Universidad de las Ciencias Informáticas

Facultad 6

Trabajo de Diploma para optar por el título de:

Ingeniero en Ciencias Informáticas

Plugin editor de consultas para la herramienta de administración de bases de datos HABD

Autores:

Yadiris Elena Fernández Martínez

Ismel Aguilar Chaveco

Tutores:

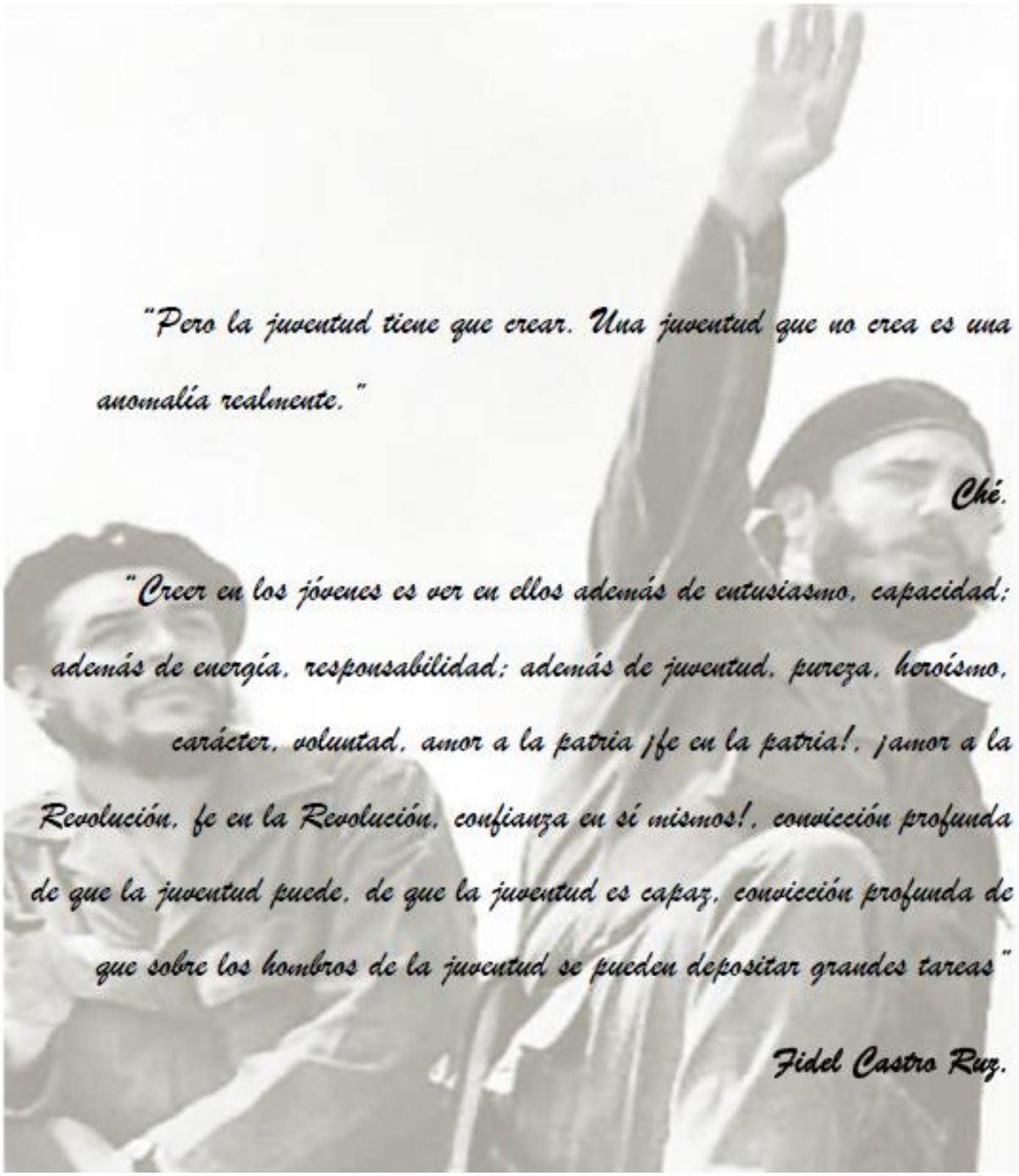
MSc.Ing. Anthony Rafael Sotolongo León

Ing. Glennis Tamayo Morales

Co-tutor:

Ing. Yunior Bauta Pentón

La Habana, junio de 2012


"Pero la juventud tiene que crear. Una juventud que no crea es una anomalía realmente."
Ché.
"Creer en los jóvenes es ver en ellos además de entusiasmo, capacidad; además de energía, responsabilidad; además de juventud, pureza, heroísmo, carácter, voluntad, amor a la patria ¡fe en la patria!, ¡amor a la Revolución, fe en la Revolución, confianza en sí mismos!, convicción profunda de que la juventud puede, de que la juventud es capaz, convicción profunda de que sobre los hombros de la juventud se pueden depositar grandes tareas"
Fidel Castro Ruz.

DECLARACIÓN DE AUTORÍA

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

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

_____

Yadiris Elena Fernández Martínez

Firma del autor

_____

Ismel Aguilar Chaveco

Firma del autor

_____

MSc. Ing. Anthony R Sotolongo León

Firma del tutor

_____

Ing. Glennis Tamayo Morales

Firma del tutor

_____

Ing. Yunior Bauta Pentón

Firma del Co-tutor

DATOS DE CONTACTO

Anthony Rafael Sotolongo: Máster en ciencias informáticas.

Correo: asotolongo@uci.cu

Años de graduados: 5 años de graduado.

Años de experiencia: 2 años de experiencia.

Categoría docente: profesor_ instructor.

Glennis Tamayo Morales: Ingeniera en ciencias informáticas.

Correo: gtamayo@uci.cu

Años de graduados: 1 año de graduado.

Años de experiencia: 1 año de experiencia.

Categoría docente:

Yunior Bauta Pentón: Ingeniero en ciencias informáticas.

Correo: ypenton@uci.cu

Años de graduados: 2 años de graduado.

Años de experiencia: 2 años de experiencia.

Categoría docente: profesor.

AGRADECIMIENTOS

YADIRIS

Quiero agradecerles a esas personas que me dieron el regalo más grande: la vida. Por enseñarme a luchar para hacer realidad mis sueños, por todo su apoyo, sacrificio y dedicación, por depositar en mi toda su confianza, y demostrarme que si se puede, por creer en mí, por todo el amor y cariño infinito que siempre me ha acompañado, por ayudar en mi formación como mujer, como persona y como profesional, con los principios que siempre me inculcaron, ***a mis padres*** *que con su amor y ternura me transformaron en la mujer que soy hoy, los amo con todo mi corazón y no me va a alcanzar la vida para retribuirles su amor. Los amo.*

A mis abuelos Ana, Arnaldo, Mirta y Jesús *que estuvieron siempre presentes en todos mis tropiezos, alegrías, tristezas y logros de la vida. Por ser cómplices de mi niñez y estar pendientes de mí en todo momento. Por dedicarme tantos años de amor, paciencia y comprensión, por estar siempre a mi lado y ser como unos padres para mí. Solo mi corazón sabe cuán grande es el amor que ciento por ustedes. No digo más, los quiero mucho.*

A mi hermana *por ser la mejor hermana de este mundo, por confiar en mí y estar siempre conmigo, por todas las cosas lindas que hemos pasado juntas. Hermanita gracias por ser tu quien más me comprendiera en algunos momentos de la vida. Espero ser esa guía que necesitas en tu vida para seguir adelante. Te adoro.*

A Yordan *por apoyarme y ayudarme todo este tiempo sé que no ha sido fácil estos 5 años pero ya ves lo logramos mi amor. Gracias te doy por ser no solo mi esposo, sino mi amigo y confidente. Por consagrar tu amor, paciencia y dedicación, porque mis sueños, preocupaciones y problemas, fueron tuyos en todo momento. Te quiero.*

A mis tías y tíos *los quiero y gracias por todo, por su ayuda y preocupación, por estirarme su mano cuando en realidad me hacía falta, les agradezco mucho.*

A toda mi familia en general y suegros Moraima y Tati *los quiero y les doy las gracias porque todos han puesto su granito de arena para que mi sueño se hiciera realidad.*

A mis tutores *por su dedicación y apoyo incondicional, por todo el tiempo que dedicaron a apoyarnos en la realización de este trabajo, a Anthony muchas gracias porque con su conocimiento y sabiduría siempre nos guió por el buen camino. A Glennis gracias por atenderme siempre que la he necesitado, por preocuparse con todo mis problemas ya sean de la escuela o personales, por su apoyo incondicional y desinteresado, por ser mi amiga, gracias y mil veces gracias.*

A mi dúo de tesis Ismel, *por ser un buen compañero, gracias por tu comprensión y colaboración, por todas las horas que puso en la realización de este trabajo, te agradezco infinitamente que hayas compartido este momento especial conmigo y te agradezco por todo, eres un magnífico amigo.*

A mis amigos Lili, Carlos, Jennys e Ibet *que en estos cinco años han sido incondicionales, han estado conmigo en los buenos y en los malos momentos, por soportarme todo este tiempo, por estar siempre presentes, y por haber aprendido a lidiar conmigo que no es tarea fácil, Lili sabes que aunque nos fajemos mucho te quiero como una hermana más, Carli siempre has sido mi ejemplo a seguir en esta escuela, por tu apoyo y consejos muchas gracias, Jennys e Ibet muchas gracias por todos los años de convivencia que juntas pasamos, por su preocupación constante y porque siempre me han brindado ayuda, consejos, fuerzas y aliento para seguir adelante.*

A los padres de Lili. *Gracias porque durante toda mi vida estudiantil universitaria me hicieron sentir un miembro más de su hermosa familia. Porque me brindaron su casa para que la convirtiera en mi hogar y me sintiera como en mi propia casa. Porque formaron parte de los momentos más difíciles de mi vida y siempre me tendieron la mano como a una hija. Estoy feliz y orgullosa de poder contar con personas como ustedes, pues son ejemplo para todos, son únicos y especiales.*

A Yeneysi *por todo su apoyo y ayuda en los momentos más difíciles de mi carrera, las pruebas de nivel y la tesis, en verdad te agradezco mucho y te considero una buena amiga. Por ser una excepcional persona, y muy buena estudiante. Gracias a tu intransigencia, tu ahínco por hacer que las cosas salgan bien y tu forma de ser, de siempre querer ayudar a los demás me hicieron vencer obstáculos que se presentaron en mi camino como estudiante universitaria. Muchísimas gracias por no decirme nunca que no cuando lo necesitaba.*

A Dailin, Yeney, Yami, Danae, Betty, Dalied *por compartir momentos extraordinarios que hicieron de mí crecer como estudiante y como persona. Por dejarme encontrar en ustedes grandes amigas, y porque en cierto momento de mi carrera siempre estaban ahí apoyándome y ayudándome a sentirme mejor en lo que me hiciera falta, muchísimas gracias en verdad les agradezco mucho.*

A todos los amigos que he conocido a lo largo de la carrera *por la ayuda que he recibido de cada uno de ellos en cierto momento de la carrera y las cosas que hemos compartido en los buenos y malos momentos (Ayeris, Alexis, Daneili, Yaimé, Yuri, Ricardo Pupo, Adrialis, Rainier, Daniel, Olivia, Erio, Jesús) a todos ellos y a los que no menciono, gracias.*

A todos los profesores *que han contribuido en mi educación, en mi formación como profesional y como persona. También quiero agradecer a la revolución, con la que me siento comprometida y a Fidel que fue el promotor de este proyecto, por darnos la oportunidad de estudiar en esta universidad.*

ISMEL

A mi madre *por haberme enseñado a ser un excelente ser humano y luchar por mí para que juntos pudiéramos lograr mis metas y objetivos. Mama, solo tú sabes encontrar la manera de comprenderme y escucharme en el momento que más me hacía falta. Por creer en mí cuando nadie lo hacía. Porque viviste mis años de universidad junto conmigo .Todas mis tristezas eran tuya, todos mis tropiezos los convertías en experiencia y junto a mi disfrutases mis momentos de triunfo. No puedo expresar en tan pocas líneas lo agradecido que estoy porque la vida me dio una madre tan excelente como tú, simplemente, te amo.*

A mi padre *por ser quien me enseñara a tener el carácter y la disciplina necesaria para ser un hombre de bien. Papa quiero agradecerte por siempre haber estado presente en los momentos más complicados de mi vida. Por ser la persona que me educara y motivara siempre ha darle frente a los problemas. Gracias porque por ti soy un hombre correcto y dispuesto a darle siempre el sí a la vida.*

A mi tío *porque más que tío ha sido siempre mi padre .Me mostraste el lado duro de la vida y me diste la enseñanza necesaria para superarlo. Eres un gran hombre y un ejemplo a seguir para todo aquel que desee ser alguien importante en la vida. La nobleza y la sencillez son tus características principales. Estoy orgulloso de ser más que tu sobrino tu hijo y no hay forma humana ni palabras coherentes que me permitan expresar el agradecimiento que siento por ti.*

A Maylen *por ser la persona que me inspira a amar y por aguantarme todos estos años. Mi amor gracias por la paciencia, la ternura, el apoyo, el cariño y el amor que siempre me has tenido. Solo tu haz sido mi confidente en momentos difíciles. Por ti me he convertido en una mejor persona, me enseñaste a ser cada día mejor y trataste de que aprendiera a comprender mejor las cosas de la vida.*

A Yadiris *mi dúo de tesis que contribuyó y se esforzó para que este sueño se hiciera realidad y para que todas las cosas siempre salieran de la mejor manera posible. Gracias porque sin ti esto no hubiéramos salido adelante, porque no hay nadie mejor capacitada que tú que hubiera logrado cumplir esta meta.*

A Frank y sus padres *por haberme acogido dentro de su hermosa familia. Ustedes se ganaron mi respeto y mi admiración, hoy quiero retribuirles los momentos que apoyaron cuando más lo necesité. En parte este logro es también de ustedes, gracias porque personas como ustedes son únicas.*

A Yeneysi *por ser una persona excepcional, por haberme ayudado y aguantado todos estos años. Gracias a ti he superado muchos obstáculos en mi vida estudiantil universitaria, con tu presencia apoyo y ahínco he logrado obtener y superar muchas metas que en cierto y determinado momento he considerado inalcanzables. Por dejarme encontrar en ti una gran amiga, más que amiga hermana y haberme dado tu apoyo y comprensión en aquellos momentos difíciles de la vida.*

A los tutores *por habernos guiado en todo momento de este trabajo .Fueron ustedes los que me animaron a que luchara por lograr cada cosa en el ámbito profesional. Gracias por el esfuerzo realizado y por la contribución que hicieron para convertirnos en grandes profesionales.*

A mis compañeros *por tantos momentos de alegrías que pasamos juntos, por brindarme su apoyo incondicional en todo momento. No puedo dejar de mencionar a: Juan Carlos Yamacho, Denys Retureta, Jorge Fernandez, Ricardo Viota, Osmel Barrera, Alejandro, Osmar, y otros que por no extender tanto esta lista, pero no por ello son menos importantes. A todos muchas gracias por su preocupación y amistad.*

DEDICATORIA

A mis padres

Este gran triunfo está dedicado a mis padres, a quienes les debo todo lo que tengo y lo que soy en esta vida porque sin ellos nada de esto hubiera sido posible, por ser lo más grande que tengo en el mundo y haberme apoyado mucho estos 5 años que llevo distantes de toda mi familia. Les dedico este trabajo ante todo porque no me han dejado claudicar en el logro de mis sueños. Son mi ejemplo a seguir, mis guías, especialmente por su amor, dedicación, comprensión y ayudarme a lograr este sueño de poder regalarles esta satisfacción de poder verme graduada.

YADIRIS

A mi madre

Tania Chaveco Martínez por ser la mujer más extraordinaria sobre la faz de la tierra. Nadie más que tú se merece este sueño hecho realidad. A ti y solo a ti te dedico el resultado del esfuerzo hecho durante todos estos años. Tú eres mi inspiración y es por ello que estuve motivado durante 5 años para alcanzar esta meta. Estoy orgulloso de ser tu hijo porque no hay amor más bello en este mundo que el que tú me has dedicado mi vida entera. Te quiero y ojalá y estés orgullosa de mi, pues todo lo que hago es siempre pensando en hacerte feliz.

ISMEL

RESUMEN

La investigación pretende dar respuesta a una problemática existente en la línea de investigación PostgreSQL de la Universidad de las Ciencias Informáticas (UCI). En esta línea se está desarrollando la herramienta de administración de bases de datos HABD, su estado actual imposibilita la creación y edición de consultas a las bases de datos, es por esto que surge la necesidad de desarrollar un plugin para editar consultas SQL.

El presente trabajo tiene como objetivo desarrollar un editor de consultas SQL, para ello se realizó una investigación acerca de las herramientas que poseen editores, seleccionando para este estudio un sistema Generador Dinámico de Reportes y cuatro herramientas de administración de bases de datos, tomando como punto de partida PgAdmin y EMS Manager for PostgreSQL. Se identificaron las funcionalidades y se realizó el diseño e implementación del sistema. El plugin desarrollado posibilita completar el código SQL, crear clases con sus atributos, relaciones, criterios de selección, de ordenamiento, proporciona poder efectuar el Insert; Delete y Update directamente a la tabla que se desea gestionar sin la necesidad de crear o implementar la consulta SQL. Permite generar el código de toda la representación visual de la consulta Select sin tener conocimientos previos de sentencias SQL.

PALABRAS CLAVES. Editor de consultas SQL, HABD, PostgreSQL.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las tecnologías aceleran su marcha a medida que el mundo se hace más próspero, trayendo consigo un auge exponencialmente durante los últimos años. Esta masificación tiene como consecuencia el crecimiento a nivel mundial de la informatización y a su vez el aumento en la generación y almacenamiento de la información.

Estos grandes volúmenes de datos no pueden ser procesados de forma manual, pues se haría el trabajo mucho más engorroso, y se afectaría la consistencia de la información. Para atender estas necesidades de un grupo de usuarios, surge la tecnología de las Bases de Datos (BD), considerada la más antigua dentro de las ciencias de la informática, utilizada para manejar gran cantidad de información. En la actualidad el enfoque de BD es extensamente utilizado por ser la única solución posible para manejar grandes volúmenes de datos. La creación de los Sistemas Gestores de Bases de Datos (SGBD)[1], permitió que las BD de hoy día sean mucho más seguras, flexibles y manejables en la mayoría de los casos.

En Cuba en los últimos años ha habido un auge en el desarrollo del *software*, donde el trabajo con las tecnologías se desempeña con mayor calidad. Es por ello que el Gobierno Cubano tiene como una de sus tareas principales, desarrollar la industria cubana del *software* con el objetivo de informatizar la sociedad. Las empresas cubanas al igual que cualquier organización del mundo necesitan de SGBD que permitan la gestión de la información que poseen. Entre las instituciones que se dedican al desarrollo de soluciones informáticas en Cuba se encuentra la Universidad de las Ciencias Informáticas (UCI).

En este contexto surge la UCI, nacida como un proyecto del gobierno cubano denominado al principio proyecto futuro, tiene como objetivos producir *software*, servicios informáticos y desarrollar la industria del *software* para contribuir al desarrollo económico del país. Con el transcurso del tiempo esta universidad se ha convertido en la vanguardia del desarrollo de las empresas cubanas de este tipo.

Encaminado hacia este objetivo, la UCI está organizada por centros de desarrollos encargados de gestionar los proyectos de producción de *software*, dando paso al Centro de Tecnologías de Gestión de Datos (DATEC) que está compuesto por varias líneas de investigación dentro de las que se encuentra PostgreSQL, la cual tiene como meta fundamental, contribuir a la soberanía tecnológica potenciando tecnologías de bases de datos libres.

[1]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.

En esta línea en el año 2011 se presentó el trabajo de diploma “Exploración y diseño de la herramienta de administración de bases de datos para PostgreSQL HABD” de la autora Ivette Rosa Teodosio Acosta. Este trabajo propone el desarrollo de una nueva herramienta de administración a partir de investigaciones realizadas sobre los inconvenientes encontrados en otras herramientas para la administración de bases de datos. La característica más importante que incorpora esta herramienta es que su arquitectura está basada en plugins, elemento a destacar, ya que de esta forma permitiría a los desarrolladores la incorporación de nuevas funcionalidades. Se ha avanzado en la implementación de un *front-end*, el cual no permite diseñar y ejecutar instrucciones de manera sencilla a las bases de datos.

Hasta la actualidad, un administrador de bases de datos que interactúe con HABD con el objetivo de crear consultas y ejecutarlas, se le imposibilita realizar esta acción debido a que carece de funcionalidades que permitan realizar este tipo de actividad, siendo una característica fundamental de las herramientas de administración de bases de datos. Por la situación planteada con anterioridad surge la siguiente interrogante:

¿Cómo contribuir al diseño y ejecución de consultas de las bases de datos PostgreSQL en la herramienta de administración HABD.?

La investigación tiene como **objeto de estudio** procesos de edición de consultas SQL **enmarcado en el campo de acción** procesos de edición de consultas SQL en PostgreSQL.

Se plantea como **objetivo general** de la investigación: desarrollar un Plugin editor de consultas para la herramienta de administración de bases de datos HABD.

En correspondencia con el objetivo general, se plantean como **objetivos específicos**:

- Caracterizar las técnicas y herramientas existentes para editar consultas en bases de datos.
- Realizar el análisis y diseño del editor de consultas para HABD.
- Realizar la implementación y validación del editor de consultas para HABD.

Para lograr el cumplimiento de estos objetivos se realizarán las siguientes **tareas investigativas:**

- Caracterización de las diferentes herramientas de administración de bases de datos existentes que contienen un editor de consultas.
- Caracterización de la metodología, herramientas y tecnologías a utilizar en el desarrollo del Plugin editor de consultas.
- Identificación de las funcionalidades del editor de consultas para HABD.

- Diseño de la solución a partir de las funcionalidades identificadas.
- Implementación de las funcionalidades identificadas.
- Integración del editor de consultas a HABD.
- Diseño de los casos de pruebas para la realización de pruebas al editor de consultas de HABD.
- Aplicación de los casos de pruebas para validar que el editor de consulta responde a las necesidades del usuario final.

Con la realización exitosa de las tareas planteadas se espera como **posible resultado**: Plugin editor de consultas para la herramienta de administración de bases de datos HABD.

El presente trabajo se ha estructurado de la siguiente manera:

Capítulo 1: Fundamento Teórica.

El presente capítulo comprende el marco teórico del trabajo, así como un análisis de distintos conceptos, metodología, herramientas y tecnologías a usar, basadas en la arquitectura definida en el trabajo de diploma *"Exploración y diseño de la herramienta de administración de bases de datos para PostgreSQL HABD",* proporcionando un entendimiento claro y preciso para el desarrollo de la investigación. Se realiza un profundo análisis de los editores de consultas en las herramientas de administración de bases de datos y en el sistema Generador Dinámico de Reportes (GDR).

Capítulo 2: Análisis y Diseño.

El contenido que se aborda en este capítulo está relacionado con el diseño del sistema. El mismo incluye el modelo de dominio del sistema, los requisitos, tanto funcionales como no funcionales a tener en cuenta en su desarrollo, recoge las tarjetas CRC (clase, responsabilidad, colaboración) y una breve descripción de las Historias de Usuario, estilo arquitectónico y patrones de diseño seleccionados.

Capítulo 3: Implementación y Prueba.

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 fue sometida la aplicación.

CAPÍTULO 1

FUNDAMENTO TEÓRICO

Introducción

El presente capítulo comprende el marco teórico del trabajo, así como un análisis de distintos conceptos, metodología, herramientas y tecnologías a usar, basadas en la arquitectura definida en el trabajo de diploma *"Exploración y diseño de la herramienta de administración de bases de datos para PostgreSQL HABD",* proporcionando un entendimiento claro y preciso para el desarrollo de la investigación. Se realiza un profundo análisis de los editores de consultas en las herramientas de administración de bases de datos y en el sistema Generador Dinámico de Reportes (GDR).

1.1 Sistemas gestores de bases de datos

Los Sistemas Gestor de Bases de Datos (SGBD) son: *"software o conjunto de programas que permiten crear y mantener una base de datos. El SGBD 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."* (1)

Los sistemas de Gestión de Bases de Datos son aplicaciones que permiten a los usuarios definir, crear y mantener las bases de datos. Proporcionan un acceso controlado a la misma de manera simultánea y garantizan que no sucedan inconvenientes con la integridad de la información. La utilización de estos sistemas es una tarea compleja ya que contienen programas que proveen grandes soluciones y requieren de gran conocimiento en el tema de administración de bases de datos.

PostgreSQL 9.1

"PostgreSQL es un potente sistema de gestión de bases de datos objeto-relacional, multiusuario, centralizado y de propósito general, que está siendo desarrollado desde 1977 y está liberado bajo la licencia Berkeley Software Distribution (BSD). Es ampliamente considerado como el sistema gestor de bases de datos de código abierto más avanzado del mundo." (2)

PostgreSQL es el gestor escogido como base para adaptarlo a las necesidades de las empresas cubanas y conformar el gestor PostgreSQL Empresarial Cubano. Permite asegurar la integridad y

seguridad de los datos. Es altamente adaptable a las necesidades del cliente, puesto que posee buenas interfaces de instalación y administración. Distribuido bajo licencia BSD (por sus siglas en inglés *Berkeley Software Distribution*). Presenta documentación pública y libre que se encuentra muy bien organizada, de esta forma se mantiene en un progresivo avance.

1.2 Herramientas de administración de bases de datos

Con el surgimiento de las herramientas de administración de bases de datos disminuyeron en un gran porciento las dificultades presentadas a los administradores en cuanto al manejo de la información, puesto que estas cuentan con interfaces gráficas haciendo más sencillo el trabajo.

En la actualidad existen varias herramientas algunas propietarias y otras libres, las primeras producidas por compañías como Microsoft y Oracle ambas norteamericanas, las segundas realizadas por grupos de Desarrollo Global de PostgreSQL (por sus siglas en inglés PGDG). Debido a la política hostil que ha presentado durante años los Estados Unidos sobre Cuba, el país ha tomado como estrategia la utilización de herramientas de código abierto, puesto que se encuentra inmerso en un proceso de migración a software libre con el propósito de alcanzar una independencia tecnológica potenciando el uso y explotación de dichas herramientas.

A continuación se presenta un estudio realizado sobre las características fundamentales que poseen algunas herramientas de administración de bases de datos y el Generador Dinámico de Reportes (GDR), para así poder seleccionar las herramientas a utilizar en una investigación más profunda sobre las peculiaridades básicas que debe cumplir un editor de consultas, posibilitando que estas particularidades sean la base y guía fundamental para la recopilación de las características esenciales y comunes que van a caracterizar el plugin a desarrollar. Para este estudio se seleccionaron cuatro herramientas existentes que utilizan el gestor PostgreSQL, dos libres PgAdmin y PgAccess y dos propietarias Navicat for PostgreSQL y EMS SQL Manager for PostgreSQL, además se investigó sobre un sistema Generador Dinámico de Reportes (GDR) el cual fue desarrollado en la línea de soluciones integrales perteneciente al centro DATEC de la facultad 6 de esta universidad, donde dicho sistema contiene un editor gráfico de consultas.

PgAdmin

PgAdmin es una aplicación gráfica para trabajar con el diseño y administración total de bases de datos PostgreSQL. Está diseñada para ejecutarse en sistemas operativos como GNU/Linux y Windows 2000, XP o 2003 con licencia *Open Source* y disponible en más de 30 idiomas. Es capaz de gestionar

diferentes versiones de PostgreSQL, incluye un constructor de consultas, un editor de SQL y un servidor del lado del editor de código.

PgAccess

PgAccess es una herramienta de software libre. Proporciona la creación, edición y navegación de tablas, vistas y funciones. No brinda la posibilidad de crear y diseñar bases de datos de forma visual. Provee la creación de consultas gráficamente. Facilita la ejecución del comando utilizado para limpiar y analizar una base de datos PostgreSQL. El ambiente visual que posee la herramienta es poco amigable y pobre, factor importante que decepciona a las personas que interactúan con PgAccess.

Navicat Premium

Navicat Premium proporciona la conexión a varios gestores de bases de datos como MySQL, SQL Server, SQLite, Oracle y bases de datos PostgreSQL simultáneamente. Está disponible para tres plataformas: Microsoft Windows, Mac OS X y Linux. Provee algunas utilidades como importación y exportación de archivos, herramienta de modelado de datos y creación de informes, monitorización de servidores y copias de seguridad. Soporta las particularidades de otros gestores de bases de datos como PostgreSQL, SQL Server, SQLite, Oracle y MySQL, incluyendo Trigger, eventos, funciones y procedimientos almacenados. Permite la creación de tablas, ejecución de consultas, constructor gráfico de consultas. Es fácil de aprender y utilizar para los usuarios que son nuevos debido a la cómoda interfaz que posee.

EMS SQL manager for PostgreSQL

EMS SQL Manager for PostgreSQL es una herramienta de alto rendimiento para la administración de bases de datos PostgreSQL con una interfaz fácil de utilizar. Posee un soporte completo de PostgreSQL hasta la versión 9.1. Funciona con cualquier versión de PostgreSQL hasta la más reciente y es compatible con las últimas características de PostgreSQL. Permite la exportación e importación de los datos. Ofrece un diseñador visual para crear bases de datos PostgreSQL, *Visual Query Builder* para crear consultas complejas, avanzadas herramientas de manipulación de datos y explorador de bases de datos mejorado para facilitar la gestión de todos los objetos PostgreSQL.

Generador Dinámico de Reportes

El Generador Dinámico de Reportes es una aplicación de suma importancia para la gestión de la información de cualquier empresa o institución, el cual contribuye a la toma de decisiones, es un componente indispensable de cualquier sistema de información o software de gestión puesto que se puede integrar con otros sistemas. Se encuentra actualmente instalado en más de 20 proyectos de la

Universidad de las Ciencias Informáticas (UCI) y entidades nacionales como la Aduana General de la República de Cuba, Oficina Nacional de Estadísticas e Información (ONEI) y Contraloría General de la República de Cuba (CGR), además ha sido contratado por varios clientes de la República Bolivariana de Venezuela. Posee un editor gráfico en el cual se pueden diseñar las consultas SQL que se desean generar, posibilitando crear consultas de tipo Select, agregando el where, Join, Group by y Order by de la consulta. Además cuenta con un editor SQL con la deficiencia que no permite editar las consultas, mostrando solamente los resultados de la consulta generada visualmente.

Luego de un estudio realizado de las diferentes herramientas de administración de bases de datos y del Generador Dinámico de Reportes, para el análisis de las características principales de los editores de consultas, se opta por una herramienta libre, una propietaria y el sistema Generador de Reportes desarrollado en la facultad. Se elige dentro de las libres estudiadas a PgAdmin, por ser la herramienta más usada para el trabajo con PostgreSQL, además se puede obtener para poder instalarla, probarla y así lograr recopilar con mayor exactitud y facilidad las características propias de su editor de consulta. No se seleccionó PgAccess puesto que a pesar de ser libre no es muy efectiva, a causa de que no se le brinda soporte frecuentemente. Carece de opciones para realizar operaciones sobre bases de datos en PostgreSQL, un ejemplo de estas es que no brinda la opción de revisar línea a línea el código en busca de errores. De las herramientas propietarias se selecciona EMS Manager for PostgreSQL, puesto que es una herramienta muy potente para la creación de consultas complejas visualmente sin un profundo conocimiento de la sintaxis SQL nombrándose *Visual Query Builder*, siendo una base y guía fundamental para la construcción del plugin. No se escogió Navicat a causa de que presenta algunas desventajas en comparación con la herramienta EMS, ejemplo de esto, cuando se ejecuta la herramienta se conecta el clúster completo de bases de datos, carga todas las bases de datos del gestor pudiendo ocasionar cualquier error no deseado en otra base de datos, a la cual no se desea consultar, modificar, actualizar o eliminar. Además presenta algunas deficiencias en su constructor gráfico de consultas, pues no brinda la opción de escoger si la consulta que se desea diseñar es un *Select, Insert, Delete o Upd*ate y en la confección del *Join* no provee la posibilidad de seleccionar a *LIKE* dentro de los operadores. Estas peculiaridades de las herramientas propietarias se pudieron probar pues cuentan con un período de prueba de 30 días sin necesidad de requerir o pedir una licencia, proporcionando la posibilidad de poder emplear y evidenciar las características propias de sus editores de consultas. Se selecciona el Generador Dinámico de Reportes puesto que es un software que contiene un editor gráfico de consultas que puede servir de mucha ayuda para el desarrollo del plugin, utilizado por la mayoría de los centros productivos de la universidad, entidades nacionales así como internacionales.

1.2.1 Editores de consultas en herramientas de administración

Los editores de consultas son el corazón de las herramientas de administración de bases datos, son el elemento más importante. Sin ellos se hace imposible gestionar las bases de datos, no se podría crear, modificar, eliminar o hacer cualquier consulta a las tablas, así como editar procedimientos almacenados y scripts SQL existentes. A continuación se evidencian las características principales que debe tener un editor, realizándose una comparación con los editores de consultas de las herramientas de administración de bases de datos seleccionadas para este estudio y el Generador Dinámico de Reportes, para poder recopilar las características que va a cumplir el Plugin editor de consultas.

Características de editores de consultas	**Editor de la herramienta PgAdmin**	**Editor de la herramienta EMS SQL Manager for PostgreSQL**	**Editor del Generador Dinámico de Reportes(GDR)**
Ejecución de consultas SQL	Si	Si	Solo las generadas Gráficamente
Análisis de la sintaxis del código	No	No	No
Consulta con conexión actual	Si	Si	Si
Cancelar ejecución de la consulta	Si	Si	No
Mostrar plan de ejecución estimado *explain analyze*	Si	Si	No
Resultados a textos	No	No	No
Resultados a cuadrícula	Si	Si	Si
Resultados a archivos	Si	Si	No
Creación de vistas desde el editor de consulta	No	Si	No
Creación de Funciones	No	No	No

Panel de código	Si	Si	Si pero no editable
Numeración de línea	No	Si	Si
Ejecución de multiconsultas	Si	Si	No
Completamiento de código	Si(débil)	Si	No
Constructor gráfico de consultas	Si	Si	Si
Permitir escoger que tipo de consulta desea diseñar (Insert, Delete, Update o Select)	No(Solo Select)	Si	No(Solo Select)
Confección del Select • Selección de criterios del where • Selección de agrupamiento • Selección de ordenamiento • Selección para el Select	 Si Si Si No	 Si Si Si Si	 Si Si Si Si
• Confección del Insert (Permitir insertar valores)	No	Si	No
• Confección del Update (Selección de criterios del where)	No	Si	No
• Confección del Delete(Selección de criterios del where)	No	Si	No
• Confeccionar el Join Permitir elegir el tipo de Join a utilizar)	Si No	Si No todos los tipos de Join	Si No
• Eliminar Tabla	Si	Si	No

Registro histórico	Si	Si	No
Código plegable	No	Si	No
Resaltar palabras reservadas	Si(algunas)	Si	Si(débil)
Importar/ Exportar archivos	Si	Si	No
Agrandar, disminuir y restaurar código	No	No	No

Tabla 1. Características principales de los editores de consultas

A partir de las revisiones bibliográficas e investigaciones sobre los editores de consultas y partiendo de la experiencia del trabajo con las herramientas de administración y el Generador Dinámico de Reportes se proponen las siguientes características que incluirá el editor de consultas a desarrollar en la presente investigación.

Ejecución de consultas SQL: Se ejecuta la consulta que se encuentra en el panel de código.

Ejecución de multiconsultas: Se ejecuta el código seleccionado o si no se ha seleccionado ningún código ejecuta todo el código del Editor de consultas.

Completamiento de código: Permite el auto-completamiento del código de la consulta SQL.

Análisis de la sintaxis del código: Muestra los errores sintácticos, léxicos o semánticos de la consulta ejecutada devueltos por el gestor PostgreSQL.

Resultados a cuadrícula: Devuelve los resultados de la consulta como una o varias cuadrículas en la ventana Resultados.

Creación de vistas: Permite la creación de vistas aprovechando la consulta Select del editor.

Numeración de línea: Muestra los números de línea a la izquierda del texto en el editor.

Constructor gráfico de consultas: Permite la creación de consultas de tipo Select mediante la agregación visual de criterios, pudiéndose agregar criterios para el Where, criterios de ordenamiento y relacionar las tablas, siendo el Join de la consulta. Posibilita poder efectuar el Insert, Update y Delete, directamente a la tabla que se desea gestionar, sin necesidad de crear o implementar la consulta.

Agrandar, disminuir y restaurar código*:* Permite aumentar, disminuir y restaurar el código escrito en el panel de código.

Panel de código: Área donde se crean las consultas SQL

Registro histórico: Guarda todo el historial de resultados y errores dados una vez abierto el editor.

Resaltar palabras reservadas: Resalta las palabras reservadas y nombres de tablas.

Importar/ Exportar archivos: Importa y exporta los archivos con formato sql.

Consulta con conexión actual: Permite realizar la consulta con la conexión que está activa en ese momento.

Después del estudio realizado no se decidió utilizar ninguno de los editores de consultas de las herramientas analizadas ni del Generador Dinámico de Reportes, debido a que no cumplen con las características básicas que se necesitan y deben cumplir para poder ser integrado a HABD, donde la particularidad principal es que tienen que tener una arquitectura basada en plugin y ninguna de las herramientas expuestas disponen de dicha peculiaridad.

1.3 Metodologías 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." (3)

Las metodologías de desarrollo de *software* se dividen en dos grandes grupos, las pesadas o tradicionales y las ágiles. Las metodologías tradicionales se basan en la idea que el éxito del producto se puede lograr si se tiene todo correctamente documentado, mientras las ágiles defienden la idea de que el proceso de desarrollo del *software*, se centra en el *softwar*e como tal y no en la documentación alrededor de este, por lo que le da mayor importancia a la programación que a la documentación, aunque no la obvia por completo, sino que toma en cuenta sólo la documentación necesaria y de forma muy sencilla.

1.3.1 Metodología Extreme Programming (XP)

La Programación Extrema (**XP** por sus siglas en inglés *e**X**treme **P**rogramming*) es una metodología ágil nacida en el verano de 1996, de la mano de Kent Beck. Las metodologías ágiles surgen como una extensión a las metodologías tradicionales para mejorar el desarrollo de sistemas según el tipo de proyecto y empresa.

"Es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizadas para proyectos de corto plazo. 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." (4)

En la presente investigación se decide utilizar XP de las metodologías ágiles que existen debido a que es dentro de ellas la más utilizada, mejor documentada y teniendo en cuenta que:

- Se utiliza para la realización de proyectos a corto plazo: para la realización del plugin se definió un tiempo de desarrollo de 10 meses de trabajo.
- Fomenta el desarrollo de equipos pequeños: XP es una metodología pensada para equipos pequeños. Para solucionar el problema de la investigación planteado el equipo cuenta con solo 2 integrantes para todo el ciclo de desarrollo del software.
- Consiste en una programación rápida o extrema: esta metodología centra la atención del equipo en desarrollar las funcionalidades y no en una profunda documentación del proceso de desarrollo. Teniendo en cuenta que se debe desarrollar el plugin en un tiempo estimado de 10 meses y que el equipo lo conforman 2 integrantes, es fundamental la utilización de esta metodología.
- Propone la programación en pares: este elemento se ve estrechamente relacionado con las características propias del equipo de desarrollo, teniendo en cuenta que son solo 2 personas que implementarán en parejas ya que con esto se garantiza que muchos errores sean detectados conforme son introducidos en el código, por lo tanto la tasa de errores del producto final es más baja, el diseño mejor y el tamaño del código menor, los problemas de programación serán resueltos más rápido y se posibilitará la transferencia de conocimientos de programación entre los miembros del equipo.
- El cliente forma parte del equipo de desarrollo: para el desarrollo del plugin el cliente estará presente y disponible todo el tiempo para el equipo. Gran parte del éxito de la utilización de esta metodología se debe a que es el cliente quien conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada.

Agregado a lo anterior un elemento importante es que el equipo que desarrollará la solución cuenta con más de 1 año de experiencia en la utilización de esta metodología pues en la línea a la que pertenece es XP la utilizada en el desarrollo de los proyectos.

1.4 Herramientas y tecnologías a utilizar

Para el desarrollo del editor de consulta quien será parte de la herramienta de administración de bases de datos es necesario tener en cuenta, las herramientas y tecnologías con las que se va a realizar el plugin.

1.4.1 Lenguaje de Modelado

El lenguaje de modelado se utiliza en combinación con una metodología de desarrollo de *software*, proporciona 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 **M**odeling **L**anguage, p*or 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. " (7)

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. Estandariza a 9 tipos de diagramas. Se utiliza UML puesto que es un lenguaje consolidado en el mundo y también en la UCI. 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.

1.4.2 Herramientas CASE (Computer-Aided Software Engineering)

"Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software." (5)

Visual Paradigm for UML 8.0

"Suite de herramientas CASE profesionales que utiliza UML como lenguaje de modelado. Es un lenguaje estándar común para todo el equipo de desarrollo que facilita la comunicación entre todos sus integrantes." (6)

Visual Paradigm for UML (VP-UML) es una herramienta CASE de diseño UML que soporta el ciclo de vida completo del desarrollo de *software.* Es una herramienta gratuita, fácil de instalar, utilizar y actualizar. Genera la documentación del proyecto automáticamente en varios formatos como son Web o PDF. Exporta a 10 lenguajes de programación incluyendo php. Se utiliza Visual Paradigm puesto que se encuentra disponible para distribuciones del sistema Operativo Linux. Proporciona disponibilidad de integrarse a los principales entornos de desarrollos. Posibilita a los programadores ayuda garantizando la generación de código para lenguajes como C++, java, php, Ada y Python. Se ha actualizado rápidamente en correspondencia con el nuevo desarrollo de técnicas del lenguaje de modelado UML.

1.4.3 Lenguaje de programación

"Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones respectivamente, los cuales pueden ser utilizados para controlar el comportamiento de una máquina, especialmente una computadora." (8)

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.

Lenguaje C++

"C++ fue inventado en 1980 por Bjarne Stroustrup en los Laboratorios Bell en Murray Hill, New Jersey. Inicialmente recibió la denominación de C. En 1983 se le dio el nombre de C++....

C++ es una versión ampliada del lenguaje C. C++ incluye todo lo que forma parte de C y añade soporte para la programación orientada a objetos (POO para abreviar). Además C + + también contiene muchas mejoras y características que sencillamente lo convierten en un C mejor, independientemente de la programación orientada a objeto." (9)

La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitieran la manipulación de objetos, añadiéndole clases, sobrecarga de operadores, plantillas, mecanismo de excepciones, funciones, utilidades adicionales de librería, tipos de datos, referencias y sistemas de espacios de nombres. C++ brinda la posibilidad de redefinir los operadores (sobrecarga de operadores) y de poder crear nuevos tipos que se comporten como tipos fundamentales. Es un lenguaje híbrido, procedural (orientado a algoritmos) y dependiente del hardware, es uno de los

lenguajes más potentes pues permite a los desarrolladores programar a alto y a bajo nivel. Se utiliza este lenguaje puesto que los niveles de ejecución de velocidad son muchos más alto que otros lenguajes y el consumo de memoria es más pequeño porque es un lenguaje compilado. El motivo fundamental por el cual se utiliza este lenguaje para el desarrollo del plugin es porque las librerías de PostgreSQL son programadas en dicho lenguaje y la herramienta HABD a la cual se va a integrar el plugin está programada también en C++ lográndose de esta manera una integración apropiada.

1.4.4 Framework

Un *framework* contiene máquinas virtuales, compiladores, bibliotecas de administración de recursos en tiempo de ejecución y especificaciones de lenguajes. Está compuesto de componentes personalizables e intercambiables para el desarrollo de una aplicación. El empleo de esta tecnología permite al programador el desarrollo rápido de las aplicaciones y la reutilización de componentes de *software*.

Framework QT 4.7.3

El nombre de QT tiene su origen en el primer nombre de la compañía que lo creó: Quasar Technologies. Después cambiaría de nombre a Troll Tech y finalmente a Trolltech. QT es un *framework* multiplataforma, de código abierto, que se utiliza para el desarrollo de aplicaciones. Está escrito en C++ por lo que responde con una rápida velocidad, sin embargo es posible utilizarlo con otros lenguajes como C#, PHP, Python, y Ruby. Permite a los desarrolladores construir aplicaciones multiplataforma a partir de una misma base de código de manera rápida y sencilla, esto posibilita que sea menos complicado el trabajo diario que realiza el programador. Provee una API (Interfaz de Programación de Aplicaciones). Cuenta con métodos para acceder y realizar operaciones a las bases de datos mediante SQL de una forma más fácil y permite que los desarrolladores tengan una alta productividad.

Se decide utilizar QT 4.7.3 puesto que la API de la biblioteca que posee cuenta con métodos para acceder a bases de datos mediante SQL de una forma más fácil. Permite una mayor reutilización de código para agilizar el proceso de desarrollo de aplicaciones visuales lo que trae consigo que sea más factible el trabajo diario del programador. Presenta un buen diseño orientado a objetos y responde con una rápida velocidad ya que está escrito en el lenguaje C++. Posibilita disponibilidad del código fuente y presenta una arquitectura lista para plugin.

1.4.5 Entorno de desarrollo

Un entorno de desarrollo integrado (por sus siglas en inglés IDE), es un programa informático 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. 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 IDEs pueden ser aplicaciones por sí solos o pueden ser parte de aplicaciones existentes, además de dedicarse en exclusiva a un solo lenguaje de programación.

Qt Creator IDE 2.2.1

Qt Creator es un IDE multiplataforma que se ajusta a las necesidades de los desarrolladores, creado por *Trolltech* para el desarrollo de aplicaciones con las bibliotecas. Se centra en proporcionar características, que ayudan a los nuevos usuarios según las necesidades y aumenta la productividad de los desarrolladores con experiencia en Qt. Los sistemas operativos que soporta en forma oficial son: GNU/Linux, Linux, Windows XP y Mac OS. Qt Creator es un prometedor IDE de desarrollo para aplicaciones. Contiene el depurador visual (*visual debugger*) para C ++, lo que aumenta la capacidad de mostrar los datos con claridad, herramientas para la rápida navegación del código, resaltado de sintaxis, auto-completado de código, control estático de código y estilo a medida que se va escribiendo. Se utiliza como IDE QT Creator pues utiliza el lenguaje de programación orientado a objetos C++ con un avanzado editor de código. Fue diseñado para hacer que el desarrollo en C++ sea más rápido y fácil ya que posee un editor de código de C++. Presenta una ayuda para los desarrolladores muy bien redactada argumentada y escrita de manera clara y sencilla. Es el IDE más idóneo para trabajar con el framework QT, ya que son realizados por la misma compañía permitiendo una mejor integración.

1.4.6 Controlador de versiones

El control de versiones no es más que la revisión o edición de un producto, es el estado en que se encuentra en un momento dado de su desarrollo, es decir la gestión de los diversos cambios que se realizan sobre los elementos del producto.

Subversion

En este trabajo se decide utilizar Subversion como controlador de versiones puesto que es universal y ayuda virtualmente en todos los aspectos a dirigir un proyecto en cuanto a: comunicación entre los desarrolladores, manejo de los lanzamientos, administración de fallos y estabilidad entre el código.

Es un *software* libre bajo una licencia de tipo *Apache/BSD*. Todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en un instante determinado. Subversion puede acceder al repositorio a través de redes, lo que permite ser usado por

personas que se encuentren en distintas computadoras. Además el repositorio del proyecto utiliza como herramienta de control de versiones a Subversion.

Conclusiones del capítulo

En la investigación realizada no se encontró ningún plugin que cumpla con las características necesarias para dar respuesta al problema planteado, aunque resaltaron diferentes herramientas que contienen editores de consultas y que cumplen con las funcionalidades principales que debe tener el editor facilitando el desarrollo del trabajo, seleccionando PgAdmin y EMS Manager for PostgreSQL como base fundamental en la construcción del editor de consulta.

Del estudio de las herramientas y tecnologías a emplear en la solución se define XP como metodología de desarrollo de *software*, Visual Paradigm 8.0 como herramienta CASE y UML como lenguaje de modelado. El lenguaje de programación a utilizar será C++, con QT Creator 2.2.1 como IDE, como *framework* de desarrollo QT 4.7.3, como Sistema Gestor de Base de Datos PostgreSQL 9.1 y como controlador de versiones Subversion.

CAPÍTULO 2

ANÁLISIS Y DISEÑO

Introducción

El contenido que se aborda en este capítulo está relacionado con el diseño del sistema. El mismo incluye el modelo de dominio del sistema, los requisitos tanto funcionales como no funcionales a tener en cuenta en su desarrollo, recoge las tarjetas CRC (clase, responsabilidad, colaboración) y una breve descripción de las Historias de Usuario, estilo arquitectónico y patrones de diseño seleccionados.

2.1 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. Es el Mecanismo fundamental para comprender el dominio del problema y para establecer conceptos comunes." (10)

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

Para proporcionar un mejor entendimiento de los principales conceptos que se manejan en el negocio y se pueda realizar este proceso de comprensión de la manera más cómoda y entendible se decidió realizar el modelo de dominio puesto que existe un solapamiento de roles lo que significa que no hay una clara diferencia entre los trabajadores y los actores del negocio ya que el cliente está dentro del equipo de desarrollo, no se consigue precisar el proceso del negocio donde se pueda evidenciar claramente quiénes son las personas que desarrollan las actividades, quiénes son las que la inician y quiénes son los beneficiados con cada uno de estos procesos y porque la herramienta constituye una solución genérica que puede desplegarse en cualquier entidad que la necesite y no es para una en específico. Se realiza este modelo aunque la metodología XP no defina una técnica específica para definir el negocio teniendo en cuenta los conceptos existentes que de una forma u otra están relacionados con el sistema a desarrollar, estableciendo las posibles relaciones que existen entre ellos. En la figura 1 se representa el modelo de dominio establecido.

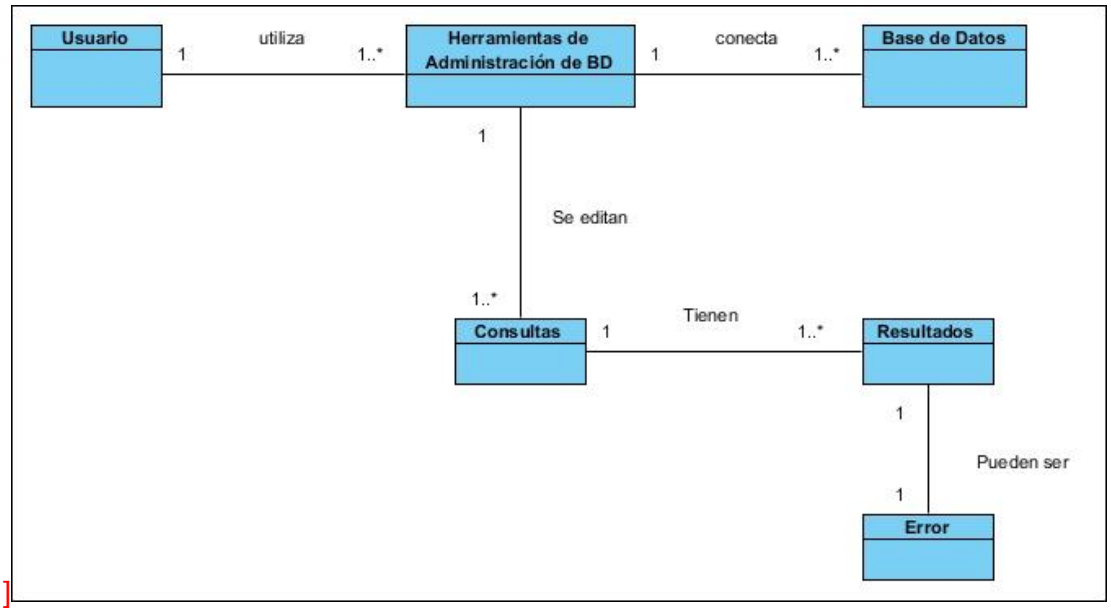


Figura 1. Modelo de Dominio.

Usuario: Persona que interactúa con las Herramientas**.**

Herramienta de Administración de BD: Herramientas existentes tomadas como punto de partida para modelar los conceptos principales y comunes de los editores de consultas.

Base de Datos: Las Bases de Datos con las cuales va a trabajar la herramienta.

Consultas: Son creadas por el usuario, para su posterior ejecución.

Resultados: Son mostrados por el editor una vez ejecutada la consulta.

Error: Son errores mostrados por el editor una vez ejecutada una consulta que tenga algún error sintáctico, léxico, semántico o que se quiera acceder a alguna tabla o atributo no existente en la Base de Datos.

2.2 Descripción del sistema propuesto

Se propone la implementación de un editor de consultas que permitirá a los usuarios, ejecutar las consultas descritas en el panel de código, a su vez podrán ir completándose, y escribir comentarios si así lo desean. Se les permitirá a los usuarios ver los resultados o los errores de las consultas ejecutadas que se guardaran en el historial una vez abierto el editor, además los usuarios podrán contar con un Editor Gráfico de Consultas (*Visual Query Builder*), donde podrán crear tablas, atributos, sus relaciones, gestionar los datos de una tabla y generar el código automáticamente de toda la representación visual sin tener conocimientos previos de sentencias SQL.

2.3 Historias de Usuario

"Uno de los artefactos más importantes que genera la metodología XP son las Historias de Usuario. Éstas tienen similar propósito que los casos de uso y son confeccionadas por el cliente. Las mismas expresan su punto de vista 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. " (11)

Las Historias de Usuarios (HU) describen las funcionalidades que debe realizar el Plugin editor de consultas SQL. Se especifica el grado de importancia (prioridad) que estas alcanzan en el desarrollo, el tiempo estimado de duración para la implementación de las mismas, se detalla la acción a realizar por cada Historia de Usuario dejando plasmado en las observaciones los inconvenientes que se pueden presentar para la no realización exitosa. Se especifica de manera visual con un esbozo como quedarán conformadas mediante el prototipo de interfaz. En sentido general proveen información al desarrollador que implementará dichas funcionalidades.

Para el presente trabajo de diploma se obtuvieron un total de 9 HU que serán implementadas. Las HU identificadas son: 1- Permitir auto-completamiento de código, 2- Ejecutar consultas SQL, 3- Mostrar resultado de la consulta ejecutada, 4- Manejar archivo, 5- Construir consultas gráficamente, 6- Mostrar historial de consultas, 7- Insertar comentarios, 8- Crear vistas y 9- Gestionar datos de forma Visual. Estas HU alcanzan un grado de importancia para el desarrollo de muy alta, alta y media, no se encuentra ninguna que la prioridad en el negocio sea baja puesto que es de suma importancia la realización exitosa de cada una de ellas.

A continuación se muestra el ejemplo de Historia de Usuario identificada Ejecutar consultas SQL, la cual será implementada en la primera iteración del sistema, puesto que presenta muy alta la prioridad del negocio.

Número: *2*	**Nombre de la Historia de Usuario:** Ejecutar consultas SQL
Cantidad de modificaciones a la Historia de Usuario: *0*	
Usuario: Yadiris E Fernández Martínez	**Iteración asignada:** 1

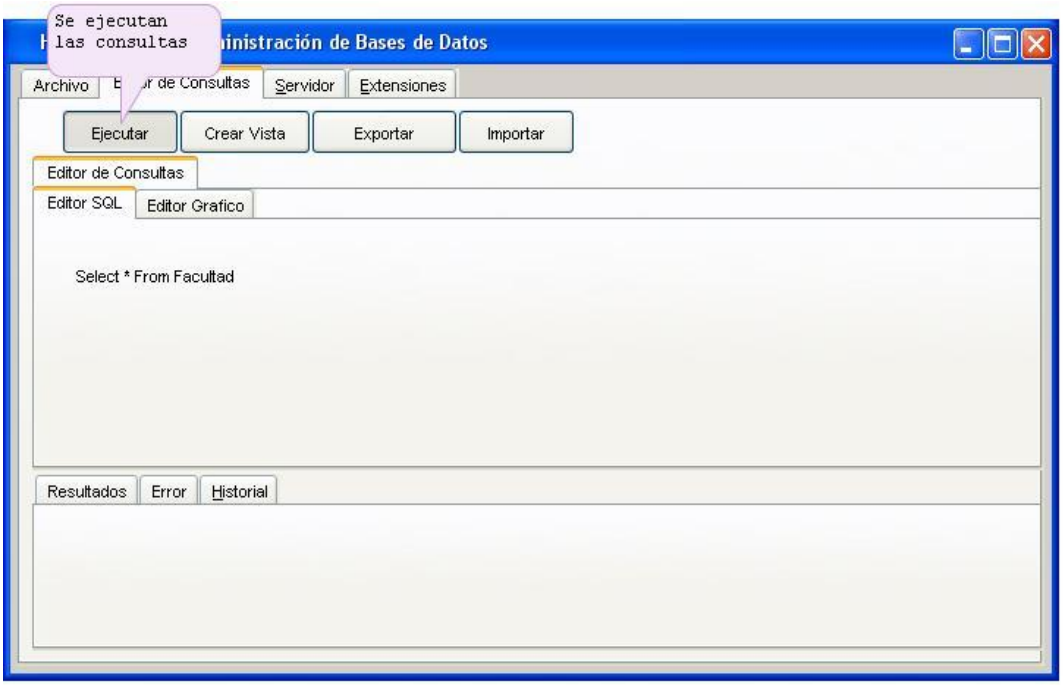
Prioridad en negocio: muy alta	**Puntos estimados:** *3*
Riesgo en desarrollo: muy alta	**Puntos reales:** 2.5
Descripción: Permite al usuario ejecutar el código seleccionado o, si no se ha seleccionado ningún código, ejecuta todo el código del Editor de consulta permitiendo la ejecución de multi-consultas.	
Observaciones:	
Prototipo de interfaces: 	

Tabla 2. Historia de Usuario "Ejecutar consultas SQL".

Para ver el resto remitirse a los documentos complementarios, Planilla de Historias de Usuario del Expediente de proyecto.

2.4 Lista de Reserva del Producto

La lista de reserva del producto agrupa los requisitos funcionales organizados por prioridad según la complejidad en el negocio y los requisitos no funcionales con una breve descripción de cada uno de ellos

"...los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir." (12)

"Los requerimientos no funcionales especifican propiedades del sistema, como restricciones del entorno o de la implementación, rendimiento, facilidad de mantenimiento, extensibilidad y fiabilidad." (13)

La etapa de definición de requerimientos es una tarea de suma importancia a la hora de desarrollar un sistema. Esta consistirá en generar una definición clara y precisa de los aspectos más relevantes del producto, cuanto más alto sea el grado de cumplimiento y aceptación por el cliente de los requerimientos, más alta será la calidad del sistema desarrollado. A continuación se muestra el listado de los requerimientos en la lista de reserva del producto.

Ítem*	Descripción	Estimación	Estimado por
Prioridad: Muy Alta			
1	Ejecutar consultas SQL.	3	analista
2	Mostrar resultados de la consulta ejecutada.	3	analista
3	Construir consultas gráficamente.	3	analista
Prioridad: Alta			
4	Permitir el auto-completamiento de código SQL.	3	analista
5	Crear vistas.	3	analista
6	Gestionar datos de forma visual.	1	analista
Prioridad: Media			
7	Exportar resultados de la consulta.	1.5	analista
8	Importar archivos.	1.5	analista
9	Mostrar historial de consultas.	1	analista
10	Insertar comentarios.	1	analista
Requisitos no funcionales			
11	**Apariencia o interfaz externa:**		analista

	El plugin tendrá una interfaz amigable, con una navegabilidad flexible y de fácil comprensión. Contará con un panel de código, barras de desplazamiento horizontal y vertical, numeración de línea y resaltado de código, las cuales se explicarán a continuación. **Panel de código** Área donde se escribe el texto del código. Contiene las características del generador de instrucciones disponibles para el lenguaje. En el panel de código se pueden establecer opciones que afecten al comportamiento del texto y que estén relacionadas con sangrías y comentarios. **Barras de desplazamiento horizontal y vertical** Permiten desplazarse por el panel de código en sentido horizontal y vertical, de forma que se pueda ver el código que se extiende más allá de los bordes visibles del panel de código. **Numeración de línea** Muestra los números de línea a la izquierda del panel de texto. **Resaltar código** Resalta con color azul las palabras reservadas del código SQL.		
12	**Facilidad de uso:** Para la utilización del editor de consultas SQL, es de vital importancia tener conocimientos de base de datos, y computación.		analista
13	**Soporte:** Se dispondrá de documentación técnica para un breve entrenamiento a los futuros usuarios que describa todas las funcionalidades del sistema.		analista

14	**Software:** Sistema Operativo: Multiplataforma. Librerías QT. Servidor de Bases de datos: SGBD PostgreSQL en cualquiera de sus versiones.		analista
15	**Hardware:** Se necesita 100 MB de memoria RAM mínimo, 100 MB de espacio libre en el disco duro.		analista

Tabla 3. Lista de Reserva del producto.

2.5 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. Este plan específica exactamente cuáles historias de usuario serán implementadas para cada iteración..." (14)

Para el desarrollo de la aplicación se definieron 3 iteraciones las cuales se detallan a continuación.

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
Iteración 1	En esta iteración se implementarán las Historias de Usuario que tengan la prioridad en el negocio MUY ALTA.	2,3,5	9 semanas
Iteración 2	En esta iteración se implementarán las Historias de Usuario que tengan la prioridad en el negocio ALTA.	1,8,9	7 semanas
Iteración 3	En esta iteración se implementarán las Historias de Usuario que tengan la prioridad en el negocio MEDIA.	4,6,7	5 semanas

Tabla 4. Plan de Iteraciones.

2.6 Modelo de Diseño

"Para el diseño de aplicaciones informáticas 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 (contenido, 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." (15)

2.6.1 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.

Para el presente trabajo se identificaron un total de 6 tarjetas CRC: 1- Consulta SQL, 2- Auto-completamiento de código, 3- Archivo, 4- Comentario, 5- Vista, 6- Datos.

A continuación se muestra el ejemplo de la tarjeta CRC Consulta SQL generada para el diseño del plugin la cual debe realizar varias funciones entre las que se encuentran: Ejecutar consultas SQL y Mostrar resultado de la consulta ejecutada. Con ella colaboran las clases Auto-completamiento de código, Datos y Comentario.

Tarjeta CRC	
Clase: Consulta SQL	
Responsabilidades	**Colaboraciones**
• Ejecutar consultas SQL. • Mostrar resultado de la consulta ejecutada. • Construir consultas gráficamente • Mostrar Historial de consulta	• Auto-completamiento de código. • Comentario • Datos

Tabla 5. Tarjeta CRC "Consulta SQL".

Para ver el resto remitirse a los documentos complementarios, Planilla Modelo de diseño del Expediente de proyecto.

2.6.2 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, y sus relaciones con los demás en el modelo." (16)

El Diagrama de Clases es el diagrama principal para el análisis y diseño. Representa las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones. Aunque la metodología XP no define un diagrama de clases del sistema, se presenta a continuación el diagrama realizado para un mejor entendimiento de la aplicación, puesto que este se puede utilizar siempre y cuando contribuya para el mejoramiento de la comunicación y comprensión del mismo ya que es una guía esencial para los desarrolladores a la hora de implementar. El diagrama cuenta con 21 clases, está compuesto por las relaciones de herencia, composición y agregación. A continuación se detallan algunas de las clases principales del diagrama que se muestra en la (Figura 2).

Principal - interface es la encargada de llevar el control sobre toda la aplicación, es la clase que construye la mayoría de los objetos más importantes del plugin, contiene el componente visual principal que se le mostrará al usuario. Dentro de ella se encuentran todos los métodos principales, que haciendo uso de las otras clases, se encargan de darle respuesta a la mayoría de las acciones que realiza el usuario sobre la aplicación.

Highlighter es la encargada de dar formato al texto introducido y mostrado en el panel de código del editor de consulta así como resaltar palabras reservadas, comentarios y otras funciones, logrando estas acciones mediante reglas que son definidas por el programador.

Text edit es la clase que su función fundamental es ejecutar el completamiento de código, independientemente que herede los componentes y atributos de la clase **texto** y por transitividad sea un QplainTextEdit. Esta clase redefine el método QKeyPressedEvent para esperar que el usuario oprima en el teclado las teclas Ctrl + Space y mostrar una lista con las posibles palabras a completar según lo que se está escribiendo en ese momento.

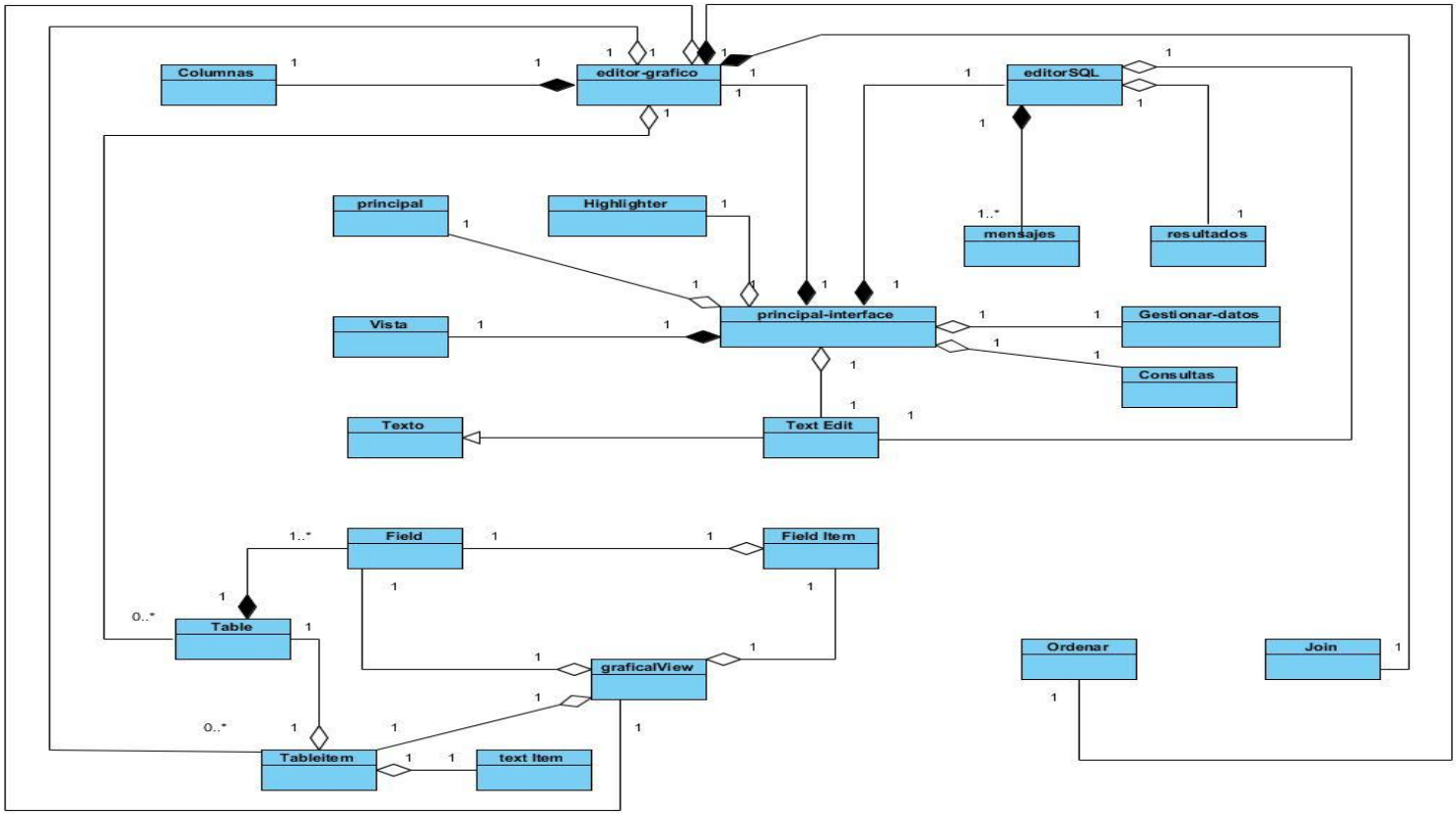


Figura 2. Diagrama de clases.

2.7 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, y los principios que orientan su diseño y evolución."* (17)

2.7.1 Estilos arquitectónicos. Patrones de Arquitectura

Involucrados en una arquitectura se encuentran los estilos arquitectónicos. Los diferentes estilos tienen sus fortalezas y debilidades, y ciertos estilos hacen que sea más fácil o más difícil trabajar con diferentes obstáculos.

"Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software." (18)

Existen numerosos estilos arquitectónicos entre los que se encuentra el Estilo de llamada y retorno y dentro de él, el patrón de arquitectura Modelo-Vista-Controlador (MVC). Este patrón de arquitectura de software separa la lógica de control, la interfaz de usuario y los datos de una aplicación en tres

componentes distintos. Se utiliza este patrón puesto que se tiene una capa de acceso a datos donde van a estar todas las clases que de una forma u otra piden información a una Base de Datos.

El paquete Modelo contiene todas las clases que tienen el código relacionado con el acceso a datos, para que este sea lo más genérico posible y se pueda reutilizar en otras situaciones y proyectos. Se incluirá consultas a las bases de datos y validaciones de entrada de datos. Se evidencia dentro del modelo 9 clases donde una de las más importante es la clase principal, la cual contiene varios métodos que envían a ejecutar consultas SQL al gestor obteniendo y devolviendo los resultados de las mismas para luego ser procesados.

El paquete Vista contiene todas las clases que poseen el código representando la parte que será visualizada en pantalla por el usuario. Se evidencia dentro de este paquete 11 clases donde la principal es editor- grafico, la misma es una interfaz visual que se le muestra al usuario para que el mismo interactúe con la aplicación.

El paquete Controlador contiene las clases que ejecutan la lógica de la aplicación, realizan llamadas al modelo para obtener los datos y se los pasa a la vista para que los muestre al usuario. Responden a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Se evidencia en este paquete la clase principal-interface donde esta clase obtiene los datos introducidos por el usuario en una de las vistas de la aplicación, los procesa pidiendo la información requerida a una de las clases del modelo para luego devolver y mostrar estos resultados al usuario.

Con el uso de este patrón se persigue mejorar la reusabilidad y que las modificaciones en las vistas impacten en menor medida en la lógica de negocio o de datos. A continuación se muestra el patrón MVC realizado.

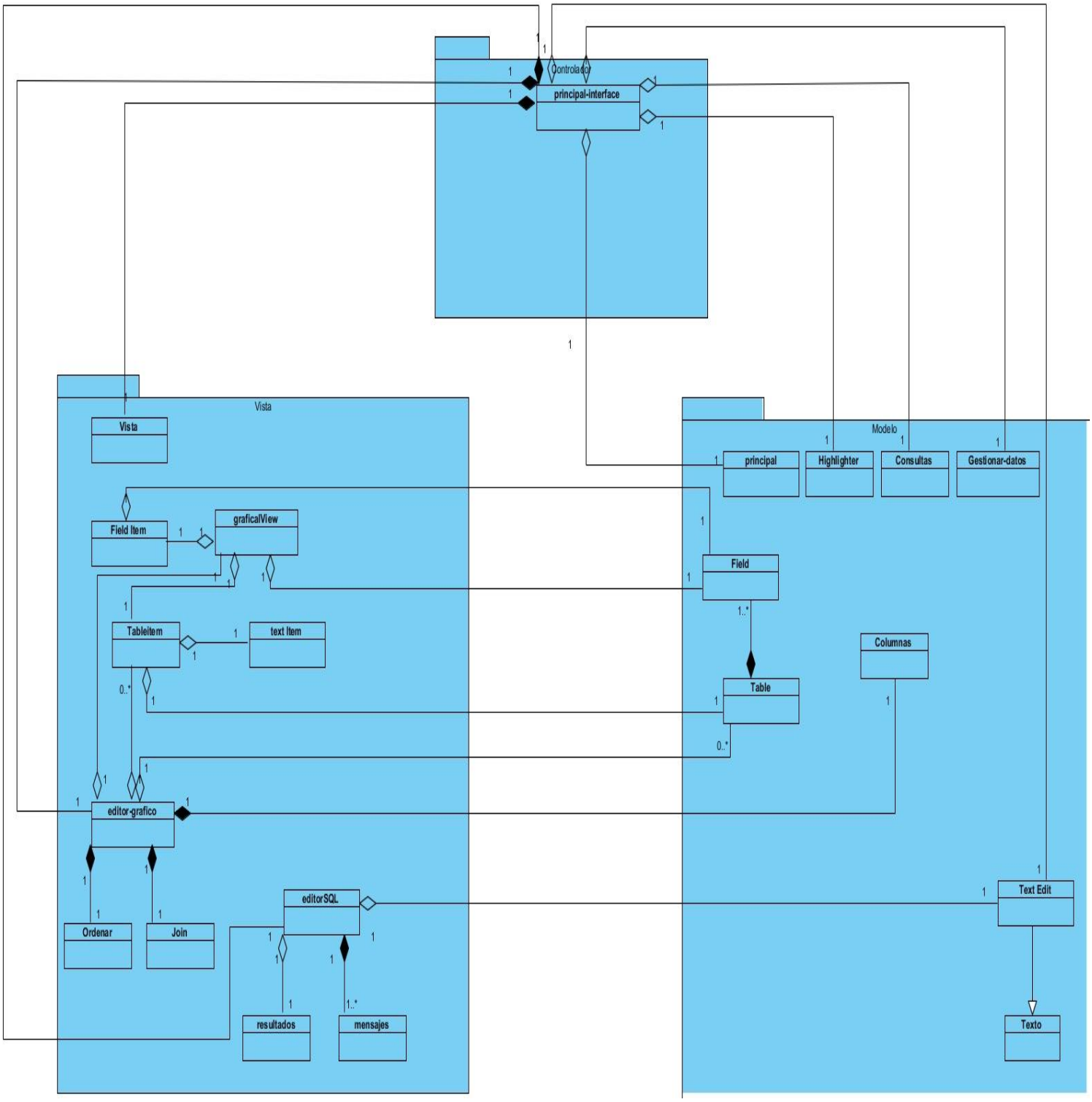


Figura 3. Patrón Modelo Vista Controlador.

2.8 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.

2.8.1 Patrones GRASP

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

Patrón Creador:

El patrón Creador guía la asignación de responsabilidades y ayuda a identificar quién debe ser el responsable de la creación de objetos. Se asigna la tarea a una clase de crear cuando, contiene, agrega, compone, almacena o usa otra clase. El propósito fundamental de este patrón es encontrar un creador que se deba conectar con el objeto producido en cualquier evento. El uso de este patrón se evidencia en las clases, editor SQL la cual crea objetos de las clases mensaje y resultado, para realizar sus responsabilidades. (Figura 4)

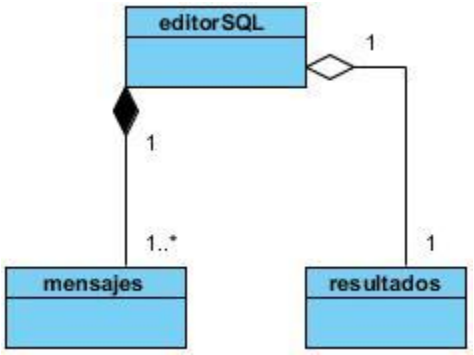


Figura 4. Fragmento del diagrama de clases donde se evidencia la utilización del patrón GRASP: Creador.

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 la clase principal-Interface la cual asigna responsabilidades a las clases editor-grafico y editorSQL. (Figura 5)

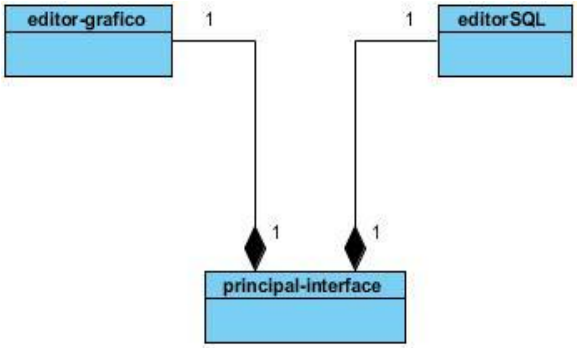


Figura 5. Fragmento del diagrama de clases donde se evidencia la utilización del patrón GRASP: Experto.

Patrón Controlador

El patrón controlador asigna la responsabilidad del manejo de los eventos del sistema a una clase. Este patrón se evidencia en la clase graficalView, la cual es la encargada de controlar y manejar todas las tareas de las clases Field Item y TableItem. (Figura 6)

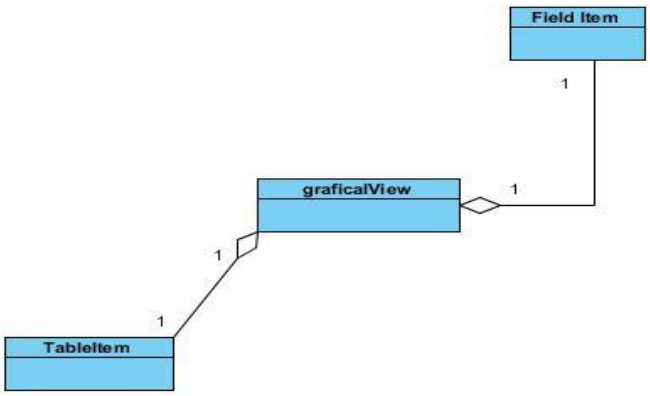


Figura 6. Fragmento del diagrama de clases donde se evidencia la utilización del patrón GRASP: Controlador.

Alta Cohesión

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. Este patrón se evidencia en las clases principal-Interface y consulta donde las mismas colaboran entre sí para lograr ejecutar la función importar consulta.

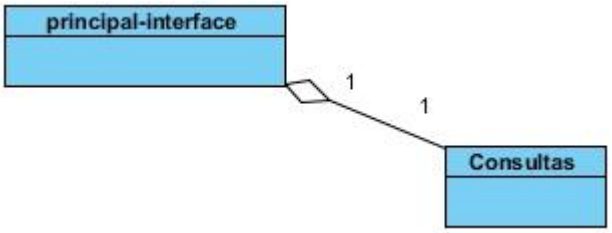


Figura 7. Fragmento del diagrama de clases donde se evidencia la utilización del patrón GRASP: Alta Cohesión.

Bajo Acoplamiento

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. Se puede evidenciar el uso de este patrón en la clase Highlighter y principal-interface puesto que Highlighter tiene el número mínimo de dependencias con otras clases.

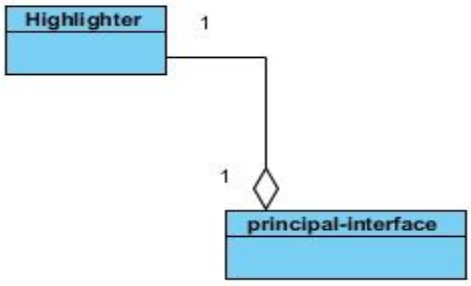


Figura 8. Fragmento del diagrama de clases donde se evidencia la utilización del patrón GRASP: Bajo Acoplamiento.

2.8.2 Patrones GOF

Los patrones GOF (*Gang of Four*, "Pandilla de los Cuatro") recopilan una serie de patrones de diseños, agrupados en tres categorías: de creación, de estructura y de comportamiento. De los diferentes patrones que ofrece GOF se ha tenido en cuenta para la modelación del plugin los siguientes:

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.

- **Facade (Fachada):** Proporciona una interfaz unificada y de alto nivel para un conjunto de interfaces de un subsistema lográndose que sea más fácil de usar.

El uso de este patrón se evidencia en la clase principal_interface la cual es la interfaz que unifica a todas las demás interfaces visuales.

Figura 9. Fragmento del diagrama de clases donde se evidencia la utilización del patrón GOF: Fachada.

Comportamiento: Contribuyen a definir la comunicación e iteración entre los objetos de un sistema.

Observer (Observador): Define una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.

El uso de este patrón se evidencia en la clase principal-interface y principal, donde al emitirse una señal en principal-interface toda la lista de objetos dependientes del objeto del cual es emitida la señal en principal-interface es actualizada en principal.

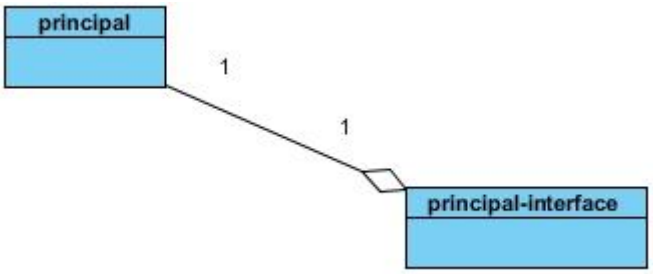


Figura 10. Fragmento del diagrama de clases donde se evidencia la utilización del patrón GOF: Observador.

Conclusiones del capítulo

Se identificaron 9 Historias de Usuario, en las cuales se agrupan un total de 10 requisitos funcionales, se planificó su implementación para 3 iteraciones comenzando por las HU de mayor prioridad del negocio. Se confeccionaron 6 tarjetas CRC. Se definieron las características no funcionales a tener en cuenta a la hora de utilizar la aplicación. Se describe la arquitectura a través de los patrones de diseño empleados, con el uso del patrón arquitectónico MVC.

CAPÍTULO 3

IMPLEMENTACIÓN Y PRUEBA

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 fue sometida la aplicación.

3.1 Implementación del sistema

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 en tareas de ingeniería las HU definidas en el capítulo dos las cuales guían la implementación, siendo así más fácil el desarrollo del sistema logrando una programación eficiente.

3.1.1 Tareas de Ingeniería

Para lograr desarrollar las HU definidas con la calidad requerida, se desglosan en Tareas de Ingeniería y se asignan a los programadores para que sean implementadas durante cada iteración, de esta forma el programador obtiene una mejor visión a la hora de implementar las HU.

A continuación se muestran las Tareas de Ingeniería número 3 y 4 vinculadas a la Historia de Usuario, Ejecutar consultas SQL, con un tiempo estimado para la realización de 3 semanas aproximadamente.

Tarea de Ingeniería	
Número Tarea: ***3***	**Número Historia de Usuario:** ***2***
Nombre Tarea: Capturar texto de la consulta	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1.5
Fecha Inicio: 18/02/2011	**Fecha Fin:** 31/10/2011

Programador Responsable: Yadiris E Fernández Martínez
Descripción: Captura el texto de la consulta que se va a ejecutar

Tabla 6. Tarea de Ingeniería "Capturar texto de la consulta".

	Tarea de Ingeniería
Número Tarea: ***4***	**Número Historia de Usuario:** ***2***
Nombre Tarea: Enviar el texto al gestor	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1.5
Fecha Inicio: 21/10/2011	**Fecha Fin:** 3/11/2011
Programador Responsable: Yadiris E Fernández Martínez	
Descripción: Envía el texto capturado al gestor PostgreSQL	

Tabla 7. Tarea de Ingeniería "Enviar el texto al gestor".

Para ver el resto remitirse a los documentos complementarios, Planilla Tarea de Ingeniería del Expediente de proyecto.

3.1.2 Estándares 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." (20)

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 fragmento del estándar definido para la implementación del plugin.

- **Identación**

La unidad de identado es de 4 espacios. El uso de la tabulación debe ser evitado pues 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. Realizar 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**

Es conveniente dejar información que pueda ser leída tiempo después por personas (posiblemente la misma persona que programó) que necesitan entender que fue lo que se hizo en el fragmento de código. Los comentarios deben ser escritos correctamente y claros. Generalmente deben usarse comentarios de una sola línea. Se debe reservar los comentarios de bloques para la documentación formal o para comentar porciones de código.

- **Declaración de variables**

Cada variable debe de ser declarada en una línea y comentada. También deben aparecer ordenadas alfabéticamente:

```
QMap<DependenceContainer*, bool> installedInstances; //plugins instalados
QList<DependenceContainer*> loadedInstancesLst; //plugins cargados
```

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

Ejemplo: loadedInstancesLst, installedInstances.

- **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. Debe haber un espacio entre el paréntesis derecho y la llave de comienzo del cuerpo de la función. Las sentencias del cuerpo deben estar en la línea siguiente. La llave que cierra debe estar alineada con la línea de declaración de la función. Los nombres de las funciones se rigen por las mismas características que las variables.

```
function ejemploPrincipal(c, d)
```

```
{
var e = c * d;
return inner(0, 1);
}
```

- **Identificadores**

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

- **Etiquetas**

Las sentencias etiquetadas son opcionales. Solo estas sentencias deben ser etiquetadas: while, do, for, foreach, switch.

- **Sentencia return**

Una sentencia return no debe utilizar paréntesis "()" alrededor del valor que se retorna. La expresión cuyo valor se retorna debe comenzar en la misma línea que la palabra reservada return, terminada con un punto y coma.

- **Espacios en blanco**

Las líneas en blanco facilitan la lectura determinando secciones de código lógicamente relacionados. Los espacios en blanco pueden ser (o no deben 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 la lista de parámetros. Esto 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 excepto cuando ese operador es una palabra como typeof.
- Cada punto y coma ";" en una sentencia de control debe ser seguido por un espacio.
- Debe dejarse un espacio luego de cada coma ",".

Para ver el resto remitirse a los documentos complementarios, Planilla Estándar de Código del Expediente de proyecto.

A continuación se presenta un ejemplo del estándar aplicado en la implementación del Plugin editor de consulta para la funcionalidad Ejecutar consultas SQL. El objetivo de este método es obtener el texto que introduce el usuario en el panel de código del editor y enviárselo al gestor, para posteriormente obtener el resultado, el cual puede ser el resultado en sí de la consulta ejecutada o un error en caso que la consulta presente algún tipo de deficiencia.

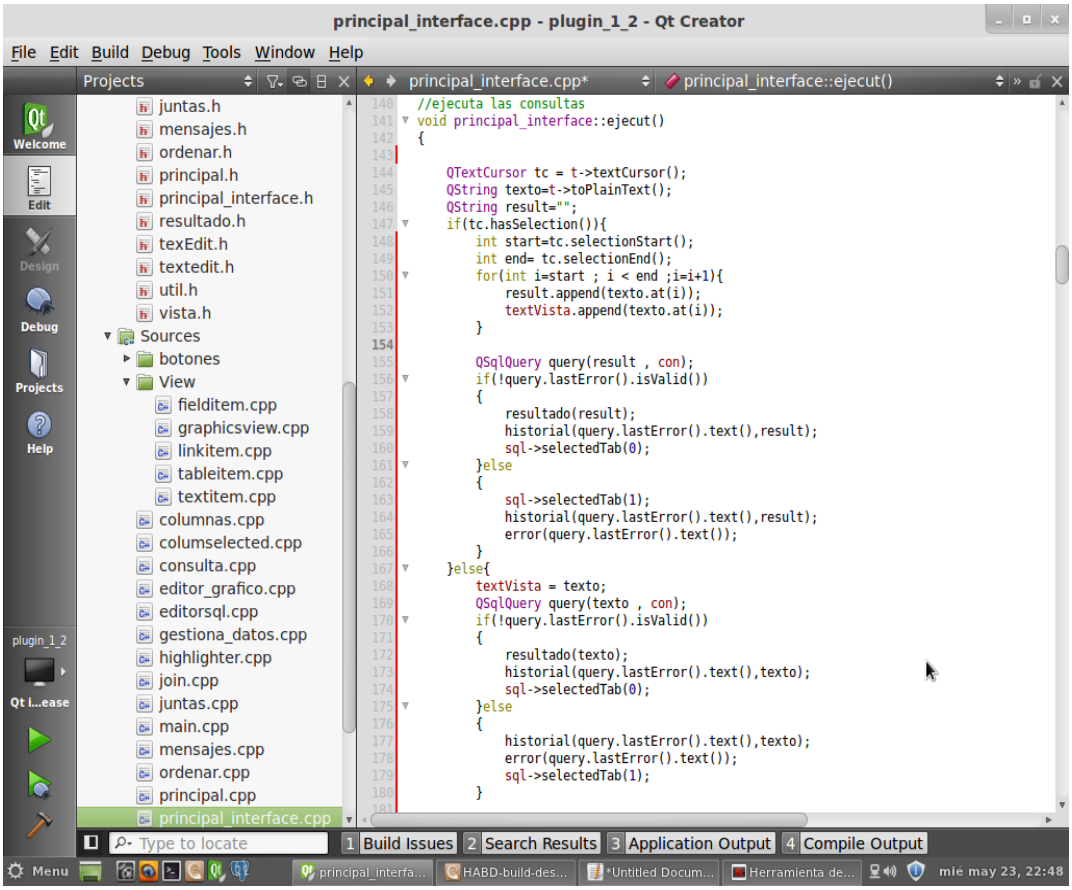


Figura 11. Ejemplo del estándar aplicado en el método ¨Ejecutar consultas SQL¨.

3.1.3 Interfaces de la aplicación

La aplicación desarrollada consta de dos interfaces principales una para la ejecución de consultas SQL y la otra para construir las consultas gráficamente.

Área de Ejecución de consultas

En esta área se realizan las consultas SQL que se van a enviar a la BD. La parte superior de la misma, brinda la opción de activar la pestaña Editor SQL, se muestra el panel de código que es donde se

implementan las consultas que se desean realizar, permitiendo el auto-completamiento de las palabras reservadas, los nombres de las tablas y vistas, posibilita la numeración de las líneas que se están implementando en el panel de código y los escritos de los comentarios que se desean efectuar. Además muestra en la parte inferior los resultados de las consultas ejecutadas, los errores y el historial de mensajes. Esta interfaz posibilita la creación de las vistas y exportar e importar las consultas SQL. En la Figura 12 se refleja los resultados obtenidos de la consulta ejecutada.

Figura 12. Ejemplo de la interfaz Editor SQL.

Área de construcción de consultas de manera visual

El plugin consta de una interfaz para la creación de consultas de manera visual, al activar la pestaña Editor Gráfico. En la parte izquierda se muestra el árbol de objetos, donde se escoge la tabla que se desea utilizar para la realización de la consulta y se desplaza hacia la parte superior derecha, permitiendo escoger los atributos que son necesarios para la ejecución de la consulta. Permite además al dar clic derecho sobre la tabla que se desea gestionar, eliminarla o editarla, posibilitando insertar datos a la tabla, actualizarlos y eliminarlos, evitando programar y diseñar las consultas de tipo, *Insert, Delete* y *Update*.

En la parte inferior de la interfaz se evidencia una serie de criterios de selección para confeccionar la consulta de tipo *Select*, posibilitando poder construir el *Where,* Order By y el Join de la consulta que se desea diseñar de manera visual. Para la confección del Where se activa la pestaña criterio,

permitiendo escoger los valores, los operadores y los conectores, estos valores se pueden ir adicionando o eliminando según la cantidad que haga falta. Se construye el Order By de la consulta al dar clic en la pestaña Ordenar, permitiendo seleccionar los criterios que aparecen en columnas disponibles en la parte inferior derecha, desplazándolo para la parte inferior izquierda y seleccionando el tipo de ordenamiento que se desea, Ascendente (ASC) o Descendente (DESC), siendo estos los criterios por los cuales se desea ordenar. Para construir el Join de la consulta se activa la pestaña Join, proporcionando la selección de las tablas a las cuales se les realizará la unión, el tipo de Join que se desea utilizar en la consulta, el atributo que es común para la tabla uno ya seleccionada y el atributo que es común para la segunda tabla ya elegida. Estos criterios de unión entre tablas se pueden ir adicionando o eliminando según las necesidades que hagan falta para el diseño de la consulta que se desea ejecutar.

Para generar el código automáticamente se activa la pestaña editor SQL donde se puede evidenciar toda la consulta diseñada. (Figura 13) y (Figura 14)

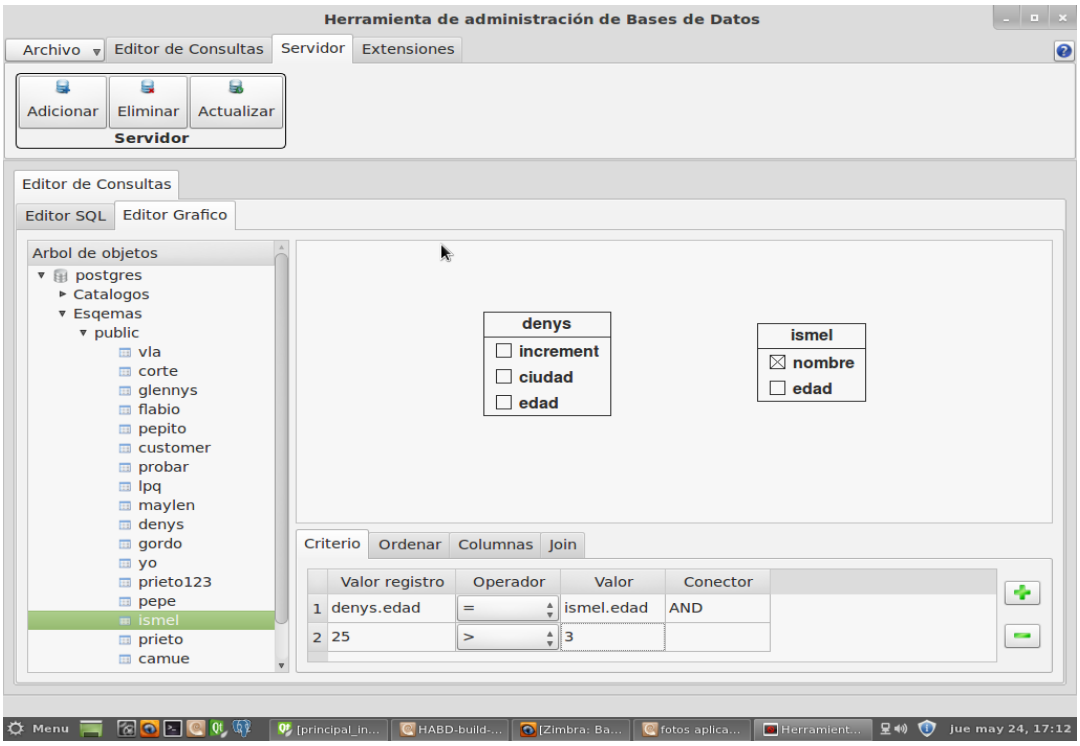


Figura 13. Ejemplo de la interfaz Editor Gráfico.

Figura 14. Ejemplo de la interfaz Editor Gráfico. Consulta generada.

3.2 Validación del sistema.

La validación y comprobación del funcionamiento del sistema antes que pueda ser liberado, permite aumentar su calidad, reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. Permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones.

Uno de los pilares fundamentales de la metodología XP, es proporcionar al cliente la posibilidad de concretar las funcionalidades de las HU y verificarlas en el proceso de pruebas.

3.2.1 Pruebas de software

"Las pruebas de software constituyen un pilar indispensable para evaluar y determinar la calidad de un software. Concretamente se puede definir pruebas de software como:

- *El proceso de ejecución de un programa con la intención de descubrir errores previos a la entrega al usuario final.*
- *Una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones específicas, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente."* (21)

"Las metodologías ágiles no consideran las pruebas como un conjunto de niveles que hay que ir superando para alcanzar la validación final del sistema que se está desarrollando. Las metodologías ágiles presentan distintos enfoques del proceso de desarrollo que vienen determinados por los tipos de pruebas que se realizan. " (22)

La metodología XP propone para validar las necesidades de los usuarios y dirigir la implementación las pruebas unitarias y las de aceptación. Las unitarias están encomendadas a verificar el código, son diseñadas por los programadores, garantizan que un determinado módulo cumpla con un comportamiento esperado antes de ser integrado al sistema. Se emplean cuando la implementación es complicada y la interfaz de un método no es clara.

Las pruebas de aceptación están destinadas a evaluar la funcionalidad requerida del software. Son definidas y diseñadas por el cliente, permitiendo validar todas las funcionalidades. Posibilitan que los programadores estén al tanto de lo que resta por realizar. Su principal objetivo es asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas.

Las pruebas de aceptación definen dos tipos de pruebas alfa y beta, para posibilitar el descubrimiento de errores que solo el usuario final podría detectar. La prueba alfa es aplicada por los usuarios finales en el área de trabajo del desarrollador en un entorno controlado, donde registra los errores y problemas detectados. La prueba Beta a diferencia de la alfa se aplica en el lugar de trabajo de los usuarios finales, en un entorno no controlado por el desarrollador. Es una aplicación en vivo del software. El usuario final es quien registra los problemas detectados y se lo informa al desarrollador, donde los ingenieros del software lo modifican y erradican los errores encontrados para preparar la liberación del mismo.

Las pruebas unitarias utilizan la técnica de caja blanca o de caja negra, mientras que las de aceptación las de caja negra. Por un lado, la técnica de caja blanca, se utiliza principalmente, desde una perspectiva interna en el desarrollo de software.

"En oposición a las pruebas de caja banca se encuentran las pruebas de caja negra (particiones de equivalencia, análisis de valores límites, pruebas transversales, etc.) tienen una visión externa del producto de software. Estas pruebas están centradas en analizar la funcionalidad. Por tanto los casos de pruebas se basan en las diferentes entradas que puede recibir el software y sus correspondientes valores de salida. Estas técnicas se pueden aplicar a cualquiera de los niveles de pruebas (unitarias, integración, aceptación) con diferentes niveles de abstracción en la definición de los casos de pruebas." (23)

Se decide utilizar para el desarrollo del plugin, la estrategia de prueba aceptación de tipo alfa, con la técnica de caja negra. Las pruebas de aceptación son más importantes que las unitarias, puesto que representan la satisfacción del cliente con el producto desarrollado, el final de una iteración y el comienzo de la siguiente. Es más efectivo que el cliente de una aceptación del producto que desea, a que el producto lo pruebe el programador. Estas pruebas conllevan al cliente a precisar lo que la aplicación debe hacer en determinadas circunstancias, por esto el cliente es la persona adecuada para diseñar las pruebas.

3.2.2 Diseño de casos de prueba. Método seleccionado

"Para desarrollar software de calidad y libre de errores, los casos de prueba son muy importantes. Un caso de prueba bien diseñado tiene gran posibilidad de llegar a resultados más fiables y eficientes…

El diseño de las pruebas especifica los detalles del método de prueba para una característica del software e identifica las pruebas correspondientes.

Un caso de prueba es un conjunto de acciones con resultados y salidas previstas basadas en los requisitos de especificación del sistema…" (24)

Se pueden obtener varios casos de prueba para determinar que una funcionalidad es completamente satisfactoria. Debe existir al menos un caso de prueba para cada HU a no ser que uno tenga requisitos secundarios. Los casos de prueba se derivan cuando se aplica alguna técnica, en este caso caja negra.

Los casos de prueba obtenidos al aplicar la técnica caja negra, pretenden demostrar que las funciones del software son operativas, que la entrada se acepta correctamente y se produce una salida correcta. Las pruebas de caja negra se aplican a la interfaz del software, son completamente indiferentes al comportamiento interno y la estructura del programa. Por tanto, están centradas en realizar pruebas del software a través de la funcionalidad.

"Las pruebas de caja negra intentan encontrar errores en las siguientes categorías.

- *Funciones incorrectas o ausentes.*
- *Errores de interfaz.*
- *Errores en estructuras de datos o en accesos a bases de datos externas.*
- *Errores de rendimiento.*
- *Errores de inicialización y de terminación. "* (25)

El método de prueba que se decide aplicar al utilizar la técnica caja negra es partición equivalente. *"La partición equivalente es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de pruebas.*

… se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de pruebas que deben desarrollarse." (26)

A continuación se presenta la Sección (SC) *"*Construir el Order By de la consulta*"*, del caso de prueba realizado a la Historia de Usuario *"*Construir consultas gráficamente*"*.

Escenario	Descripción	Variable 5	Variable 6	Variable 7	Respuesta del sistema	Flujo central
EC 1.4 Adicionar criterios para el order by	Permite adicionar criterios para la confección del order by de la consulta	I()	I()	I()	El sistema muestra mensaje de error: Debe Seleccionar la fila que desea adicionar.	*1-* Seleccionar el criterio por el cual se desea ordenar en columnas disponibles 2- Pasarlo para columnas 3- Escoger el orden 4- Dar clic en la pestaña Editor SQL para ver la consulta generada
		V(denis.edad)	V(denis.edad)	V(DESC)	Se adicionó correctamente el criterio del order by en la consulta generada en el editor sql.	
EC 1.5 Eliminar criterios para el order by	Permite eliminar criterios del order by	NA	I()	NA	El sistema muestra mensaje de error: Debe Seleccionar la fila que desea eliminar.	1-Seleccionar el criterio que se desea eliminar en columnas 2- Pasar el criterio seleccionado para columnas disponibles 3- Dar clic en la pestaña Editor SQL para ver los
		NA	V(denis.edad)	NA		

					Se eliminó correctamente el criterio del order by	cambios efectuados

Tabla 8. Caso de prueba "Construir consultas gráficamente". SC construir el Order By.

Para ver el resto remitirse a los documentos complementarios, Planilla de Casos de Pruebas del Expediente de proyecto.

La tabla que se muestra a continuación describe las variables que se encuentran asociadas al caso de prueba representado, SC "Construir el Order By de la consulta".

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
5	Columnas Disponibles	Item	No	En este campo el usuario selecciona los criterios de ordenamiento que desea adicionar
6	Columnas	Item	No	En este campo el usuario selecciona los criterios de ordenamiento que desea eliminar
7	Orden	Item	No	En este campo el usuario selecciona el tipo de orden si es ASC O DESC

Tabla 9. Caso de prueba "Construir consultas gráficamente¨. Variables de la Sección"Construir el Order By de la consulta".

Para ver el resto remitirse a los documentos complementarios, Planilla de Casos de Pruebas del Expediente de proyecto.

Las pruebas realizadas proporcionaron resultados satisfactorios en más del 80 porciento de los casos de pruebas efectuados. Los resultados que no fueron satisfactorios pasaron a ser no conformidades y se emitieron en el registro de dificultades encontradas. A continuación se muestra un ejemplo de las no conformidades más importantes encontradas, en la HU "Construir consultas gráficamente".

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del Equipo Desarroll
Aplicación	1	En la adición de criterios de la confección del Order By no elimina el criterio seleccionado en columnas disponibles.	Construir consultas gráficamente	Prueba	S	14/05/12 PD 15/05/12 RA	Se corrigió el error, encontrado, ya se elimina el criterio seleccionado para la confección del order by, de columnas disponibles.
Aplicación	2	No adiciona en columnas disponibles el criterio seleccionado en columnas que se desea eliminar, para la construcción de Order By.	Construir consultas gráficamente	Prueba	S	14/05/12 PD 15/05/12 RA	Se corrigió el erro encontrado, ya se adiciona el criterio seleccionado en columnas disponibles que se desea eliminar
Aplicación	3	En la adición de criterios de la confección del JOIN, al insertar un nuevo criterio existiendo campos vacíos en el anterior, no muestra el mensaje de error: Existen campos en blanco.	Construir consultas gráficamente	Prueba	NS	14/05/12 PD 16/05/12 RA	Se corrigió el error, encontrado, ya se muestra el mensaje de error al insertar un nuevo criterio para la confección del JOIN

Tabla 10. Ejemplo de no conformidades encontradas y resueltas en la HU "Construir consultas gráficamente".

Para ver el resto remitirse a los documentos complementarios, Planilla de No Conformidades del Expediente de proyecto.

Se evidencia a continuación los resultados de las no conformidades de las pruebas efectuadas en las tres iteraciones realizadas, donde en la primera iteración de descubrieron un total de 4 no conformidades, 1 error ortográfico, 2 errores de validación y 1 de interfaz, en la segunda iteración se encontraron 1 error ortográfico y 1 de interfaz, y para culminar en la tercera iteración no se halló ninguna no conformidad.

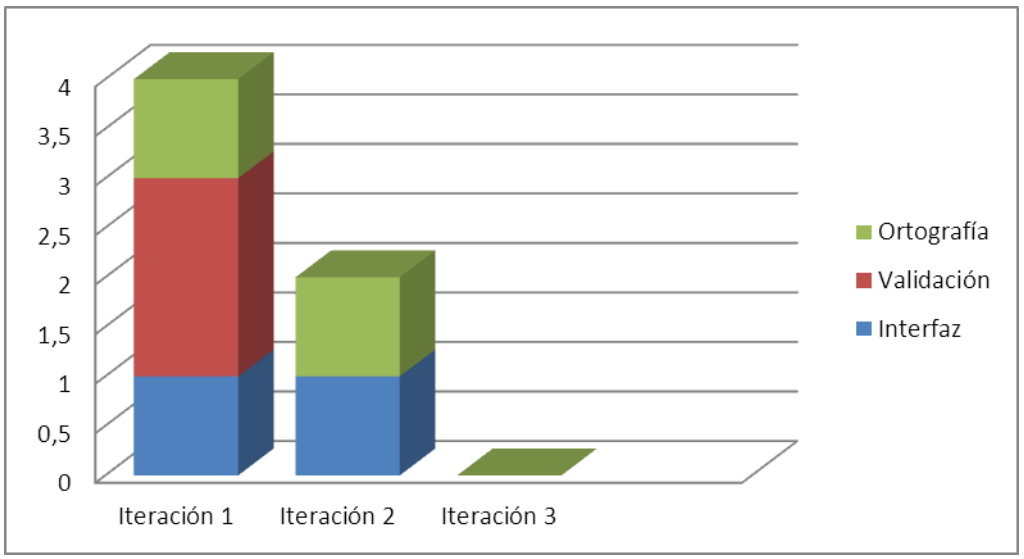


Figura 15. Representación de los tipos de no conformidades encontradas en las iteraciones.

Se muestra a continuación los resultados de las clasificaciones de las no conformidades encontradas en la realización de los casos de pruebas por iteraciones.

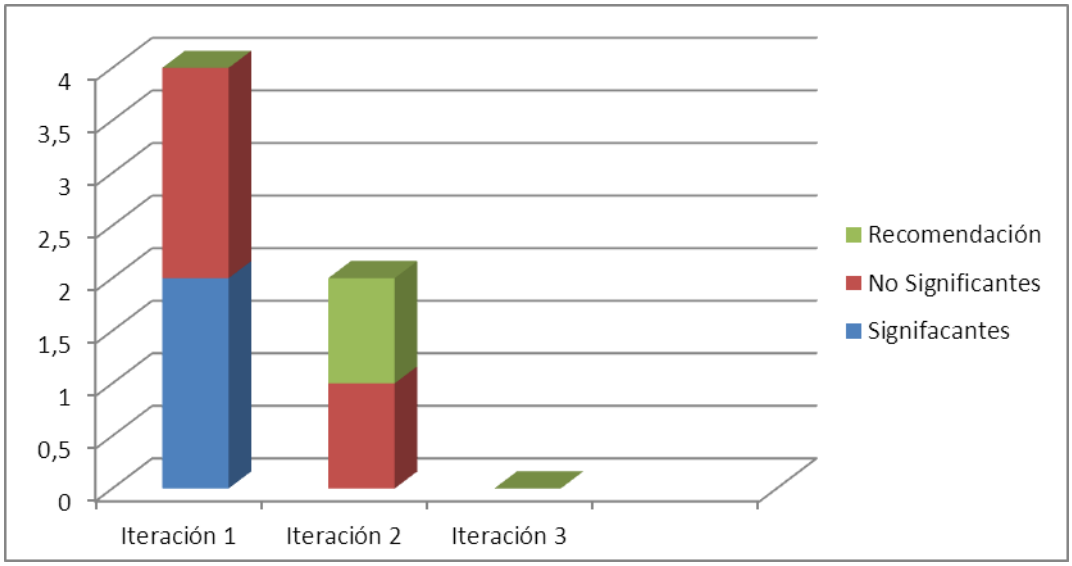


Figura 16. Representación de las clasificaciones de no conformidades encontradas en las iteraciones.

Para que conste la aceptación de los resultados de las pruebas, luego de ser identificadas y corregidas las no conformidades encontradas y por tanto la aceptación del Plugin editor de consultas SQL para la herramienta de administración de bases de datos HABD en su versión 1.0, se realizó la carta de aceptación del cliente al producto desarrollado. Ver (Anexo 7).

Conclusiones del capítulo

De las 9 HU definidas en la fase de análisis y diseño se logró implementar el 100% de las mismas haciendo uso de QT Creater con C++ como lenguaje. En el desarrollo de este capítulo se realizaron las Tareas de Ingeniería, se definió el estándar de codificación, se analizaron diferentes aspectos como el significado de las pruebas de *software*. Se realizaron las pruebas de aceptación a través del diseño de casos de pruebas, uno por cada Historia de Usuario, utilizando el método caja negra con la técnica, partición de equivalencia, arrojaron como resultado 4 no conformidades en la primera iteración y 2 en una segunda iteración para un total de 6 no conformidades, que fueron solucionadas demostrando que la herramienta desarrollada cumple con las características especificadas por el cliente.

CONCLUSIONES

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

- Al realizar el análisis se obtuvieron 9 HU de las que se identificaron 10 requisitos funcionales.
- Se obtuvieron durante el diseño del plugin 6 tarjetas CRC así como 22 clases que conformaron el diagrama de clases.
- Se implementó el Plugin editor de consultas permitiendo la creación y ejecución de sentencias SQL en la herramienta de administración de bases de datos para PostgreSQL HABD.
- Se validó la solución mediante la confección de casos de prueba, uno por cada Historia de Usuario demostrando que la aplicación cumple con las funcionalidades identificadas.

RECOMENDACIONES

Después de lograr los objetivos que se trazaron al principio de este trabajo, se plantea la siguiente recomendación:

- Continuar el desarrollo de la herramienta con el objetivo de incorporar otras funcionalidades, como la realización de analizador semántico a la hora de escribir las consultas en el editor y revertir el proceso del editor gráfico de consultas.

REFERENCIAS BIBLIOGRÁFICAS

1. **Cobo, Angel Yera.** Diseño y programación de bases de datos. Colección didáctica escolar. Madrid, España : Vision Libro. 978-84-9821-459-8.

2. **Alvarez, Aldo Cristiá, Valmaseda, Marcos Luis Ortiz y Ortiz, Yudisney Vázquez.** Personalización con funcionalidades de análisis de datos, monitoreo, administración, desarrollo o seguridad. Centro de Tecnologías de Gestión de Datos(DATEC),UCI. Ciudad de la Habana, Cuba : s.n., 2010.

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

4. **Fernández, Hector Arturo Florez.** Procesos de ingeniría de software. 2009.

5. **Alfaro, Felix Murillo.** El mejor soporte para el proceso de desarrollo de software. Colección cultura informática.

6. **Ortiz, Yudisney Vázquez y Reyes, Yunior Mesa.** Sitemas de Bases de Datos. Espacio de comunicación e intercambio para la comunidad técnica cubana de PostgeSQL. PostgreSQL, Universidad de las Ciencias Informáticas (UCI). Ciudad de la Habana, Cuba : s.n., 2011. pág. 4. 1994-1536.

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

8. **Aguilera, Carmen P.** Sistema de información para el registo y control de procesos de gestión de higiene ocupacional. Universidad de Oriente, núcleo de monagas.Comisión de trabajo de grado. Maturin, Monagas, Venezuela : s.n., 2011. pág. 91.

9. **Schildt, Herbert.** C++ Guía de autoenseñanza. Madrid : s.n. 84-481-3203-3.

10. **Gómez, Gloria Lucia Giraldo.** Ingeniería de software clase 5, Actores y sus roles. Modelo de Dominio. Escuela de sistemas, universidad nacional de Colombia, sede Merlin. Merlin, Colombia : s.n.

11. **Reyes, Yunior Mesa y Ortiz, Yudisney Vázquez.** Sitemas de Bases de Datos. Espacio de comunicación e intercambio para la comunidad técnica cubana de PostgeSQL. PostgreSQL. Ciudad de la Habana, Cuba : s.n., 2011. pág. 6. 1994-1536.

12. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** El proceso unificado del software. 2000. pág. 107, Cap 6, Epígrafe 6.3.

13. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** El proceso unificado del software. 2000. pág. 107 Cap 6, Epígrafe 6.2.

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

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

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

17. **Guerrero, Lugo Manuel Barbosa.** Arquitectura de software como eje temático de investigación. 2006.

18. **Reynoso, Carlos Billi.** introducción a la arquitectura de software. versión 1.0 . Universidad de Buenos Aires. Buenos Aires : s.n., 2004.

19. **Prieto, felix.** Programacion III. Sistemas tema 7. Patrones de Diseño. Universidad de Valladolid. 2005.

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

21. **Sánchez, Linet Lores y Roque, Diana Monné.** Aplicación de las pruebas de libración al sistema informático de genética médica. Universidad de las Ciencias informáticas(UCI). Ciudad de la Habana, Cuba : s.n., 209.

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

23. Revista Española de Innovación, Calidad e Ingeniería del Software(REICIS). **Yague, Agustin y Garbajosa, Juan.** [ed.] Luis Fernández Sanz y Juan José Cuadrado Gallego. pág 4, Barcelona, España : s.n., 2009, Vol. 5. 1885-4486.

24. Los casos de pruebas en la prueba del software.Revista digital LAMPSAKOS. **Cristegui, Jose Luis.** 3, 2010.

25. **Pressman, Roger S.** Ingeniería del software. Un enfoque práctico. 6ta edición cap 13 .

26. **Pressman, Roger S.** Ingeniería del Software. Un enfoque práctico. 6ta edición, Cap 14. pág. 19.

BIBLIOGRAFÍA

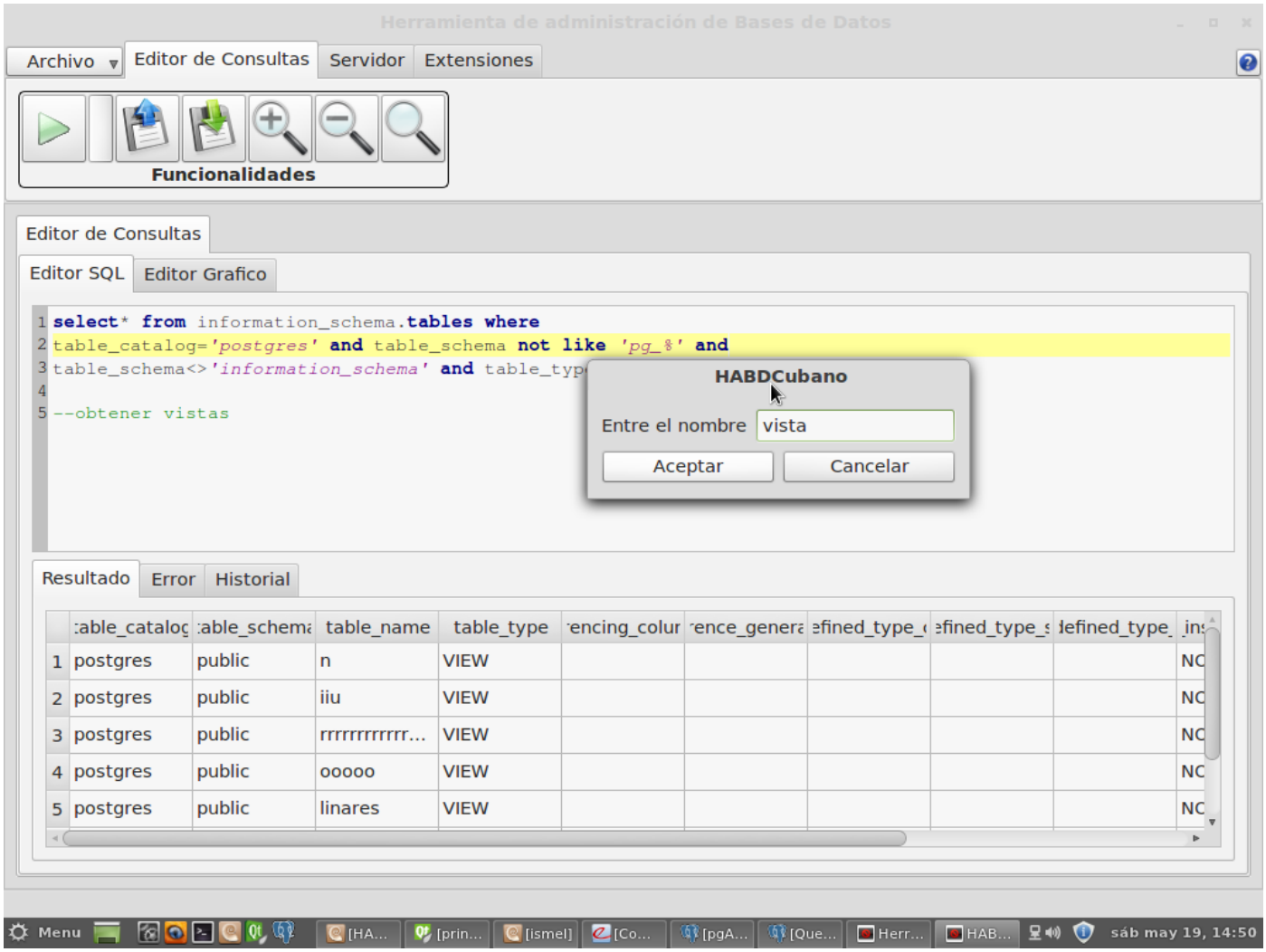
1. Actas de los talleres de las jornadas del software y bases de datos. . **Yague, Agustin y Garbajosa, Juan.** Madrid : s.n., 2009. Vols. Vol 3, Num4. Universidad Politécnica de Madrid(UPM).
2. **Aguilera.M, Carmen. P.** Sistema de información para el registo y control de procesos de gestión de higiene ocupacional. UNIVERSIDAD DE ORIENTE . MATURÍN / MONAGAS / VENEZUELA : s.n., 2011. pág. 260.
3. **Alvarez, Aldo Cristiá, Valmaseda, Marcos Luis Ortiz y Ortiz, Yudisney Vázquez.** Personalización con funcionalidades de análisis de datos, monitoreo, administración, desarrollo o seguridad. Centro de Tecnologías de Gestión de Datos(DATEC),UCI. Ciudad de la Habana, Cuba : s.n., 2010.
4. **Alfaro, Felix Murillo.** El mejor soporte para el proceso de desarrollo de software. Colección cultura informática.
5. **Beck, Kent, Fowler, Martin y Wesley, Addison.** Planning Extreme Programming. 2000. 0-201-71091-9.
6. **Beck, Kent.** Extreme Programing. 1999. 0201616416.
7. **Canós, José H., Letelier, Patricio y Penadés, Mª Carmen.** Métodologías Ágiles en el Desarrollo de Software. DSIC -Universidad Politécnica de Valencia.
8. **Cobo, Angel Yera.** Diseño y programación de bases de datos. Colección didáctica escolar. Madrid, España : Vision Libro. 978-84-9821-459-8.
9. **Collins-Sussman, Ben, Fitzpatrick, Brian W. y Pilato, C. Michael.** Version Control with Subversion. Stanford California USA. USA, California : s.n., 2006. 94305.
10. **Dapena, Martha D. Delgado.** Definición del modelo del negocio y del dominio utilizando.
11. **Fernández, Hector Arturo Florez.** Procesos de ingeniría de software. 2009.
12. **Gómez, Gloria Lucia Giraldo.** Ingeniería de software clase 5, Actores y sus roles. Modelo de Dominio. Escuela de sistemas, universidad nacional de Colombia, sede Merlín. Merlín, Colombia : s.n.
13. **Guadarrama, Addiel Rodríguez y Mazorra, Thaymí.** Análisis y diseño de una herramienta web para la gestión de la información. Universidad de la Ciencias Informáticas (UCI). 2010.
14. **Guerrero, Lugo Manuel Barbosa.** Arquitectura de software como eje temático de investigación. 2006.

15. **Hugo, Serrano Cuayahuitl Victor.** Pruebas de unidad, Estándares de codificación y generación automática de código. Facultad de Ciencias básicas, ingeniería y tecnología. Universidad Autónoma de Tlaxcala .
16. INGENIERÍA DE SOFTWARE—CALIDAD DEL PRODUCTO—PARTE 1: MODELO DE LA CALIDAD (ISO/IEC 9126-1:2001, IDT) Software engineering—Product quality—Part 1: Quality Model. 2005. 35.080.
17. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** El proceso unificado de software. Apéndice A1. 2000.
18. **Kaisler, Stephen H.** SOFTWARE PARADIGMS. George Washington University. United States of America : s.n., 2005. 0-471-48347-8.
19. **Küng, Stefan, Onken, Lübbe y Large, Simon.** TortoiseSVN. Un cliente de Subversion para Windows. Versión 1.4.1. 2006.
20. **Letelier, Patricio.** Metodologías Ágiles y XP. Departamento de Sistemas informáticos y Computación, Universidad Politécnica de Valencia. Valencia : s.n.
21. Los casos de pruebas en la prueba del software.Revista digital LAMPSAKOS. **Cristegui, Jose Luis.** 3, 2010.
22. **Mestras, Juan Pavón.** Estructura de las Aplicaciones Orientadas a Objetos. El patrón Modelo-Vista-Controlador (MVC). Dep. Ingeniería del Software e Inteligencia Artificial, Universidad Complutense Madrid, Facultad de Informática. Madrid, España : s.n., 2008-2009.
23. **Ortiz, Yudisney Vázquez y Reyes, Yunior Mesa.** Sitemas de Bases de Datos. Espacio de comunicación e intercambio para la comunidad técnica cubana de PostgeSQL. PostgreSQL, Universidad de las Ciencias Informáticas (UCI). Ciudad de la Habana, Cuba : s.n., 2011. pág. 4. 1994-1536.
24. **Pampillo, Yulenia Manso y Cuadra, Yuleidis Rodríguez.** Propuesta de modelo para probar la confiabilidad de los productos desarrollados con PostgreSQL en el Centro de Tecnología y Gestión de Datos (DATEC). Universidad de las Ciencias Informáticas. Cuidad de la Habana, Cuba : s.n., 2010.
25. **Peñalver, G., Meneses, A. y García, S.** SXP, METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE. Universidad de las Ciencias Informáticas. Ciudad de La Habana, Cuba : s.n., 2010. 1er Congreso Iberoaméricano de ingeniería de Proyectos Antofagasta -Chile.
26. **Pressman, Roger S.** Ingeniería de Software. Un enfoque práctico. 6ta edición, Cap 14. pág. 19.
27. **Prieto, felix.** Programación III. Sistemas tema 7. Patrones de Diseño. Universidad de Valladolid. 2005.

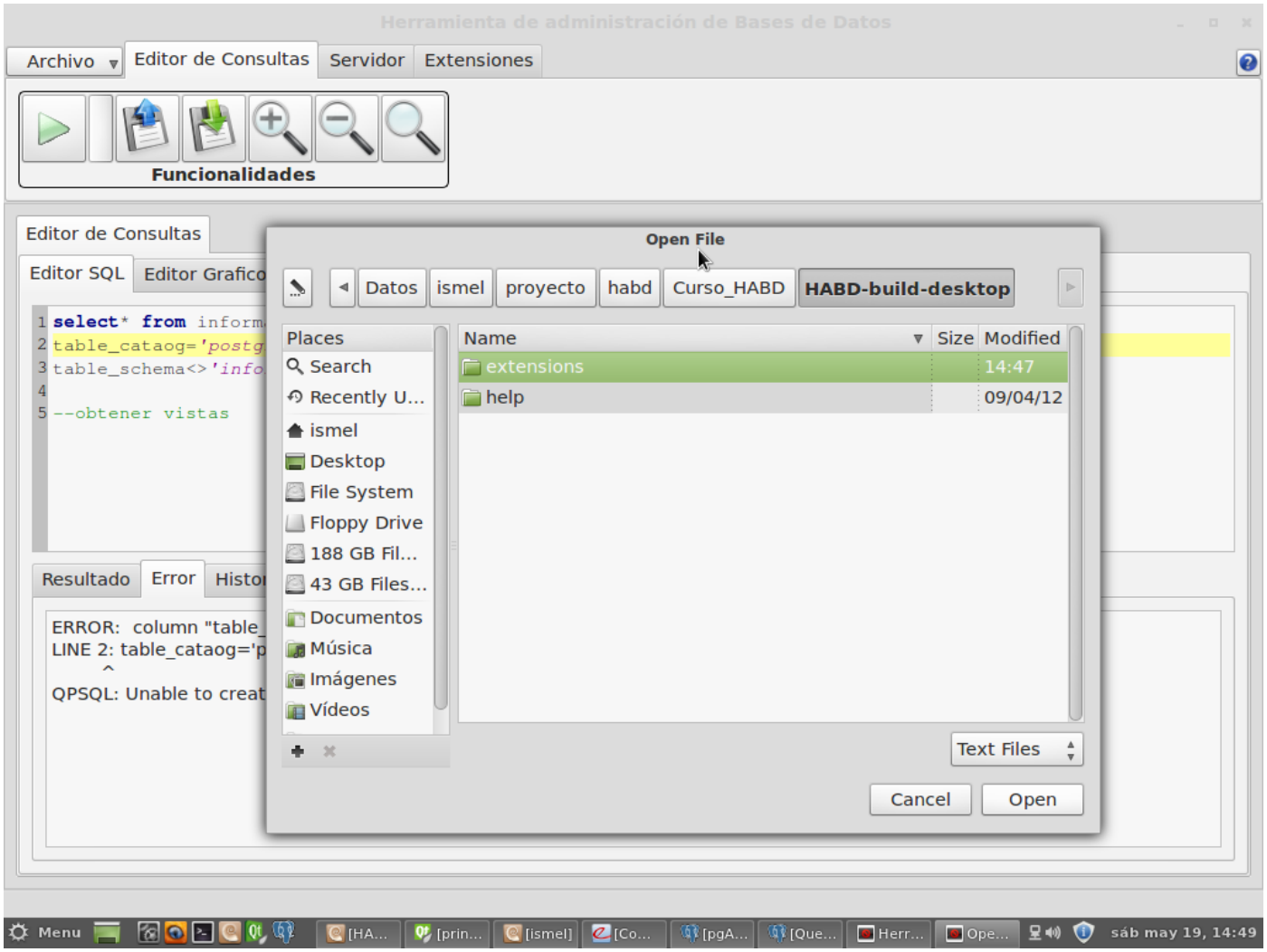
28. Revista Española de Innovación, Calidad e Ingeniería del Software(REICIS). **Yague, Agustin y Garbajosa, Juan.** [ed.] Luis Fernández Sanz y Juan José Cuadrado Gallego.pág 4, Barcelona, España : s.n., 2009, Vol. 5. 1885-4486.
29. **Reyes, Yunior Mesa y Ortiz, Yudisney Vázquez.** Sitemas de Bases de Datos. Espacio de comunicación e intercambio para la comunidad técnica cubana de PostgeSQL. PostgreSQL. Ciudad de la Habana, Cuba : s.n., 2011. pág. 7. 1994-1536.
30. **Reynoso, Carlos Billi.** introducción a la arquitectura de software. versión 1.0 . Universidad de Buenos Aires. Buenos Aires : s.n., 2004.
31. **Reynoso, Carlos.** Alternativas metodológicas. UNIVERSIDAD DE BUENOS AIRES.
32. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** El proceso unificado del software. 2000. pág. 107 Cap 6, Epígrafe 6.2.
33. **Sánchez, Linet Lores y Roque, Diana Monné.** Aplicación de las pruebas de libración al sistema informático de genética médica. Universidad de las Ciencias Informáticas(UCI). Ciudad de la Habana, Cuba : s.n., 209.
34. **Sande, Martín.** Programación en C++ con Qt bajo Entorno GNU/Linux. 2004. #281622.
35. **Schildt, Herbert.** C++ Guía de autoenseñanza. Madrid : s.n. 84-481-3203-3.
36. **Schildt's, Herb.** C++ Programming CookBook. New York, Chicago, San Francisco : s.n., 2008. 0-07-164385-0.
37. **Serradilla, Juan Luis.** Control de Versiones con Subversion y TortoiseSVN. Sección de Metodología, Normalización y Calidad del Software ATICA, Universidad de Murcia,Vicerrectorado de Investigación y Nuevas Tecnologías. Murcia : s.n., 2007.
38. **Suárez, Pablo y Fontela, Carlos.** Documentación y pruebas antes del paradigma de objetos. 2003.
39. **Thelin, Johan.** Foundations of QT Development. [ed.] Jason Gilmore y Kelly Winquist. [trad.] Dina Quan. United States of America,New York : Springer-Verlag, 2007. 978-1-59059-831-3.
40. **Wesley, Addison.** Testing Extreme Programing. 2002. 0-321-11355-1.
41. **Zamudio, Esmeralda Villegas y Méndez, Alejandra Virrueta.** Investigación documental. Metodologías de desarrollo de software. Instituto tecnológico superior de Apatzingán, Michoacan. Apatzingán Michoacan : s.n., 2010.

ANEXOS

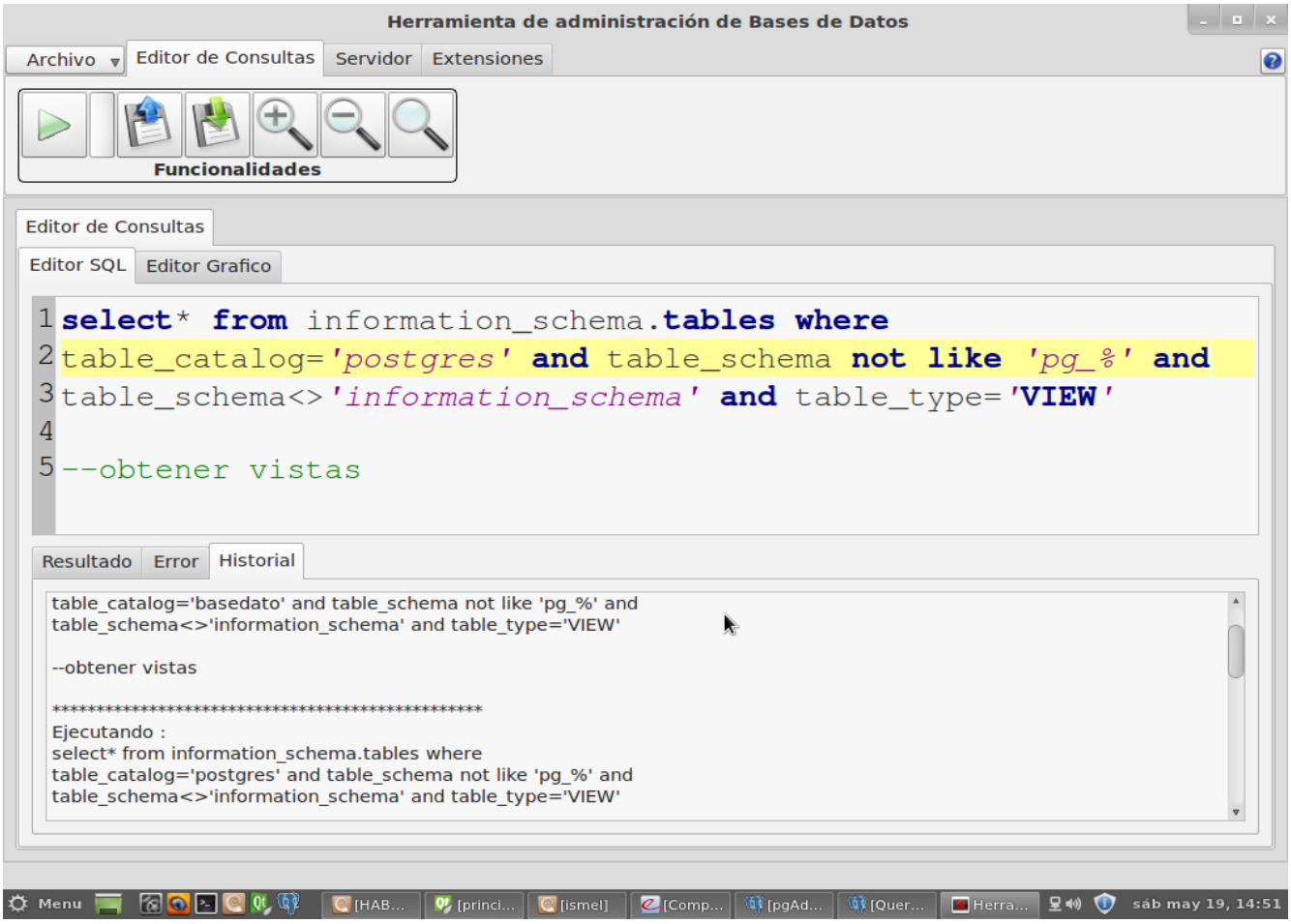
Anexo 1: Interfaz de usuario Crear vistas.



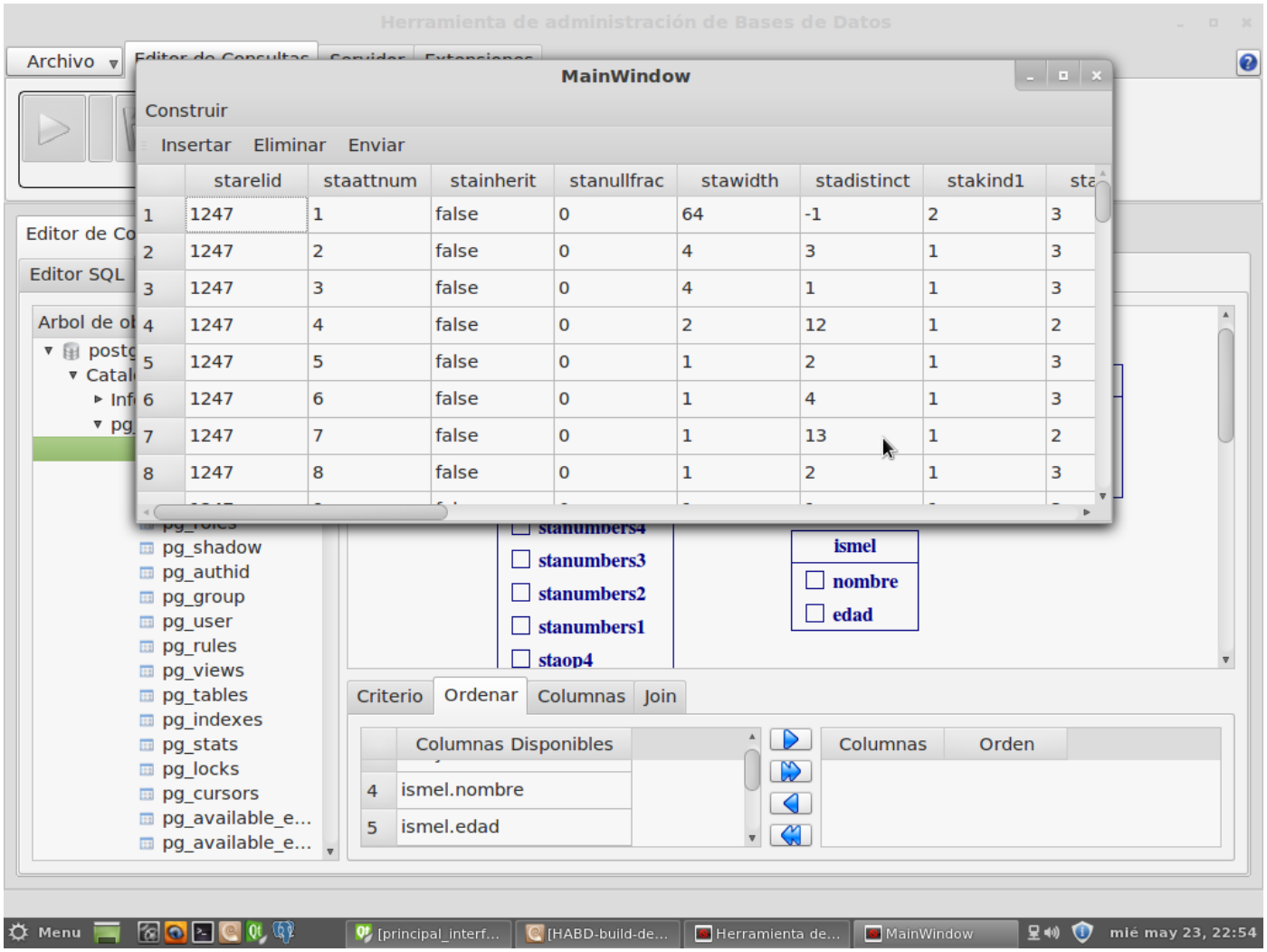
Anexo 2: Interfaz de usuario Manejar archivos.



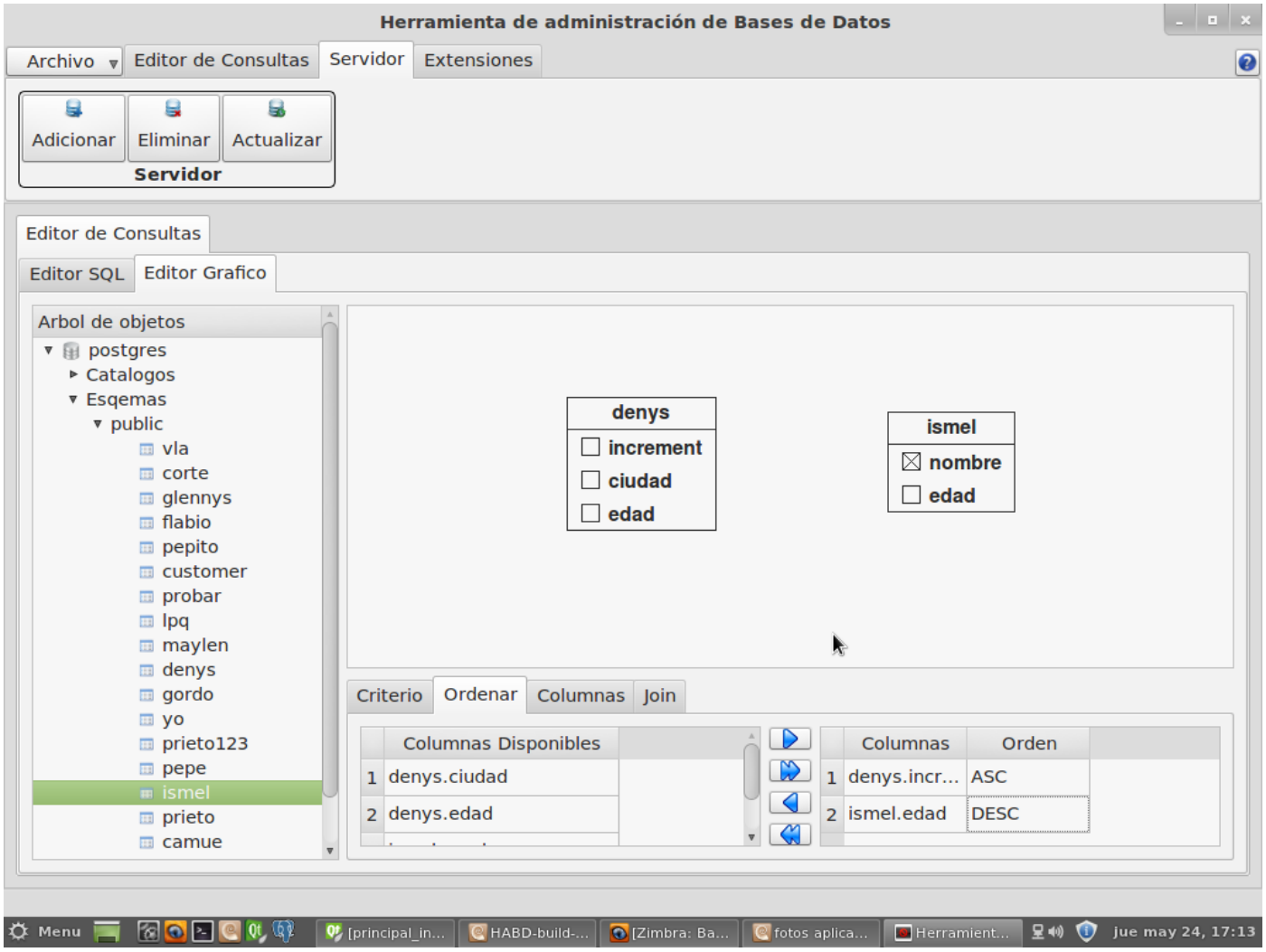
Anexo 3: Interfaz de usuario con el código aumentado.



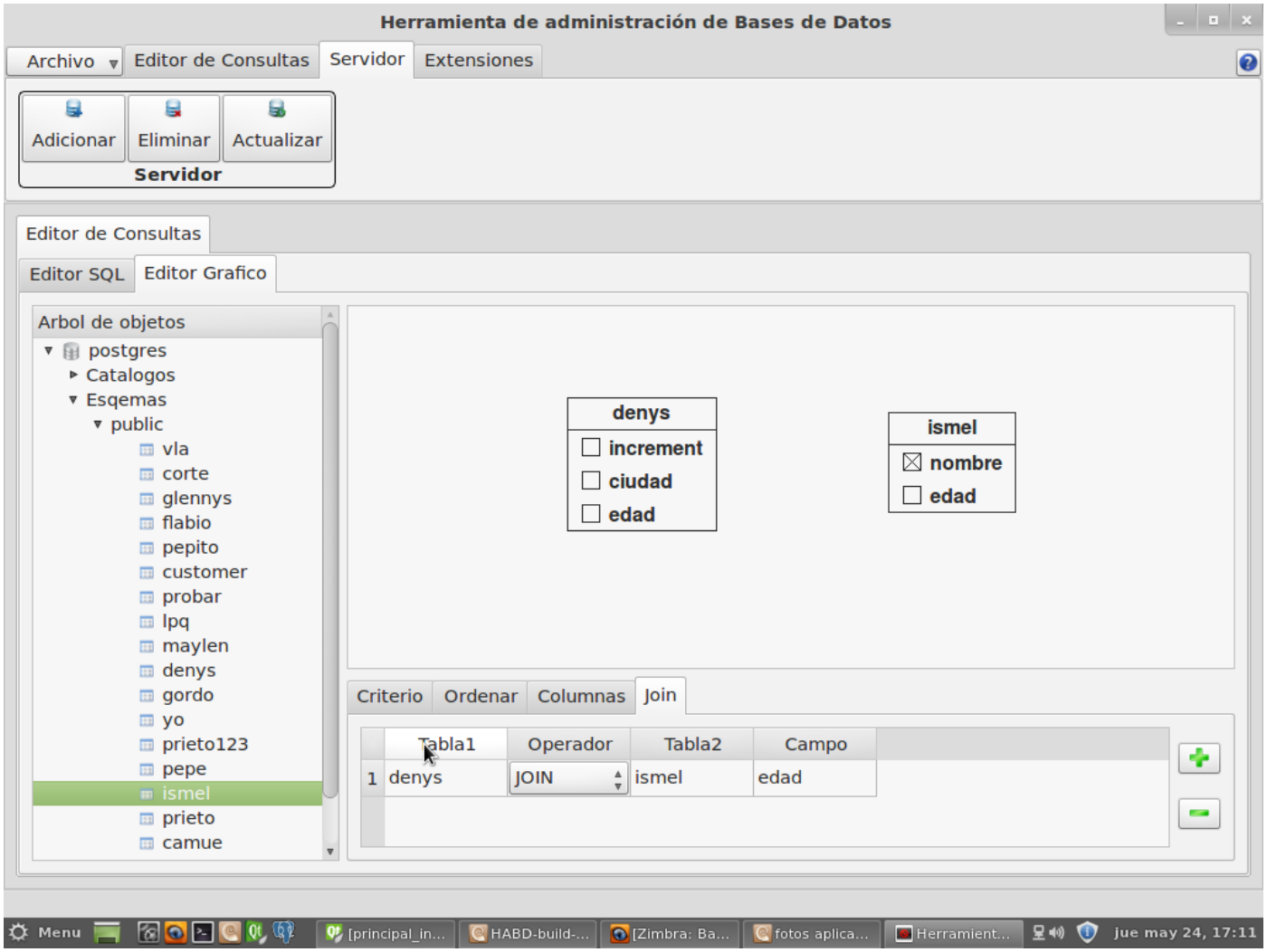
Anexo 4: Interfaz de usuario Gestionar datos.



Anexo 5: Construcción del Order By.



Anexo 6: Construcción del Join.



Anexo 7: Carta de aceptación del cliente al producto desarrollado.

Acta de Aceptación

Proyecto: PostgreSQL

Categoría de las pruebas: Revisión a la aplicación.

Fecha de conciliación: 5 de junio de 2012.

Firma de los involucrados en el proceso:

- **Por la parte del Cliente (PostgreSQL): Ing.** Flavio Enrique Roche Rodríquez
- **Por la parte Jefe de Departamento (PostgreSQL): Ing.** Edgar Roja Ricardo

Observaciones del proceso:

Durante el proceso de pruebas se detectaron un conjunto de incidencias que quedaron registrados adecuadamente en el correspondiente documento Informe Final de No Conformidades (NC), con sus respectivas observaciones. Teniendo en cuenta que las No Conformidades han sido debidamente resueltas por el Equipo de Desarrollo y validada la eficacia de la corrección por los clientes, se ha tomado el acuerdo de Aceptar la aplicación **Plugin editor de consultas para la herramienta de administración de bases de datos HABD en** su versión **1.0**, integrado en la herramienta de administración de bases de datos **HABD** en su versión **1.0** , con fecha 5 de junio de 2012.

Para que conste la Aceptación de los resultados de las pruebas y por tanto la Aceptación **del Plugin editor de consultas para la herramienta de administración de bases de datos HABD** en su versión **1.0**, dando fe del acuerdo, se extiende la presente Acta en un (1) ejemplar.

Ing. Flavio Enrique Roche Rodríquez

(Nombre y apellidos)
Cliente
(PostgreSQL)

Ing. Edgar Roja Ricardo

(Nombre y apellidos)
Jefe de Departamento
(PostgreSQL)

Referencia:

GLOSARIO DE TÉRMINOS

API: Interfaz de Programación de Aplicaciones.

BD: Base de Datos.

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.

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).

MVC: Patrón arquitectónico Modelo Vista Controlador.

Plugin: Software que se relaciona con otro para portar una función nueva, generalmente muy específico que se ajusta a la aplicación principal.

QT: Nombre de la compañía que lo creó: Quasar Technologies.

SC: Abreviatura de la palabra Sección utilizada para enumerar las secciones de los casos de pruebas.

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.