para verificar si el programa cumple con lo que especifica la historia de usuario. Para el presente trabajo se obtienen un total de siete (HU) que serán realizadas en dos iteraciones, a continuación se muestran las HU correspondientes a esta investigación (Ver tablas siguientes 3 - 9).

Tabla 3. Historia de Usuario "Conectar con fuente de datos Excel"

Historia de Usuario	
Número 1	**Nombre de la Historia de Usuario:** Conectar con fuente de datos Excel.
Cantidad de modificaciones a la Historia de Usuario: 1	
Usuario: Angel Manuel López Acosta	**Iteración asignada:** 1
Prioridad en negocio: Muy Alto	**Puntos estimados:** 2
Riesgo en desarrollo: Alto	**Puntos reales:** 2
Descripción: Permite a Multicorn conectarse a fuentes de datos Excel.	
Observaciones: Se necesitan poseer conocimientos mínimos sobre tablas de Microsoft Excel.	
Prototipo de interfaces:	

Tabla 4. Historia de Usuario "Conectar con fuente de datos Access"

Historia de Usuario	
Número 2	**Nombre de la Historia de Usuario:** Conectar con fuente de datos MongoDB.
Cantidad de modificaciones a la Historia de Usuario: 1	
Usuario: Angel Manuel López Acosta	**Iteración asignada:** 1
Prioridad en negocio: Muy Alto	**Puntos estimados:** 2
Riesgo en desarrollo: Alto	**Puntos reales:** 2
Descripción: Permite a Multicorn conectarse a fuentes de datos de MongoDB.	
Observaciones: Se necesitan poseer conocimientos mínimos sobre el SGBD MongoDB.	
Prototipo de interfaces:	

Tabla 5. Historia de Usuario "Conectar con fuente de datos MongoDB"

Historia de Usuario	
Número 3	**Nombre de la Historia de Usuario:** Conectar con fuente de datos Access.
Cantidad de modificaciones a la Historia de Usuario: 1	
Usuario: Angel Manuel López Acosta	**Iteración asignada:** 2
Prioridad en negocio: Muy Alto	**Puntos estimados:** 2
Riesgo en desarrollo: Alto	**Puntos reales:** 2
Descripción Permite a Multicorn conectarse a fuentes de datos Access.	
Observaciones: Se necesitan poseer conocimientos mínimos sobre bases de datos Microsoft Access.	

Prototipo de interfaces:

Tabla 6. Historia de Usuario "Conectar con fuente de datos CouchDB"

Historia de Usuario	
Número 4	**Nombre de la Historia de Usuario:** Conectar con fuente de datos CouchDB.
Cantidad de modificaciones a la Historia de Usuario: 1	
Usuario: Angel Manuel López Acosta	**Iteración asignada:** 3
Prioridad en negocio: Muy Alto	**Puntos estimados:** 2
Riesgo en desarrollo: Alto	**Puntos reales:** 2
Descripción Permite a Multicorn conectarse a fuentes de datos de CouchDB.	
Observaciones: Se necesitan poseer conocimientos mínimos sobre el SGBD CouchDB.	
Prototipo de interfaces:	

Tabla 7. Historia de Usuario "Conectar con fuente de datos DBF"

Historia de Usuario	
Número: 5	**Nombre de la Historia de Usuario:** Conectar con fuentes de datos DBF
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Angel Manuel López Acosta	**Iteración asignada:** 3
Prioridad en negocio: Media	**Puntos estimados:** 2
Riesgo en desarrollo: Medio	**Puntos reales:** 2
Descripción: Permite a Multicorn conectarse a fuentes de datos DBF.	

Observaciones: Se necesitan poseer conocimientos mínimos sobre archivos dbf.
Prototipo de interfaces:

Tabla 8. Historia de Usuario "Integrar Conectores"

Historia de Usuario	
Número: 6	**Nombre de la Historia de Usuario:** Integrar conectores.
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Angel Manuel López Acosta	**Iteración asignada:** 4
Prioridad en negocio: Alto	**Puntos estimados:** 1.8
Riesgo en desarrollo: Medio	**Puntos reales:** 1.8
Descripción: Integra los componentes desarrollados a la extensión Multicorn.	
Observaciones: Esta historia de usuario actualiza Multicorn para que además agregue los cambios al núcleo de Python.	
Prototipo de interfaces:	

Tabla 9. Historia de Usuario "Generar Ayuda"

Historia de Usuario	
Número: 7	**Nombre de la Historia de Usuario:** Generar ayuda.
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Angel Manuel López Acosta	**Iteración asignada:** 4

Prioridad en negocio: Media	**Puntos estimados:** 0.2
Riesgo en desarrollo: Medio	**Puntos reales:** 0.2
Descripción: Manual que guía al usuario para un mejor entendimiento de la extensión y del trabajo con DataWrappers.	
Observaciones: Se genera una ayuda para el usuario.	
Prototipo de interfaces:	

2.4 Lista de reserva del Producto

La lista de reserva del producto es una tabla (Ver tabla 10) que contiene los requisitos funcionales que debe cumplir la aplicación que se desea realizar, ordenados según la prioridad de implementación, ubicados en Muy Alta, Alta, Media y Baja, en esta última aparecen los requisitos funcionales de menor complejidad, además de los requisitos no funcionales del sistema a desarrollar. Indica la estimación de cada uno de ellos su implementación por semanas y quien hizo la estimación.

Tabla 10. Lista de Reserva del Producto

Ítem *	Descripción	Estimación	Estimado por
Prioridad: Muy Alta			
1	Conectarse a fuentes de datos CouchDB.	2	Programador
2	Conectarse a fuentes de datos Excel.	2	Programador
3	Conectarse a fuentes de datos Access	2	Programador
4	Conectarse a fuentes de datos MongoDB.	2	Programador
5	Conectarse a fuentes de datos DBF.	2	Programador
Prioridad: Alta y Media			
6	Integrar conectores a Multicorn	1.8	Programador

7	Generar Ayuda	0.2	Programador
Prioridad: Baja			
8	Facilidad de uso. Para utilizar esta extensión es necesario poseer conocimientos básicos de PostgreSQL.		
9	Soporte. Se dispondrá de documentación técnica que describa el funcionamiento de Multicorn, de modo que existirá un manual para el uso de la misma orientado a clientes y usuarios finales.		
10	Portabilidad. Extensión multiplataforma, lo que permitirá poder disponer de la misma en cualquier sistema operativo.		
11	Software. Para poder hacer uso de esta extensión se debe tener instalado las siguientes dependencias: libpq-dev postgresql-server-dev-9.1 o superior python-dev python-setuptools		
12	Hardware. Se necesita 512 MB de memoria RAM		

13	mínimo recomendable, 1GB de espacio libre en el disco duro para su instalación y el micro a 300 MHz. Disponibilidad. Se podrá hacer uso de la extensión, siempre que estén instaladas todas dependencias, y las librerías o módulos necesarios de Python: python-xlrd, dbfpy, pyodbc, python-couchdb y python-pymongo		
14	Restricciones de diseño e implementación. Para el desarrollo de la extensión se usará el lenguaje de programación Python con el IDE JetBrains PyCharm 2.7.		

2.5 Tareas de la Ingeniería

El equipo de desarrollo evalúa cada historia de usuario descrita y las divide en tareas, donde cada una de estas representa una característica del sistema. A continuación se muestran las tareas de la ingeniería (TI) (Ver tablas siguientes 11 - 17) donde se muestra el encargado de programarlas así como una breve descripción de la misma.

Tabla 11. Tarea de la Ingeniería "Implementar Excel DataWrapper"

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Implementar Excel DataWrapper	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2

Fecha Inicio: 1/02/2013	**Fecha Fin:** 14/02/2013
Programador Responsable: Angel Manuel López Acosta	
Descripción: Es la encargada de Importar datos desde ficheros Excel hacia el gestor PostgreSQL utilizando el módulo nombrado python-xlrd que permite la lectura de ficheros de Excel en Python.	

Tabla 12. Tarea de la Ingeniería "Implementar Access DataWrapper"

	Tarea de Ingeniería
Número Tarea: 2	**Número Historia de Usuario:** 2
Nombre Tarea: Implementar Access DataWrapper	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 15/2/2013	**Fecha Fin:** 28/02/2013
Programador Responsable: Angel Manuel López Acosta	
Descripción: Es la encargada de Importar datos desde Access hacia el gestor PostgreSQL utilizando el módulo nombrado python-pyodbc que permite el uso de ODBC para conectarse a bases de datos, entre las cuales se encuentra Access, en Python.	

Tabla 13. Tarea de la Ingeniería "Implementar MongoDB DataWrapper"

	Tarea de Ingeniería
Número Tarea: 3	**Número Historia de Usuario:** 3
Nombre Tarea: Implementar MongoDB DataWraper	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 1/3/2013	**Fecha Fin:** 14/3/2013
Programador Responsable: Angel Manuel López Acosta	
Descripción: Es la encargada de Importar datos desde MongoDB hacia el gestor PostgreSQL utilizando el módulo nombrado python-pymongo que permite acceder a Bases de Datos MongoDB en Python.	

Tabla 14. Tarea de la Ingeniería "Implementar CouchDB DataWrapper"

	Tarea de Ingeniería
Número Tarea: 4	**Número Historia de Usuario:** 4
Nombre Tarea: Implementar CouchDB DataWrapper	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 15/03/2013	**Fecha Fin:** 28/03/2013
Programador Responsable: Angel Manuel López Acosta	
Descripción: Es la encargada de Importar datos desde CouchBD hacia el gestor PostgreSQL utilizando el módulo nombrado python-pycouchdb que permite acceder a datos de CouchDB en Python.	

Tabla 15. Tarea de la Ingeniería "Implementar DBF DataWrapper"

	Tarea de Ingeniería
Número Tarea: 5	**Número Historia de Usuario:** 5
Nombre Tarea: Implementar DBF DataWrapper	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Fecha Inicio: 29/03/2013	**Fecha Fin:** 11/04/2013
Programador Responsable: Angel Manuel López Acosta	
Descripción: Es la encargada de Importar datos desde ficheros DBF hacia el gestor PostgreSQL utilizando el módulo nombrado python-dbfpy que permite acceder a ficheros DBF en Python.	

Tabla 16. Tarea de la Ingeniería "Integrar Conectores"

Tarea de Ingeniería	
Número Tarea: 6	**Número Historia de Usuario:** 6
Nombre Tarea: Integrar conectores	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1.8
Fecha Inicio: 12/04/2013	**Fecha Fin:** 23/04/2013
Programador Responsable: Angel Manuel López Acosta	
Descripción: Se integran los conectores desarrollados a Multicorn	

Tabla 17. Tarea de la Ingeniería "Generar Ayuda"

Tarea de Ingeniería	
Número Tarea: 7	**Número Historia de Usuario:** 7
Nombre Tarea: Generar ayuda.	
Tipo de Tarea: Soporte	**Puntos Estimados:** 0.2
Fecha Inicio: 24/04/2013	**Fecha Fin:** 25/04/2013
Programador Responsable: Angel Manuel López Acosta	
Descripción: Manual que guía al usuario para trabajar con los conectores desarrollados.	

2.6 Plan de Iteraciones

Después de ser descritas e identificadas las historias de usuario, se procede a la planificación de la etapa de implementación del sistema, es necesario crear un plan donde se indiquen las historias de usuario que se crearán para cada versión del programa. Un plan de iteraciones es una planificación donde los desarrolladores y clientes establecen los tiempos de implementación ideales de las historias de usuario. Este plan refleja exactamente cuáles HU serán implementadas en cada iteración. (28)

Para el desarrollo de la aplicación se definieron 4 iteraciones las cuales se detallan a continuación:

Tabla 18. Plan de Iteraciones

Relase	Descripción de la Iteración	Orden de la HU a implementar	Duración total
1	En esta iteración se realizarán dos historias de usuarios de prioridad muy alta, estas historias de usuarios se encargarán de realizar los conectores hacia las diferentes fuentes de datos Excel y MongoDB.	1-2	4 semanas
2	En esta iteración se realizará la historia de usuario “Conectar con fuente de datos Access” de prioridad muy alta. El objetivo de esta iteración es continuar con la implementación de los conectores.	3	2 semanas
3	En esta iteración se realizarán dos historias de usuario de prioridad muy alta. Los objetivos de esta iteración son continuar la implementación de los conectores CouchDB y DBF hacia las respectivas fuentes de datos.	4-5	4 semanas
4	En esta iteración se realizarán dos historias de usuario, una de prioridad alta y otra de prioridad media. Los objetivos de esta iteración son integrar los conectores implementados y generar una ayuda.	6-7	2 semanas

2.7 Modelo del Diseño

La implementación de un proyecto de desarrollo requiere de un buen diseño de sus clases para de esta forma realizarlo con la mejor calidad posible logrando una mayor satisfacción del cliente. La metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML. En su lugar se usan otras técnicas como las tarjetas CRC (clase, responsabilidad y colaboración). No obstante el uso de estos diagramas puede aplicarse siempre y cuando influyan en el mejoramiento de la comunicación, no sea un peso su mantenimiento, no sean extensos y se enfoquen en la información importante. (29)

2.7.1 Tarjetas CRC

Las tarjetas CRC trabajan con una metodología basada en objetos, estas garantizan que el equipo completo contribuya en la tarea del diseño.

Las tarjetas CRC están divididas en 3 partes:

Clase: Representa una colección de objetos similares.

Responsabilidades: Describen las funciones que debe realizar una clase, es aquello que la clase sabe o hace.

Colaboraciones: Describen las demás clases con las que trabaja una clase en conjunto para llevar a cabo sus responsabilidades.

Estas tarjetas sirven para diseñar el sistema en conjunto con todo el equipo de desarrollo permitiendo enfocar el modo de pensar y apreciar la tecnología de objetos. A continuación se muestran las tarjetas CRC del diseño de los conectores. (Tablas 19-24)

Tabla 19. Tarjeta CRC "ForeignDataWrapper"

Tarjeta CRC	
Clase: ForeignDataWrapper	
Responsabilidades	**Colaboraciones**
Conectar con datos externos	-

Tabla 20. Tarjeta CRC "AccessFdw"

Tarjeta CRC	
Clase: AccessFdw	
Responsabilidades	**Colaboraciones**
Importar datos desde Access	ForeignDataWrapper

Tabla 21. Tarjeta CRC "ExcelFdw"

Tarjeta CRC	
Clase: ExcelFdw	
Responsabilidades	**Colaboraciones**
Importar datos desde Excel	ForeignDataWrapper

Tabla 22. Tarjeta CRC "MongoDBFdw"

Tarjeta CRC	
Clase: MongoDBFdw	
Responsabilidades	**Colaboraciones**
Importar datos desde MongoDB	ForeignDataWrapper

Tabla 23. Tarjeta CRC "CouchDBFdw"

Tarjeta CRC	
Clase: CouchFdw	
Responsabilidades	**Colaboraciones**
Importar datos desde CouchDB	ForeignDataWrapper

Tabla 24. Tarjeta CRC "DBFFdw"

Tarjeta CRC	
Clase: DBFFdw	
Responsabilidades	**Colaboraciones**
Importar datos desde ficheros DBF	ForeignDataWrapper

2.7.2 Diagrama de Clases

El diagrama de clases es una herramienta esencial durante el proceso de análisis y diseño del sistema. Representa de una manera estática la estructura de información del sistema y la visibilidad que tiene cada una de las clases así como sus relaciones con los demás en el modelo. (30)

A la hora de programar es necesario realizar un diagrama con todas o algunas de las clases que el programador tiene en mente para empezar la implementación. Esto hace que se dificulte menos el trabajo y se realice de manera organizada. A continuación se muestra el diagrama de clases de los conectores a implementar para un mejor entendimiento.

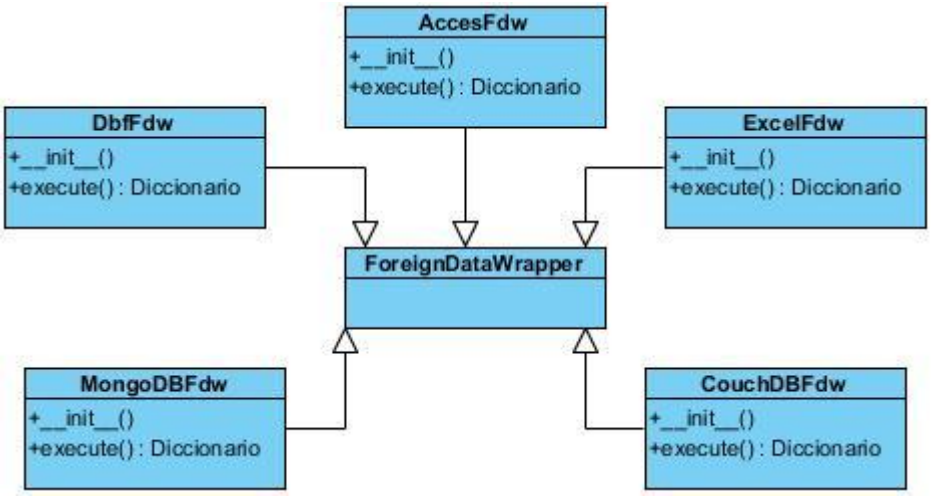


Figura 3. Diagrama de Clases del Diseño

El Diagrama de Clases del Diseño para los conectores a implementar está conformado por cinco clases correspondiente a cada uno de los conectores: *AccessFdw*, *ExcelFdw*, *CouchDBFdw*, *MongoDBFdw* y *DBFFdw* los cuales heredan de la interfaz *ForeignDataWrapper*.

2.8 Arquitectura de Software

La definición oficial de arquitectura del software es de la IEEE (Instituto de Ingenieros Eléctricos y Electrónicos) que plantea: "*La arquitectura del software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán de conjunto con los principios que orientan su diseño y evolución*." (17)

Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software como a continuación se especifica.

2.8.1 Estilos arquitectónicos. Patrones de Arquitectura

Involucrados en una arquitectura se encuentran los estilos arquitectónicos. Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica. Los diferentes estilos tienen sus fortalezas y debilidades, haciendo que sea más fácil o más difícil trabajar con diferentes obstáculos en alguna que otra situación. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software. (31)

Existen numerosos estilos arquitectónicos entre los que se encuentra el estilo de llamada y retorno y dentro de él las arquitecturas estratificadas (capas). Las capas dentro de una arquitectura son un conjunto de servicios especializados que pueden ser accesibles por múltiples clientes y que deben ser fácilmente reutilizables.

La arquitectura en capas es en realidad un estilo de programación donde el objetivo principal es separar los diferentes aspectos del desarrollo, tales como las cuestiones de presentación, lógica de negocio, mecanismos de almacenamiento, etc.

El modelo en N-capas posee numerosas ventajas:

- Desarrollos paralelos (en cada capa)
- Aplicaciones más robustas debido al encapsulamiento
- Mantenimiento y soporte más sencillo (es más sencillo cambiar un componente que modificar una aplicación monolítica)
- Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad)
- Alta escalabilidad. La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. El crecimiento es casi lineal y no es necesario añadir más código para conseguir esta escalabilidad.

Para el desarrollo de los conectores se dividió al gestor PostgreSQL en varias capas, en las mismas se encuentran diferentes tablas, dependiendo de su uso. En este análisis se divide en 4 capas, en la capa 1 se agrupan las tablas del sistema, en la capa 2 las tablas locales, en la capa 3 las tablas foráneas, las cuales se comunican con la capa 4 en donde se encuentran las FDW, las cuales son utilizadas para acceder a datos que no se encuentren en PostgreSQL como se muestra en la figura 4.

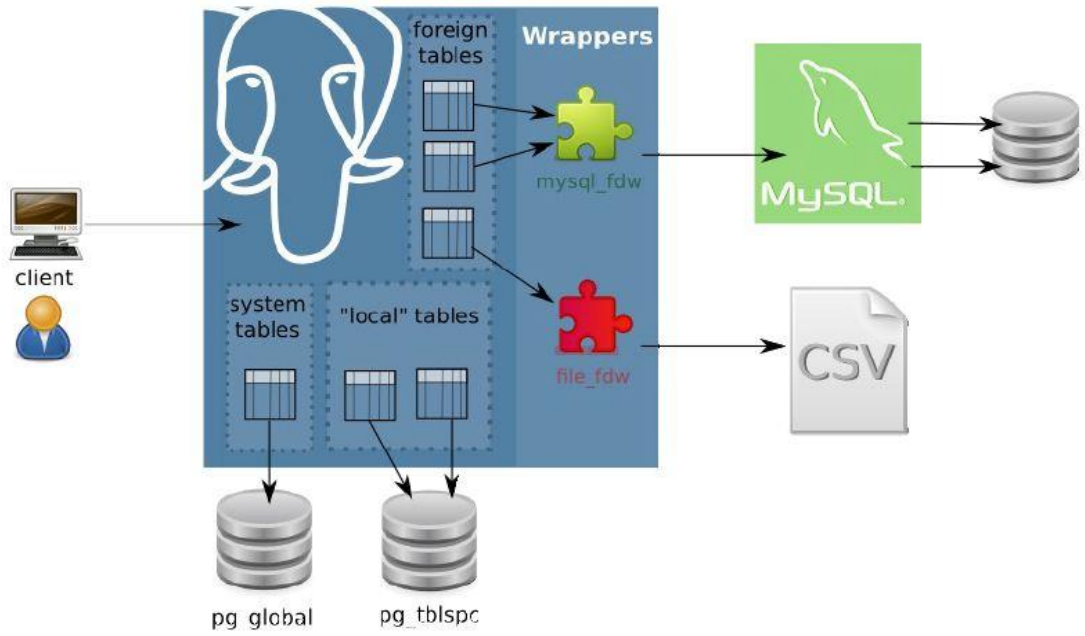


Figura 4. Arquitectura de los conectores de datos de Multicorn

2.10 Patrones de Diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema general de diseño en un contexto particular. (19)

Con el uso de patrones de diseño se evita la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente. Permite formalizar un vocabulario común entre diseñadores y estandarizar el modo en que se realiza el diseño.

2.10.1 Patrones GRASP

Los patrones GRASP (Patrones de Software para la asignación General de Responsabilidad) constituyen un apoyo para el programador, pues ayudan a entender el diseño de objetos. De los diferentes patrones que ofrece GRASP se ha tenido en cuenta para la modelación de los conectores los siguientes:

Patrón Experto

El patrón experto se usa al asignar responsabilidades. Ofrece como solución asignar las responsabilidades a las clases que tienen la información necesaria para cumplir con estas para las cuales son creadas sin depender de ninguna otra. Es un principio básico que suele utilizarse en el diseño orientado a objetos. El uso de este patrón se evidencia en cada uno de los conectores pues cada uno tiene la responsabilidad de conectar con una fuente de datos diferente.

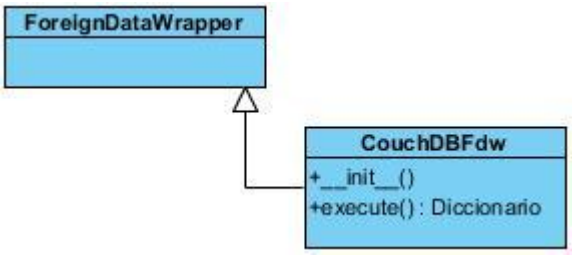


Figura 5. Fragmento del diagrama de clases donde se evidencia la utilización del patrón GRASP: Experto

Alta Cohesión Bajo Acoplamiento

El patrón alta cohesión es la meta principal que ha de buscarse en todo momento, se debe tener presente en todas las decisiones de diseño. El alto nivel de cohesión es presentado por las clases que tienen responsabilidades moderadas en un área funcional y colaboran con otras para llevar a cabo las tareas, no donde dichas clases abarcan el volumen de las responsabilidades a realizar sin importar su complejidad. El bajo acoplamiento estimula la asignación de responsabilidades de forma tal que la inclusión de éstas no incremente el acoplamiento, es decir, significa asignar una responsabilidad para mantener pocas dependencias entre las clases. Ambos patrones son complementarios, como se muestra en la siguiente figura se evidencia alta cohesión entre *ForeigDataWrapper*, *MongoDBFdw* y *CouchDBFdw*; se evidencia bajo acoplamiento entre *MongoDBFdw* y *CouchDBFdw*.

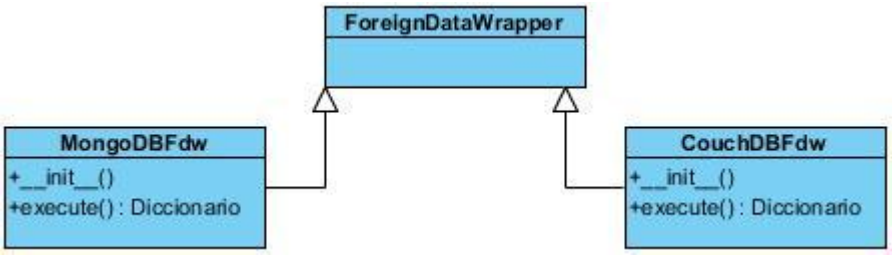


Figura 6. Fragmento del diagrama de clases donde se evidencia la utilización de los patrones GRASP: Alta Cohesión y Bajo Acoplamiento

2.10.2 Patrones GOF

Los patrones GOF (*Gang of Four*, "Pandilla de los Cuatro") recopilan una serie de patrones de diseño, agrupados en tres categorías: de creación, de estructura y de comportamiento. De los diferentes patrones que ofrece GOF se ha tenido en cuenta en la modelación de los conectores para Multicorn el siguiente patrón de estructura.

Estructurales: Describen las clases y objetos que pueden ser combinados para establecer grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos. Entre estos patrones se utiliza el patrón:

- *Bridge*: Desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente, permite tener diferentes implementaciones de una abstracción

El uso de este patrón se evidencia entre interfaz *ForeigDataWrapper* y los restantes *Wrappers* como se muestra en la figura 7.

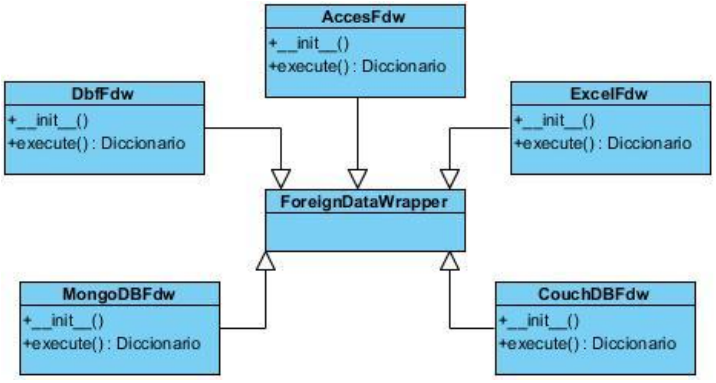


Figura 7. Diagrama que demuestra utilización del patrón GOF: Bridge

Conclusiones del Capítulo

En el capítulo han sido analizados los elementos que describen las características y diseño de los conectores propuestos para Multicorn. Se identificaron 7 historias de usuarios que describen los requisitos funcionales de los conectores y se estableció en que iteración y prioridad se desarrollarían. Se definió un diagrama de clases del diseño y modelo de dominio, artefactos que no son generados por la metodología utilizada, pero son necesarios para lograr un mejor entendimiento en la implementación. Se identificaron los patrones de diseño, el patrón de arquitectura a utilizar y se definieron 7 tareas de la ingeniería.

Capítulo 3: Implementación y Pruebas

Introducción:

El contenido que se aborda en este capítulo está relacionado con la descripción de la implementación del sistema, para darle solución a las Historias de Usuario definidas en el capítulo anterior, se realizan las tareas de Ingeniería, se define el estándar de codificación y se especifican las pruebas a las que fueron sometidos los conectores y la extensión en su conjunto.

3.1 Implementación de los conectores para Multicorn

La implementación tiene como objetivo principal desarrollar la arquitectura y el sistema como un todo, así como definir la organización del código. Para llevarla a cabo se desglosan las HU en tareas de ingeniería, que fueron descritas en el capítulo anterior, las cuales guían la implementación, siendo así más fácil el desarrollo del producto logrando una programación eficiente.

3.1.1 Estándar de codificación

Los 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. (32)

Los estándares de codificación permiten entender de manera rápida, fácil y sencilla, el código empleado en el desarrollo de un software. Es de gran importancia el usar técnicas de codificación y realizar buenas prácticas de programación con vistas a generar un código de alta calidad. Si se aplica de forma continua un estándar de codificación bien definido, y posteriormente se efectúan revisiones del código, caben muchas posibilidades de que un proyecto de software se convierta en un sistema fácil de comprender y de mantener, garantizando un mantenimiento óptimo de dicho código por parte del programador. A continuación se presenta un estándar definido para la implementación de los conectores.

Identación

La unidad de identado es de 4 espacios. El uso de la tabulación debe ser evitado porque (tal como se escribía en el siglo pasado) no existe un estándar que determine con precisión el ancho que va a producir la tabulación.

Longitud de la Línea

Evitar líneas con más de 80 caracteres. Cuando una sentencia sobrepase una línea simple del editor, esta deber ser separada en otras. La separación después de un operador, preferentemente luego de una coma. Realizar la ruptura después de un operador disminuye la probabilidad de que al realizar un copiado del código se cometa un error por olvido de la inserción del punto y coma. La siguiente línea debe ser identada con 5 espacios.

Comentarios

Un modo de aumentar la legibilidad de un programa consiste en intercalar comentarios que expliquen su finalidad. Los comentarios sólo tienen por objeto facilitar la legibilidad de los programas para los programadores, pueden escribirse en el idioma que desees. Los comentarios en Python se inician con el símbolo “#”, todo texto desde la almohadilla hasta el final de la línea se considera comentario, en consecuencia, es omitido por Python.

Declaración de variables

El nombre de una variable es su identificador. Hay unas reglas precisas para construir identificadores, si no se siguen, el identificador no es válido. En Python, la primera operación sobre una variable debe ser la asignación de un valor. No se puede usar una variable a la que no se ha asignado previamente un valor.

Declaración de funciones

No debe haber espacio entre el nombre de la función y el paréntesis izquierdo, ni entre este y la lista de parámetros. La línea comienza con la palabra reservada “def” que es la cabecera de la función. Las sentencias del cuerpo deben estar en la línea siguiente. Los nombres de las funciones se rigen por las mismas características que el de las variables. No es necesario utilizar símbolos como: {,}, '\n'

Identificadores

Un identificador puede estar formado por letras minúsculas, mayúsculas, dígitos y/o el carácter de subrayado “_”, con una restricción: que el primer carácter no sea un dígito. Un identificador no puede coincidir con una palabra reservada o palabra clave. Una palabra reservada es una palabra que tiene un significado predefinido y es necesaria para expresar ciertas construcciones del lenguaje. 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.

Sentencias y bloques

Sentencias

Cada línea debe contener como máximo una sentencia. Cada sentencia es terminada con una nueva línea, no es necesario un punto y coma ";" al final de cada sentencia simple. Una sentencia de asignación puede resultar en la asignación de una función o de un objeto como literal. Para expresar más de una sentencia en una línea, se deben separar con ";" .Las líneas en blanco son ignoradas.

Etiquetas

Las sentencias etiquetadas son opcionales. Solo las sentencias while, do, for deben ser etiquetadas.

Sentencia yield

Una sentencia yield no utiliza paréntesis "()" alrededor del valor que se retorna. La expresión cuyo valor se retorna debe comenzar en la misma línea que la palabra reservada yield.

Espacios en blanco

Las líneas en blanco facilitan la lectura determinando secciones de código lógicamente relacionada. Los espacios en blanco pueden o no ser utilizados en las siguientes circunstancias:

- No se debe utilizar un espacio en blanco entre el identificador de una función y el paréntesis que abre a la lista de parámetros. Ello ayuda a distinguir entre palabras reservadas y llamadas a funciones.
- Para cualquier operador binario excepto el punto ".", el paréntesis que abre "(" y el corchete que abre "[" todos deben ser separados por un espacio entre operandos y operador.
    - No se debe utilizar el espacio para separar un operador unario de su operando.
    - Cada punto y coma ";" en una sentencia de control debe ser seguido por un espacio.
    - Debe dejarse un espacio luego de cada coma ",".

Declaraciones de clases

No debe existir un fichero con más de una clase declarada.

A continuación se presenta un ejemplo del estándar aplicado en la implementación de los conectores en este caso para el conector hacia MongoDB. (Figura 8)

```
def __init__(self, options, columns):
    super(MongoForeignDataWrapper, self).__init__(options, columns)
    self.columns = columns
    self.servidor=Connection(str(options.get('server',None)), int(options.get('port',None)))
    self.bd =self.servidor[str(options.get('db',None))]
    self.coleccion=self.bd[str(options.get('col',None))]


def obtenertodos (self):
    for resultado in self.coleccion.find():
        yield resultado

def obteneralgunos (self,condicion):
    for resultado in self.coleccion.find(condicion):
        yield resultado
```

Figura 8.Ejemplo de aplicación del Estándar de Codificación

3.2 Características del sistema

- **Dependencias de software**

Definido el estándar de codificación se procede a implementar los conectores teniendo en cuenta las características descritas en el capítulo anterior. Como se explica en capítulos anteriores Multicorn es una extensión PostgreSQL desarrollada en lenguaje Python la cual se basa en conectores para acceder a datos externos a PostgreSQL. Esta extensión para ser instalada requiere de la instalación previa de una serie de paquetes de software, estos son:

- libpq-dev: Es un paquete de archivos que constituyen una biblioteca estática para la compilación de programas en C, Multicorn los utiliza para comunicarse con una base de datos PostgreSQL.
- postgresql-server-dev-9.1 ó superior: Es un paquete de archivos de desarrollo para la programación del lado del servidor PostgreSQL. Este paquete contiene también los archivos necesarios para la construcción de módulos adicionales para PostgreSQL.
- python-dev: Es un paquete de archivos y biblioteca estática para Python.

- python-setuptools: Es una colección de mejoras para el módulo *distutils* de Python que permite al desarrollador construir y distribuir paquetes Python de forma sencilla, en especial cuando dependen de otros paquetes Python para funcionar.

Cabe destacar que para PostgreSQL en su versión 9.1 debe utilizarse la última versión de Multicorn existente que es la 0.9.1 y para PostgreSQL 9.2 la versión 1.0.0 actualmente se encuentra disponible como la versión 1.0.0beta1, siendo integrados los conectores la versión 0.0.8 última versión disponible para el desarrollador.

- **Instalación de la extensión Multicorn**

Después de haber instalado las dependencias de software descritas anteriormente se procede a la instalación de Multicorn ejecutándose para ello una serie de acciones por consola con permisos de superusuario, que luego de estar posicionados en la carpeta raíz donde se encuentran los archivos de la extensión se ejecutan los comandos <*make*> y <*make install*> quedando de esta forma instalado Multicorn listo para realizar consultas a las fuentes de datos disponibles.

3.2.1 Conectores

A continuación se describe por la complejidad en su desarrollo, el conector AccessFdw el cual permite acceder a fuentes de datos de Microsoft Access.

AccessFdw

Para acceder a las fuentes de datos MS Access se implementa el conector AccessFdw que importa el contenido de las tablas de Access hacia una tabla foránea en PostgreSQL. Este conector utiliza el módulo pyodbc de Python que en ocasiones se encuentra incompleto, por esa razón se explica detalladamente cómo se procede. Es necesario instalar otras dependencias de software:

- mdbtools: Es una herramienta para visualizar bases de datos Access en GNU/Linux.
- libmdbodbc1: Es el controlador de software para acceder a archivos de bases de datos JET/MS Access a través de ODBC.

Después se procede a configurar los siguientes archivos de configuración de esta manera:

- sudo nano /etc/odbcinst.ini
- sudo nano /etc/odbc.ini

```
[MDBTools]
Description = MDBTools Driver
Driver = libmdbodbc.so.1
Setup = libmdbcodbc.so.1
FileUsage = 1
UsageCount = 1
```

```
[MiConexion]
Description = Microsoft Access Try DB
Driver = MDBTools
Database = /path/database.mdb
Servername = localhost
Username = User
Password = Password
```

A continuación se describe el código del conector Implementado:

```
import pyodbc
from multicorn import ForeignDataWrapper
class AccessFdw(ForeignDataWrapper):
        def __init__(self, fdw_options, fdw_columns):
        super(AccessFdw, self).__init__(fdw_options, fdw_columns)
                self.table = fdw_options['table']
                self.columns = fdw_columns
                self. consuta = fdw_options[' consulta']
                self.dns = fdw_options['dns']

        def execute(self, quals, columns):
                c = pyodbc.connect('self.dns')
                csr = c.cursor()
                csr.execute(self.consulta)
                col = self.columns
                lines = csr.fetchall()
                for i in range(len(lines)):
                        line={}
                        cont = 0
                        for j in col:
                        line [j] = lines [i][cont]
                        cont+=1
                yield line
```

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. Hay multitud de conceptos (y palabras clave) asociadas a las pruebas, clasificarlas es difícil, pues no son mutuamente disjuntas, sino muy entrelazadas. La metodología XP propone para validar las necesidades de los usuarios y dirigir la implementación las pruebas unitarias y las de aceptación. (30)

3.3.1 Estrategias de pruebas

Como ya se ha señalado anteriormente existen disímiles estrategias de pruebas, el estudio de este trabajo se ha centrado específicamente en la investigación de las pruebas correspondientes a la metodología de desarrollo de software empleada en el presente trabajo de diploma. XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final. (31)

Pruebas unitarias

Una prueba unitaria es la verificación de un módulo (unidad de código) determinado dentro de un sistema. Son llevadas a cabo por los programadores encargados de cada módulo asegurando que un determinado módulo cumpla con un comportamiento esperado en forma aislada antes de ser integrado al sistema. (32)

Las pruebas unitarias son una de las piedras angulares de XP. Estas pruebas deben ser definidas antes de realizar el código ("*Test-driven programming*"). Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código. Los programadores deben realizar estas pruebas cuando la interfaz de un método de la aplicación no es clara, la implementación es complicada, para probar entradas y condiciones inusuales, luego de modificar algo. Éstas deben contemplar cada módulo del sistema que pueda generar fallas. En XP los programadores deben escribir las pruebas unitarias para cada módulo. Los casos de este tipo de pruebas no hay que escribirlos para todos los módulos, sino destacar en aquellos que exista la posibilidad de fallar.

Pruebas de aceptación

Las pruebas de aceptación, al igual que las de sistema, se realizan sobre el producto terminado e integrado; pero a diferencia de aquellas, están creadas para que sea un usuario final quien detecte los posibles errores. (28)

Estas pruebas son definidas por el cliente para cada historia de usuario, y tienen como objetivo asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. Las pruebas de aceptación corresponden a una especie de documento de requerimientos en XP, ya que marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo hacia donde tiene que ir y en qué puntos o funcionalidades debe poner el mayor esfuerzo y atención. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable al cliente; sino que se realizan sobre el producto terminado o pudiera ser una versión del producto o una iteración pactada previamente con el cliente. Existen dos tipos de pruebas de aceptación:

La prueba alfa

Se lleva a cabo, por un cliente, en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario. Las pruebas alfa se llevan a cabo en un entorno controlado. Para que tengan validez, se debe primero crear un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente. Una vez logrado esto, se procede a realizar las pruebas y a documentar los resultados.

La prueba beta

Se lleva a cabo por los usuarios finales del software en los lugares de trabajo de los clientes. A diferencia de la prueba alfa, el desarrollador no está presente normalmente. Así, la prueba beta es una aplicación "en vivo" del software en un entorno que no puede ser controlado por el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador.

Con el estudio realizado este trabajo decide llevar a cabo la estrategia de pruebas de aceptación debido a que con esta estrategia la presentación de los resultados es importante, en cambio para las unitarias no tiene mucho sentido ya que siempre se requiere un total de efectividad en los módulos más críticos. Esta estrategia en la metodología XP tienen como propósito indicarle al equipo cuando las funcionalidades de

una iteración han sido completadas exitosamente. Significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente. Es casi imposible obtener una implementación libre de fallas, es por ello que se debe tener bien definido un criterio de aprobación para saber cuándo el software está listo para ser liberado.

3.3.2 Técnica de prueba seleccionada

Cualquier proyecto que se trace una estrategia de prueba debe contar con métodos y técnicas para la aplicación de cada una de estas pruebas. A continuación se realiza una breve caracterización de dos técnicas importantes para de esta forma seleccionar una de estas técnicas para la correcta realización de las pruebas.

Pruebas estructurales o Caja blanca

La prueba de la caja blanca es una técnica de diseño de casos de prueba que realiza las pruebas al código de la aplicación para derivar los casos de prueba. Ellas garantizan que el programador pueda ejecutar los caminos independientes de cada módulo al menos una vez y que utilice todas las estructuras de datos internas.

En las pruebas a realizar se seleccionan en función del conocimiento que se tiene de la implementación del módulo. Se suelen aplicar a módulos pequeños, el probador analiza el código y deduce cuántos y qué conjuntos de valores de entrada han de probarse para que al menos se ejecute una vez cada sentencia del código. Se pueden refinar los casos de prueba que se identifican con pruebas de caja negra. (29)

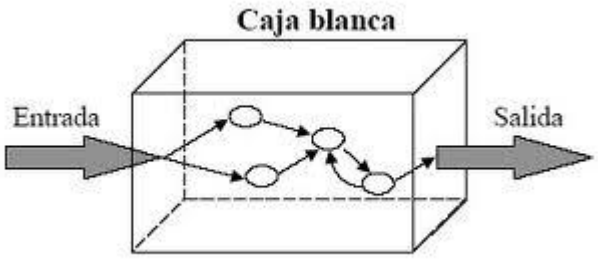


Figura 9. Técnica Caja Blanca

Pruebas de funcionalidad o 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. (30)

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.

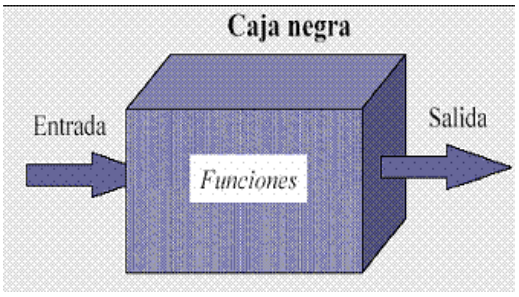


Figura 10. Técnica Caja Negra

3.3.3 Casos de pruebas basados en historias de usuario

Usando técnicas de caja negra en el diseño de los Casos de Prueba, 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. (31)

A través de los casos de pruebas se pueden probar si los conectores realmente funcionan, 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. Se diseñaron un total de seis casos de prueba que permitieron comprobar la totalidad de las funcionalidades definidas en las HU, a continuación se muestran dos de ellos.

Tabla 25. Caso de prueba aplicado al conector CouchDBFdw.

Escenario	Descripción	V 1	V 2	V 3	V 4	Respuesta del sistema	Flujo central
EC 1.1 Realizar consulta con parámetros correctos	Se realiza la consulta SQL con los datos correctos	V dato	V dato	V dato	V dato	Se genera una tabla con el nombre de las columnas y el contenido de las mismas	1- El Usuario ejecuta el script SQL. 2- El Sistema retorna una tabla generada con los resultados de la consulta realizada
EC 1.2 Realizar consulta con parámetros incorrectos	Se realiza la consulta SQL con los datos correctos	I	I	I	I	El sistema muestra errores en la consulta, no se crea la tabla con el contenido de las mismas	1- El Usuario ejecuta el script SQL. 2- El Sistema retorna errores en la consulta y no se crea la Tabla Foránea.

Tabla 26. Caso de Prueba aplicado al conector MongoDBFdw.

Escenario	Descripción	V 1	V 2	V 3	V 4	V 5	Respuesta del sistema	Flujo central
EC 1.1 Realizar consulta con parámetros correctos	Se realiza la consulta SQL con los datos correctos	V dato	V dato	V dato	V dato	V dato	Se genera una tabla con el nombre de las columnas y el contenido de las mismas	1- El Usuario ejecuta el script SQL. 2- El Sistema retorna una tabla generada con los resultados de la consulta realizada
EC 1.2 Realizar consulta con parámetros incorrectos	Se realiza la consulta SQL con los datos correctos	I	I	I	I	I	El sistema muestra errores en la consulta, no se crea la tabla con el contenido de las mismas	1- El Usuario ejecuta el script SQL. 2- El Sistema retorna errores en la consulta y no se crea la Tabla Foránea.

3.3.4 Presentación de los resultados de las pruebas funcionales

Como define la metodología que guía esta investigación, al concluir cada iteración del desarrollo se realizan pruebas a las HU implementadas en cada una de estas iteraciones con el objetivo de minimizar el número de errores al concluir la implementación de los conectores.

Las pruebas han sido aplicadas a través de los Casos de Prueba correspondientes a las 7 HU, permitiendo comprobar que los conectores desarrollados funcionan acorde a las especificaciones funcionales y requisitos definidos por el cliente.

Luego de concluida la implementación de las HU agrupadas en las cuatro iteraciones definidas se detectaron un total de 13 no conformidades, de las cuales 3 de ellas fueron detectadas en la primera iteración, 6 en la segunda y 4 en la tercera. Cabe destacar que en la cuarta iteración del desarrollo no fueron detectados errores, siendo esta la única iteración donde fue necesario realizar una única iteración de pruebas, mientras que en la segunda iteración del desarrollo se hizo necesaria la realización de una tercera iteración de pruebas; en todos los casos las no conformidades fueron resueltas en un período corto de tiempo. A continuación se muestran las gráficas que representan los resultados de las pruebas anteriormente descritas. (Figura 11)

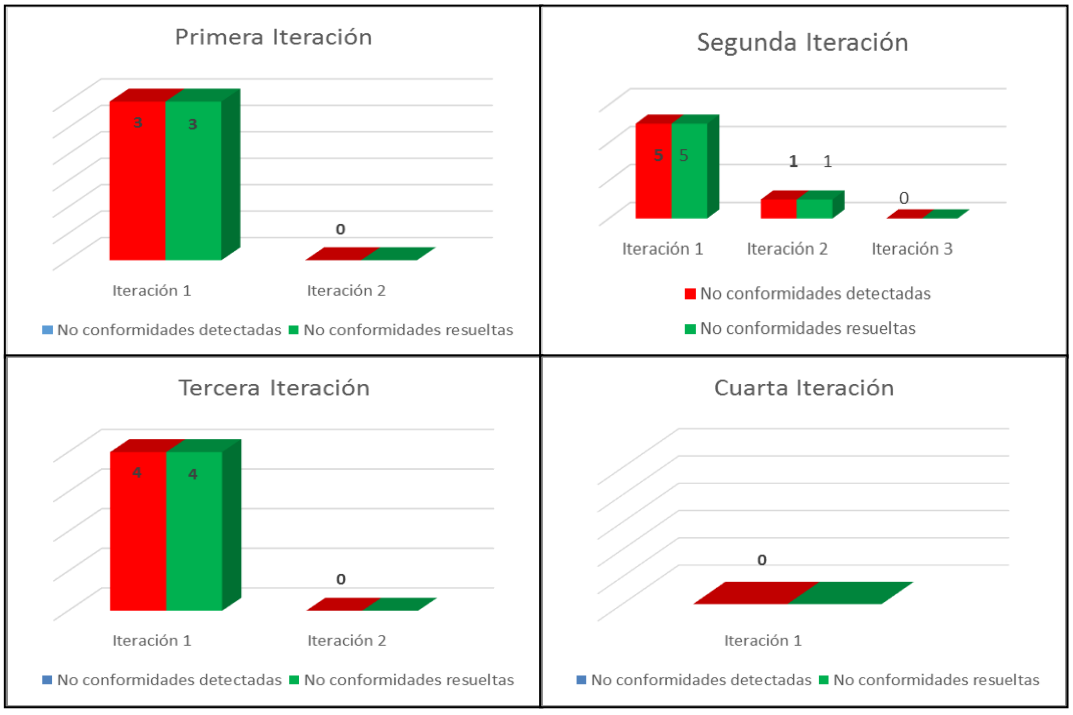


Figura 11. Resultados de las pruebas funcionales por iteraciones del desarrollo.

Conclusiones del Capítulo

En este capítulo se realizó un estudio de las estrategias de pruebas en la metodología XP para finalmente aplicar la que más se ajuste a los CDE. Fueron probadas todas las funcionalidades definidas por el usuario mostrando las no conformidades encontradas, dándole solución en todos los casos en un corto período de tiempo. Se logró presentar los resultados arrojados en cada iteración logrando obtener una aplicación funcional de alta calidad posibilitando su futura comercialización.

Conclusiones Generales

La realización del presente trabajo posibilitó cumplir con los objetivos y tareas propuestas para su desarrollo, arribándose a las siguientes conclusiones:

- Al realizar el análisis se obtuvieron 7 HU de las que se identificaron 7 requisitos funcionales, basado en este artefacto generado se diseñaron los casos de pruebas aplicados en las pruebas a los conectores.
- Se obtuvieron durante el diseño de los conectores 6 tarjetas CRC que describen las clases, sus responsabilidades y colaboraciones así como 6 clases que conformaron el diagrama de clases.
- Se implementaron e integraron los CDE que permiten consultar las fuentes de datos Excel, Access, CouchDB, MongoDB, DBF.

Se validó la solución mediante los casos de prueba diseñados, uno por cada Historia de Usuario demostrando que los conectores cumplen con las funcionalidades identificadas.

Recomendaciones

Después de lograr los objetivos que se trazaron al inicio de la investigación, se proponen las siguientes recomendaciones:

- Se recomienda continuar la investigación sobre las Envolturas de Datos Externas así como la extensibilidad de PostgreSQL y la extensión Multicorn.
- Implementar nuevos conectores que aumenten la extensibilidad del gestor PostgreSQL.

Referencias Bibliográficas

1. **Cristiá Álvarez, Aldo, Ortiz Valmaseda, Marcos Luis y Ortiz Vázquez, Yudisney.** Propuesta del gestor de bases de datos cubano basado en PostgreSQL. [en línea] Vol.: 3, No. 9, (2010). [Consulta el 25 de octubre de 2012]. Disponible en http://publicaciones.uci.cu/index.php/SC

2. **Mesa Reyes, Ing. Yunior.** INFLUENCIA DE LA COMUNIDAD TÉCNICA CUBANA DE POSTGRESQL Y SU PORTAL WEB EN EL IMPACTO DEL USO DEL POSTGRESQL EN LAS EMPRESAS CUBANAS. [Consultado el 3 de octubre de 2012.] Disponible en: http://ccia.cujae.edu.cu/index.php/siia/siia2012/paper/view/1657

3. **Yunta Rodríguez, Luis.** BASES DE DATOS DOCUMENTALES: ESTRUCTURA Y PRINCIPIOS. MALDONADO, Angeles: s.n. ISSN 1988-3455, 2001. [Consultado el: 6 de octubre de 2012.] Disponible en: http://www.unav.es/dpp/documentacion/proteger/lryunta.pdf

4. **Yague, Agustin y Garbajosa, Juan**, Actas de los talleres de las jornadas del software y bases de datos. Madrid: s.n., 2009. Vols. Vol 3, Num4. Universidad Politécnica de Madrid (UPM). [Consultado el: 7 de octubre de 2012.] Disponible en: http://www.sistedes.es/ficheros/actas-talleres-JISBD/Vol-2/No-4/PRIS08.pdf

5. **White Twon.** [en línea] 20 de diciembre de 2011. [Consultado el 1 de enero de 2013] Disponible en: http://www.whitetown.com/es/misc/dbf-format/

6. **White Twon.** [en línea] 20 de diciembre de 2011. [Consultado el 1 de enero de 2013] Disponible en: http://www.whitetown.com/es/misc/dbf/

7. **Balter, Alison**. Alison Balter's Mastering Microsoft® Office Access 2007 Development Copyright © 2007. ISBN 0-672-32932-8 [Consultado el 6 de febrero de 2013] Disponible en: ftp://10.0.0.22/documentacion/Base%20de%20Datos/DBMS/Access/Access 2007 Programming By Example 1i - Korol, J 2008.pdf

8. **Walkenbach, John.** MICROSOFT EXCEL 2010 BIBLE. J WALKENBACH - WILEY PUBLISHING INC.2010

Referencias Bibliográficas

9. **Introducción a MongoDB**. [en línea] 18 de febrero de 2010. [Consultado el 6 de febrero de 2013] Disponible en: http://www.genbetadev.com/bases-de-datos/una-introduccion-a-mongodb

10. **Guía CouchDB.** [en línea] 18 de febrero de 2010. [Consultado el 6 de febrero de 2013] Disponible en: http://guide.couchdb.org/editions/1/en/index.html

11. **Guía CouchDB**. [en línea] 18 de febrero de 2010. [Consultado el 6 de febrero de 2013]Disponible en: http://guide.couchdb.org/editions/1/en/why.html

12. **Garzón Pérez, Teresa.** Sistemas Gestores de Bases de Datos. 2010. 1988-6047. [Consultado el 8 de febrero de 2013] Disponible en: http://www.csi-csif.es/andalucia/modules/mod_ense/revista/pdf/Numero_30/TERESA_GARZON_1.pdf

13. **Sitio Oficial de la Comunidad PostgreSQL** [en línea] 21 de enero 2010. [Consultado el 5 de febrero de 2013]Disponible en: http://www.postgresql.org/docs/9.1/static/extend-extensions.html

14. **PostGIS — Spatial and Geographic Objects for PostgreSQL.** [en línea] 30 de enero 2010. [Citado el 5 de febrero de 2013]. Disponible en: http://postgis.net/

15. **Multicorn.** [en línea] 30 de septiembre 2011. [Consultado el 5 de febrero de 2013]. Disponible en: http://multicorn.org/

16. **PostgreSQL.** [en línea] 30 de septiembre 2011. [Consultado el 5 de febrero de 2013]. Disponible en: http://wiki.postgresql.org/wiki/SQL/MED

17. **Zamudio, Esmeralda Villegas y Méndez, Alejandra Virrueta**. Investigación documental. Metodologías de desarrollo de software. Instituto tecnológico superior de Apatzingán, s.n., 2010.

18. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** El proceso unificado de software. Apéndice A1. 2000.

19. **Mendoza Sánchez, María A.** Metodologías de desarrollo de Software [en línea] 7 de junio de 2004. [Consultado el 10 de enero de 2013.] Disponible en:

ftp://ucistore.uci.cu/documentacion/Ingenieria%20Software/Metodologias/caracteristicas%20breves%20de%20RUP,%20XP,%20MSF.pdf

20. **Pressman**, **Roger S**. Ingeniería de Software, un enfoque práctico. Quinta edición. S.l.: McGraw-Hill Companies, 2002. ISBN: 8448132149.

21. **Sierra, María.** [en línea] [Citado el: 24 de enero de 2012.] Disponible en: http://personales.unican.es/ruizfr/is1/doc/lab/01/is1-p01-trans.pdf.

22. **González Duque, Raúl**. Python para todos. 2008. Disponible en http://mundogeek.net/tutorial-python/.

23. **Sitio Oficial de la Compañía JetBrains**. [en línea] 15 de enero 2011. [Citado el 5 de febrero de 2013] Disponible en: http://www.jetbrains.com/pycharm/

24. **Delgado Dapena, Martha D.** Definición del modelo del negocio y del dominio utilizando UML.

25. **Flores Fernández, Héctor Arturo**. Procesos de ingeniería de software. 2009

26. **Letelier, Patricio y Pnadés, María del Carmen.** Metodologías ágiles para el desarrollo de software, eXtreme Programing (XP). 26, Valencia: s.n., 2006, Vol. 05. 1666-1680.

27. **Rodríguez Guadarrama, Addiel y Mazorra, Thaymí**. Análisis y diseño de una herramienta web para la gestión de la información. Universidad de la Ciencias Informáticas (UCI). 2010.

28. **Mesa Reyes, Yunior y Vázquez Ortiz, Yudisney.** Sistemas de Bases de Datos. Espacio de comunicación e intercambio para la comunidad técnica cubana de PostgreSQL. PostgreSQL. Ciudad de la Habana, Cuba: s.n., 2011. pág. 7. 1994-1536.

29. **Aguilera. M, Carmen P.** Sistema de información para el registro y control de procesos de gestión de higiene ocupacional. UNIVERSIDAD DE ORIENTE. MATURÍN / MONAGAS / VENEZUELA: s.n., 2011. pág. 260.

Referencias Bibliográficas

30. **Serrano Cuayahuitl, Victor Hugo.** Pruebas de unidad, Estándares de codificación y generación automática de código. Facultad de Ciencias básicas, ingeniería y tecnología. Universidad Autónoma de Tlaxcala.

31. **Abran, Alain, y otros. SWEBOK.** Guide to the software Engineering Body of Knowledge. 2004. 0-7695-2330-7.

32. **Gutiérrez, J.J, Escalona, M. Mejías M.J, Torres J.** Pruebas del sistema en Programación Extrema. Sevilla: s.n.

33. **Suárez Pablo y Fontela, Carlos.** Documentación y pruebas ante el paradigma de objetos. 2003.

34. **Drake José M y Patricia López.** Verificación y Validación. 2009.

Bibliografía

1. **Abran, Alain, y otros. SWEBOK.** Guide to the software Engineering Body of Knowledge. 2004. 0-7695-2330-7.

2. **Aguilera. M, Carmen P.** Sistema de información para el registro y control de procesos de gestión de higiene ocupacional. UNIVERSIDAD DE ORIENTE. MATURÍN / MONAGAS / VENEZUELA: s.n., 2011. pág. 260.

3. **Balter, Alison**. Alison Balter's Mastering Microsoft® Office Access 2007 Development Copyright © 2007. ISBN 0-672-32932-8 [Consultado el 6 de febrero de 2013] Disponible en: ftp://10.0.0.22/documentacion/Base%20de%20Datos/DBMS/Access/Access_2007_Programming_By_Example_1i_-_Korol,_J_2008.pdf

4. **Cristiá Álvarez, Aldo, Ortiz Valmaseda, Marcos Luis y Ortiz Vázquez, Yudisney.** Propuesta del gestor de bases de datos cubano basado en PostgreSQL. [en línea] Vol.: 3, No. 9, (2010). [Consulta el 25 de octubre de 2012]. Disponible en: http://publicaciones.uci.cu/index.php/SC

5. **Delgado Dapena, Martha D.** Definición del modelo del negocio y del dominio utilizando UML.

6. **Drake José M y Patricia López.** Verificación y Validación. 2009.

7. **Flores Fernández, Héctor Arturo**. Procesos de ingeniería de software. 2009

8. **Garzón Pérez, Teresa.** *Sistemas Gestores de Bases de Datos.* 2010. 1988-6047.

9. **González Duque, Raúl**. Python para todos. 2008. Disponible en http://mundogeek.net/tutorial-python/.

10. **Guía CouchDB.** [en línea] 18 de febrero de 2010. [Consultado el 6 de febrero de 2013] Disponible en: http://guide.couchdb.org/editions/1/en/index.html

11. **Guía CouchDB**. [en línea] 18 de febrero de 2010. [Consultado el 6 de febrero de 2013] Disponible en: http://guide.couchdb.org/editions/1/en/why.html

12. **Gutíerrez J.J, Escalona M.J, Mejías M., Torres, J.** Pruebas del sistema en Programación Extrema. Sevilla: s.n.

13. **Introducción a MongoDB**. [en línea] 18 de febrero de 2010. [Consultado el 6 de febrero de 2013] Disponible en: http://www.genbetadev.com/bases-de-datos/una-introduccion-a-mongodb

14. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** El proceso unificado de software. Apéndice A1. 2000.

15. **Letelier, Patricio, Pnadés, María del Carmen.** Metodologías ágiles para el desarrollo de software, eXtreme Programing (XP). 26, Valencia: s.n., 2006, Vol. 05. 1666-1680.

16. **Mendoza Sánchez, María A.** Metodologías de desarrollo de Software [en línea] 7 de junio de 2004. [Consultado el 10 de enero de 2013.] Disponible en: ftp://ucistore.uci.cu/documentacion/Ingenieria%20Software/Metodologias/caracteristicas%20breves%20de%20RUP,%20XP,%20MSF.pdf

17. **Mesa Reyes, Yunior y Vázquez Ortiz, Yudisney.** Sistemas de Bases de Datos. Espacio de comunicación e intercambio para la comunidad técnica cubana de PostgreSQL. PostgreSQL. Ciudad de la Habana, Cuba: s.n., 2011. pág. 7. 1994-1536.

18. **Mesa Reyes, Ing. Yunior.** INFLUENCIA DE LA COMUNIDAD TÉCNICA CUBANA DE POSTGRESQL Y SU PORTAL WEB EN EL IMPACTO DEL USO DEL POSTGRESQL EN LAS EMPRESAS CUBANAS. [Citado el: 3 de octubre de 2012.] Disponible en: http://ccia.cujae.edu.cu/index.php/siia/siia2012/paper/view/1657

19. **Multicorn.** [en línea] 30 de septiembre 2011. [consultado el 5 de febrero de 2013]. http://multicorn.org/

20. **PostGIS — Spatial and Geographic Objects for PostgreSQL.** [en línea] 30 de enero 2010. [Consultado el 5 de febrero de 2013]. Disponible en: http://postgis.net/

21. **PostgreSQL.** [en línea] 30 de septiembre 2011. [Consultado el 5 de febrero de 2013]. Disponible en: http://wiki.postgresql.org/wiki/SQL/MED

22. **Pressman**, **Roger S**. Ingeniería de Software, un enfoque práctico. Quinta edición. S.l.: McGraw-Hill Companies, 2002. ISBN: 8448132149.

23. **Rodríguez Guadarrama, Addiel y Mazorra, Thaymí**. Análisis y diseño de una herramienta web para la gestión de la información. Universidad de la Ciencias Informáticas (UCI). 2010.

24. **Serrano Cuayahuitl, Victor Hugo.** Pruebas de unidad, Estándares de codificación y generación automática de código. Facultad de Ciencias básicas, ingeniería y tecnología. Universidad Autónoma de Tlaxcala.

25. **Sierra, María.** [en línea] [Citado el: 24 de enero de 2012.] Disponible en: http://personales.unican.es/ruizfr/is1/doc/lab/01/is1-p01-trans.pdf.

26. **Sitio Oficial de la Compañía JetBrains**. [en línea] 15 de enero 2011. [Consultado el 5 de febrero de 2013] Disponible en: http://www.jetbrains.com/pycharm/

27. **Sitio Oficial de la Comunidad PostgreSQL** [en línea] 21 de enero 2010. [Consultado el 5 de febrero de 2013] Disponible en: http://www.postgresql.org/docs/9.1/static/extend-extensions.html

28. **Suárez Pablo y Fontela, Carlos.** Documentación y pruebas ante el paradigma de objetos. 2003.

29. **Walkenbach, John.** MICROSOFT EXCEL 2010 BIBLE. J WALKENBACH - WILEY PUBLISHING INC.2010

30. **White Twon.** [en línea] 20 de diciembre de 2011. [Consultado el 1 de enero de 2013] Disponible en: http://www.whitetown.com/es/misc/dbf-format/

31. **White Twon.** [en línea] 20 de diciembre de 2011. [Consultado el 1 de enero de 2013] Disponible en: http://www.whitetown.com/es/misc/dbf/

32. **Yunta Rodríguez, Luis.** BASES DE DATOS DOCUMENTALES: ESTRUCTURA Y PRINCIPIOS. MALDONADO, Angeles: s.n. ISSN 1988-3455, 2001. Disponible en: http://www.unav.es/dpp/documentacion/proteger/lryunta.pdf

33. **Yague, Agustin y Garbajosa, Juan**, Actas de los talleres de las jornadas del software y bases de datos. Madrid: s.n., 2009. Vols. Vol 3, Num4. Universidad Politécnica de Madrid (UPM).

34. **Zamudio, Esmeralda Villegas y Méndez, Alejandra Virrueta**. Investigación documental. Metodologías de desarrollo de software. Instituto tecnológico superior de Apatzingán, s.n., 2010.

Anexos

Anexo 1. Tabla de Microsoft Access Importada mediante la extensión Multicorn

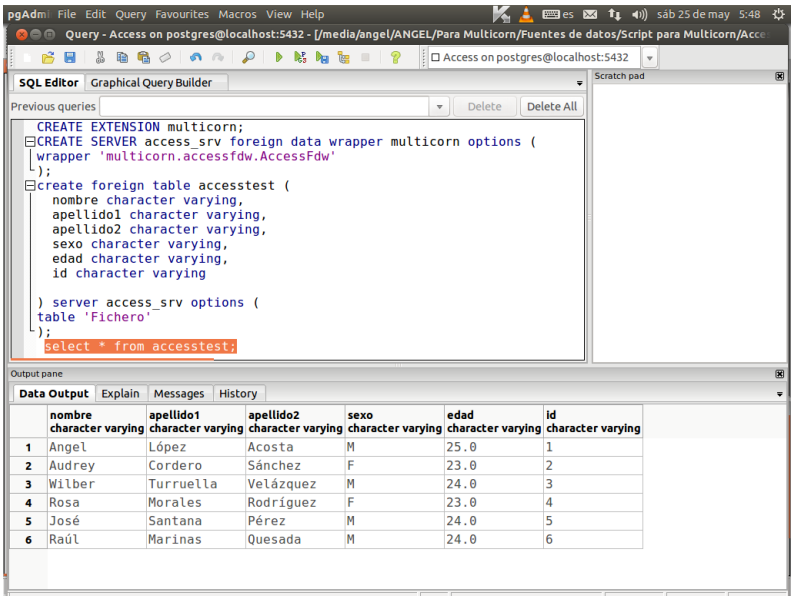


Figura 12. Tabla Creada con los datos importados de un fichero Excel.

Anexo 2. Datos del SGBD CouchDB importada mediante la extensión Multicorn

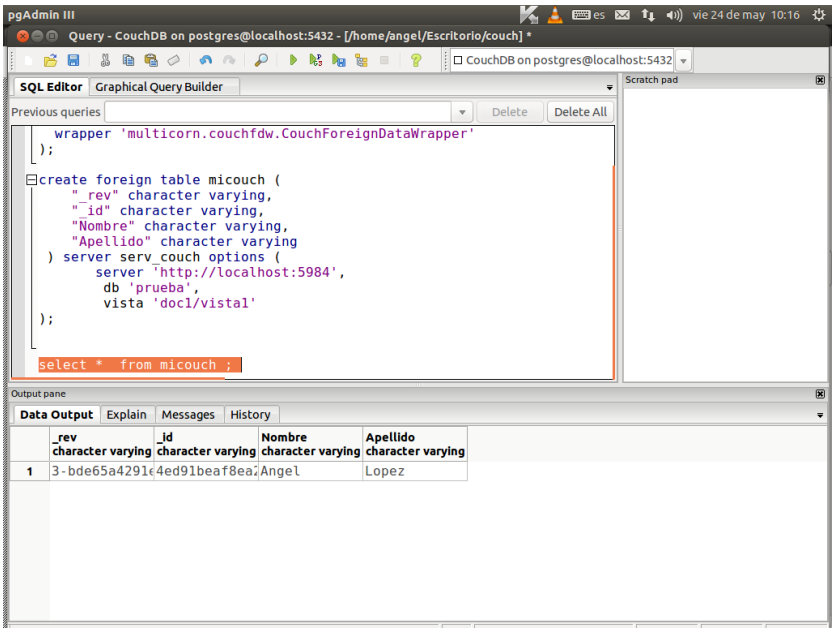


Figura 13. Tabla creada con los datos contenidos en una base de datos CouchDB.

Anexo 3. Datos del SGBD MongoDB importada mediante la extensión Multicorn

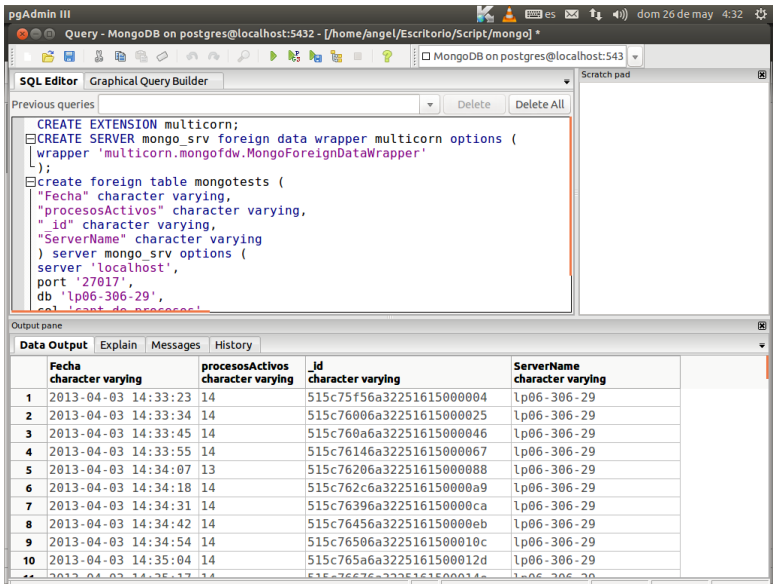


Figura 14. Tabla creada con los datos contenidos en una base de datos CouchDB

Anexo 4. Datos un fichero Excel importada mediante la extensión Multicorn

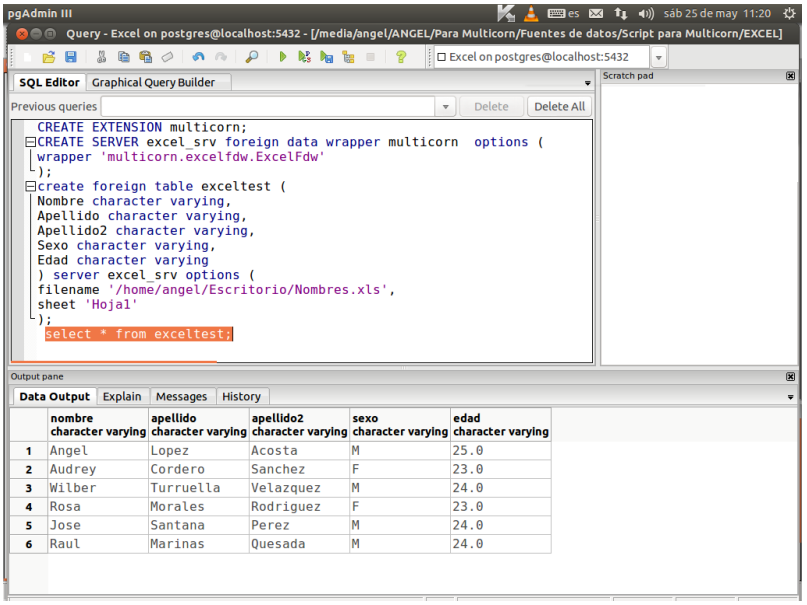
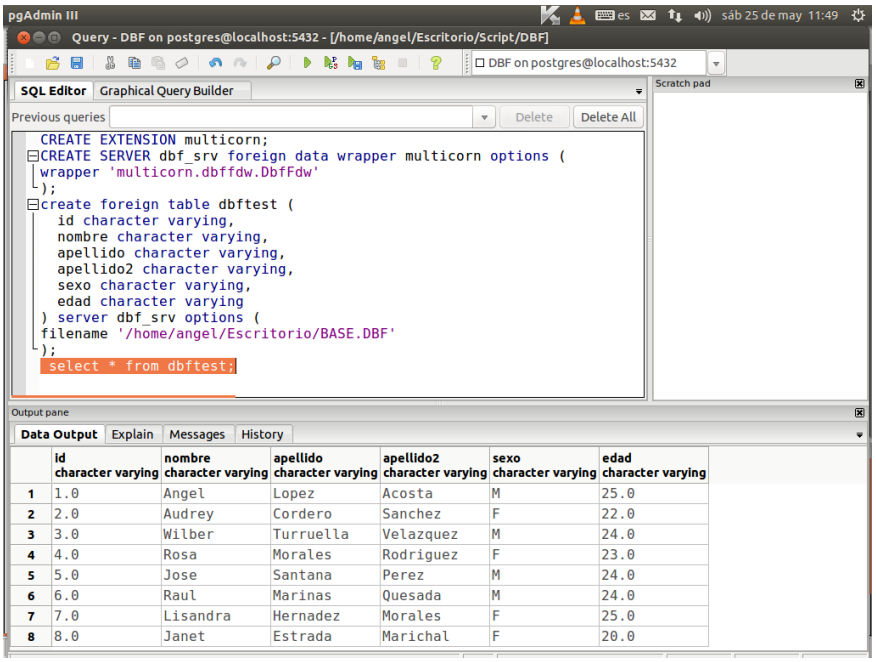


Figura 15. Tabla creada con los datos contenidos en un fichero Excel

Anexo 5. Datos un fichero DBF importada mediante la extensión Multicorn



	id character varying	nombre character varying	apellido character varying	apellido2 character varying	sexo character varying	edad character varying
1	1.0	Angel	Lopez	Acosta	M	25.0
2	2.0	Audrey	Cordero	Sanchez	F	22.0
3	3.0	Wilber	Turruella	Velazquez	M	24.0
4	4.0	Rosa	Morales	Rodriguez	F	23.0
5	5.0	Jose	Santana	Perez	M	24.0
6	6.0	Raul	Marinas	Quesada	M	24.0
7	7.0	Lisandra	Hernadez	Morales	F	25.0
8	8.0	Janet	Estrada	Marichal	F	20.0

Figura 16. Tabla creada con los datos contenidos en un fichero DBF

Glosario de Términos

API: Interfaz de Programación de Aplicaciones.

Bare Metal: Software que se ejecuta directamente sobre el hardware

C: Lenguaje de Programación

C++: Lenguaje de programación.

CASE: Ingeniería de software asistida por computadoras.

CRC: Artefacto generado por la metodología XP, es una tarjeta dividida en tres partes Clase, Responsabilidad y Colaboración.

DATEC: Centro de Tecnologías de Gestión de Datos

GOF*:* Patrones de diseños, *Gang of Four* (Pandilla de los Cuatro).

GRASP: Patrones de *Software* para la Asignación General de Responsabilidad.

HU: Historia de Usuario.

IDE: Entorno de Desarrollo Integrado.

Licencia BSD: La licencia BSD es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution).

ODBC: Es un estándar de acceso a las bases de datos desarrollado por SQL Access Group en 1992.

Pycharm: Entorno de Desarrollo Integrado.

SGBD: Es un conjunto de programas que permite a los usuarios crear y mantener una base de datos, facilitando el proceso de definir, construir y manipular bases de datos.

UCI: Universidad de las Ciencias Informáticas.

UML: Lenguaje de Modelado Unificado.