Universidad de las Ciencias Informáticas
"Facultad 6"

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Título: Plugin CRUD-PG para la herramienta de administración de bases de datos HABD.

Autores:

Evelyn Rodríguez Rojas

Yaciel López Pérez

Tutores:

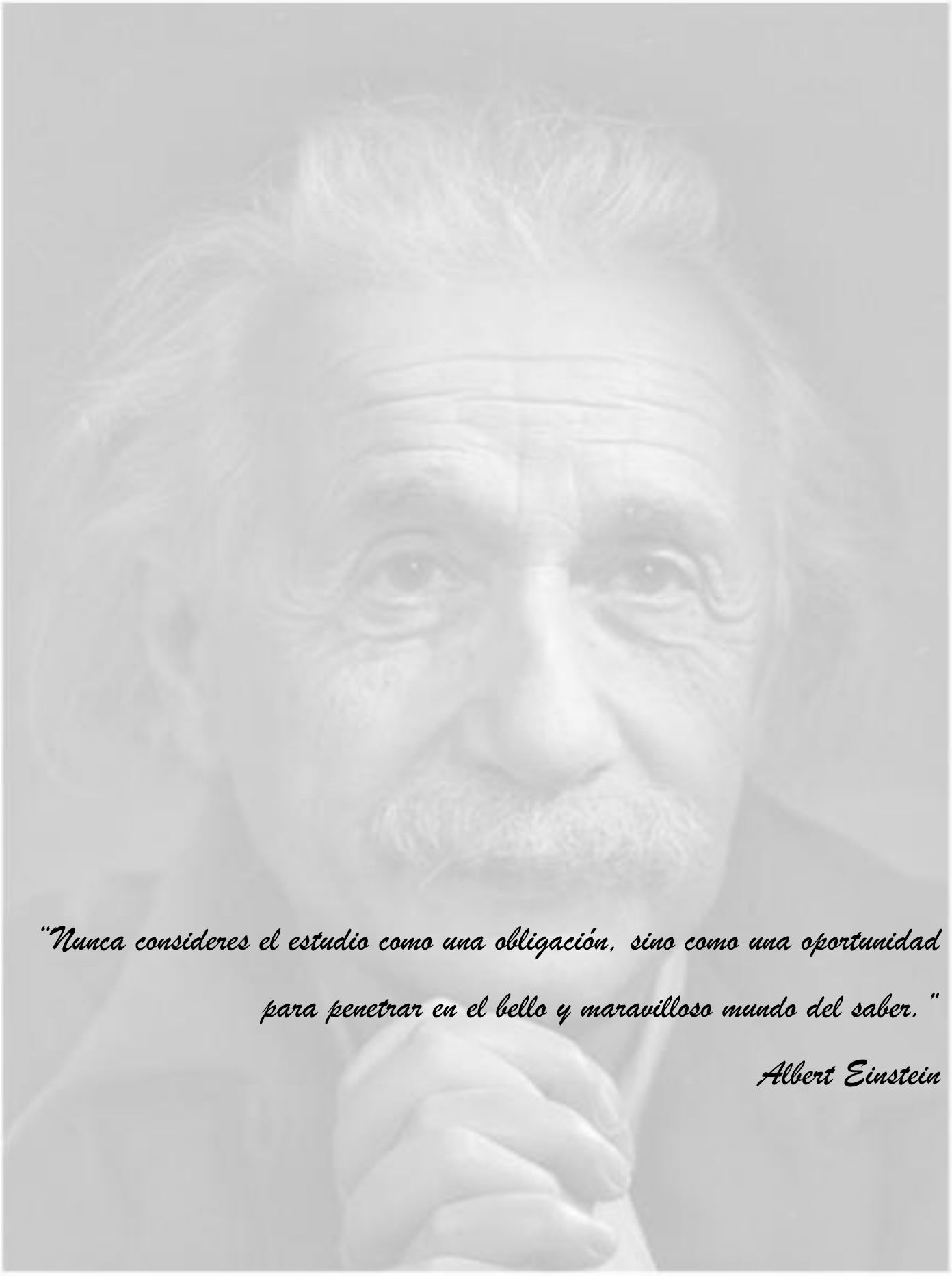
Msc. Anthony R. Sotolongo León.

Ing. Flavio Enrique Roche.

Ciudad de la Habana, Junio del 2012

"Año 53 de la Revolución Cubana"

Frase



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

Albert Einstein

Declaración de autoría

Declaramos ser autores del presente trabajo “Plugin CRUD-PG para la herramienta de administración de bases de datos HABD” y reconocemos a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año 2011.

_____	_____
Autor	**Autor**
Evelyn Rodríguez Rojas	**Yaciel López Pérez**
_____	_____
Tutor	**Tutor**
Msc. Anthony R. Sotolongo León.	**Ing. Flavio Roche Rodríguez**

Datos de contactos

Tutores:

Msc. Anthony R. Sotolongo León.

Especialidad de graduación: Ingeniero en Ciencias Informáticas

Años de experiencia en el tema: 2 años

Años de graduado: 2 años

e-mail: asotolongo@uci.cu

Ing. Flavio Enrique Roche.

Especialidad de graduación: Ingeniero en Ciencias Informáticas

Años de experiencia en el tema: 1 años

Años de graduado: 1 años

e-mail: feroche@uci.cu

Agradecimientos

De Evelyn:

Fueron muchas las personas que tuvieron que ver con este gran logro, pero la primera y más importante es una personita que si le debo todo lo que soy y esa es mi mamá Estrella, que es la luz que guía mi vida y que siempre me acompaña donde quiera que voy, la persona más importante de mi vida y a la que amo con la vida.

Le agradezco a mi novio Jorge por estar ahí siempre para mí, por confiar y aguantar todas mis malcriadeces y mis malos ratos. A mi papá Carlos por ayudarme siempre que pudo. A mis tutores Flavio Enrique y Anthony Rafael por ayudarme a lograr mi gran sueño, pero muy especial a mi tutor Flavio.

Agradezco también a mi hermano Roniel por ayudarme a lo largo de mi carrera, a pesar de nuestras discusiones siempre estuvo ahí para mí, lo quiero mucho, a mi abuela Nenita que aunque ella no esté conmigo sé que siempre me cuida donde quiera que esté, porque es mi angelito de la guarda y sé que estaría muy orgullosa de mi, a mis suegros Mirtha y Jorge, mis cuñadas Nataliek y Kirenia, a mis sobrinitos Renier y Keiny que son la chispita de mi vida, a mi padrastro Carlos por ser muy bueno conmigo y con mi mamá, a mi compañero de tesis Yaciel que sin él no habría sido posible todo esto, a mi prima Yanet por ayudarme mucho en el primer año de mi carrera que fue el más difícil, a mi prima Yamitza, mi tía Mirtha, mi tío Tony, en fin a toda mi familia.

Le agradezco a mi grupo 307, el grupo que me acompañó hasta mi tercer año de la carrera, con él que pase mis mejores años. También quiero agradecerle a las nuevas amistades que encontré

Agradecimientos

aquí en la UCI, Héctor Alcides (ojos bellos) y Abel, a ellos y a todas aquellas personas que siempre confiaron en mí les agradezco mucho.

Por último, les agradezco también a las personas que en algún momento quisieron y me hicieron tanto daño, eso me hizo entender que no todo es color de rosa y que hay que ser fuerte y mirar siempre hacia adelante, porque los años pasan, pero los recuerdos quedan.

Muchas Gracias

Dedicatoria

De Evelyn:

Le dedico este trabajo de diploma y todo lo que soy a la persona más importante de mi vida, esa personita que siempre me apoyó y creyó en mi todo el tiempo, la que me aconsejaba cuando las cosas no me salían bien, la que me decía no te preocupes de todo se sale, la que se desveló por mi día y noche. En fin le dedico todo esto por tantas y tantas cosas, pero la principal y más importante porque la quiero mucho.

Realmente creo que cada hijo en este mundo tiene que estarle agradecido a su madre por cada cosa, quizás haya niños que ya no la tengan, pero yo le doy gracias a Dios porque la tengo a ella y porque todavía puedo compartir cada minuto de mi vida con ella; por eso le dedico todo lo que soy, para poder de alguna forma expresarle todo lo que siento por ella. Esa persona de quien hablo es mi mamá, la merecedora de todas mis victorias, a ella va dedicado todo lo que soy. La quiero mucho.

RESUMEN

El desarrollo de la informática en la actualidad avanza vertiginosamente, cada vez más se requieren de aplicaciones que hagan las acciones más fáciles al usuario, con el menor costo de tiempo y esfuerzo posible. Una herramienta que facilita esto es la herramienta de administración de base de datos HABD, la cual provee una interfaz amigable y una arquitectura basada en plugins que le permite a los desarrolladores agregar nuevos servicios de forma fácil, evitando así que los usuarios necesiten de varias aplicaciones para resolver un único problema. Independiente de esta herramienta se creó otra llamada CRUD-PG 1.0, la cual permite implementar lógica de negocio dentro del servidor de base de datos mediante funciones o procedimientos almacenados, facilitando a los programadores que la utilicen el trabajo con las base de datos y además permite establecer un estándar para el nombre de las funciones. Esta nueva herramienta presenta algunas deficiencias que atentan contra su utilización para este tipo de tareas, además no puede ser integrada a la herramienta de administración de base de datos HABD, por haber sido desarrollada en otro lenguaje de programación y por no contar con una arquitectura basada en plugin como esta herramienta de administración lo exige.

En el presente trabajo se desarrolla una nueva versión de dicha herramienta donde se erradican los inconvenientes detectados. También se le añaden nuevas funcionalidades y se logra tener una interfaz más amigable y fácil de interactuar para el usuario.

PALABRAS CLAVES: herramienta de administración de base de datos, HABD, CRUD-PG.

Índice

Índice de tabla

Índice de figura

INTRODUCCIÓN

En la actualidad la información ha llegado hacer el eje principal que mueve a las grandes empresas, a raíz de esto surge la necesidad de almacenar gran cantidad de información de forma rápida y fácil para tenerla organizada de tal manera que esta permita ser accedida en cualquier momento por todas aquellas personas que la necesiten. Estos grandes volúmenes de datos no pueden ser procesados de forma manual por el hombre debido a la gran dificultad que esto implica, por lo que existen métodos que facilitan un rápido almacenamiento de estos. Para poder recopilar toda esta información es que surge la tecnología de base de datos.

Debido a la gran evolución de las base de datos, estas tienen mayor calidad, flexibilidad y manejo desde la creación de los gestores de base de datos (SGBD). De esta forma, la gestión de datos se libró de los grandes ordenadores centrales, para así distribuirse según las necesidades de los usuarios, permitiéndole crear y mantener sus bases de datos, utilizando herramientas para transformar lo lógico en conjuntos de datos.

A través de los años el gran desarrollo de los gestores de base de datos lo han tenido importantes compañías como ORACLE, Informix, entre otras, pero a veces se hace un poco difícil su obtención al ser estas empresas privadas, lo que trae consigo que se dificulte acceder a sus productos.

Debido a este problema se han desarrollado otros gestores de bases de datos gratuitos que tienen gran calidad, dentro de los cuales se encuentra PostgreSQL, siendo actualmente el sistema libre más avanzado, que permite una fácil gestión de los usuarios y de las bases de datos que contiene. Por la gran escalabilidad que posee puede ser ejecutado en la mayoría de los sistemas operativos y brinda una mayor confianza a la hora de integrar los datos.

La Universidad de las Ciencias Informáticas (UCI) además de ser un centro educativo, también está dedicada a la producción de software y servicios informáticos, permitiendo así vínculos entre el estudio y el trabajo como guía de formación personal de cada estudiante. En esta existen diferentes centros productivos dentro de los que se destaca el Centro de Tecnologías de Gestión de Datos (DATEC), el mismo está compuesto por varios departamentos dentro de los que se encuentra PostgreSQL.

La UCI también forma parte de “La Comunidad Técnica Cubana de PostgreSQL”, la cual cuenta con especialistas de alto nivel que potencian el empleo de tecnologías de base de datos de código abierto. En su gran mayoría las herramientas que esta Comunidad posee son destinadas para administrar, monitorizar y dar mantenimiento a distintas bases de datos y aunque son libres, se imposibilita en muchas ocasiones obtenerlas, por ser Cuba un país bloqueado.

A raíz de esto el departamento de PostgreSQL se ha encaminado en la creación de herramientas netamente implementadas en el país, posibilitando que puedan ser reutilizadas a lo largo del tiempo. Estas herramientas necesitan de complementos que colaboren al desarrollo de nuevas y especificas funciones.

En el curso 2010-2011 en este departamento se presentó el trabajo de diploma Exploración y diseño de la Herramienta de Administración de Bases de Datos para PostgreSQL (HABD). Este trabajo propone el desarrollo de una nueva herramienta de administración, el cual surge a partir de las investigaciones realizadas sobre los inconvenientes encontrados en los principales software para la administración de bases de datos que existen en el mundo. Esta aplicación presenta una arquitectura basada en plugins, lo que permite agregar un gran número de funcionalidades evitando así que los usuarios necesiten de varias aplicaciones para resolver un único problema.

Además de esta aplicación se desarrolló otra llamada CRUD-PG que se encarga de generar funciones o procedimientos almacenados de forma automática para realizar las operaciones de escritura, lectura, actualizar y eliminar (por sus siglas en inglés CRUD) sobre las tablas de las bases de datos de PostgreSQL.

Generalmente estas operaciones se implementan en la capa de negocio de las aplicaciones. Incluso existen algunos framework que generan estas operaciones de forma automática, liberando así a los programadores de realizar estas tareas, pero esto tiene la desventaja de que crean las funciones en el mismo lenguaje con el que están implementadas, imposibilitando a otras aplicaciones realizadas en otro lenguaje, acceder a las funciones que este genera. Otra desventaja es que tiene que procesar resultados intermedios en el cliente para obtener el resultado final, lo cual ralentiza el proceso de intercambio de datos entre el cliente y el servidor.

De esta forma se desaprovecha una de las principales ventajas que tiene el gestor de base de datos PostgreSQL, el cual permite la implementación de procedimientos almacenados o funciones, posibilitando así, realizar operaciones dentro del propio gestor, es decir incluir la lógica de negocio dentro de las bases de datos, lo que posibilita lograr mejores tiempos de desarrollo de las aplicaciones y respuestas en servidores de base de datos PostgreSQL.

CRUD-PG no permite su integración a HABD ya que no cuenta con una arquitectura orientada a plugins como este lo exige y además está implementada en otro lenguaje de programación. También presenta inconvenientes a la hora de generar las funciones con respecto a la seguridad y optimización del código, no permite la implementación de cursores para perfeccionar las funciones que trabajen con

grandes cantidades de datos; carece de facilidades visuales para el usuario como son: la selección de las funciones que el usuario desee generar o eliminar y la visualización previa del código de las funciones que serán generadas.

Por todo lo antes expuesto se propone como **problema de la investigación**: ¿Cómo contribuir al desarrollo de operaciones CRUD sobre las tablas de datos PostgreSQL en la herramienta de administración de base de datos HABD?

Con el desarrollo de este trabajo se pretende dar solución al problema antes citado. Para la obtención de estos resultados se plantea como **objeto de estudio:** Proceso de programación en PostgreSQL enmarcado en el **campo de acción:** Operaciones CRUD en lenguajes procedurales.

El **objetivo general** del trabajo: Desarrollar un plugin que permita operaciones CRUD para la herramienta de administración de base de datos HABD.

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

- Analizar y diseñar el plugin para la generación de operaciones CRUD en las tablas de base de datos PostgreSQL.
- Implementar el plugin para la generación de operaciones CRUD en las tablas de base de datos de PostgreSQL.
- Probar el plugin para generar las operaciones CRUD en las tablas de bases de datos de PostgreSQL.

Para dar cumplimiento a los objetivos específicos se plantean las siguientes **tareas de la investigación:**

- Caracterización de las diferentes herramientas de administración de bases de datos existentes en el mundo que permiten la generación de operaciones CRUD en tablas de base de datos.
- Caracterización de las metodologías, herramientas y tecnologías a utilizar en el desarrollo del plugin para la generación de operaciones CRUD en tablas de base de datos.
- Diseño del plugin a partir de las funcionalidades identificadas.
- Identificación de las funcionalidades del plugin para la generación de operaciones CRUD en tablas de base de datos.

- Implementación de las funcionalidades identificadas.
- Aplicación de pruebas funcionales para validar que el plugin desarrollado se integra a la herramienta HABD.
- Diseño de los casos de pruebas para validar el plugin para la generación de operaciones CRUD en tablas de bases de datos.

Estructura del trabajo

El presente trabajo está formado por: introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía, anexos y glosario de términos. A continuación se muestra una breve descripción de cada capítulo.

Capítulo 1: Fundamento teórico.

En el capítulo se describe el marco teórico del trabajo. Contiene un análisis de los distintos conceptos que permiten un entendimiento claro y preciso del trabajo a desarrollar. Este brinda una breve reseña de las características fundamentales que va a tener este complemento, además de la metodología aplicada, herramientas y lenguajes en las que se apoyará la realización del mismo.

Capítulo 2: Características del sistema propuesto.

En el capítulo se realiza un estudio preliminar del negocio y de las necesidades de información. Además se definen los requisitos funcionales y no funcionales, entre otros artefactos que ayudan a entender aún más el flujo de información.

Capítulo 3: Validación del sistema propuesto.

En este capítulo se realizan las pruebas pertinentes para poder validar la implementación del plugin y dar una solución concreta y precisa para después valorar los resultados obtenidos. Además se muestran las no conformidades encontradas de la aplicación.

CAPÍTULO 1: Fundamento teórico

Introducción

En el presente capítulo se abordarán los conceptos referentes al dominio del problema, los cuales son imprescindibles para comprender el objetivo y el alcance del trabajo. Además se define y describe la metodología a utilizar. Se realiza la descripción de las características de cada una de las herramientas utilizadas para el desarrollo de la solución propuesta, proporcionando fundamentos que justifiquen su selección.

1.1 Conceptos asociados a la investigación

Desde su surgimiento, las bases de datos se convirtieron en la herramienta fundamental para almacenar gran cantidad de información. Debido a esto, en muy poco tiempo la información almacenada por las grandes empresas y negocios alcanzaron una dimensión considerablemente voluminosa. Con la acumulación de esta información se presentó la problemática de cómo darle un fin útil.

¿Qué es una Base de Datos?

Las bases de datos son un conjunto de datos almacenados entre los que existen relaciones lógicas y han sido diseñadas para satisfacer los requerimientos de información de una empresa u organización. En una base de datos, además de los datos, también se almacena su descripción. (1)

Herramienta de administración de base de datos

Las herramientas de administración de base de datos pueden conceptualizarse de varias maneras:

- Son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. (2)
- Son un conjunto de programas que administran y gestionan la información contenida en una base de datos. (3)

Según los conceptos detallados previamente, se puede decir que las herramientas de administración de bases de datos son sistemas que permiten administrar mediante una interfaz de usuario los gestores de bases de datos, también garantizan que la información almacenada esté segura, dando permiso a usuarios o grupos de usuarios que tengan acceso al servidor.

Sistema Gestor de Base de Datos

Debido al aumento de información y del personal que accede sobre ella, surge la necesidad de contar con sistemas de administración para controlar todos los datos y los usuarios. Para la administración de las bases de datos se utilizan los Sistemas de Gestión de Base de Datos (SGBD).

Los sistemas de gestores de bases de datos son un conjunto de aplicaciones que permiten a los usuarios definir, crear y mantener la base de datos y proporciona un acceso controlado a la misma. Los SGBD es la aplicación que interactúa con los usuarios de los programas de aplicación y la base de datos. (4)

Después del concepto visto anteriormente se puede decir que un sistema gestor de base de datos no es más que una herramienta para asegurar el mantenimiento de los datos existentes en una aplicación. Es por esto que uno de los principales objetivos de los SGBD es facilitarle al usuario las herramientas necesarias para manipular los datos siempre y cuando sea de una manera práctica y eficiente, de modo que no sea necesario que el usuario conozca la manera de en qué se esté almacenando la información en la computadora.

Plugin

Frecuentemente se hace necesario requerir ciertos complementos que amplíen las funcionalidades de las aplicaciones, para lograr esto se han creado los plugins.

El Dr. Harvey M. Deitel[1] los define como: Pequeñas aplicaciones que añaden alguna función a otros programas, habitualmente de mayor tamaño. Un programa puede tener uno o más conectores. Son muy utilizados en los programas navegadores para ampliar sus funcionalidades. (5)

[1] Harvey M. Deitel, consejero delegado de Deitel & Associates, Inc., tiene 40 años de experiencia en el campo de la computación, incluida la industria y experiencia académica. El Dr. Deitel obtuvo una licenciatura y una maestría en el Massachusetts Institute of Technology y un doctorado la Universidad de Boston. Él tiene 20 años de experiencia en la universidad la enseñanza, incluida la tenencia de ganar y que actúa como Presidente del Departamento de Informática de la Universidad de Boston antes de la fundación Deitel & Associates. El Dr. Deitel ha entregado seminarios profesionales a nivel internacional para las grandes corporaciones, organizaciones gubernamentales y varias ramas de las fuerzas armadas.

Otro de los conceptos de plugin más utilizados es el del escritor Paul J. Deitel[2] que los define como módulos de hardware o software que añaden características o servicios específicos a un sistema más grande. (6)

Según los conceptos vistos anteriormente por dos escritores muy reconocidos se puede decir que los plugin son complementos que se crean para brindar beneficios a los desarrolladores que trabajan con aplicaciones, para de esta forma ampliar sus capacidades.

Lenguaje de programación del lado del servidor

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página al cliente. Estos son independientes del navegador utilizado, que no necesitará plugin especiales para visualizar correctamente cualquier página. (7)

Como ejemplos se puede citar algunos de ellos, entre los que se encuentran ASP, PHP, PERL, los cuales se ejecutan en servidores web. Los que se usarán durante el desarrollo de la aplicación son SQL y PL/pgSQL, los cuales son ejecutados en servidores de bases de datos PostgreSQL.

SQL surge en el año 1974, bajo las investigaciones en los laboratorios de IBM. A principios de los años 80, gracias al gran éxito de este, se convierte en el estándar industrial de las bases de datos relacionales. Es un lenguaje estándar usado para sistemas administradores de bases de datos. Tiene un proceso de construcción de consultas y funciones basado en comandos, parámetros y predicados.

El lenguaje PL/pgSQL fue escrito por Jan Wieck[3]. Es comparable al lenguaje procedural de Oracle, PL/SQL, es procedural cargable, provisto por el sistema de bases de datos PostgreSQL, fácil de aprender, potente y siempre está disponible.

1.2 Herramienta CRUD-PG

La herramienta CRUD-PG fue desarrollada por un grupo de personas del departamento de PostgreSQL del centro DATEC de la Universidad de Ciencias Informáticas (UCI). Esta permite desarrollar parte de la lógica de negocio en la base de datos PostgreSQL mediante procedimientos

[2] Es autor o co-autor de docenas de libros y paquetes multimedia y actualmente está escribiendo muchas más. Con las traducciones publicadas en japonés, ruso, español, italiano, chino básico, chino tradicional, coreano, francés, polaco y portugués, los textos de la Deitel se han ganado el reconocimiento internacional.

[3] Contribuyó al segundo lenguaje procedural, PL/pgSQL, para ir con el original lenguaje procedural PL/pgTCL con el que el contribuyó la última versión.

almacenados o funciones programadas en diferentes lenguajes como son PL/pgSQL y SQL, realizando así operaciones de escritura, lectura, actualización y eliminación (por sus siglas en inglés CRUD) lo que posibilita una mayor rapidez en la interacción con la base de datos. Para lograr esto CRUD-PG genera una serie de funciones estándares capaces de realizar dichas operaciones y de esta forma implementar parte de la lógica del negocio en la base de datos.

En cada base de datos donde se utiliza la herramienta, esta generará una serie de funciones capaces de operar los datos, permitiendo a los programadores hacer uso de las mismas sin tener que preocuparse por implementarlas en sus aplicaciones, ya que en cada base de datos donde se aplique la misma se generarán funciones capaces de operar los datos. Además estas funciones tienen la característica de ser independientes del lenguaje en que están programadas dichas aplicaciones.

1.2.1 Integración del CRUD-PG con la herramienta de administración de base de datos HABD

CRUD-PG es una herramienta para generar funciones automáticas mediantes procedimientos almacenados, esta quiere ser integrada con la herramienta de administración de base de datos HABD, pero no es posible porque dicha herramienta está implementada en otro lenguaje de programación el cual es Java. Actualmente existen varias formas de integrarla, mediante plugin, módulos y librerías, pero esta será integrada a través de un plugin, ya que es una herramienta hecha para la gestión de los mismos.

Ventajas de los plugin

Cada plugin en forma particular tiene sus propias ventajas, pero independientemente se puede decir que estas en su generalidad son:

- Aumentan las funcionalidades y el rendimiento del programa al que se integre.
- Ahorran tiempo y trabajo con las funciones que le son implementadas.
- Se integran a los programas sin interferir en el desarrollo de los mismos.
- Pueden llevar a cabo efectos en tiempo real como módulos de software.

Ventajas de integrar el plugin CRUD-PG a la herramienta de administración de base de datos HABD

Existen diferentes ventajas al integrar el pulgin CRUD-PG a la herramienta HABD, entre las cuales se encuentran:

- Para programar funciones o procedimientos almacenados que realicen operaciones CRUD, un programador con habilidades medias utiliza como promedio 90 segundos para cada una. Con la integración del plugin CRUD-PG a la herramienta HABD este tiempo disminuye considerablemente, pues el mismo genera estas operaciones de forma automática con un nombre estándar para su ejecución y en dependencia de las características de hardware del servidor el tiempo de implementación de las mismas es de milisegundos.
- Otro de los beneficios que trae consigo esta integración es que el plugin permanece inactivo hasta que el usuario seleccione una tabla de la base de datos. Esto permite que no sobrecargue la aplicación a la cual se integra, posibilitando un mejor uso de los recursos del sistema.
- No depende de la integración de otros plugins para su funcionamiento.

1.3 Metodología de desarrollo de software

Una metodología se refiere a los métodos de investigación en una ciencia, esta se entiende como la parte del proceso de investigación que permite vincular los métodos y las técnicas necesarios para llevarla a cabo. (8)

Las metodologías son la vía a seguir para desarrollar un software de forma consecuente, estas estrechan tres prioridades principales que conducen a una mejor calidad de aplicaciones, permiten un transcurso de desarrollo controlado y un proceso normalizado en una organización, es decir, no depende del personal.

Las metodologías pretenden guiar a los desarrolladores en el proceso de desarrollo de software definiendo artefactos, roles e indicando paso a paso las actividades que se van a realizar mediante prácticas y técnicas recomendadas. Su principal meta es lograr un producto final de calidad y que cumpla con los requisitos del usuario, pero los requisitos son tan diversos y cambiantes, que han dado lugar al surgimiento de una variada cantidad de metodologías de desarrollo.

Dentro de estas se encuentra Extreme Programming (XP), la cual es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo del software, promoviendo así el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un clima agradable y amistoso de trabajo. (8)

La metodología XP incluye al cliente en el equipo de desarrollo, garantizando naturalidad en las soluciones implementadas y una buena actitud del personal del proyecto para enfrentar los cambios. Es apropiada para proyectos de alto riesgo técnico para los cuales precisa de cuatro variables: coste, tiempo, calidad y ámbito. Una de las grandes primacías de XP es que garantiza a toda costa el cumplimiento de integrar todas las necesidades del cliente en el proyecto.

Características de la metodología Programación Extrema

El desarrollo bajo XP tiene características que lo distinguen de otras metodologías, estas son:

- Comunicación efectiva en tiempo real entre el cliente y los desarrolladores.
- Se liberan varias entregas del software en la medida que se va desarrollando.
- Los objetivos planteados en características, tiempo y costos son reajustados inalterablemente en función del avance real obtenido.
- Añade funcionalidad con retroalimentación continua en correspondencia con la magnitud que va alcanzando el proyecto.
- Los cambios son parte sustantiva del proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierte en miembro del equipo.

Se decide utilizar XP porque es una metodología que se adecúa perfectamente a las características que presenta el proyecto como son: requisitos muy cambiantes, existe un alto riesgo técnico, posee un equipo pequeño y poco tiempo de desarrollo. Además el cliente forma parte del equipo de desarrollo, lo que permite establecer un mejor vínculo de comunicación entre este y los desarrolladores, elevando así la calidad del sistema a desarrollar.

1.4 Tecnologías y herramientas a utilizar

Para el desarrollo del plugin se realizó un estudio exhaustivo acerca de las tecnologías y herramientas existentes, a raíz de esta investigación se definieron las más adecuadas para cumplir con lo establecido por el del departamento y su línea de desarrollo, las cuáles se describirán a continuación.

1.4.1 Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML) es el lenguaje estándar para modelar software que permite la visualización, especificación y documentación de los modelos que se crean durante la construcción y desarrollo de un software. (9)

UML es un lenguaje fácil y descriptivo, ya que permite modelar aplicaciones en cualquier dominio y contiene generación automática de código, ajustando su utilización para todas aquellas personas que no tengan conocimientos de lenguajes de programación.

Este provee al usuario, la posibilidad de que reflexione antes de invertir y gastar grandes cantidades de recursos en proyectos de riesgo.

Propiedades de UML como lenguaje de modelado estándar

UML posee grandes propiedades que lo hacen ser el lenguaje de modelado de sistemas de software más conocido en la actualidad, estas son:

- Concurrencia, es un lenguaje distribuido y adecuado a las necesidades de conectividad actuales y futuras.
- Ampliamente utilizado por la industria desde su adopción por OMG.
- Reemplaza a decenas de notaciones empleadas con otros lenguajes.
- Modela estructuras complejas.
- Las estructuras más importantes que soportan tienen su fundamento en las tecnologías orientadas a objetos, tales como objetos, clase, componentes y nodos.
- Emplea operaciones abstractas como guía para variaciones futuras, añadiendo variables si es necesario.
- Comportamiento del sistema: casos de uso, diagramas de secuencia y de colaboraciones, que sirven para evaluar el estado de las máquinas. (10)

El UML tiene como objetivo principal y más importante resolver los requisitos guiados por casos de usos. Además de ser un lenguaje diseñado con un propósito general que puede ser utilizado por todos los modeladores y de código libre.

1.4.2 Herramienta de Ingeniería de Software Asistida por Computación

Las Herramientas CASE (*Computer Aided Software Engineering*; y en su traducción al Español significa Ingeniería de Software Asistida por Computación) son 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. Como es sabido, los estados en el ciclo de vida de desarrollo de un software son: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación. Estas se pueden ver como la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales. (11)

Ventajas de las herramientas de Ingeniería de Software Asistida por Computación

Las herramientas CASE poseen grandes ventajas que permiten aumentar la productividad en el desarrollo de un software:

- Verificar el uso de todos los elementos en el sistema diseñado.
- Automatizar el dibujo de diagramas.
- Ayudar en la documentación del sistema.
- Ayudar en la creación de relaciones en la Base de Datos. (11)

La principal ventaja que ofrece una herramienta CASE es que facilita crear productos con mayor calidad, lo que permite aumentar su rendimiento. Para lograr esto hay que contar con una metodología de desarrollo adecuada, además de contar con la propia herramienta.

Existen diferentes tipos de herramientas CASE, dentro de las cuales se encuentra Visual Paradigm, la cual es una herramienta que es aplicada a lo largo de todo el ciclo de vida de un software. En estas son incluidas actividades como la gestión de proyectos y la estimación. También ofrece un conjunto completo de herramientas de los equipos de desarrollo de software necesario para la captura de requisitos, software de planificación, la planificación de controles, el modelado de clases y de datos, entre otras.

Características de Visual Paradigm:

- Visual Paradigm For UML es una Herramienta Case que soporta las últimas versiones del mismo, (Lenguaje de Modelado Unificado) y la Notación y Modelado de Procesos de Negocios. Desde un Grupo Administrador de Objetos.
- En adición al soporte de Modelado UML esta herramienta provee el modelado de procesos de negocios, además de un generador de mapeo de objetos-relacionales para los lenguajes de programación Java .NET y PHP.
- Para desarrolladores independientes existe una versión llamada *Community Edition* en la que se caracteriza por ser de uso No Comercial.
- Se integra con las siguientes herramientas Java: Eclipse/IBM WebSphere, JBuilder, NetBeans IDE, Oracle JDeveloper, BEA Weblogic.
- Está disponible en varias ediciones, cada una destinada a unas necesidades: *Enterprise, Professional, Community, Standard, Modeler y Personal.* (12)

Ventajas de Visual Paradigm

- Navegación intuitiva entre código y el modelo.
- Poderoso generador de documentación y reportes UML PDF/HTML/MS Word.
- Demanda en tiempo real, modelo incremental de viaje redondo y sincronización de código fuente.
- Superior entorno de modelado visual.
- Soporte completo de notaciones UML.
- Diagramas de diseño automático sofisticado.
- Análisis de texto y soporte de tarjeta CRC.
- Proporciona soporte a varios lenguajes en generación de código e ingeniería inversa a través de plataformas java. (12)

Se utiliza el Visual Paradigm para realizar el diseño de diagrama de clases, las clases conceptuales del modelo de dominio y el diagrama de paquetes, ya que mediante estos se obtiene un mejor entendimiento de la implementación y diseño de la aplicación, además de ser la herramienta a utilizar definida por el propio proyecto.

1.4.3 Lenguaje de programación

Los Lenguajes de Programación (LP) son un conjunto de símbolos junto a un conjunto de reglas para combinar dichos símbolos que se usan para expresar programas. Constan de un léxico, una sintaxis y una semántica. (13)

Los LP le facilitan las tareas al programador, ya que cuentan con formas adecuadas que pueden ser escritas y leídas por personas. Son herramientas que permiten crear programas y software, entre ellos se tiene Delphi, Visual Basic, Pascal, Java, C++, etc.

El lenguaje de programación utilizado es el lenguaje C++, es orientado a objetos que le permite al programador diseñar aplicaciones desde un punto de vista más como una comunicación entre los objetos que en una secuencia estructurada de código, aunque permite la programación estructurada de igual manera. (14)

C++ posee muchas características y ventajas importantes que facilitan el trabajo de la aplicación a implementar, a continuación se mostrarán algunas de estas facilidades:

Características del lenguaje de programación C++

- El programador tiene el control total de lo que está haciendo, permitiendo una máxima eficiencia al no incorporar verificación de errores en tiempo de ejecución.
- El código generado por los compiladores del lenguaje no deben incluir una sobrecarga de recursos computacionales, minimizando la implementación de recursos tales como el polimorfismo y la expansión de patrones.
- Es un lenguaje de nivel intermedio, pudiéndose utilizar tanto para escribir software de bajo nivel (drivers, componentes de sistemas operativos, etc.) como para el desarrollo rápido de aplicaciones, según el marco de trabajo con el que se disponga, como VCL de Borland C++ Builder.
- Los compiladores de C++ generan código nativo con un alto grado de optimización en memoria y velocidad, lo que lo convierte en uno de los lenguajes más eficientes.
- El lenguaje apoya el desarrollo de clases genéricas con parámetros de tipo y de tamaño a través de los patrones de clase y de funciones. (15)

Ventajas del lenguaje de programación C++

- El uso de constructores de alto nivel.
- Generar programas eficientes, gracias a la posibilidad de la gestión de memoria que posee mediante punteros.
- Es un lenguaje muy robusto en cuanto a la creación de sistemas complejos se refiere.
- Posee la programación orientada a objetos, lo que permite al programador diseñar aplicaciones desde un punto de vista más como una comunicación entre los objetos, que en una secuencia estructurada de código.
- Por ser un lenguaje compilado, su velocidad de ejecución es más rápida que la de otros lenguajes como por ejemplo Java que utiliza una máquina virtual para poder ejecutarse.
- La posibilidad de poder ser compilado en una variedad de computadoras, con pocos cambios (portabilidad). (6)

La razón por la que se utiliza el lenguaje C++ es porque el plugin que se va a desarrollar necesita ser integrado a la herramienta de administración de base de datos HABD, que fue implementada en el framework de desarrollo Qt que emplea este lenguaje.

1.4.4 Framework de desarrollo

Un framework no es más que un esquema o patrón para el desarrollo o la implementación de una aplicación, no necesariamente están ligados a un lenguaje de programación concreto, aunque sea así en muchas ocasiones. (16)

Los frameworks son diseñados para facilitar el desarrollo de software, ofreciéndole así a los programadores y diseñadores tener más tiempo para identificar los requerimientos del software, además representan una arquitectura de software que mediante esta se pueden modelar las relaciones generales de las entidades del dominio.

Ventajas de los frameworks

Los frameworks tienen diferentes ventajas, pero las principales que no podrían dejar de mencionarse son las que se muestran a continuación:

- El programador no necesita plantearse una estructura global de la aplicación, sino que el framework le proporciona un esqueleto que hay que "rellenar".
- Facilita la colaboración. Cualquiera que haya tenido que "pelearse" con el código fuente de otro programador sabrá lo difícil que es entenderlo y modificarlo; por tanto, todo lo que sea definir y estandarizar va a ahorrar tiempo y trabajo a los desarrollos colaborativos.
- Es más fácil encontrar herramientas adaptadas al framework concreto para facilitar el desarrollo. (16)

Existen diversos frameworks para el trabajo con el lenguaje de programación seleccionado anteriormente, entre estos destaca Qt, este es un *toolkit* (conjunto de herramientas) de plataforma cruzada (*cross-platform*) desarrollado por *Trolltech* (empresa Noruega) para construir GUIs en C++. Es un software multiplataforma ajustable a Microsoft Windows, Linux, Mac OS X. Las primeras versiones datan de 1996 y desde entonces se han desarrollado aplicaciones tales como: Google Earth, Adobe Photoshop Elements, Skype y el entorno de escritorio de Linux: KDE.

Este es distribuido bajo ediciones comerciales o bien *Open Source*. Incluye una extensa librería de clases C++ y utilidades para construir las aplicaciones de forma rápida y sencilla.

Su lenguaje funcional es C++, permitiendo sobre él implementaciones de funcionalidades. Se aplica en las principales plataformas y proporciona una excelente documentación y soporte gracias a las comunidades que lo apoyan.

Características de Qt

- Se puede utilizar para el diseño de interfaz gráfica de usuario (GUI), o para crear aplicaciones completas con su apoyo a la integración con los entornos de desarrollo integrados (IDEs).
- Tiene un excelente soporte para gráficos en 2D y 3D.
- Permite crear aplicaciones independientes de la plataforma de base de datos utilizando estándares bases de datos.
- Incluye controladores nativos para Oracle, SQL Server, PostgreSQL, MySQL, SQLite, y bases de datos compatibles con ODBC.
- Puesto que la arquitectura de Qt se aprovecha de la plataforma subyacente, muchos clientes los utilizan para el desarrollo individual de la plataforma de Windows, Mac OS X y Unix.

- Las aplicaciones Qt pueden ampliar su funcionalidad mediante plugins y librerías dinámicas. Los plugins proporcionan *codecs* (funcionalidades) adicionales, conductores de base de datos, formatos de imagen, estilos y *widgets*, estos al igual que las bibliotecas pueden ser vendidos como productos en sí mismos. (17)

Ventajas de Qt

- Es un framework de código abierto completamente gratuito.
- Un programa desarrollado en él puede ser compilado en cualquier plataforma, sin necesidad de realizar cambios en el código y funcionará de igual manera que lo hizo en la plataforma que fue desarrollada anteriormente.
- Contiene una amplia gama de clases y herramientas orientada a objetos lo cual hace de la programación de interfaces gráficas una aventura placentera. (18)

Se utiliza Qt porque es el framework en el que se desarrolló la herramienta de administración de base de datos HABD a la que será integrado el plugin. También posee una Interfaz de Aplicación de Programación (API) muy fácil de utilizar, que permite una mejor productividad y calidad del trabajo, consta con diferentes herramientas muy potentes y eficientes para lograr un buen producto. Además de ser el framework escogido por el departamento y el proyecto.

1.4.5 Entorno de desarrollo integrado

Un entorno de desarrollo integrado, por sus siglas en inglés *Integrated Development Enviroment* (IDE) es un programa que está compuesto por un conjunto de herramientas hechas para los programadores. Este puede ser usado por varios lenguajes de programación o por uno en específico, proveen un ambiente de trabajo muy amigable.

El entorno de desarrollo utilizado es QtCreator, porque se ajusta perfectamente a las necesidades de la aplicación que se va a desarrollar, también es un editor de código avanzado.

QtCreator posee un depurador visual (*visual debugger*) para C++ consciente de la estructura de muchas clases de Qt, lo que aumenta la capacidad de mostrar los datos de Qt con claridad, además muestra la información en bruto procedente de SGBD de una manera clara y concisa.

QtCreator se centra en proporcionar características que ayudan a los nuevos usuarios de Qt aprender y comenzar a desarrollar rápidamente, también aumenta la productividad de los desarrolladores con experiencia en Qt.

- Editor de código con soporte para C++, QML y ECMAscript.
- Herramientas para la rápida navegación del código.
- Resaltado de sintaxis y auto-completado de código.
- Control estático de código y estilo a medida que se escribe.
- Ayuda sensitiva al contexto.
- Plegado de código (code folding).
- Además soporta los lenguajes: C#/.NET Languages (Mono), Python: PyQt y PySide, Ada, Pascal, Perl, PHP y Ruby.
- Posee también una GUI integrada y diseñador de formularios.
- Soporte para refactorización de código. (19)

Se utiliza como entorno de desarrollo integrado QtCreator, ya que es el IDE que hace posible la integración con la herramienta de administración de base de datos HABD, en el cual la misma fue desarrollada, además de ser el IDE de desarrollo predefinido por el proyecto.

1.4.6 Sistema de control de versiones

Un sistema de control de versiones es un sistema de gestión de archivos y directorios, cuya principal características es que mantiene la historia de los cambios y modificaciones que se han realizado sobre ellos a lo largo del tiempo. De esta forma, el sistema es capaz de “recordar” las versiones antiguas de los datos, lo que permite examinar el histórico de cambios o recuperar versiones anteriores de un fichero, incluso aunque haya sido borrado. (20)

Algunas operaciones más comunes que un cliente puede realizar dentro del sistema de control de versiones, se tienen:

- Mediante la operación *Check-out* se obtiene una copia local de la última revisión que se realizó, esta puede ser modificada por el cliente.
- Permite añadir, crear, modificar, borrar, etc.
- Con la operación commit se completan al repositorio los últimos cambios que se realizaron en la copia local.

El sistema de control de versión que se tiene implantado en el proyecto es el sistema de control subversión, el cual es está preparado para funcionar perfectamente en red y está distribuido bajo la licencia libre de tipo Apache.

Las principales características de Subversion son:

- Mantiene versiones no sólo de archivos, sino también de directorios.
- También se mantienen versiones de los metadatos asociados a los directorios.
- Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.
- Atomicidad de las actualizaciones. Una lista de cambios constituye una única transacción o actualización del repositorio. Esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.
- Posibilidad de elegir el protocolo de red. Además de un protocolo propio (svn), puede trabajar sobre http (o https).
- Soporte tanto de ficheros de texto como de binarios.
- Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos. (20)

Se utiliza como Sistema de Control de Versiones Subversion por las numerosas características y funcionalidades que este brinda, además es el sistema de control establecido por el proyecto.

1.4.7 Sistema Gestor de Base de Datos

Mundialmente existen diversos Gestores de Bases de Datos (GBD) como Oracle, MySQL, SQLite, Informix, etc. entre estos se destaca PostgreSQL ya que este es un Sistema de Gestión de Bases de Datos del modelo objeto-relación que ha sido desarrollado de varias formas desde la década de 1980. Sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto.

Características de PostgreSQL

El GBD PostgreSQL tienen numerosas características que lo diferencian de los restantes gestores, a continuación se muestran algunas de las más importantes:

- Permite tener un punto en el tiempo de recuperación, *tablespaces*, replicación asincrónica, transacciones anidadas, *backups* en caliente, un optimizador de consultas sofisticado y escribir por delante de registro para la tolerancia a fallos.
- Su administración se basa en usuarios y privilegios.
- Control de concurrencia multi-versión, lo que mejora sensiblemente las operaciones de bloqueo y transacciones en sistemas multi-usuario.
- Es compatible con conjuntos de caracteres internacionales, las codificaciones de caracteres multibyte, unicode, y es las configuraciones regionales para la ordenación, mayúsculas y minúsculas, y el formato.
- Es altamente escalable, tanto en la gran cantidad de datos que puede manejar y en el número de usuarios concurrentes que puede acomodar.
- Implementa un subconjunto extendido de los estándares SQL92 y SQL99.
- Permite el paso entre dos estados consistentes manteniendo la integridad de los datos.
- Reducción de la Redundancia.
- Permite la programación a usuarios avanzados.
- Funciones de compatibilidad para ayudar en la transición desde otros sistemas menos compatibles con SQL.
- Sistema de tipos de datos extensible para proveer tipos de datos definidos por el usuario, y rápido desarrollo de nuevos tipos. (21)

Ventajas de PostgreSQL

PostgreSQL ofrece muchas ventajas respecto a otros sistemas de gestores de bases de datos, algunas de estas son:

- **Instalación Ilimitada:** Con PostgreSQL, nadie puede demandarlo por violar acuerdos de licencia, puesto que no hay costo asociado a la licencia del software.
- **Ahorros considerables en costos de operación:** Ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que otros productos, conservando todas las características, estabilidad y rendimiento.

- **Multiplataforma:** PostgreSQL está disponible en casi cualquier Unix (34 plataformas en la última versión estable), y ahora en versión nativa para Windows.
- **Estabilidad y Confiabilidad Legendarias:** Es extremadamente común que compañías reporten que PostgreSQL nunca ha presentado caídas en varios años de operación de alta actividad. Ni una sola vez. Simplemente funciona.
- **Extensible:** El código fuente está disponible para todos sin costo.
- **Diseñado para ambientes de alto volumen:** PostgreSQL usa una estrategia de almacenamiento de filas llamada MVCC para conseguir una mejor respuesta en ambientes de grandes volúmenes. (21)

Se escogió como Sistema de Gestor de Base de Datos, PostgreSQL por sus numerosas ventajas vistas anteriormente y principalmente porque el plugin que se desarrolla está hecho para generar las funciones de dicho gestor.

1.5 Conclusiones del capítulo.

Durante la realización del presente capítulo se analizaron los principales conceptos sobre bases de datos, herramientas de administración, gestor de base de datos, plugins y lenguajes de programación. Se seleccionó XP como metodología de desarrollo ya que es la que más se adapta a las necesidades del proyecto. También se identificaron las principales herramientas y tecnologías a usar para el desarrollo de este tipo de aplicación, donde se eligió C++ como lenguaje de programación, Qt como framework de desarrollo, UML como lenguaje de modelado, Visual Paradigm como herramienta CASE, Subversion como sistema de control de versiones, QtCreator como entorno de desarrollo y PostgreSQL como gestor de base de datos.

CAPÍTULO 2: Características del sistema propuesto.

Introducción

En este capítulo se puntualiza la propuesta de solución al problema de la investigación planteado, estableciendo así, las funcionalidades más importantes con las que contará el plugin. Además se muestran explícitamente las historias de usuarios y los requisitos no funcionales, también serán generados los artefactos que propone la metodología XP correspondientes a la etapa de diseño y los diferentes patrones de arquitectura y de diseño que serán utilizados.

2.1 Descripción de la solución propuesta

Para darle respuesta al problema de la investigación antes mostrado, se propone el desarrollo de un plugin para integrarlo a la herramienta de administración de base de datos HABD que genere funciones automáticas CRUD en las base de datos de PostgreSQL.

El plugin contendrá el mismo nivel de privilegio que contiene la aplicación a la que se integrará. Permitirá al usuario seleccionar la tabla a la que desee generarle las funciones CRUD, además de que brindará la posibilidad de seleccionar las funciones que el usuario desee generar, obtener una vista previa de las mismas y eliminarlas después de generadas. También posibilitará la visualización de todas las funciones que han sido generadas para todas las tablas.

2.2 Modelo de dominio

La metodología XP no define un modo específico para definir el negocio, es así que se realiza el llamado modelo de dominio, ya que mediante este se realiza el proceso de comprensión y entendimiento de la manera más cómoda para aquellas personas que van a trabajar con la aplicación.

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

Las clases del dominio aparecen en tres formas típicas:

- Objetos del negocio que representan cosas que se manipulan en el negocio.
- Objetos del mundo real y conceptos de los que el sistema debe hacer un seguimiento.

- Sucesos que ocurrirán o han ocurrido. (22)

A continuación se muestra el modelo de dominio conformado, el cual muestra la relación de las clases conceptuales relacionadas con el plugin.

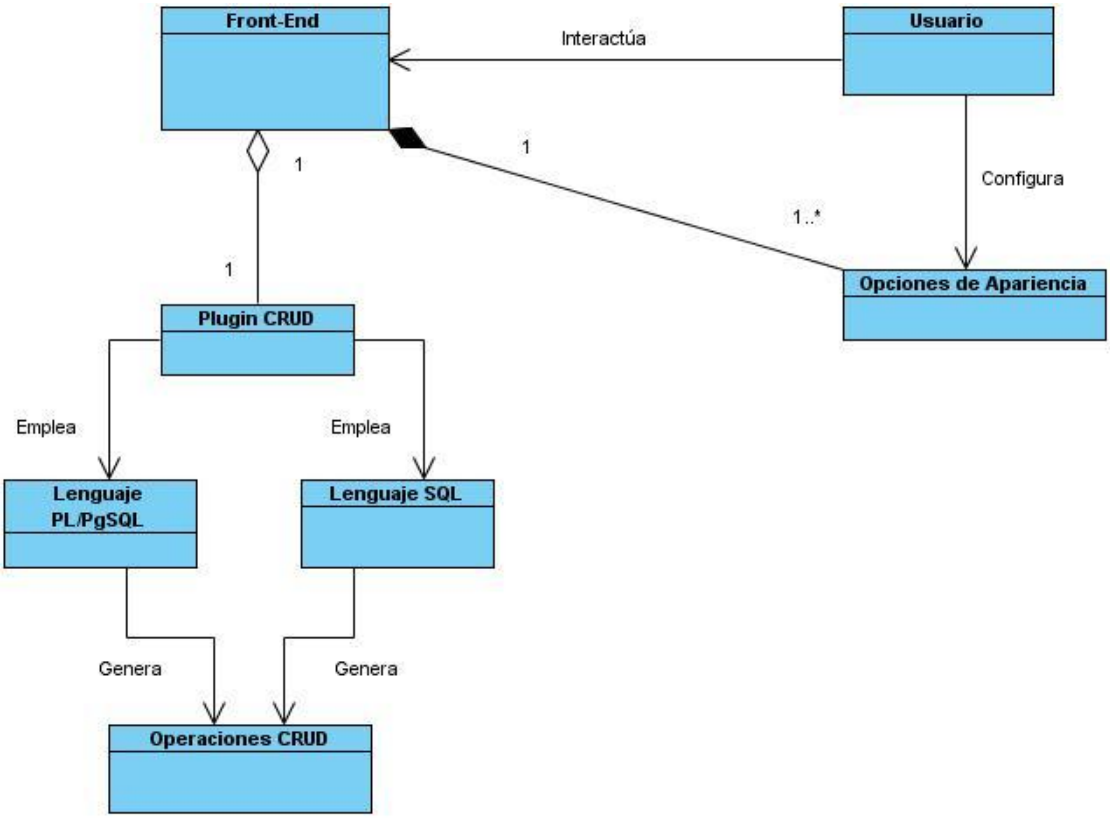


Figura 1: Modelo de Dominio

Usuario: es la persona que interactúa con la herramienta.

Front-End: caparazón que será capaz de cargar y manipular los plugins que sean adicionados por el usuario.

Opciones de Apariencia: Esta clase permite obtener una navegabilidad flexible y de fácil comprensión, tratando así de conseguir rápidamente que se cumpla el objetivo principal, el cual es tratar de que el cliente se sienta a gusto con la aplicación. Además de que su diseño gráfico esté de acorde con las pautas de diseño de la Universidad.

Plugin CRUD: es un módulo que se agrega para añadir nuevas características y funcionalidades al Front-End.

Lenguaje PL/pgSQL: Con el empleo del lenguaje PL/pgSQL se generan las operaciones CRUD: insertar, seleccionar, seleccionar condicional, seleccionar general, actualizar, eliminar y eliminar truncar. Además de la función cursor para optimizar la función seleccionar.

Lenguaje SQL: Con el empleo del lenguaje SQL se genera la función CRUD seleccionar.

Operaciones CRUD-PG: esta permiten generar las operaciones de escritura, lectura, actualizar y eliminar, por sus siglas en inglés CRUD en los lenguajes de programación del lado del servidor PL/pgSQL y SQL.

2.3 Historias de usuarios

Las Historias de Usuarios (HU) son la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. (23)

En la metodología XP las historias de usuarios sustituyen a los documentos de especificación funcional y a los casos de usos. Se usan para elaborar estimaciones de tiempo de desarrollo de la aplicación. Tienen la facilidad de permitir respuestas de forma rápida a los requisitos cambiantes que puedan surgir durante el desarrollo del software, gracias a esto no se derrocha tiempo en la gestión de los mismos y se pueden administrar cómodamente los requisitos de los usuarios.

Las historias de usuarios deben tener un tiempo de desarrollo de programación estimado entre una y tres semanas, si esta estimación es mayor de tres semanas, esta historia debe ser dividida en dos o más, en el caso de que sea menos de una semana debe ser combinada con otra historia.

Se identificaron y redactaron diez historias de usuarios donde se describen las funcionalidades con que deberá contar la aplicación en su versión final. A continuación se mostrará como ejemplo la historia de usuario Generar función Insertar en el lenguaje PL/pgSQL, las demás se encuentran en el expediente de proyecto.

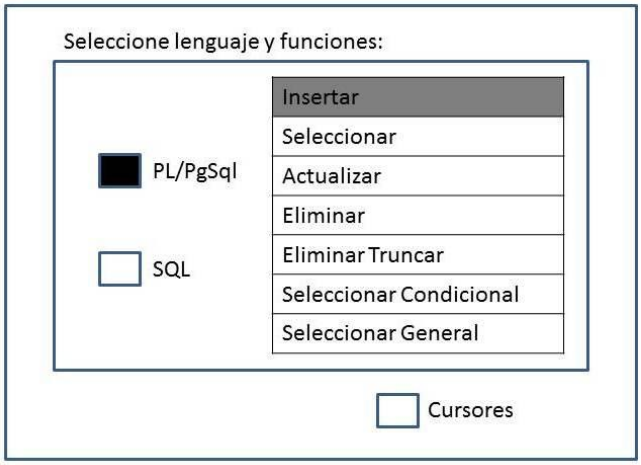
<table>
<tr><th colspan="2">Historia de Usuario</th></tr>
<tr><td>Número: 1</td><td>Nombre de la Historia de Usuario: Generar función Insertar en el lenguaje PL/pgSQL.</td></tr>
<tr><td colspan="2">Cantidad de modificaciones a la Historia de Usuario: 1</td></tr>
<tr><td>Usuario: usuario registrado en la base de datos</td><td>Iteración asignada: 1era iteración.</td></tr>
<tr><td>Prioridad en negocio: Muy Alta</td><td>Puntos estimados: 2.0</td></tr>
<tr><td>Riesgo en desarrollo: Muy Alto</td><td>Puntos reales: 2 semanas.</td></tr>
<tr><td colspan="2">Descripción: El usuario selecciona la tabla, posteriormente precederá a escoger el lenguaje y la función Insertar que se generará para dicha tabla, al concluir esto debe presionar el botón Generar para que dé cumplimiento a la funcionalidad.</td></tr>
<tr><td colspan="2">Observaciones: El usuario registrado en la base de datos debe seleccionar una tabla.</td></tr>
<tr><td colspan="2">Prototipo de interfaces:</td></tr>
<tr><td colspan="2"> 
</td></tr>
</table>

Tabla 1: HU: Generar función Insertar en el lenguaje PL/pgSQL

2.4 Lista de reserva del producto

La Lista de Reserva del Producto (LRP) es una tabla, la cual contiene el listado de las historias de usuarios, el cual se encuentran ordenado por su prioridad de implementación, las cuales se clasifican en Muy Alta, Alta, Media y Baja, en esta última aparecen los requisitos de menor complejidad, además de los requisitos no funcionales del sistema a desarrollar. También aparece la estimación del tiempo de cada uno de ellos y su implementación por semanas, además del rol que hizo dicha estimación.

Esta lista de reserva del producto recoge los requisitos correspondientes al plugin a implementar, los cuales demuestran la capacidad del mismo para poder ser integrado a la herramienta de administración HABD.

Ítem *	Descripción	Estimación	Estimado por
Prioridad: Muy Alta			
1	Generar función Insertar en el lenguaje PL/pgSQL.	2	Analista
2	Generar función Seleccionar en los lenguajes PL/pgSQL y SQL	2	Analista
3	Generar función Actualizar en el lenguaje PL/pgSQL.	2	Analista
4	Generar función Eliminar en el lenguaje PL/pgSQL.	2	Analista
5	Generar función Eliminar Truncar en el lenguaje PL/pgSQL.	2	Analista
6	Generar función Seleccionar Condicional en el lenguaje PL/pgSQL.	3	Analista
7	Generar función Seleccionar General en el lenguaje PL/pgSQL.	2	Analista
Prioridad: Alta			
8	Generar función Cursor para Select.	1	Analista

9	Eliminar funciones generadas.	1	Analista
Prioridad: Media			
10	Generar Ayuda.	1	Analista
Prioridad: Baja			
1	**Apariencia o interfaz externa:** Interfaz intuitiva, amigable, organizada, con una navegabilidad flexible y de fácil comprensión, de forma que el objetivo del sistema para con los clientes, se pueda conseguir rápidamente. Debe estar diseñada para verse en cualquier resolución igual o inferior a 1024x768.		
2	**Facilidad de uso:** Para utilizar el sistema es necesario poseer conocimientos elementales de computación, así como de base de datos.		
3	**Soporte**: Se dispondrá de una página web en la cual están descritas todas las funcionalidades del sistema, permitiendo al usuario tener un mejor conocimiento y manejo de la aplicación.		
4	**Software**: Sistema Operativo: Multiplataforma. Librerías QT. Servidor de Bases de datos: SGBD PostgreSQL 8.4.x o versiones superiores.		
5	**Hardware:** Se necesita 128 MB de memoria RAM como mínimo, 1GB de espacio libre en el disco duro para su instalación y el micro a 300 MHz.		
6	**Seguridad:** Garantizar que cada conexión que se realice al servidor de base de datos solicite autenticación. Permitir que las funcionalidades del sistema se muestren de acuerdo al tipo de usuario que esté activo. Permitir hacer salvas y restauración cuando se caiga la conexión con el servidor.		

Tabla 2: Lista de Reserva del Producto

2.5 Tareas de la ingeniería

Las Tareas de la Ingeniería (TI) se usan para describir las tareas que se realizan sobre el proyecto. Estas pueden ser de: desarrollo, corrección, mejora, etc. Las TI tienen relación con una historia de usuario, aunque de una historia de usuario se puede derivar más de una tarea de la ingeniería; en estas se especifica la fecha de inicio y fin de la tarea, se nombra al programador responsable de

cumplirla, también aparece el tiempo estimado que se demorará en realizar las tareas, además se realiza un pequeña descripción de lo que se tratará de hacer en la tarea.

A continuación se mostrará un ejemplo de la tarea de ingeniería, esta es correspondiente con la historia de usuario mostrada anteriormente, el resto se encuentra en el expediente de proyecto.

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario: 1**
Nombre Tarea: Generar función Insertar en el lenguaje PL/pgSQL.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2 semanas
Fecha Inicio: 02/12/2011	**Fecha Fin: 16/12/2011**
Programador Responsable: Yaciel López Pérez	
Descripción: Recibirá como parámetro la tabla y su esquema correspondiente en que se generará la función Insert que posibilitará la inserción de datos posteriormente.	

Tabla 3: Tarea de la Ingeniería: Generar función Insertar

2.6 Plan de iteraciones

El plan de iteraciones está compuesto por iteraciones que no superan a las tres semanas. Este posee una arquitectura del sistema que puede ser establecida en la primera iteración y utilizada durante la trayectoria del proyecto. Para que se efectúe la creación de esta arquitectura se deben escoger las historias correctas que la exijan, pero como el cliente es el que decide que o cuales historias se implementarán en cada iteración, muchas veces la implementación de esta arquitectura no es posible. El sistema estará listo para su utilización al completarse la última iteración.

En el transcurso de la elaboración del plan de iteraciones se deben tomar en cuenta elementos, tales como:

- Velocidad del proyecto.
- Pruebas de aceptación no superadas de la iteración precedente.
- Tareas no terminadas de la iteración precedente.
- Historias de usuarios no abordadas. (8)

Cada programador es responsable del número de tareas de programación que le sean asignadas, aunque todas son llevadas a cabo por parejas de programadores.

Ya identificadas las historias de usuarios se establecieron dos iteraciones para el desarrollo de la aplicación. Para realizar la selección de las historias a implementar se tuvo en cuenta la prioridad que estas representan para el negocio y la relación de sus funcionalidades.

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
Iteración 1	En esta iteración se implementarán las historias de usuarios desde la uno hasta la siete, estas se encargan de realizar las operaciones básicas CRUD. Estas historias son muy imprescindible para la aplicación, ya que sin estas no funciona el plugin y por tal motivo se le asignó la prioridad muy alta.	HU #1 HU #2 HU #3 HU #4 HU #5 HU #6 HU #7	15 semanas
Iteración 2	En esta otra iteración se desarrollarán las restantes historias, es decir la ocho, nuevo y diez, estas historias se encargan de generar las funciones con cursores, deshacer las operaciones generadas si así lo desea y facilitar la ayuda de la aplicación, para que los usuarios que interactúen con ella tengan un mejor conocimiento de la misma, estas tienen como prioridad alta, alta y media respectivamente.	HU #8 HU #9 HU #10	3 semanas

Tabla 4: Plan de iteraciones

2.7 Modelo de diseño

La metodología XP propone que, mientras no se afecten las funcionalidades especificadas por el cliente, el diseño de la aplicación debe ser lo más natural posible. Cualquier código extra o cualquier tipo de complejidad deben ser eliminadas del sistema. Esta metodología hace mayor énfasis en los diseños simples y claros.

2.7.1 Diagrama de clases

Para una descripción más puntualizada del plugin se escogió un artefacto definido en la metodología de RUP (*Rational Unified Process*) tal como es el caso del diagrama de clases. Además se incluye el rol de analista por el trabajo que tiene en la producción de un software de calidad y que XP no posee. A continuación se muestra el diagrama de clases perteneciente a la aplicación.

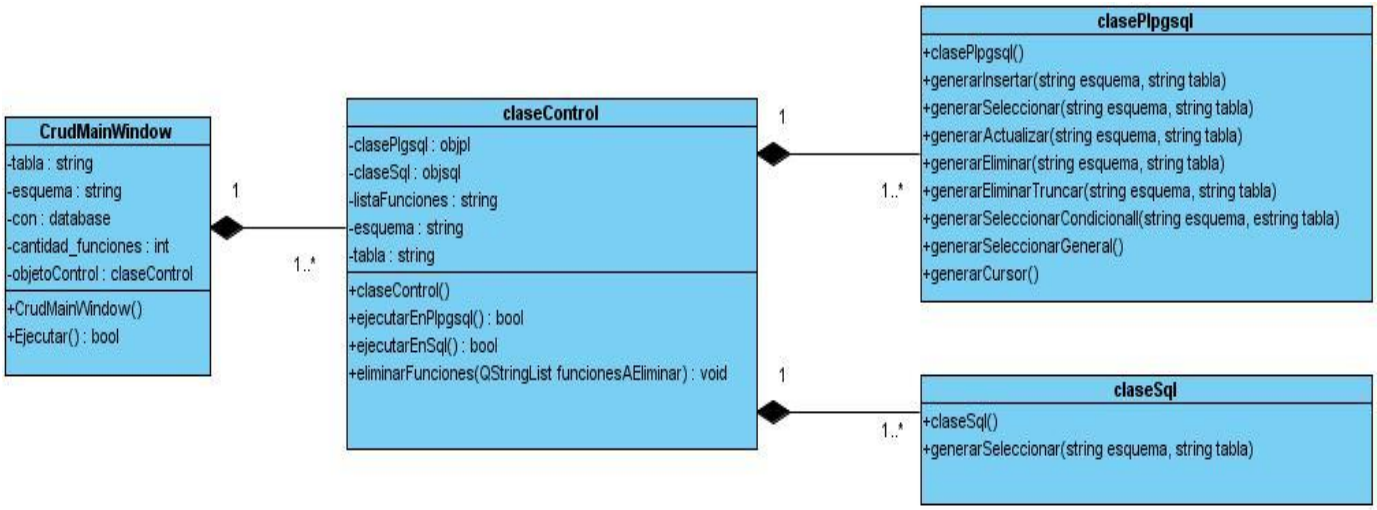


Figura 2: Diagrama de clases

CrudMainWindow: esta se encarga de proveer los datos de entrada que el usuario debe enviar a la clase controladora, cuando esta lo solicite.

Control: Esta clase es la encargada de controlar el sistema, además de obtener la conexión del HABD y ejecuta en el servidor de base de datos las consultas que devuelven los métodos asociados a las clases clase_pl_SQL y clase_SQL.

PL/pgSQL: Genera las consultas correspondientes a las funciones implementadas en el lenguaje PL/pgSQL.

SQL: Genera la consulta correspondiente a la función seleccionar implementada en el lenguaje SQL.

2.7.2 Tarjetas Clases-Responsabilidades-Colaboración

Las tarjetas CRC (*Clases-Responsabilidades-Colaboración*) fueron presentadas por Beck y Cunningham en 1989 para la enseñanza de la Programación Orientada a Objetos (POO). Estas son una técnica de modelado orientado a objetos que permite identificar las clases y sus responsabilidades, además de servir de herramienta de ayuda al refinamiento de clases.

Estas tarjetas se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores. En ellas se expresa el diseño del sistema, la sencillez de esta tarjeta hace que del diseño una tarea fácil.

- **Clase**: Cualquier persona, cosa, evento, concepto, pantalla o reporte. Además consiste en crear una clase para cada tarjeta.
- **Responsabilidades**: son las cosas que conocen y las que realizan, sus atributos y métodos, es decir son todos los servicios que proporciona para todos los contratos que soporta la lista de servicios que una instancia de una clase puede pedir a una instancia de otra, también incluyen el conocimiento que tiene la clase y las acciones que puede realizar un objeto de la misma.
- **Colaboraciones**: son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades, además representan las peticiones por parte de un cliente a los servidores. (24)

La característica más sobresaliente de las tarjetas CRC es su simpleza y adaptabilidad. Su principal beneficio es que alientan la disertación animada entre los desarrolladores. Son especialmente eficaces cuando se está en medio de un caso de uso para ver cómo lo van a implementar las clases. Los desarrolladores escogen tarjetas a medida que cada clase colabora en el caso de uso. Conforme se van formando ideas sobre las responsabilidades, se pueden escribir en las tarjetas. Es importante pensar en las responsabilidades, ya que evita pensar en las clases como simples depositarias de datos, y ayuda a que el equipo se centre en comprender el comportamiento de alto nivel de cada clase. (24)

Durante el proceso de diseño de la aplicación se elaboraron un total de diez tarjetas CRC. A continuación se muestra una de las más significativas:

Tarjeta CRC	
Clase: Generar función Insertar en el lenguaje PL/pgSQL.	
Responsabilidades	**Colaboraciones**
Generar función Insertar en el lenguaje PL/pgSQL.	

Tabla 5: Tarjeta CRC: Generar función Insertar

2.8 Patrón arquitectónico

Los patrones arquitectónicos aconsejan la arquitectura global que debe seguir una aplicación. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes.

Para lograr una arquitectura más robusta y consistente de la aplicación se utilizó el patrón Modelo-Vista-Controlador (MVC), el cual es un patrón de arquitectura para las aplicaciones de software, consiste en separar la lógica de negocio de la interfaz de usuario. Además divide la aplicación en tres capas: el modelo, la vista y el controlador, incrementando la reutilización y flexibilidad de la misma.

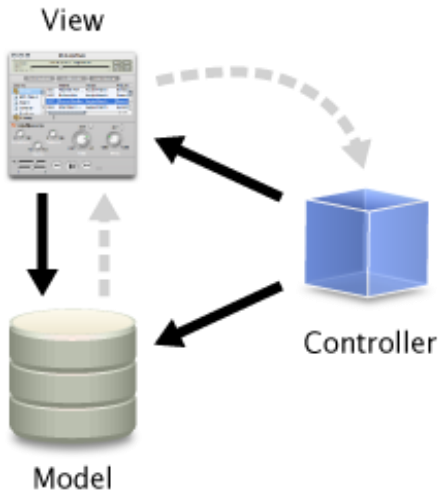


Figura 3: Patrón Modelo-Vista-Controlador

Modelo

El modelo son las clases que representan la información específica con la cual el sistema maneja los datos, es decir su lógica de negocio.

Vista

Las vistas se encargan de mostrar toda la información que se encuentra contenida en el modelo al usuario, es decir mediante esta se generan las respuestas visuales. Además una vista está asociada a un modelo, aunque pueden existir varias vistas asociadas a un mismo modelo.

Controlador

El controlador es el encargado de procesar las interacciones del usuario y realiza los cambios adecuados en el modelo o en la vista. Además se encarga de dirigir todo el flujo del control de la aplicación mediante mensajes externos.

A continuación se muestra cómo se evidencia este patrón arquitectónico dentro de la aplicación:

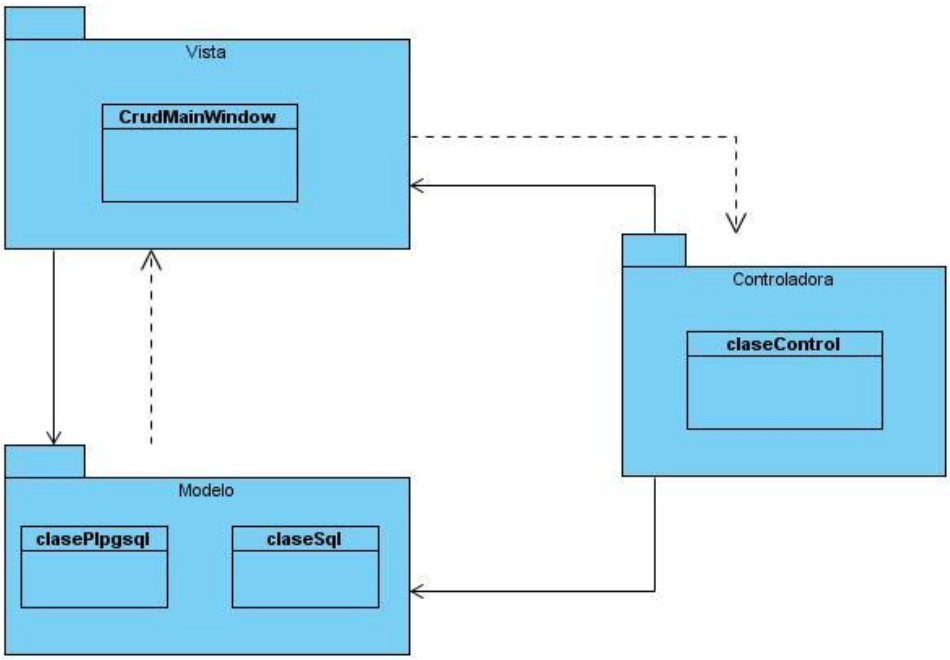


Figura 4: Patrón Arquitectónico Modelo-Vista-Controlador

2.9 Patrones de diseños

Los patrones de diseño son el soporte de las soluciones a problemas comunes en el desarrollo de un software y de otros contornos referentes al diseño de interacción o interfaces. A su vez brindan una solución ya aprobada y documentada de problemas de desarrollo del software que están sometidos a contextos equivalentes.

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. (25)

Para obtener un mejor diseño de la aplicación se tuvieron en cuenta varios patrones de diseño, dentro de los cuales se encuentran los patrones GRASP (patrones generales de software para asignar responsabilidades), los cuales describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones, se utilizaron los siguientes:

- **Alta Cohesión:** Cada elemento debe realizar una labor única dentro del sistema, es decir que no debe estar desempeñada por el resto de los elementos. Un ejemplo donde se evidencia este patrón en la aplicación es mediante la clase clasePlpgsql, que es la única encargada de generar funciones en el lenguaje PL/pgSQL.
- **Bajo Acoplamiento:** Propone que el nivel de acoplamiento entre los objetos sea bajo. El principal objetivo de este patrón es tratar que las clases estén lo menos relacionadas, de tal forma que si sufre alguna modificación se tenga la mínima repercusión posible en las clases restantes. Este patrón se evidencia en la aplicación mediante la independencia que existe entre las clases clasePlpgsql y claseSql, que cada una es la encargada de realizar sus propias funcionalidades sin depender de la otra, lo que permite una mayor reutilización del código.
- **Experto:** Asigna la responsabilidad de realizar una tarea determinada a aquel objeto que tiene la información necesaria para ello. El cumplimiento de una responsabilidad solicita a menudo información distribuida en varias clases de objetos. Esto se evidencia en las clases claseControl y clasePlpgsql, en este caso la clase claseControl es la encargada de asignarle la responsabilidad a la clasePlpgsql de generar la función que el usuario haya solicitado.
- **Controlador:** Asigna la responsabilidad de controlar el flujo de sucesos del sistema, a las clases más específicas. Este patrón no realiza estas actividades, las encomienda a otras clases con las que mantiene un modelo de alta cohesión. En la aplicación se refleja este patrón de

diseño en la clase claseControl, la cual se encarga de controlar el flujo de información dentro de la misma.

- **Creador:** Este patrón asigna responsabilidades relacionadas con la creación de objetos, escogiendo como objeto creador aquél que necesite conectarse al objeto creado. Este patrón es demostrado en la aplicación mediante la creación de un objeto o instancia de la clase clasePlpgsql en la clase controladora claseControl.

2.10 Estándares de codificación

En la metodología XP la codificación debe realizarse siguiendo estándares de codificación ya creados, ya que trabajar bajo estos estándares mantiene el código consistente y facilita su comprensión y escalabilidad, lo que permite que cualquiera persona del equipo pueda modificar cualquier parte del código. Un estándar de codificación es necesario para soportar otras prácticas de la XP.

El estándar de codificación que se utiliza es el definido anteriormente para realizar la aplicación HABD, es utilizado porque mediante él se tiene un mejor entendimiento para aquellas personas que deseen trabajar con la aplicación y permite optimizar el código y tenerlo más organizado, para que de esta forma permita realizar cambios futuros a los programadores que no tengan conocimiento de la aplicación. A continuación se muestra como se evidencia el estándar de codificación en la aplicación:

Identación

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

```
QString consulta = objetoPlpgsql->generaTruncate(esquemaDeLaTabla);
QSqlQuery ejecutar(consulta, conexionHabd);
verificar = true;
```

Figura 5: Ejemplo de estándar de identación

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. Realice 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 cinco espacios.

```
QString completo = "CREATE OR REPLACE FUNCTION ""\""+esquema+"\""+".";
completo += tabla+"_crud_insertar("+tipos+") \n";
```

Figura 6: Ejemplo de estándar de longitud de línea

Comentarios

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

```
QSqlDatabase con; //Variable que obtiene la conexión del HABD.
```

Figura 7: Ejemplo de estándar de comentarios

Declaración de variables

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

```
QString esquemaDeLaTabla; //Variable que almacena el esquema correspondiente
                          //a la tabla seleccionada.

QStringList funcionesSeleccionadas; //Variable que almacena las funciones
                                    //seleccionadas por el usuario.

QString tablaSeleccionada; //Variable que almacena la tabal seleccionada.
```

Figura 8: Ejemplo de estándar de declaración de variables

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 el de las variables.

Identificadores

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

```
clasePlpgsql *objetoPlpgsql;
```

Figura 9: Ejemplo de estándar de identificadores

Sentencias

Sentencias Simples

Cada línea debe contener como máximo una sentencia. Debe poner un punto y coma ";" al final de cada sentencia simple. Tenga en cuenta que una sentencia de asignación puede resultar en la asignación de una función o de un objeto como literal y en todos los casos como sentencia de asignación debe estar finalizada con un punto y coma.

```
listaFuncionesQueFaltan = funcionesQueFaltan;
```

Figura 10: Ejemplo de estándar de sentencia simple

Sentencias Compuestas

Las sentencias compuestas son aquellas sentencias que contienen una lista de sentencias encerradas entre llaves:

- Las sentencias encerradas deben ser identada a tres espacios.
- La llave que inicia la lista de sentencias debe estar al final de línea de la sentencia compuesta.
- La llave que termina la lista de sentencias debe estar al comienzo de una línea y guardar la misma identación que la sentencia compuesta en correspondencia con la llave que inicia.
- Las llaves siempre serán usadas para listar todas las sentencias, aunque se trate de una sola, cuando son parte de una estructura de control como if o for. Ello facilita agregar nuevas sentencias sin la introducción accidental de errores.

```
if(ui->checkBox_cursor->isEnabled() && ui->checkBox_cursor->isChecked()){
    ui->listWidget_funciones->addItem(tr("Seleccionar"));
} else {
    if(ui->checkBox_sql->isChecked()){
        ui->checkBox_sql->setChecked(false);
        llenarListaFuncionesQueFaltan(listaFuncionesQueFaltan);
    }
}
```

Figura 11: Ejemplo de estándar de sentencias compuestas

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.

```
return camposOrganizados;
```

Figura 12: Ejemplo de estándar de sentencia return

Sentencia if

```
if(stringDeFuncionesQueEstan.contains("_seleccionar_estandar")==false){

    listaFuncionesQueFaltan.append(("Seleccionar"));
}
```

Figura 13: Ejemplo de estándar de sentencia if

Espacios en blanco

Las líneas en blanco facilitan la lectura determinando secciones de código lógicamente relacionada. 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 a la lista de parámetros. Ello ayuda a distinguir entre palabras reservadas y llamadas a funciones.
- Para cualquier operador binario excepto el punto “.”, el paréntesis que abre “(” y el corchete que abre “[” todos deben ser separados por un espacio entre operandos y operador.
- No se debe utilizar el espacio para separar un operador unario de su operando 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 “,”.

2.11 Interfaces de la aplicación.

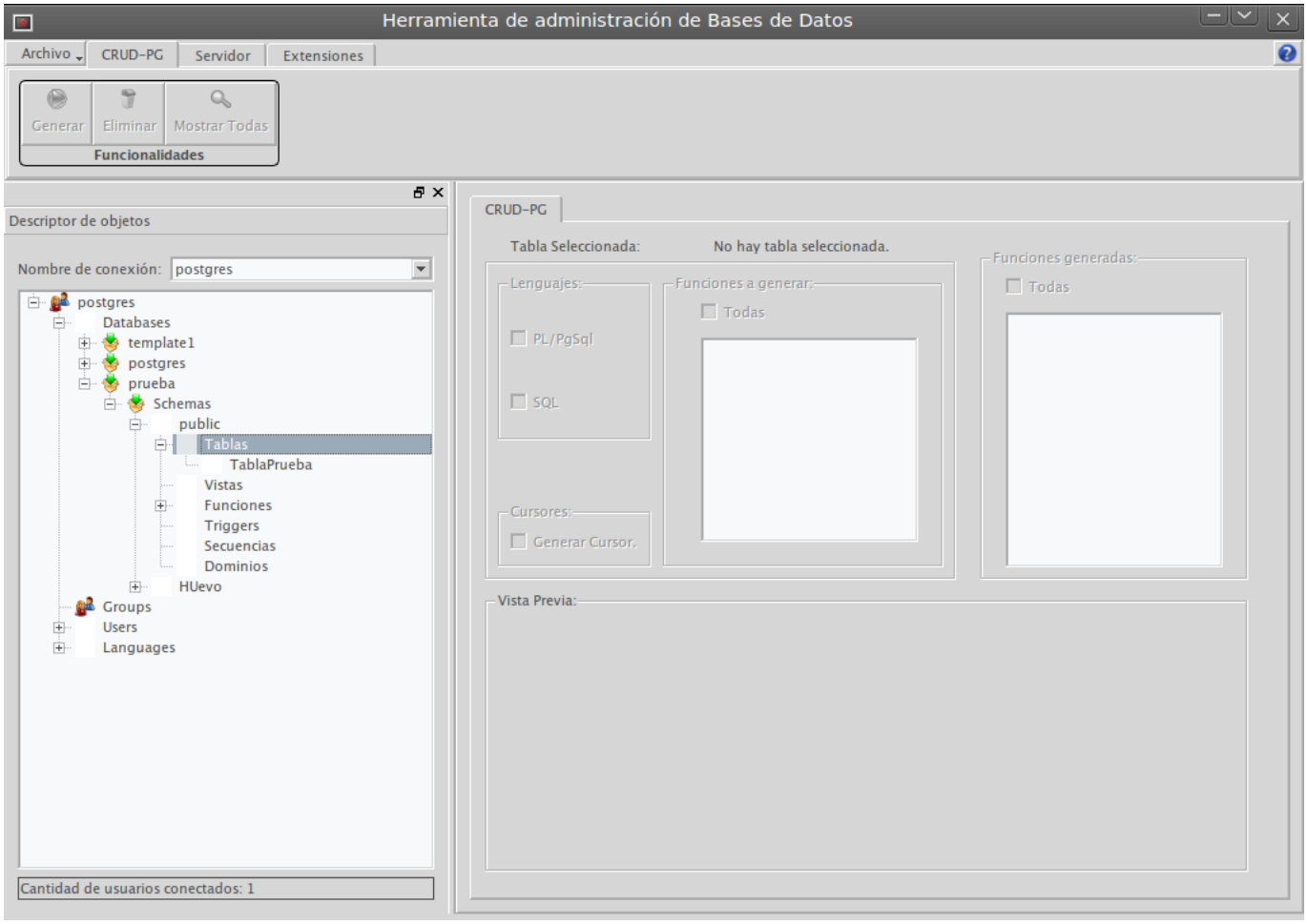


Figura 14: Interfaz de la aplicación (Plugin desactivado)

Esta es la interfaz principal con la que interactúa el usuario. En ella se muestra como el plugin se encuentra desactivado, este se activa al usuario seleccionar una tabla de la base de datos ver (Figura 15) y le proporciona la información acerca del estado de las funciones generadas y las que faltan por generarse para dicha tabla.

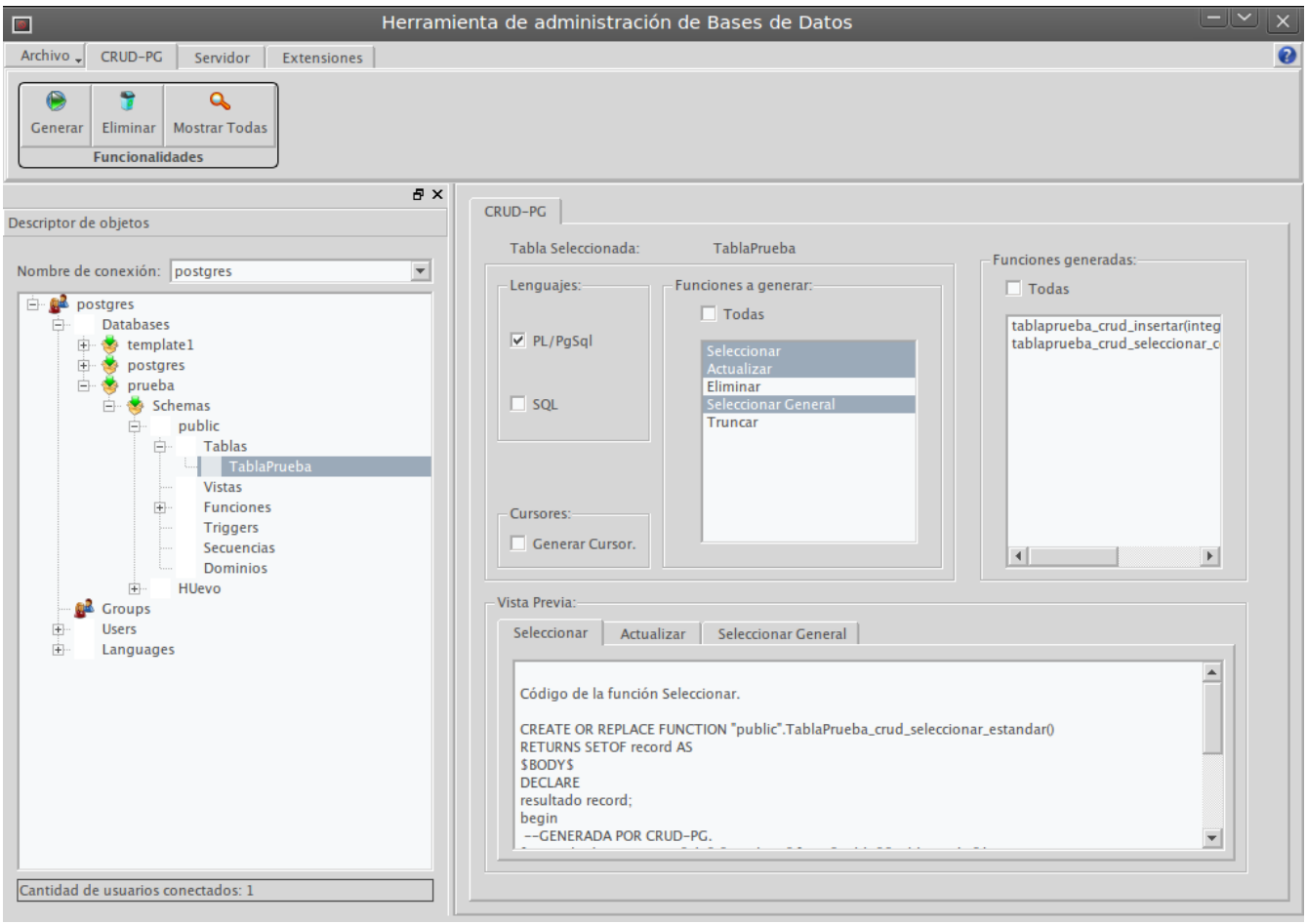


Figura 15: Interfaz de la aplicación (Plugin activado)

El usuario debe seleccionar una de las casillas del recuadro Lenguajes o bien si desea generar los cursores para optimizar la función seleccionar en el lenguaje PL/pgSQL. Posteriormente debe seleccionar las funciones que desee generar, al marcarlas en el recuadro que dice Vista Previa, se muestra el código de las mismas de forma independiente, finalmente se presiona el botón Generar y se muestra un mensaje (ver Figura 16).

Para eliminar una o más funciones el usuario debe seleccionarlas del recuadro Funciones Generadas y presionar el botón Eliminar ver (Figura 17). Para mostrar todas las funciones generadas por el plugin el usuario debe presionar el botón Mostrar Todas ver (Figura 18).

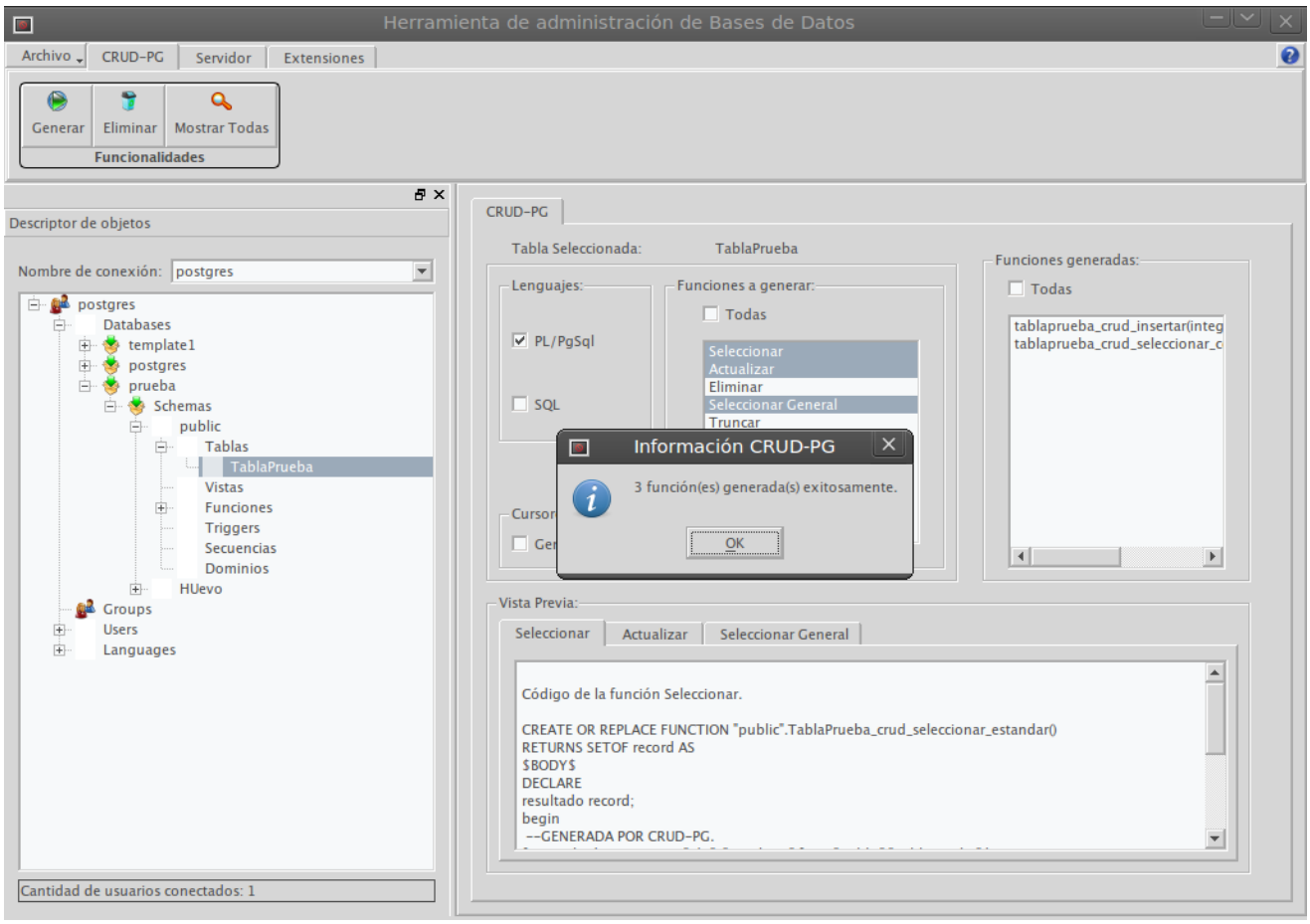


Figura 16: Interfaz de la aplicación (Funciones generadas)

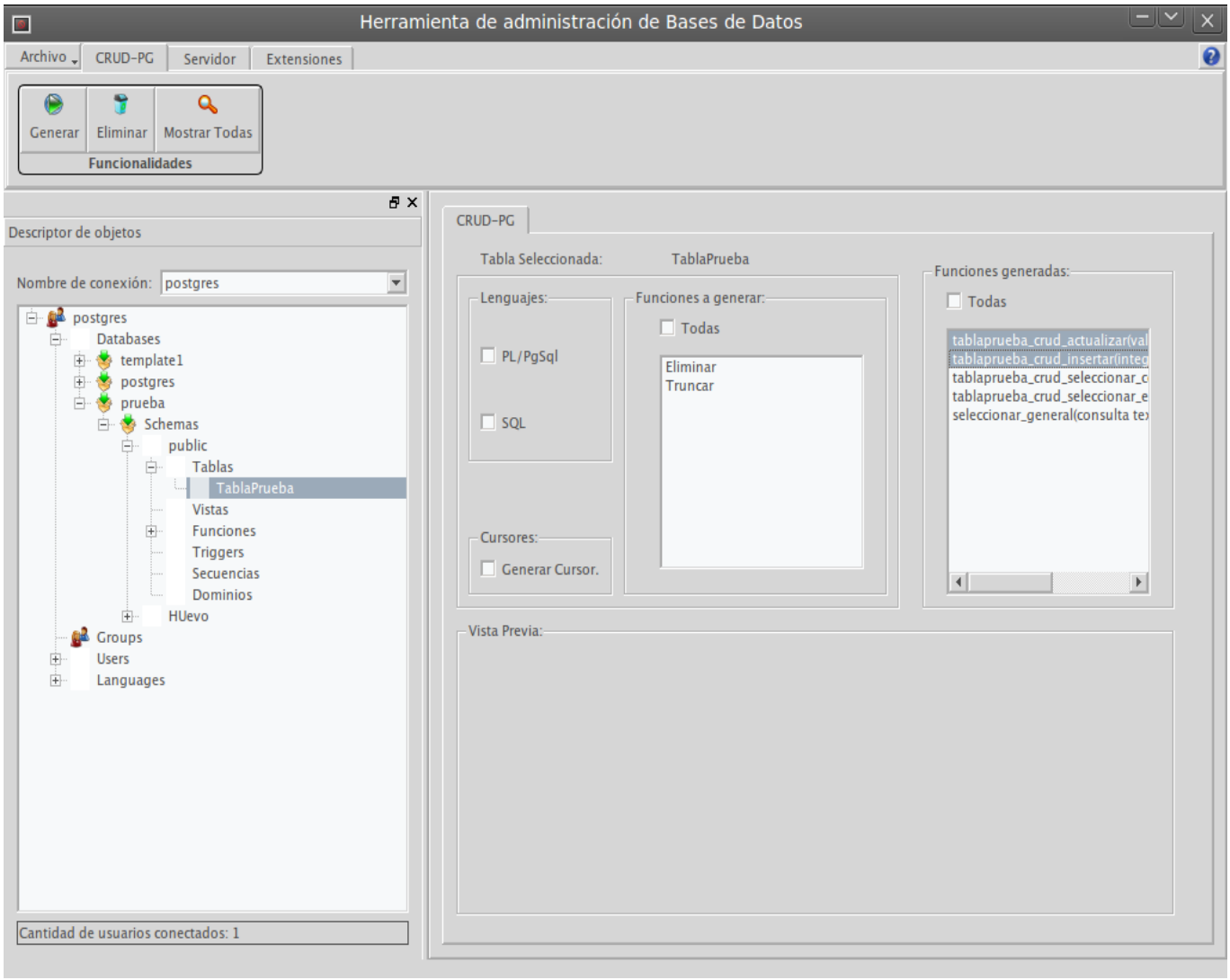


Figura 17: Interfaz de la aplicación (Funciones a eliminar)

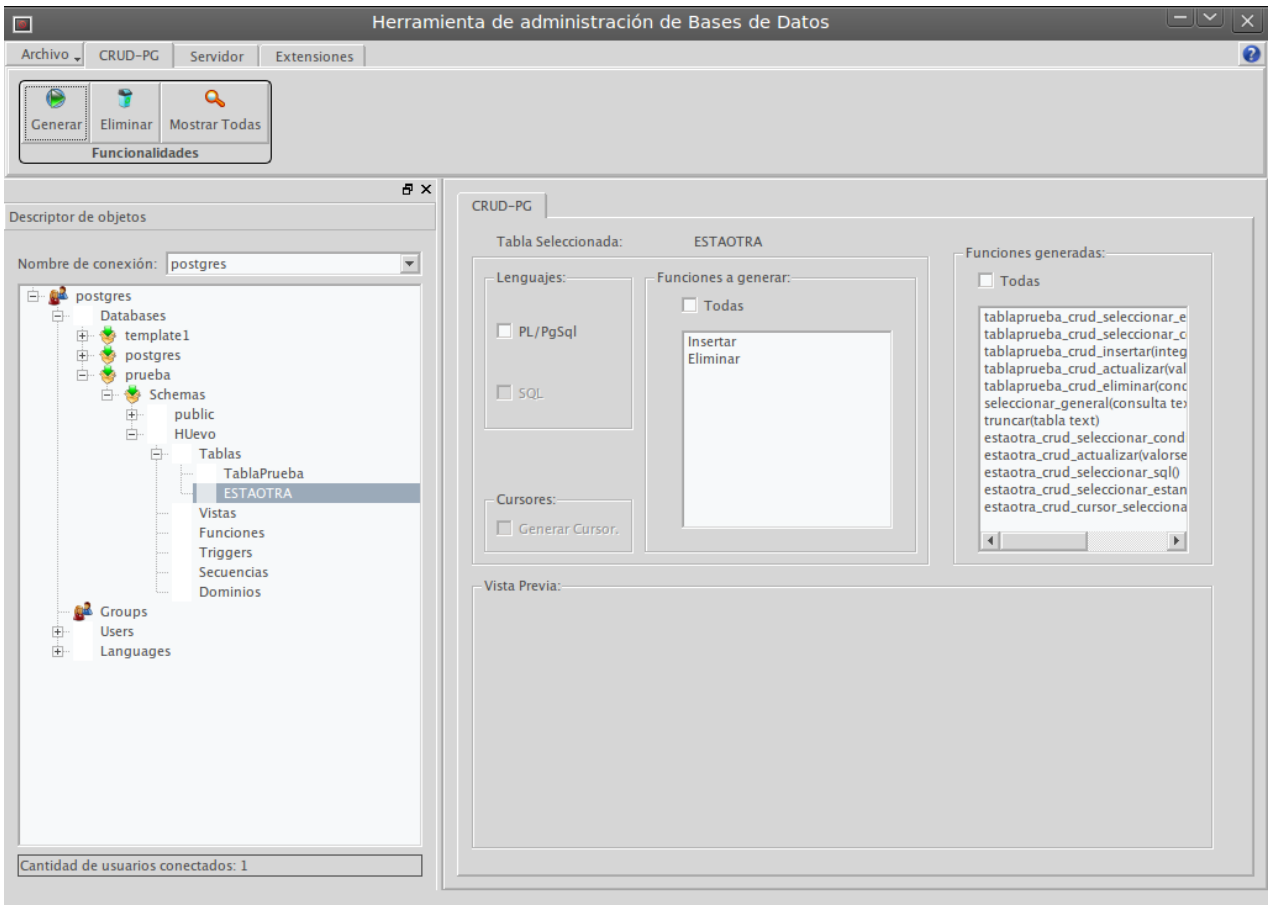


Figura 18: Interfaz de la aplicación (Funciones mostradas todas)

2.12 Conclusiones del capítulo

En el presente capítulo se describió la solución propuesta para el problema de la investigación, se identificaron los requisitos no funcionales y las diez historias de usuario. Además se generaron los artefactos que concibe la metodología XP como son: la lista de reserva del producto, las tareas de la ingeniería y el plan de iteraciones, los cuales son muy importantes ya que establecerán el avance de la aplicación. También se creó el modelo de dominio, para así lograr un mejor entendimiento del desarrollo de la aplicación. Se nombraron los diferentes patrones de diseño y arquitectónicos que se utilizarán en la implementación del software. Fueron generadas las tarjetas CRC donde en ellas se reflejan las diferentes clases que intervendrán en la aplicación con sus responsabilidades.

CAPÍTULO 3: Validación del sistema propuesto.

Introducción

En el capítulo se describen los enfoques de diseños de casos de pruebas que existen para probar un software, además de explicar las distintas pruebas que realiza la metodología escogida. El método que se utiliza es el de caja negra y su técnica es la de partición equivalente. También se encuentran los casos de pruebas realizados a las distintas historias de usuarios, conjuntamente a estos se encuentra la tabla de las no conformidades encontradas en la aplicación.

3.1 Enfoques de diseños de pruebas

¿Qué es probar un Software?

Probar es el proceso de ejecución de un programa con el propósito de encontrar errores, se dice que una prueba tiene éxito si descubre un error no detectado hasta el momento. Las pruebas deberían planificarse mucho antes de que empiecen, ya que a estas se le deberían hacer un pequeño seguimiento hasta los requisitos del cliente.

Existen tres enfoques de diseños de casos de pruebas, los cuáles son:

- **Enfoque estructural o de caja blanca:** esta consiste en centrarse en la estructura interna (implementación) del programa para elegir los casos de prueba.
- **Enfoque funcional o de caja negra:** consiste en estudiar la especificación de las funciones, la entrada y la salida para derivar los casos.
- **Enfoque aleatorio:** consiste en utilizar modelos (en muchas ocasiones estadísticos) que representen las posibles entradas al programa para crear a partir de ellos los casos de prueba. (26)

3.1.1 Método seleccionado

La metodología XP se basa en probar el sistema a implementar continuamente como sea posible, reduciendo así el número de los errores, lo que permite aumentar la calidad de los sistemas.

Esta metodología divide las pruebas del sistema en dos grupos: pruebas unitarias, las cuales se encargan de comprobar el código y son diseñadas por los programadores, y el otro grupo son las

pruebas de aceptación o funcionales como también se les llama, estas pruebas se encargan de evaluar el producto al final de una iteración, para así verificar que este cumpla con las condiciones diseñadas por el cliente final. Las pruebas unitarias suelen ser pruebas de caja blanca, mientras que las de aceptación son de caja negra.

Las pruebas de caja negra también conocidas como pruebas funcionales tienen como objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados. También se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a proveer datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro.

Estas pruebas de caja negra están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario, apoyándose así en los requisitos del módulo. El que realiza estas pruebas no sabe cómo está estructurado por dentro el programa o bien no necesita saber nada de programación, solo necesita saber cuáles pueden ser las posibles entradas sin necesidad de entender cómo se deben obtener las salidas, donde se trata de encontrar errores en la interfaz mientras se está usando. (27)

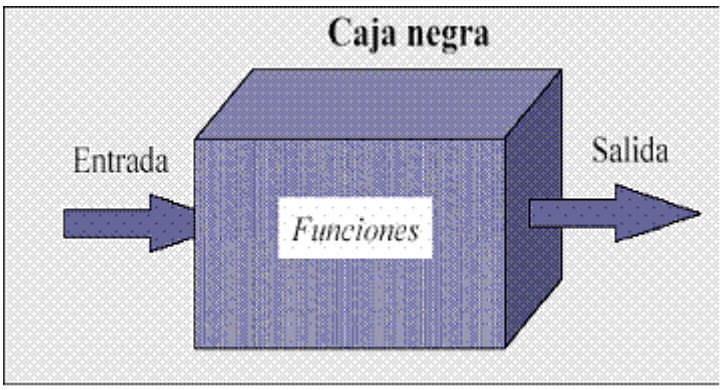


Figura 19: Método de caja negra

A continuación se muestran algunas variantes de pruebas de caja negra:

- **Métodos de prueba basados en grafos:** en este método se debe entender los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas.

- **Partición equivalente:** se presenta la partición equivalente como un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. El objetivo de partición equivalente es reducir el posible conjunto de casos de prueba en uno más pequeño, un conjunto manejable que evalúe bien el software.
- **Análisis de valores límite:** los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límites (AVL) como técnica de prueba. El análisis de valores límite lleva a una elección de casos de prueba que ejerciten los valores límite. El análisis de valores límite es una técnica de diseño de casos de prueba que completa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida. (28)

Para verificar que el sistema desarrollado cumple con las funciones específicas para las que se ha creado se realizan pruebas de aceptación, utilizando el enfoque de prueba funcional o de caja negra y empleando la técnica de partición equivalente, ya que mediante estas pruebas se obtiene un producto final con gran calidad y principalmente se logra cumplir con las especificaciones tratadas con el cliente.

3.2 Caso de pruebas basadas en historias de usuarios

Para probar las funcionalidades se diseñaron un total de diez casos de pruebas correspondientes a las historias de usuarios identificadas. A continuación se muestra un ejemplo de casos de prueba para la HU: Generar función Insertar en el lenguaje PL/pgSQL.

Condiciones de ejecución: El plugin tiene que estar integrado al HABD y haber al menos una conexión a un servidor de base de datos.

Escenario	Descripción	Variable 1	Variable 2	Variable 3	Variable 4	Respuesta del sistema	Flujo central
EC 1.1 Generar la función Insertar con los datos correctos.	Se genera la función satisfactoriamente	V	V	V	N/A	El sistema muestra un mensaje: ¨ # función(es) generada(s) exitosamente¨.	1- El usuario debe seleccionar la tabla. 2- El usuario debe seleccionar la función Insertar. 3- El usuario debe presionar el botón Generar.
		tabla de la base de datos	PL/pgSQL	Tipo de función: Insertar	No se selecciona		
EC 1.2 Generar la función Insertar con los datos incorrectos.		V	I	V	N/A	El sistema muestra un mensaje: ¨Debe seleccionar un lenguaje¨.	
		tabla de la base de datos	No se selecciona	Tipo de función: Insertar	No se selecciona		
		V	V	I	NA	El sistema muestra un mensaje: ¨Debe seleccionar al menos una función¨.	
		tabla de la base de datos	PL/pgSQL	No se selecciona	No se selecciona		

Tabla 6: Sección a probar de la HU: Generar función Insertar en el lenguaje PL/pgSQL

La tabla que se muestra a continuación describe las variables que se encuentran asociadas a las interfaces de la HU Generar función Insertar en el lenguaje PL/pgSQL.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
variable 1	Tablas	Campo de selección(*tree*)	No	Son las tablas de la base de datos donde se generan las funciones.
variable 2	Lenguajes	Campo de selección (*Checkbox*)	No	Son los lenguajes en el que pueden ser generadas las funciones.
variable 3	Tipo de funciones	Campo de selección (*Listbox*)	No	Son los disímiles tipos de funciones que pueden ser generadas.
variable 4	Cursores	Campo de selección (*Checkbox*)	Si	Es una optimización a la función Seleccionar del lenguaje PL/pgSQL.

Tabla 7: Descripción de las variables

Los resultados no satisfactorios obtenidos de las pruebas realizadas pasaron a ser no conformidades y se emitieron en el registro de defectos y dificultades detectados. La tabla que se muestra a continuación muestra las no conformidades que se encontraron, además de las fechas en la que se detectaron y en la que se resolvieron.

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del Equipo Desarrollo
Aplicación	1	Se muestra el mensaje: 1 funciones generadas exitosamente, pero no se genera.	Generar función Insertar en el lenguaje PL/pgSQL.	Prueba	S	04/05/2012 PD 06/05/2012 RA	Se generó la función seleccionada, mostrando el mensaje: 1 funciones generadas exitosamente.
Aplicación	2	Se muestra el mensaje: 1 funciones generadas exitosamente pero no se genera.	Generar función Eliminar en el lenguaje PL/pgSQL	Prueba	S	04/05/2012 PD 08/05/2012 RA	Se generó la función seleccionada, mostrando el mensaje: 1 funciones generadas exitosamente.
Aplicación	3	Se muestra el mensaje: 1 funciones generadas exitosamente, pero no se genera.	Generar función Truncar en el lenguaje PL/pgSQL	Prueba	S	11/05/2012 PD 12/05/2012 RA	Se generó la función seleccionada, mostrando el mensaje: 1 funciones generadas exitosamente.

Tabla 8: Tabla no conformidades

3.3 Conclusiones del capítulo

Se analiza el método de caja negra y sus técnicas, como resultado de este estudio se diseñaron un total de diez casos de pruebas basados en las HU que se obtuvieron en la fase de exploración de la aplicación. Se verificó el funcionamiento del plugin aplicándole los casos de pruebas al funcionamiento de este. De las pruebas ejecutadas al sistema se obtiene un total de tres no conformidades y todas funcionales. Fueron resueltas en un período de siete días.

CONCLUSIONES GENERALES

Al finalizar la investigación puede concluirse lo siguiente:

- El proceso de desarrollo de la aplicación informática estuvo guiado por la metodología XP, utilizando C++ como lenguaje de programación y Qt como framework para acceder al contenido del plugin.
- Se definieron las funcionalidades de la aplicación reflejadas en diez historias de usuarios y se elaboraron un total de diez tarjetas CRC que guiaron el proceso de implementación.
- Se implementó una aplicación informática que generara las operaciones CRUD mediante funciones o procedimientos almacenados, para de esta forma hacer más fácil el trabajo con estas operaciones.
- Se realizaron pruebas de validación de software al plugin “CRUD-PG para la herramienta de administración de base de datos HABD”, para verificar que funciona correctamente y que los requisitos se cumplen de forma satisfactoria.

RECOMENDACIONES

Durante el desarrollo del trabajo han surgido algunas ideas que podrían ser incorporadas en un futuro con el objetivo de fortalecer el plugin, por lo que se recomienda:

- Generar funciones en otros lenguajes soportados por PostgreSQL, como pljava, plperl, entre otros.
- Brindar la posibilidad al usuario de editar o realizar cambios menores en el código de las funciones que genere el plugin y ajustarlas a su conveniencia, para la futura utilización de las mismas con frameworks y entornos de desarrollo integrados.

REFERENCIAS BIBLIOGRÁFICAS

1. **Marqués, Mercedes.** *Apuntes de Ficheros y Bases de Datos.* Castellón de la Plana, España : s.n., 2009. Ingeniería Técnica en Informática de Gestión de la Universidad Jaume.

2. **Wesley, Addison.** *Database Administration : The Complete Guide to Practices and Procedures .* June 14, 2002 .

3. **Álvarez, Sara.** desarrollo web. [En línea] 31 de Julio de 2007. [Citado el: 21 de Octubre de 2011.] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

4. **Navathe, Ramez Elmasri y Shamkant B.** *"Sistemas de bases de Datos Conceptos Fundamentales".* Segunda edición. Addison Wesley Iberoamericana : s.n., 1997.

5. **Harvey M.Deitel, P.J.Deitel.** *Como programar en C/C++ y Java.*

6. **González, Luis.** [En línea] Proyecto EAST, 17 de Enero de 2007. [Citado el: 22 de Octubre de 2011.] http://eats.wordpress.com/2007/01/17/lenguajes-del-lado-servidor-y-del-lado-cliente/.

7. **Beck, K.** *"Extreme Programming Explained. Embrace Change", Pearson Education.* Addison Wesley. 2000. 1999.

8. **Rumbaugh, James.** *El Lenguaje Unificado de Modelado.* California : s.n., 1998.

9. **Burkhardt, Rainer.** *UML: Unified Modeling Language.* Addison-Wesley. 1997.

10. **INSTITUTO NACIONAL DE ESTADISTICA INFORMATICA.** *Herramientas CASE.* 1999.

11. Visual Paradigm Internaciona. *visual paradigm.* [En línea] [Citado el: 26 de Octubre de 2011.] http://www.visual-paradigm.com/product/vpuml/.

12. Lenguaje de Programación. *Lenguaje de Programación.* [En línea] [Citado el: 26 de Octubre de 2011.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

13. cplusplus. *cplusplus.* [En línea] [Citado el: 05 de Noviembre de 2011.] http://www.cplusplus.com.

14. **Wesley, Addison.** *Comparative Programming Languages.* 1993.

15. Global Mentoring. *Global Mentoring.* [En línea] 6 de Mayo de 2011. [Citado el: 29 de Octubre de 2011.] http://globalmentoring.com.mx/portal/es/blog/item/que-es-un-framework.html.

16. Qt nokia. *Qt nokia.* [En línea] [Citado el: 5 de Noviembre de 2011.] www.trolltech.com.

17. openSUSE.org. *openSUSE.org.* [En línea] [Citado el: 5 de Noviembre de 2011.] http://es.opensuse.org/Qt_Toolkit.

18. Qt Nokia. *Qt Nokia.* [En línea] [Citado el: 13 de Noviembre de 2011.] http://developer.qt.nokia.com/wiki/QtCreator_Spanish.

19. **Instituto Nacional de Tecnologías Educativas y Formación del Profesorado.** Observatorio Tecnológico. *Observatorio Tecnológico.* [En línea] 17 de Enero de 2008. [Citado el: 16 de Noviembre de 2011.] http://recursostic.educacion.es/observatorio/web/ca/software/software-general/548-luis-garcia.

20. PostgreSQL. *PostgreSQL.* [En línea] [Citado el: 20 de Noviembre de 2011.] http://www.postgresql.org.

21. MSDN Magazine . *MSDN Magazine .* [En línea] [Citado el: 16 de Enero de 2011.] http://msdn.microsoft.com/en-us/magazine/ee236415.aspx.

22. **Joskowicz, Ing. José.** *Reglas y Prácticas en eXtreme Programming .*

23. **Fowler, Martin.** *UML, gota a gota.* Addison Wesley Longman . México : s.n., 1999.

24. Geek the Planet. *Geek the Planet.* [En línea] 10 de Mayo de 2011. [Citado el: 18 de Enero de 2012.] http://geektheplanet.net/5461/patrones-de-diseno-%C2%BFque-son-y-para-que-sirven.xhtml.

25. **Eduardo Fernández, Medina Patón.** Alarcos. *Alarcos.* [En línea] [Citado el: 10 de Abril de 2012.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema09.pdf .

26. **Elena, Ciolli Maria.** *Testing de Migracion de Aplicaciones Distribuidas a Entornos Web.* Mayo 2008.

27. **Johanna Rojas, Emilio Barrios ARQUISOFT.** [En línea] 13 de Abril de 2011. [Citado el: 15 de Abril de 2012.] http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html.

BIBLIOGRAFÍAS

Álvarez, Sara. 2007. desarrollo web. [En línea] 31 de Julio de 2007. [Citado el: 21 de Octubre de 2011.] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

Beck, K. 2000. 1999. *"Extreme Programming Explained. Embrace Change", Pearson Education.* Addison Wesley. 2000. 1999.

Burkhardt, Rainer. 1997. *UML: Unified Modeling Language.* Addison-Wesley. 1997.

cplusplus. *cplusplus.* [En línea] [Citado el: 05 de Noviembre de 2011.] http://www.cplusplus.com.

Eduardo Fernández, Medina Patón. Alarcos. *Alarcos.* [En línea] [Citado el: 10 de Abril de 2012.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema09.pdf .

Elena, Ciolli Maria. Mayo 2008. *Testing de Migracion de Aplicaciones Distribuidas a Entornos Web.* Mayo 2008.

Fowler, Martin. 1999. *UML, gota a gota.* Addison Wesley Longman . México : s.n., 1999.

2011. Geek the Planet. *Geek the Planet.* [En línea] 10 de Mayo de 2011. [Citado el: 18 de Enero de 2012.] http://geektheplanet.net/5461/patrones-de-diseno-%C2%BFque-son-y-para-que-sirven.xhtml.

2011. Global Mentoring. *Global Mentoring.* [En línea] 6 de Mayo de 2011. [Citado el: 29 de Octubre de 2011.] http://globalmentoring.com.mx/portal/es/blog/item/que-es-un-framework.html.

González, Luis. 2007. [En línea] Proyecto EAST, 17 de Enero de 2007. [Citado el: 22 de Octubre de 2011.] http://eats.wordpress.com/2007/01/17/lenguajes-del-lado-servidor-y-del-lado-cliente/.

Harvey M.Deitel, P.J.Deitel. *Como programar en C/C++ y Java.*

INSTITUTO NACIONAL DE ESTADISTICA INFORMATICA. 1999. *Herramientas CASE.* 1999.

Instituto Nacional de Tecnologías Educativas y Formación del Profesorado. 2008. Observatorio Tecnológico. *Observatorio Tecnológico.* [En línea] 17 de Enero de 2008. [Citado el: 16 de Noviembre de 2011.] http://recursostic.educacion.es/observatorio/web/ca/software/software-general/548-luis-garcia.

Johanna Rojas, Emilio Barrios ARQUISOFT. 2011. [En línea] 13 de Abril de 2011. [Citado el: 15 de Abril de 2012.]

http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html.

Joskowicz, Ing. José. *Reglas y Prácticas en eXtreme Programming .*

Lenguaje de Programación. *Lenguaje de Programación.* [En línea] [Citado el: 26 de Octubre de 2011.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

Marqués, Mercedes. 2009. *Apuntes de Ficheros y Bases de Datos.* Castellón de la Plana, España : s.n., 2009. Ingeniería Técnica en Informática de Gestión de la Universidad Jaume.

Mora, Roberto Canales. 2003-2005. *GRASP.* 2003-2005.

MSDN Magazine . *MSDN Magazine .* [En línea] [Citado el: 16 de Enero de 2011.] http://msdn.microsoft.com/en-us/magazine/ee236415.aspx.

Navathe, Ramez Elmasri y Shamkant B. 1997. *"Sistemas de bases de Datos Conceptos Fundamentales".* Segunda edición. Addison Wesley Iberoamericana : s.n., 1997.

openSUSE.org. *openSUSE.org.* [En línea] [Citado el: 5 de Noviembre de 2011.] http://es.opensuse.org/Qt_Toolkit.

Penadés, Patricio Letelier y Mª Carmen. *Métodologías ágiles para .* Valencia : s.n.

PostgreSQL. *PostgreSQL.* [En línea] [Citado el: 20 de Noviembre de 2011.] http://www.postgresql.org.

Qt nokia. *Qt nokia.* [En línea] [Citado el: 5 de Noviembre de 2011.] www.trolltech.com.

Qt Nokia. *Qt Nokia.* [En línea] [Citado el: 13 de Noviembre de 2011.] http://developer.qt.nokia.com/wiki/QtCreator_Spanish.

Rumbaugh, James. 1998. *El Lenguaje Unificado de Modelado.* California : s.n., 1998.

Visual Paradigm Internaciona. *visual paradigm.* [En línea] [Citado el: 26 de Octubre de 2011.] http://www.visual-paradigm.com/product/vpuml/.

Wesley, Addison. 1993. *Comparative Programming Languages.* 1993.

—. June 14, 2002 . *Database Administration : The Complete Guide to Practices and Procedures .* June 14, 2002 .

ANEXOS

Anexo 1: HU #2 Generar función Seleccionar.

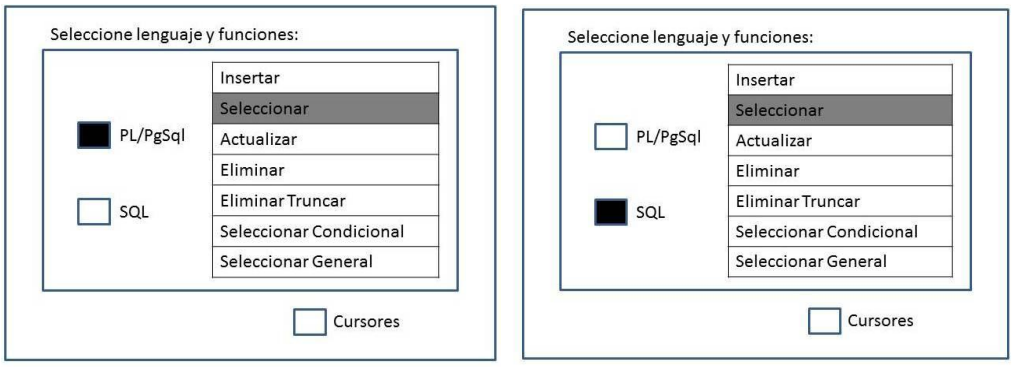
Historia de Usuario	
Número: 2	**Nombre de la Historia de Usuario:** Generar función Seleccionar en los lenguajes PL/pgSQL y SQL
Cantidad de modificaciones a la Historia de Usuario: 1	
Usuario: usuario registrado en la base de datos	**Iteración asignada:** 1era iteración.
Prioridad en negocio: Muy Alta	**Puntos estimados:** 2.0
Riesgo en desarrollo: Muy Alto	**Puntos reales:** 2 semanas.
Descripción: El usuario selecciona la tabla, posteriormente precederá a escoger uno de los dos lenguajes y la función Seleccionar que se generará para dicha tabla, al concluir esto debe presionar el botón Generar para que dé cumplimiento a la funcionalidad.	
Observaciones: El usuario registrado en la base de datos debe seleccionar una tabla.	
Prototipo de interfaces:	
	

Tabla 9: HU Generar función Seleccionar en los lenguajes PL/pgSQL y SQL

Anexo 2: HU #3 Generar función Actualizar.

Historia de Usuario	
Número: 3	**Nombre de la Historia de Usuario:** Generar función Actualizar en el lenguaje PL/pgSQL.
Cantidad de modificaciones a la Historia de Usuario: 1	
Usuario: usuario registrado en la base de datos	**Iteración asignada:** 1era iteración.
Prioridad en negocio: Muy Alta	**Puntos estimados:** 2.0
Riesgo en desarrollo: Muy Alto	**Puntos reales:** 2 semanas.
Descripción: El usuario selecciona la tabla, posteriormente precederá a escoger el lenguaje y la función Actualizar que se generará para dicha tabla, al concluir esto debe presionar el botón Generar para que dé cumplimiento a la funcionalidad.	
Observaciones: El usuario registrado en la base de datos debe seleccionar una tabla.	
Prototipo de interfaces:	

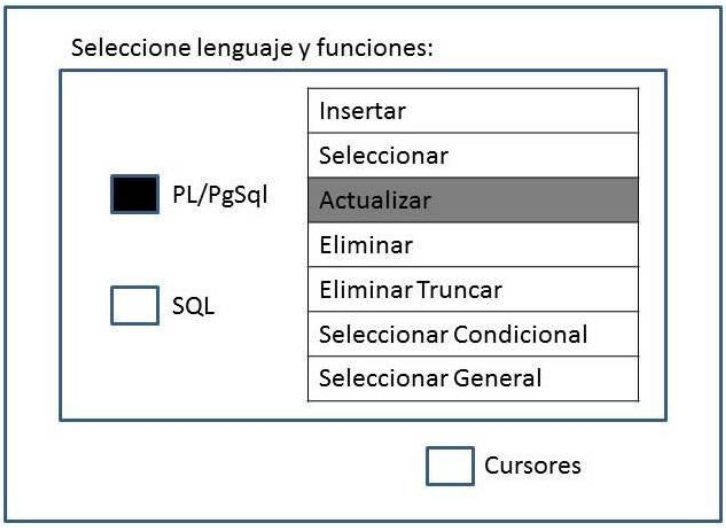


Tabla 10: HU Generar función Actualizar en el lenguaje PL/pgSQL

Anexo 3: HU #4 Generar función Eliminar.

Historia de Usuario	
Número: 4	**Nombre de la Historia de Usuario:** Generar función Eliminar en el lenguaje PL/pgSQL.
Cantidad de modificaciones a la Historia de Usuario: 1	
Usuario: usuario registrado en la base de datos	**Iteración asignada:** 1era iteración.
Prioridad en negocio: Muy Alta	**Puntos estimados:** 2.0
Riesgo en desarrollo: Muy Alto	**Puntos reales:** 2 semanas.
Descripción: El usuario selecciona la tabla, posteriormente precederá a escoger el lenguaje y la función Eliminar que se generará para dicha tabla, al concluir esto debe presionar el botón Generar para que dé cumplimiento a la funcionalidad.	
Observaciones: El usuario registrado en la base de datos debe seleccionar una tabla.	
Prototipo de interfaces:	

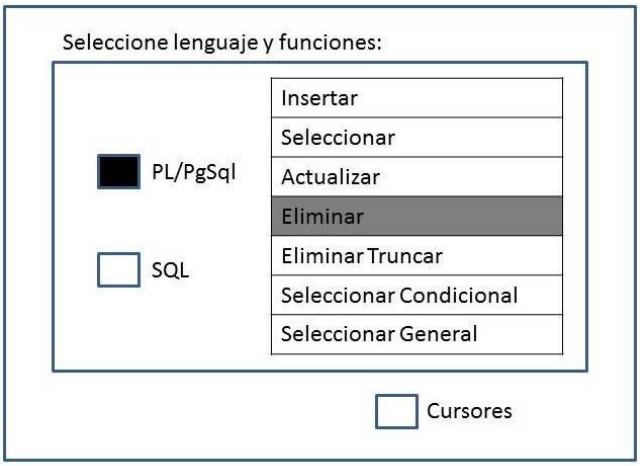


Tabla 11: Generar función Eliminar en el lenguaje PL/pgSQL.

Anexo 4: Tarea de ingeniería #2.

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario: 2**
Nombre Tarea: Generar función Seleccionar en los lenguajes PL/pgSQL y SQL	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2 semanas
Fecha Inicio: 09/01/2012	**Fecha Fin: 23/01/2012**
Programador Responsable: Yaciel López Pérez	
Descripción: La funcionalidad recibirá como parámetro una tabla y sus campos, además de su esquema correspondiente en que se generará la función que permitirá hacer un Select, para mostrar todos los datos posteriormente contenidos en la tabla.	

Tabla 12: Generar función Seleccionar en los lenguajes PL/pgSQL y SQL

Anexo 5: Tarea de ingeniería #3.

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario: 3**
Nombre Tarea: Generar función Actualizar en el lenguaje PL/pgSQL.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2 semanas
Fecha Inicio: 24/01/2012	**Fecha Fin: 07/02/2012**
Programador Responsable: Yaciel López Pérez	
Descripción: Recibirá como parámetro la tabla y su esquema correspondiente en que se generará la función Update que posibilitará la actualización de los datos en la tabla, basada en las condiciones que el usuario le introduzca.	

Tabla 13: Generar función Actualizar en el lenguaje PL/pgSQL

Anexo 6: Tarea de ingeniería #4.

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario: 4**
Nombre Tarea: Generar función Eliminar en el lenguaje PL/pgSQL.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2 semanas
Fecha Inicio: 08/02/2012	**Fecha Fin: 22/02/2012**
Programador Responsable: Yaciel López Pérez	
Descripción: Esta función recibirá por parámetro una tabla y su esquema correspondiente, a la cual se le generará la función Delete, que posibilita al usuario eliminar los datos de una tabla mediante una condición.	

Tabla 14: Generar función Eliminar en el lenguaje PL/pgSQL

Anexo 7: Tarjeta CRC #2.

Tarjeta CRC	
Clase: Generar función Seleccionar en los lenguajes PL/pgSQL y SQL	
Responsabilidades	**Colaboraciones**
Generar función Seleccionar en los lenguajes PL/pgSQL y SQL	

Tabla 15: Generar función Seleccionar en los lenguajes PL/pgSQL y SQL

Anexo 8: Tarea CRC #3.

Tarjeta CRC	
Clase: Generar función Actualizar en el lenguaje PL/pgSQL.	
Responsabilidades	**Colaboraciones**
Generar función Actualizar en el lenguaje PL/pgSQL.	

Tabla 16: Generar función Actualizar en el lenguaje PL/pgSQL

Anexo 9: Tarea CRC #4.

Tarjeta CRC	
Clase: Generar función Eliminar en el lenguaje PL/pgSQL.	
Responsabilidades	**Colaboraciones**
Generar función Eliminar en el lenguaje PL/pgSQL.	

Tabla 17: Generar función Eliminar en el lenguaje PL/pgSQL

Anexo 10: Caso de prueba de la HU: Generar función Seleccionar en los lenguajes PL/pgSQL o SQL.

Escenario	Descripción	Variable 1	Variable 2	Variable 3	Variable 4	Respuesta del sistema	Flujo central
EC 1.1 Generar la función Seleccionar con los datos correctos.	Se genera la función satisfactoriamente	V	V	V	N/A	El sistema muestra un mensaje: ¨ # función(es) generada(s) exitosamente¨.	1- El usuario debe seleccionar la tabla. 2- El usuario debe seleccionar la función Seleccionar. 3- El usuario debe presionar el botón Generar
		tabla de la base de datos	PL/pgSQL o SQL	Tipo de función: Seleccionar	No se selecciona		
EC 1.2 Generar la función Seleccionar con los datos incorrectos.		V	I	V	N/A	El sistema muestra un mensaje: ¨Debe seleccionar un lenguaje¨.	
		tabla de la base de datos	No se selecciona	Tipo de función: Seleccionar	No se selecciona		
		V	V	I	NA	El sistema muestra un mensaje: ¨Debe seleccionar al menos una función¨.	
		tabla de la base de datos	PL/pgSQL	No se selecciona	No se selecciona		

Tabla 18: Sección a probar de la HU: Generar función Seleccionar en los lenguajes PL/pgSQL o SQL

Anexo 11: Caso de prueba de la HU: Generar función Actualizar en el lenguaje PL/pgSQL.

Escenario	Descripción	Variable 1	Variable 2	Variable 3	Variable 4	Respuesta del sistema	Flujo central
EC 1.1 Generar la función Actualizar con los datos correctos.	Se genera la función satisfactoriamente	V	V	V	N/A	El sistema muestra un mensaje: ¨ # función(es) generada(s) exitosamente¨.	1- El usuario debe seleccionar la tabla. 2- El usuario debe seleccionar la función Actualizar. 3- El usuario debe presionar el botón Generar.
		tabla de la base de datos	PL/pgSQL	Tipo de función: Actualizar	No se selecciona		
EC 1.2 Generar la función Actualizar con los datos incorrectos.		V	I	V	N/A	El sistema muestra un mensaje: ¨Debe seleccionar un lenguaje¨.	
		tabla de la base de datos	No se selecciona	Tipo de función: Actualizar	No se selecciona		
		V	V	I	NA	El sistema muestra un mensaje: ¨Debe seleccionar al menos una función¨.	
		tabla de la base de datos	PL/pgSQL	No se selecciona	No se selecciona		

Tabla 19: Generar función Actualizar en el lenguaje PL/pgSQL.

Anexo 12: Caso de prueba de la HU: Generar función Eliminar en el lenguaje PL/pgSQL.

Escenario	Descripción	Variable 1	Variable 2	Variable 3	Variable 4	Respuesta del sistema	Flujo central
EC 1.1 Generar la función Eliminar con los datos correctos.	Se genera la función satisfactoriamente	V	V	V	N/A	El sistema muestra un mensaje: ¨ # función(es) generada(s) exitosamente¨.	1- El usuario debe seleccionar la tabla. 2- El usuario debe seleccionar la función Eliminar. 3- El usuario debe presionar el botón Generar.
		tabla de la base de datos	PL/pgSQL	Tipo de función: Eliminar	No se selecciona		
EC 1.2 Generar la función Eliminar con los datos incorrectos.		V	I	V	N/A	El sistema muestra un mensaje: ¨Debe seleccionar un lenguaje¨.	
		tabla de la base de datos	No se selecciona	Tipo de función: Eliminar	No se selecciona		
		V	V	I	NA	El sistema muestra un mensaje: ¨Debe seleccionar al menos una función¨.	
		tabla de la base de datos	PL/pgSQL	No se selecciona	No se selecciona		

Tabla 20: Generar función Eliminar en el lenguaje PL/pgSQL

GLOSARIO DE TÉRMINOS

A

Apache: Es un software (libre) servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etcétera), Windows y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

API: Interfaz de programación de aplicaciones o API (del inglés Application Programming Interface).

ASP: Active Server Pages es una tecnología de Microsoft del tipo lado del servidor para páginas web generadas dinámicamente.

G

GUI: La interfaz gráfica de usuario, es conocida también como GUI es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

H

Hardware: Componentes físicos que constituyen las computadoras y demás dispositivos periféricos.

M

Multiplataforma: Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas. Una plataforma es una combinación de hardware y software usada para ejecutar aplicaciones.

P

PHP: Es un lenguaje de programación usado generalmente para la creación de contenido para sitios web.

Plugins: Es una aplicación informática que añade funcionalidades específicas a un programa principal. Los plugins no son parches ni actualizaciones, sino propiedades añadidas a los programas originales.

S

Software: Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.