Universidad de las Ciencias Informáticas

Facultad 6

Título: Plugin de recuperación del estado de entidades para la herramienta de administración de bases de datos HABD

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

Autores:

Jennys Almaguer Zaldivar

Alexis Camué Hernández

Tutor:

Ing. Marcos Luis Ortiz Valmaseda

La Habana, junio de 2012

"Año 54 de la Revolución"

"A ustedes les corresponderá vivir el siglo más difícil y decisivo de la historia humana. Para ello, prepararse es el más sagrado deber; profundizar en los conocimientos profesionales y políticos es requisito indispensable. La cultura general integral masiva, algo jamás soñado por sociedad alguna, es hoy una posibilidad real al alcance de todos los cubanos.

Una profunda formación ética, humanitaria, solidaria e internacionalista es parte esencial de esa cultura."

Fidel Castro Ruz

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2012.

Jennys Almaguer Zaldivar

_____

Firma del Autor

Alexis Camué Hernández

_____

Firma del Autor

Ing. Marcos Luis Ortiz Valmaseda

_____

Firma del tutor

Tutor:

Ing. Marcos Luis Ortiz Valmaseda

Especialidad de graduación: Ingeniería en Ciencias Informáticas

Categoría docente: Instructor

Cargo: Asesor Técnico Docente

Área: Centro de Tecnologías de Gestión de Datos (DATEC)

Años de graduado: 3

Correo Electrónico: mlortiz@uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba

Agradecimientos

"El agradecimiento es la memoria del corazón" según el filósofo taoísta Lao-tsé, si se nos escapara alguien en la turbiedad de nuestra memoria, no significa que se nos escape de la memoria del corazón."

De Jennys

A mi mami querida Luz Zaldivar: Por su amor infinito e incondicional, por su sacrificio, tenacidad y coraje, por sus preocupaciones y desvelos, por su perseverancia en mi educación inculcándome valores humanos y éticos, por ser un ejemplo de mujer, madre, profesional, amiga y esposa. Por su apoyo en todos los momentos difíciles, por su fuerza y sensibilidad. Te quiero mami.

A mi papá Javier Almaguer: Por todo su apoyo incondicional en todas las etapas de la vida, por su amor y cariño, por hacerme reír y compartir momentos inolvidables. Por su ímpetu y presencia en los momentos más difíciles. A su esposa Anita por acogerme en su hogar y cuidar mucho de él.

A mi padre de crianza Reynaldo Fidalgo: Por ser mi ejemplo de profesional guiándome hacia esta carrera, por haber sido como un padre en las tristezas, alegrías, enfermedades y en la salud. Por sus preocupaciones y enseñanzas, por todo su cariño implícito en todo lo que ha hecho por mí.

A mis hermanitos Loandis y Javierito, por existir y darme recuerdos bellos en momentos difíciles.

A mi abuela querida Magalis Hechavarría, por conllevar mis malacrianzas y sin embargo preocuparse por mí, por todo su cariño. A la tía más linda del mundo Yanexy Hechavarría, por ser más que tía, una amiga.

A mi novio y compañero de tesis, Alexis Camué por aguantar hasta ciertos puntos mi estrés, y mis arranques de locura. Por hacerme reír, por todas las canciones compartidas, por ser cómplices, por todo su amor y por lo que hemos hecho juntos.

Agradezco a mis suegros, Olga Lidia Hernández y Ángel Andrés Camué por todo el apoyo y acogerme como si fuera un miembro más de la familia, a Titi, Doris a toda la familia.

Agradezco a mis amigas de toda la vida, Yani, Dari, y a mi prima Yinet.

A mis compañeras Yadiris por compartir gran parte de las vicisitudes vividas en la universidad, por ser

amiga, banco de crédito y tutora a la vez, a Liliannys por su sensibilidad, desinterés y su amistad eterna a Dalied por su confianza, su amistad noble y sincera y por su paciencia para conmigo, a mi segunda tutora Ivet, a mi compañeras Betty, Yeneisi, Yasnelis, Yadira, a Yasna, Aliu, y Liset y Sandrita.

A los profes, *mi tutor Marcos Luis por su ayuda, al profe Flavio por su apoyo incondicional, a Yunior Mesa por toda su disponibilidad a ayudar, a Glennis por su asistencia y a Edgar por ayudarnos en todo y hacer del proyecto PostgreSQL una familia.*

A la Revolución y a nuestro Comandante en Jefe por la creación de esta Universidad.

De Alexis

Agradezco primeramente a mi abuela Teodora Martínez, quien dijo desde su lecho de muerte "…sigue estudiando, no dejes de hacerlo…", convirtiéndose en mi principal inspiración.

A mis padres *por todo el apoyo, la confianza y la educación que me han dado. Por haberme guiado por el camino correcto sin importar la situación.*

A mi hermana *por ser incondicional en todos los momentos juntos.*

A mi familia *por ser motivo de paz, alegría y desinterés.*

A mi compañera de tesis y novia *Jennys por ser una persona tan especial y diferente.*

A Flavio*, por su apoyo incondicional.*

A Marcos Luis *por su ayuda.*

A mis amigos *y compañeros de estudios.*

A la FEU *por enseñarme valores y un nuevo modo de ver la vida.*

A Fidel y a la Revolución *por haberme permitido estudiar en esta universidad.*

""La gratitud, como ciertas flores, no se da en la altura y mejor reverdece en la tierra buena de los humildes."" José Martí

Dedicatoria

Dedicamos el esfuerzo y sacrificio, todos nuestros éxitos y vicisitudes de tantos años de estudios y vida, esperando ser un orgullo para ustedes:

De Jennys:

A mis padres, Luz Zaldivar, Javier Almaguer y Reynaldo Fidalgo.

De Alexis:

A mi madre Olga Lidia Hernández y a mi padre Ángel Andrés Camué.

Resumen

El presente trabajo surge de la necesidad de corregir los errores de usuario que provocan la eliminación involuntaria de una entidad o varias de ellas, de forma tal que se asegure la persistencia y recuperación de la información contenida en las bases de datos (BD) PostgreSQL, evitando así las molestas consecuencias por pérdidas de datos. Por dicha razón se decide desarrollar un plugin que permita la recuperación del estado de entidades, para integrar en la herramienta de administración de bases de datos HABD y esta a su vez formará parte del conjunto de herramientas que conforman el PostgreSQL Cubano el cual se desarrolla en el Departamento de PostgreSQL perteneciente al Centro de Tecnologías de Gestión de Datos de la Universidad de las Ciencias Informáticas. Para ello se realizó un estudio profundo del funcionamiento de las copias de seguridad en bases de datos PostgreSQL y una investigación exhausta de herramientas que permitan la recuperación del estado de entidades en los sistemas gestores de bases de datos. La realización del plugin de recuperación del estado de entidades constituirá una alternativa para que la pérdida de datos por error de usuario sea reversible de una manera rápida en correspondencia con el procedimiento que utiliza. Pretende colaborar con la persistencia de los datos y evitar la interferencia en la confidencialidad e integridad de la información almacenada en las bases de datos, facilitando el trabajo a los desarrolladores de la comunidad de PostgreSQL.

Palabras claves: entidad, plugin, procedimiento

INTRODUCCIÓN

El desarrollo audaz y avanzado de las Tecnologías de la Información y las Comunicaciones (TIC) ha generalizado el empleo de los medios digitales; materializándose significativamente en la forma de registrar el cúmulo de información que generan las grandes empresas y los polos investigativos. Las bases de datos solucionan el problema de almacenar la información de forma manual, pues organizan y estructuran los datos digitalmente. Desde la creación de los Sistemas Gestores de Bases de Datos (SGBD), software que permite a los usuarios manipular la información almacenada, las bases de datos son más seguras, flexibles y manejables.

En la actualidad, los sistemas informáticos demandan el empleo de los gestores de bases de datos relacional. En el mercado del software existen múltiples variantes propietarias para suplir dicha demanda, entre ellas se encuentran Oracle y SQLServer. Igualmente prevalecen opciones libres, las cuales tienen el riesgo latente de que dejen de serlo en algún momento, como ocurrió con FoxPro y MySQL. (1) Uno de los mejores exponentes de los gestores libres es PostgreSQL, pues posee características que tradicionalmente solo se podían encontrar en productos comerciales de alta trascendencia, además de ser un software de código abierto y un sistema gestor objeto-relacional.

Las aplicaciones de software en Cuba generalmente emplean una amplia gama de gestores privativos y libres, por este motivo el país se encamina hacia los objetivos siguientes:

- ✓ Contribuir al fortalecimiento de la soberanía tecnológica cubana, desde el enfoque del desarrollo de tecnologías de bases de datos adoptando como base a PostgreSQL.
- ✓ Proveer soluciones integrales y consultorías relacionadas con la migración y la explotación de PostgreSQL.
- ✓ Contribuir a la formación de especialistas de alto nivel, potenciando el empleo de tecnologías de bases de datos de código abierto. (1)

Para materializar las misiones mencionadas se sitúa en la vanguardia de la revolución tecnológica cubana a la Universidad de la Ciencias Informáticas (UCI), principal fuente de exportación de productos informáticos nacionales. Este centro contribuye de manera activa y progresiva con la independencia tecnológica, potenciando la práctica y explotación del gestor libre PostgreSQL; para ello el Departamento de PostgreSQL, perteneciente al Centro de Tecnologías de Gestión de Datos (DATEC), se ha enfocado hacia el desarrollo de aplicaciones que faciliten el trabajo con el gestor PostgreSQL. Actualmente se lleva a cabo la creación de una herramienta de administración de bases

de datos HABD, que adopta como punto de partida a PgAdmin3; interfaz gráfica de código abierto para la administración del gestor de bases de datos PostgreSQL.

La herramienta solamente posee un Front-end, el cual constituye el resultado de la tesis precedente "Exploración y diseño de la herramienta de administración de bases de datos para PostgreSQL (HABD)." Además posee una arquitectura basada en plugin[1], lo que permite que varios componentes sean desarrollados por distintos equipos de forma independiente, para una vez implementados integrarlos en la aplicación principal o Front-end; de esta forma los desarrolladores pueden integrar funcionalidades y nuevos servicios por la flexibilidad que posee.

La herramienta de administración PgAdmin3 carece de funcionalidades para recuperar datos perdidos causados por errores humanos. Cuando un usuario borra o actualiza una entidad de una base de datos como resultado de un error involuntario o accidental, no queda otra alternativa para deshacer la acción que emplear las herramientas de copias de seguridad o salvas (backup) que provee el SGBD PostgreSQL, las cuales solucionan indirectamente el problema. Se realiza una copia de la base de datos cada cierto tiempo, en dependencia de su configuración. Luego se restaura la copia de la base de datos, recuperando el último estado guardado de cada una de las entidades.

El procedimiento mencionado no solo retrocede la acción que se necesita invertir, sino también otras que realizaron posibles usuarios desde que ocurrió el incidente hasta el momento en que se realizó la última salva de la base de datos, originando la pérdida de información. El trabajo se torna inevitablemente engorroso pues los usuarios deben determinar las acciones que tendrían que ejecutar nuevamente.

Dada la complejidad de los sistemas actuales y las exigencias cada vez más críticas en su disponibilidad, recuperar un error humano crea problemas, pues el proceso y resultado que ocasiona la recuperación del estado de entidades es generalmente complejo y tiende a incurrir en un gasto innecesario de recursos.

De lo expresado con anterioridad surge el siguiente **problema de investigación**: ¿Cómo recuperar el estado de entidades en bases de datos PostgreSQL?

Se define como **objeto de estudio**: La recuperación del estado de entidades en bases de datos PostgreSQL, enmarcando la investigación en el **campo de acción**: Aplicaciones informáticas para la recuperación del estado de entidades en bases de datos PostgreSQL.

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

En correspondencia con la problemática planteada y con el propósito de solucionarla se determina como **objetivo general:** Desarrollar un plugin de recuperación del estado de entidades para la herramienta de administración de bases de datos HABD.

Para cumplir este objetivo, se desglosan en los siguientes **objetivos específicos:**

- ✓ Caracterizar los procedimientos y herramientas existentes en la recuperación del estado de entidades enmarcado en bases de datos PostgreSQL.
- ✓ Realizar análisis y diseño del plugin de recuperación del estado de las entidades para bases de datos PostgreSQL.
- ✓ Realizar implementación y pruebas al plugin de recuperación del estado de entidades para bases de datos PostgreSQL.

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

1. Revisión bibliográfica de los procedimientos y herramientas existentes para la recuperación del estado de entidades en bases de datos relacionales.
2. Caracterización de las metodologías, herramientas y tecnologías a utilizar en el desarrollo del plugin de recuperación del estado de las entidades para bases de datos PostgreSQL.
3. Elaboración del análisis del plugin de recuperación del estado de entidades para HABD.
4. Definición de la arquitectura del plugin de recuperación del estado de entidades para HABD.
5. Implementación del plugin de recuperación del estado de las entidades para HABD.
6. Integración del plugin de recuperación del estado de las entidades en HABD.
7. Confección de la ayuda del plugin de recuperación del estado de entidades para HABD.
8. Realización del diseño de casos de pruebas.
9. Evaluación del plugin de recuperación del estado de las entidades a partir de los casos de pruebas diseñados.

Resultados Esperados:

Con la realización exitosa de las tareas planteadas se espera como **posible resultado**: Plugin de recuperación del estado de entidades para la herramienta de administración de bases de datos HABD.

El presente trabajo se ha estructurado de la siguiente manera:

Capítulo I: **Fundamentos Teóricos**

El capítulo comprende el estudio acerca de las bases teóricas que constituyen el soporte del plugin a implementar. Es el resultado de la investigación sobre los procedimientos y herramientas existentes para la recuperación del estado de entidades en bases de datos PostgreSQL. Se centra la atención en los principales términos y tendencias relacionadas con la recuperación del estado de entidades, conjuntamente se analiza y selecciona el lenguaje, la metodología, tecnologías y herramientas que son necesarias en el desarrollo del plugin de recuperación del estado de entidades.

Capítulo II: **Análisis y Diseño**

La metodología XP deja en manos del equipo de desarrollo los artefactos a utilizar para la implementación, dependiendo de las capacidades de comunicación, la decisión de generar tantos tipos de diagramas UML como se necesiten y así facilitar el proceso de desarrollo. En el presente capítulo se expondrán las principales características del sistema, su diseño, su arquitectura y los requisitos tanto funcionales como no funcionales a tener en cuenta en su desarrollo. Está integrado por las fases: Exploración, donde se describe la herramienta que se propone a través de las historias de usuario, obteniéndose la lista de reserva del producto y la fase Diseño que recoge las tarjetas CRC (clase, responsabilidad, colaboración) así como la arquitectura de software.

Capítulo III: **Implementación y Pruebas**

La implementación en el proceso de desarrollo de un software adquiere suma relevancia debido a que le provee funcionalidad al producto que se desarrolla. Del mismo modo, son significativas las pruebas que se le realizan al mismo para validar su correcto funcionamiento. En el presente capítulo se desarrolla la descripción de la implementación del sistema y se especifican las pruebas a las que fue sometido el plugin de recuperación del estado de entidades en cada una de las iteraciones, proceso que guía la identificación y corrección de fallos cometidos en las HU, así como su verificación y materialización lo que contribuye a elevar la calidad de los productos desarrollados y la seguridad en los programadores para efectuar modificaciones.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS

Introducción

El capítulo comprende el estudio acerca de las bases teóricas que constituyen el soporte del plugin a implementar. Es el resultado de la investigación sobre los procedimientos y herramientas existentes para la recuperación del estado de entidades en bases de datos PostgreSQL. Se centra la atención en los principales términos y tendencias relacionadas con la recuperación del estado de entidades, conjuntamente se analiza y selecciona el lenguaje, la metodología, tecnologías y herramientas que son necesarias en el desarrollo del plugin de recuperación del estado de entidades.

1.1. Conceptos asociados a la investigación

Sistema gestor de bases de datos

Un sistema gestor de bases de datos según C. J Date básicamente es *"un sistema computarizado para guardar registros; cuya finalidad general es almacenar información y permitir a los usuarios recuperar y actualizar con base en peticiones. La información en cuestión puede ser cualquier cosa que sea de importancia para el individuo u organización; en otras palabras, todo lo que sea necesario para auxiliarle en el proceso general de su administración.*" (2)

Sistema gestor de bases de datos PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD (*Berkeley Software Distribution*) y con su código fuente disponible libremente. Funciona perfectamente con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema por su gran escalabilidad. Soporta gran parte del estándar *SQL* (*Structured Query Language*[2] en español Lenguaje de Consulta Estructurado) y considerable variedad de tipos de datos, además del almacenamiento de grandes objetos binarios como imágenes, sonidos y videos.

La tendencia hacia la implantación del SGBD PostgreSQL en las universidades y organismos del país es cada vez más creciente, pues se caracteriza por su gran potencia y ser de código abierto. Para administrar dicho gestor, es necesario utilizar una interfaz que lo permita. Actualmente existen varias interfaces gráficas de administración de bases de datos PostgreSQL, libres y comerciales. En Cuba se ha generalizado el uso de PgAdmin3 por ser software libre y contar con disímiles prestaciones.

[2] **El lenguaje de consulta estructurado** (*SQL*): Es un lenguaje de base de datos normalizado. Está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

Herramienta de administración de bases de datos PgAdmin3

PgAdmin3 es una herramienta de código abierto, de diseño y manejo de bases de datos para su uso con PostgreSQL. La interfaz gráfica está escrita en C++ usando la librería gráfica wxWidget lo que permite que sea multiplataforma. Puede ver y trabajar con casi todos los objetos de las bases de datos, examinar sus propiedades y realizar tareas administrativas. Soporta versiones de servidores 7.3 y superiores. Fue diseñado para responder a las necesidades de todos los usuarios, desde la escritura de simples consultas SQL a la elaboración de bases de datos complejas. (3)

Actualmente PgAdmin3 es la herramienta de administración más usada en los proyectos productivos que trabajan con PostgreSQL, incluyendo el Departamento de PostgreSQL, esto se debe a que las aplicaciones gráficas propietarias para administrar el gestor de bases de datos PostgreSQL, a pesar de su potencia son exageradamente costosas, además adquirir las licencias de software privativo se dificulta en Cuba, por pertenecer a la lista de países embargados. Los usuarios que actualmente trabajan con PgAdmin3 deben realizar mayor esfuerzo para explotar sus funcionalidades, pues posee deficiencias en las interfaces visuales al no permitir interactuar fácilmente con el gestor, por ejemplo, el usuario no posee la opción de personalizar el ambiente de trabajo de manera que se muestren las ventanas de la aplicación según sus necesidades. Además resulta complicado incorporar nuevas funcionalidades por la extensión y poca flexibilidad que posee.

Debido a ello en el departamento de PostgreSQL se lleva a cabo el desarrollo de la herramienta de administración de bases de datos HABD que adopta como punto de partida al PgAdmin3, aunque se emplea una estrategia para avanzar en sentido al desarrollo de software basado en plugin, lo que permite a los desarrolladores incorporar nuevos servicios de manera sencilla.

PostgreSQL Empresarial Cubano

El gestor PostgreSQL Empresarial Cubano es basado en PostgreSQL, el cual está dirigido a solucionar las necesidades de las aplicaciones empresariales cubanas. El gestor está conformado por varios componentes: el núcleo es PostgreSQL y en el contorno a él se encuentran una serie de herramientas; materiales para el entrenamiento y soporte que conforman en conjunto, un producto a la altura de las necesidades de las empresas y organismos cubanos.

Entre las herramientas que se hacen necesarias y que conforman el conjunto de aplicaciones, que proporcionan valor agregado al gestor PostgreSQL Empresarial Cubano, se encuentra HABD, herramienta de administración de bases de datos PostgreSQL, esta brinda una interfaz entre el usuario y el gestor, lo que proporciona su mejor aprovechamiento por parte de los usuarios finales. HABD provee una interfaz amigable y una arquitectura que permite a los desarrolladores incorporar nuevos

servicios de manera sencilla por su potente flexibilidad.

Herramienta de administración de bases de datos HABD

La herramienta de administración de bases de datos PostgreSQL HABD provee una interfaz amigable y su arquitectura está basada en plugin, de ahí que una vez creado un componente para solucionar una tarea específica exista un bajo acoplamiento o dependencia con otros componentes, de manera que admita realizar modificaciones al código sin necesidad de transformar y afectar el funcionamiento del sistema. Provee mayor calidad en el software dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización. De esta forma se le pueden agregar gran número de funcionalidades, lo que evita que los usuarios necesiten de varias aplicaciones para resolver un único problema.

En el presente, HABD solo posee un Front-end; estructura que permite instalar y desinstalar los plugins, que constituyen las funcionalidades en las que se encuentran trabajando actualmente el grupo de desarrolladores del proyecto PostgreSQL, imprescindibles para brindar un servicio general. Una vez integrado cada plugin, la aplicación es capacitada para contribuir al intercambio entre el usuario y el gestor PostgreSQL facilitando el trabajo de ambos. HABD carece de una funcionalidad que permita la recuperación del estado de entidades, para llevar a cabo su desarrollo es imprescindible el estudio de herramientas y procedimientos en la realización de este tipo de recuperación enmarcado esencialmente en el SGBD PostgreSQL.

1.2. Procedimientos y herramientas existentes para la recuperación del estado de entidades en bases de datos relacionales

La tecnología de base de datos permite que la información que manejan las empresas y organizaciones se almacenen de forma organizada, facilitando la persistencia de las mismas. Según C.J. Date "*las bases de datos están integradas y por lo regular son compartidas; se emplean para almacenar datos persistentes. Dichos datos pueden considerarse, de manera útil aunque informal, como una representación de entidades*. *El término "entidad" es empleado comúnmente en los círculos de bases de datos para referirse a cualquier objeto distinguible que va a ser representado en la base de datos o cualquier objeto acerca del cual se necesita registrar información*." (2)

Las entidades y atributos del mundo real, se convierten en registros y campos de una base de datos relacional, es decir, los datos se muestran en forma de tablas y relaciones. Las entidades poseen atributos y mantienen relaciones entre ellas, pueden ser tanto objetos materiales como libros o fotografías, se incluyen personas e incluso, conceptos, ideas abstractas, lugar, cosa, o evento de

interés informacional. (2)

Un sistema gestor de bases de datos es el software que gestiona las entidades que contiene una base de datos. El objetivo principal es proporcionar un entorno eficiente a la hora de almacenar y recuperar la información. Entre las acciones que permiten gestionar las entidades se encuentran modificar y eliminar, las cuales tienen la capacidad de cambiar el estado de las entidades[3] de disponible a no disponible. El empleo erróneo o equívoco de estas acciones por un usuario, en determinadas circunstancias pueden comprometer la persistencia de los datos y la capacidad con que se retrocede la acción realizada, que significa la capacidad con que se recupera la entidad perdida.

Comúnmente la forma de evitar dichos errores es restringir el acceso de un usuario a los datos y servicios. Para ello los SGBD ofrecen herramientas de seguridad preventivas para controlar el acceso de usuarios a los datos de las aplicaciones otorgando a los usuarios solo aquellos privilegios requeridos para realizar sus tareas. A pesar de la existencia de estas herramientas preventivas, los usuarios que son autorizados suelen cometer errores.

Las bases de datos constituyen una de las piezas susceptibles de una organización, pues pueden ser afectadas irreversiblemente por pérdida de datos causadas por los mencionados errores de usuario, que provocan generalmente la eliminación involuntaria de una entidad o varias de ellas. Para recuperar errores humanos, es necesario el empleo de herramientas que permitan corregirlos, es decir que recuperen el estado perdido de la entidad cambiándoles los estados específicamente de no disponible a disponible.

La aplicación gráfica PgAdmin3 se comporta ineficazmente al realizar dicha tarea, por carecer de funcionalidades que recuperen automáticamente la entidad que se perdió con la acción realizada, sin embargo se pueden emplear las herramientas de copias de seguridad que posee el SGBD PostgreSQL, pues mediante los procedimientos correspondientes se soluciona indirectamente el problema. En el caso particular del SGBD Oracle es posible emplear un conjunto herramientas que conforman la tecnología para la corrección de errores humanos denominada Flashback.

1.2.1. Copias de seguridad en PostgreSQL

Las copias de seguridad poseen la capacidad de cambiar el estado de las entidades, pues una vez que se realiza una copia, al restaurarla recupera el último estado guardado de cada una de las entidades. Se deben realizar con determinada periodicidad, la cual está en función del tamaño de los datos y de la

[3] **Estado de entidad**: Situación o modo en la que se encuentran una determinada entidad.

frecuencia con que son modificados, esto marca la estrategia de la copia de seguridad. Existen tres formas principales de realizar copias de seguridad:

- ✓ Copias de seguridad de ficheros del sistema operativo.
- ✓ Salvas (Backup) mediante volcados SQL.
- ✓ *Point in Time Recovery*. (4)

Copias de seguridad de ficheros del sistema operativo

El procedimiento de las copias de seguridad de ficheros, se basa en la duplicación de los clúster[4]del Sistema Operativo (SO), no del software instalado. Es un método sencillo e ineficaz, aunque es conveniente tener una copia del mismo. (4)

Desventajas

- ✓ El mayor inconveniente del procedimiento es que obligatoriamente el servidor PostgreSQL debe estar apagado.
- ✓ No se puede recuperar partes del clúster (bases de datos, esquemas, tablas). La recuperación consiste en borrar todo el clúster y descomprimir el fichero de copia de seguridad, por lo que se pierden los datos que se hayan modificado a partir la última copia de la base de datos.

Salvas mediante volcados SQL

Los procedimientos de salva mediante volcados SQL se realizan empleando las herramientas que proporciona el gestor PostgreSQL, estas son muy flexibles y de gran utilidad. Permiten hacer copias de seguridad y restauración de toda la base de datos o partes de ella, clasificadas por dicha razón en copias completas o parciales. (4)

Características de las copias completas:

- ✓ La copia de seguridad lleva a cabo un volcado completo de la base de datos en un instante de tiempo y contiene la información hasta el instante en que comenzó el proceso de backup.
- ✓ Sencillo de realizar.
- ✓ La base de datos puede estar abierta.
- ✓ En bases de datos pequeñas es muy rápido.

[4]**Un clúster**(o unidad de asignación): Es un conjunto contiguo de sectores que componen la unidad más pequeña de almacenamiento de un disco. Los archivos se almacenan en uno o varios clústeres, dependiendo de su tamaño de unidad de asignación. Sin embargo, si el archivo es más pequeño que un clúster, éste lo ocupa completo.

- ✓ Se restaura "todo" o "nada".
- ✓ En bases de datos grandes puede no ser efectivo.

Características de las copias parciales:

- ✓ Se realiza a una parte de la base de datos bien sea a una tabla o esquema.
- ✓ Sencillo de realizar.
- ✓ La base de datos puede estar abierta.
- ✓ Es rápido.
- ✓ Se restaura "todo" o "nada".

El procedimiento para realizar copias y recuperaciones a una base de datos mediante volcados *SQL* es el siguiente*:*

1. El proceso comienza salvando la base de datos mediante la herramienta *pg_dump*, que vuelca una base de datos completa o parcial, bien en texto plano o en un formato propio de PostgreSQL. Se puede recuperar cualquier objeto que se encuentre en el fichero aisladamente. El servidor debe estar en marcha. Es un programa cliente de la base de datos (como *psql*[5]*)*, lo que significa que se puede utilizar para hacer copias de bases de datos remotas, siempre que se tengan privilegios para acceder a todas sus tablas, es decir, ser el usuario administrador de la base de datos. (4)
2. Para restaurar la copia de la base de datos realizada con *pg_dump* se emplea la herramienta *pg_restore*, que recupera los objetos volcados en una copia de seguridad que se realizó en un formato propio de PostgreSQL. (4)
3. La herramienta *pg_dump_all* se utiliza para copiar todas las bases de datos de un clúster. Incluye una opción interesante que permite exportar únicamente los objetos globales, con la que se puede generar un script que contenga la creación de roles de usuarios y grupos. (4)
4. La herramienta *psql* se emplea para recuperar los ficheros de volcados en texto plano con sentencias *SQL*. Si lo que se necesita recuperar es un clúster completo o la parte global (roles/usuarios), siempre se debe restaurar en la base de datos. (4)

5**Psql**: Es la herramienta canónica para trabajar en modo línea de comandos con PostgreSQL. Herramienta completa para poder manipular las bases de datos pues permite escribir consultas de forma interactiva y ver los resultados de la consulta.

Point in Time Recovery

Point in Time Recovery (PITR) es un tipo de backup avanzado utilizado en sistemas PostgreSQL que trabajan con datos significativos, los cuales no pueden perderse en caso de fallo. (5) PITR no es más que el almacenamiento y copia continua de todas las transacciones producidas por PostgreSQL a partir del último backup realizado a nivel de sistema de fichero[6]. Estas copias de seguridad se podrán usar en caso de un fallo grave de hardware con pérdida de la zona de datos o por necesidad de restaurar una base de datos en un determinado momento del pasado.

A partir de la versión 8.0, PostgreSQL permite actualizar la copia de seguridad con los cambios que proporcionan en los ficheros *log*[7] que hayan sobrevivido. Para poder utilizar esta opción tiene que estar habilitado el archivado *WAL* y en funcionamiento. (6)

El procedimiento de Point in Time Recovery cuando está activado se realiza de la siguiente manera:

1. PostgreSQL almacena todos los ficheros *WAL (WriteAhead Log* - información sobre las transacciones realizadas) generados por el sistema. Este almacenamiento es continuo y no se detiene una vez activado.
2. Cada cierto tiempo se debe realizar la denominada copia de seguridad base. Esta copia de seguridad se realiza a nivel del sistema de ficheros, sin apagar PostgreSQL aunque se comporte inconsistentemente.
3. Una vez terminada la copia de seguridad base, se borran todos ficheros *WAL* antiguos que no se necesiten.
4. En caso necesario, se puede utilizar la copia de seguridad inconsistente, copia de seguridad base, más todos los ficheros *WAL* archivados desde el término de esta copia hasta el momento del incidente, para restaurar la base de datos a un estado consistente y sin pérdida de datos. (5)

De la misma manera se puede devolver a la base de datos al estado en que se encontraba en un determinado momento, de ahí que se llame “recuperación a un punto del tiempo”. De tal forma que si

[6] **Sistema de ficheros**: Es la organización lógica de un dispositivo que permite almacenar y recuperar información en forma de fichero. Son los métodos y estructuras de datos que un sistema operativo utiliza para seguir la pista de los archivos de un disco o partición, es decir, es la manera en la que se organizan los archivos en el disco.

[7] **Ficheros *log***: Son ficheros de texto plano que almacenan todo lo que sucede mientras se está empleando el SO Linux. Son escritos por el núcleo, los servicios o las aplicaciones. Están ubicados en el directorio /var/log.

existe un fallo por borrado de una tabla importante con transacción confirmada, se soluciona recuperando la copia de seguridad y las transacciones hasta que se borró la tabla.

Las herramientas *pg_start_backupypg_stop_ backup* permiten recuperaciones PITR.

Según la experiencia adquirida con la investigación, los procedimientos de las copias de seguridad en PostgreSQL demandan significativos conocimientos de sistemas generalmente basados en Unix, del tipo de salva que se emplee y del gestor PostgreSQL. Además, el método tradicional restablece la base de datos desde un backup y recupera hasta el momento anterior al error, si el tamaño de la base de datos es monumental, puede tardar horas o incluso días restaurarla.

En términos de seguridad y confiabilidad se reconoce al tipo de copia de seguridad PITR, pues posee la capacidad de recuperar a partir de un punto en el tiempo, lo que soluciona el problema de la pérdida de datos desde que ocurrió el fallo hasta que se realizó la última salva de la base de datos, pues PITR guarda transacciones después de dicha salva, sin embargo de la misma forma que las demás salvas, implica a la base de datos en su totalidad y puede ocasionar la pérdida de las acciones realizadas después del fallo.

El procedimiento de las copias de seguridad mediante PITR es fácil de entender teóricamente, pero la práctica y administración son complicadas y engorrosas si se quiere asegurar la calidad y utilidad de este tipo de copia de seguridad. El gestor PostgreSQL proporciona la infraestructura necesaria para poder realizar salvas del tipo PITR, pero son los usuarios los que deben crear todo lo necesario para ponerla en práctica, a partir de un extensivo conocimiento de su correcto procedimiento. Para implementar PITR se necesitan conocimientos de administración de sistemas Linux/Unix y como trabajar con LVM (Linux Volumen Manager) para administrar discos y particiones. (5)

1.2.2. Tecnología Oracle Flashback

La tecnología Flashback es un grupo de herramientas de bases de datos Oracle que proporciona una interfaz SQL para analizar y reparar rápidamente los errores humanos, pues permite ver y devolver los últimos estados de los objetos de una base de datos. Provee varias alternativas para ver el estado pasado de los datos, “rebobinándolos” hacia atrás y adelante sin necesidad de una restauración de la base de datos desde un backup. Es fácil de utilizar, mediante un solo comando breve para recuperar toda la base de datos, en lugar de seguir algún procedimiento complejo. (7)

De esta manera la tecnología *Flashback* puede revertir los cambios no deseados más rápidamente y con menos impacto en la disponibilidad de la base de datos. Las características de Flashback ofrecen la posibilidad de consultar las versiones anteriores de objetos de esquema, consulta de datos

históricos, el análisis del cambio, y llevar a cabo auto-servicio de reparación para recuperarse de la corrupción lógica, mientras que la base de datos está en línea. Soporta la recuperación en todos los niveles, incluso las filas, transacciones, tablas, espacios de tabla y bases de datos. (7)

Aunque las ventajas de Flashback son muchas, tiene algunas limitaciones:

- ✓ Si la corrupción de la base de datos es a nivel físico, no se puede usar Flashback como medio de recuperación.
- ✓ Puede volver atrás cambios no deseados realizados en un pasado cercano lo cual es configurable y limitado según la cantidad de transacciones que se realicen en la base de datos y el espacio disponible para guardarlas. (7)

Las herramientas de alternativas de uso son las siguientes:

- ✓ *Oracle Flashback Query (Flashback Consulta*): consulta cualquier dato del pasado. Esta característica se emplea para ver y reconstruir los datos corruptos que se han eliminando o cambiado involuntariamente. (7)
- ✓ *Oracle Flashback Version Query (Flashback Versión Consulta: p*osee la capacidad de recuperar diferentes versiones de una fila a través de un intervalo de tiempo especificado. El administrador tiene visibilidad de los valores en la medida en que fueron modificados por diferentes transacciones a lo largo del período, por lo que le otorga la capacidad de detectar exactamente cuándo y cómo se han cambiado los datos, esto proporciona un gran valor, tanto en la depuración de aplicaciones como en la reparación de datos. (7)
- ✓ O*racle Flashback Transaction Query (Flashback Consulta de Transacciones)*: permite consultar todos los cambios realizados por una transacción específica. Se puede obtener el código SQL, para deshacer los cambios en las filas particulares afectadas por la transacción de modo que permite volver atrás cada cambio realizado, así como recuperar los datos históricos para una operación determinada o para todas las transacciones dentro de un intervalo de tiempo dado. (7)
- ✓ *Oracle Flashback Table* (Flashback Tabla): restaura una tabla a su estado, en un punto anterior. Proporciona una solución rápida, en línea para la recuperación de una tabla o un conjunto de tablas que han sido erróneamente modificadas por un usuario o aplicación. En la mayoría de los casos, alivia la necesidad de los administradores para llevar a cabo la complicada recuperación PITR, incluso después de un retroceso, se puede volver al estado original de los datos. (7)

- ✓ *Oracle Flashback Database* (Flashback Base de Datos): restablece toda una base de datos hasta un punto específico. Provee una alternativa directa y eficiente para realizar recuperaciones PITR. Con la tecnología Flashback esta tarea se realiza más rápido porque no es necesario restaurar desde un backup. Es rápido, pues solo restablece bloques que han cambiado, además fácil de utilizar y eficiente, porque puede literalmente, restablecer una base de datos en cuestión de minutos. (7)
- ✓ *Oracle Flashback Drop*: recupera una tabla eliminada. Esta función anula los efectos del *Drop Table* (Borrado de tabla). (7)

Ejemplo de la herramienta para recuperar una tabla mediante la sentencia *DROP*. El procedimiento es el siguiente:

1. Cuando se borra una tabla, la base de datos no libera inmediatamente el espacio asociado con la tabla. La base de datos renombra la tabla y la coloca junto sus objetos asociados en el *recycle bin.*
2. En caso de que se haya borrado por error, puede ser recuperada posteriormente. Mediante la opción llamada *Flasback Drop* y se utiliza la sentencia *Flasback Table* para restaurar la tabla. (7)

El *recycle bin* es una tabla del diccionario de datos que contiene información sobre objetos borrados. Las tablas borradas y sus objetos asociados no se borran, siguen ocupando espacio, solo que no aparecen en el listado de objetos disponibles. Cada usuario puede consultar sus objetos en *recycle bin* mediante comandos.

El *Flashback Table* mantiene automáticamente todos los atributos de la tabla, como por ejemplo los índices, *triggers y constraints*.

Conclusiones

Los procedimientos de salva son eficientes y útiles para el objetivo con que fueron creados, sin embargo, para recuperar el estado de una entidad es necesario comprometer la base de datos completa o parte de ella, en algunos casos, para lograr la recuperación. En desventaja dependen en gran medida del tamaño de la base de datos, pues si esta es colosal puede llevarse considerable tiempo en la restauración y recuperación; debido a ello resulta un proceso engorroso que arriesga la integridad y persistencia de los datos almacenados.

Oracle posee el grupo de herramientas llamadas *Oracle Flashback Technology* que puede ser usada para volver una BD atrás en un punto en el tiempo opuesto al tradicional PITR. En general, las

características *flashback* son más eficientes y menos perjudiciales que la media de restauración en la mayoría de las situaciones en las que se aplican, pues poseen la ventaja de “rebobinar” los datos hacia atrás y hacia delante sin necesidad de una restauración de la base desde un backup, aunque sea utilizando PITR.

Estas herramientas se acercan en gran medida al objetivo del presente trabajo, pues su procedimiento se caracteriza por ser menos dificultoso con respecto a las copias de seguridad en PostgreSQL, sin embargo es inaccesible la investigación del código fuente, del funcionamiento o procedimiento que emplea por pertenecer al privativo y comercial SGBD Oracle.

Debido al estudio realizado sobre los procedimientos y herramientas que recuperan el estado de una entidad en bases de datos PostgreSQL, la investigación asume la inexistencia de una aplicación que permita este tipo de recuperación de forma automatizada y sin tener que comprometer la persistencia de los demás datos, en consecuencia se decide desarrollar un plugin para integrar en HABD que le incorpore dicha funcionalidad, prescindiendo de las herramientas que provee PostgreSQL, para ello se define un nuevo procedimiento para guiar la implementación del sistema.

1.3. Metodología de desarrollo de software

La metodología de desarrollo de software es un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. (8) Guían su ciclo de vida y son indispensables para garantizar la satisfacción del cliente, lo que justifica su extendida utilización.

Metodología *eXtreme Programming*

La metodología ágil XP (eXtreme Programming en español Programación Extrema) es un conjunto de valores, principios y prácticas para el desarrollo vertiginoso de un software de alta calidad que proporciona el valor más alto para el cliente en el menor tiempo posible. Fue creada por Kent Beck, en esta se utiliza un enfoque orientado a objetos y con su empleo se logra la confección de productos fáciles de usar, rápidos, fiables y robustos contra los fallos. (9) El objetivo principal que persigue XP es la satisfacción del cliente, que debe estar disponible todo el tiempo para convertirse de esta forma en miembro del equipo. Otro objetivo es potenciar al máximo el trabajo en equipo y enfatizar la convivencia con el cambio, por lo se ajusta en proyectos con requisitos imprecisos y muy cambiantes y donde existe un alto riesgo técnico.

Entre los principios básicos de XP, incluyen simplificar el diseño para agilizar el desarrollo, pues XP aboga por el diseño evolutivo en lugar del diseño planeado, empezando por lo más sencillo que

funcione en el presente, luego se transforma con más complejidad si muestra insuficiencias, de esta forma se facilita el mantenimiento mediante la refactorización del código. XP hace especial énfasis en equipos de desarrollo pequeños con el coraje suficiente para mantener la simplicidad lo que permite diferir decisiones de diseño. (10). Además solicita a cada miembro que construya lo más sencillo que funcione, pues se basa en hacer algo simple en ese momento y crear un ambiente donde el coste del cambio sea lo más bajo posible.

Se establece el empleo de la metodología XP en el desarrollo del plugin de recuperación del estado de entidades pues sus características se ajustan a las necesidades de esta investigación, además es diferente al resto de las metodologías por su énfasis en la adaptabilidad, por lo que está diseñada para ofrecer el software que el usuario necesita, en el momento requerido. Esta es una de las necesidades imprescindibles del equipo de investigación debido a su inexperiencia y a su enfrentamiento en la creación de una herramienta desconocida e innovadora. XP adopta un enfoque “extremo” para el desarrollo iterativo, por lo que se centra más en el código que en la documentación formal, porque se enfoca en el producto final, una de las premisas que guían el presente trabajo.

En la metodología empleada hay que destacar una serie de peculiaridades que la diferencian de otras. Los procesos empiezan con la definición de las “tarjetas de historias” o historias de usuario, las cuales se definen de forma conjunta con el cliente, enmarcando desde el proceso inicial la relevancia que tiene su participación. En el presente trabajo el cliente pertenece al proyecto PostgreSQL, por lo se encuentra disponible para colaborar con la realización de un producto a la altura de las expectativas.

Otra particularidad que introducen los procesos basados en XP es el trabajo en parejas que tiene como ventaja la detección de errores en la medida en que son introducidos. El objetivo principal es repartir la propiedad y responsabilidad del código entre todos los integrantes del equipo de desarrollo, de forma tal que tanto los méritos como los errores no se atribuyan de forma individual, esta particularidad se adapta perfectamente por estar formado por dos integrantes que se proponen lograr un producto de alta calidad con un diseño óptimo.

1.4. Tecnologías y herramientas asociadas al desarrollo del plugin de recuperación del estado de entidades

Para el desarrollo del plugin de recuperación del estado de entidades, se tuvo en cuenta la selección de las herramientas y tecnologías con las que se diseñó e implementó para lograr una integración adecuada con HABD.

1.4.1. Sistema gestor de bases de datos PostgreSQL 9.1

El plugin de recuperación del estado de entidades es desarrollado para su uso con bases de datos PostgreSQL, debido a ello es necesario el empleo del gestor en la implementación. El plugin se crea sobre la base de la última versión del gestor PostgreSQL, por ser recomendable dicha práctica, sin embargo puede ser utilizado en cualquiera de las versiones inferiores de PostgreSQL, pues la implementación es transparente y homogénea. La última versión se encuentra instalada en los proyectos productivos de la universidad, donde se explotan las principales características que se le han añadido.

Entre estas se hallan la replicación sincrónica, las tablas de tipo Unlogged y la implementación del estándar SQL/MED: Foreign Data Wrappers. Provee la definición del lenguaje por columnas y expresiones comunes de tablas modificables. Incluye las nuevas utilidades para la replicación: vista pg_stat_replication que muestra todas las réplicas y sus estados y el comando pg_basebackup para la clonación de bases de datos usando el puerto 5432. Una de sus nuevas funciones es Serialized Snapshot Isolation (SSI) que permite el aislamiento real de transacciones concurrentes, lo cual mejora considerablemente el rendimiento del gestor, al haber menos bloques de tipo explícito o los llamados deadlocks. Conjuntamente posibilita los triggers en vistas y la reducción del tamaño del tipo de dato NUMERIC.

1.4.2. Lenguaje de programación

Es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente. (11)

Lenguaje de programación C++

El lenguaje C++ es imperativo derivado del C, es procedural (orientado a algoritmos) y orientado a objetos. Una de sus propiedades es la reutilización del código en forma de librerías de usuario. (12) Proporciona un acceso a bajo nivel de hardware solamente igualado por el ensamblador y es un lenguaje de propósito general que se adapta a múltiples situaciones. El C++ mantiene las ventajas del C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia; sin embargo, ha eliminado algunas de las dificultades y limitaciones del C original. El lenguaje C++ proporciona un gran salto cualitativo frente a C, al aportar nuevas características útiles en diversos contextos. Por ejemplo, la sintaxis de clases y objetos permite manipular convenientemente diversas estructuras de datos y operaciones y las excepciones permiten procesar de un modo claro los casos de error.

Se decide emplear C++ en el desarrollo del plugin por ser un lenguaje versátil, potente y para lograr una integración apropiada:

- ✓ Con el SGBD PostgreSQL pues sus librerías fueron programadas en el lenguaje C, antecesor de C++, y las últimas funcionalidades en C++.
- ✓ Con la tesis precedente, pues su desarrollo estuvo enmarcado con dicho lenguaje, además para obtener una homogeneidad entre los plugins desarrollados definiéndose en la arquitectura de proyecto el empleo del lenguaje por dicha razón.

1.4.3. Framework de desarrollo

En el desarrollo de un software, un framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos concretos, sobre esa base otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Framework de desarrollo Qt 4.7

Qt es un framework libre y de código abierto para el desarrollo de aplicaciones multiplataforma y se utiliza para crear interfaces gráficas. Posee un conjunto de herramientas para facilitar su empleo, una excelente documentación, ofrece una suite de aplicaciones para proveer y agilizar las tareas de desarrollo y significativamente tiene una arquitectura capacitada para el desarrollo basado en plugin. (13)

Se emplea el framework en el desarrollo del plugin de recuperación del estado de entidades, debido a la característica de ser rápido para desarrollar desde la salida de Qt Creator 2.0, su entorno integrado de desarrollo. La utilización de C++ de forma nativa le hace menos vulnerable a los fallos y responde con una rápida velocidad por estar escrito en dicho lenguaje. (14) Provee varias clases para facilitar ciertas tareas de programación como la comunicación con bases de datos, pues entre sus módulos se encuentra *QtSQL,* que permite la integración con bases de datos e incluye los modelos de datos que permiten editar las tablas de éstas.

Presenta un buen diseño orientado a objetos y una mayor reutilización de código para agilizar el proceso de desarrollo de aplicaciones visuales, lo que trae consigo que sea más fácil el trabajo diario del programador, razones por la cual fue seleccionado en la arquitectura del proyecto.

1.4.4. Entorno integrado de desarrollo

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

Entorno integrado de desarrollo Qt Creator 2.0

Qt Creator es un excelente IDE, multiplataforma para desarrollar aplicaciones en C++ de manera sencilla y rápida pues posee un avanzado editor de código C++, característica que determina su empleo en el desarrollo del plugin de recuperación del estado de entidades. Es basado en la librería Qt y posee herramientas para la administración y construcción de proyectos, depurador visual, GUI (Interfaces gráficas de usuario) integrada, diseñador de formularios y auto-completado de código. (15) Qt Creator se integra con la mayoría de los sistemas de control de versiones incluyendo *Subversion*, Perforce, y otros. Fue seleccionado en la arquitectura del proyecto.

1.4.5. Lenguaje de modelado

El lenguaje de modelado está formado por un conjunto de símbolos que permiten modelar parte de un diseño de software orientado a objetos. Se utiliza extensivamente en combinación con una metodología de desarrollo de software para avanzar de una especificación inicial a un plan de implementación y para mostrar y comunicar dicho plan a todo un equipo de desarrolladores. (16)

Lenguaje Unificado de Modelado

"UML" son las siglas de *Unified Modeling Language* significa en español Lenguaje Unificado de Construcción de Modelos, notación (esquemática en su mayor parte) con que se construyen sistemas por medio de conceptos orientados a objetos. UML se define como un lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software orientados a objetos. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas en todo el ciclo de vida de desarrollo de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. (17)

1.4.6. Herramienta de modelado

Una herramienta CASE (*Computer Aided Software Engineering* significa en español Ingeniería de Software Asistida por Computadora) es un conjunto de programas y ayudas que proporcionan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo

de vida de desarrollo de un software. (18) Entre las tareas se encuentran el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente a partir del diseño, compilación automática, documentación o detección de errores entre otras.

Visual Paradigm 8.0

Visual Paradigm es una herramienta CASE libre, multiplataforma de modelado *UML*, caracterizada por su significativa potencia. Soporta el ciclo de vida completo del desarrollo de software. Permite dibujar todos los tipos de diagramas de clases. Posee un diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad, uso de un lenguaje estándar común al equipo de desarrollo que facilita la comunicación, modelo y código que permanece sincronizado en todo el ciclo de desarrollo y compatibilidad entre ediciones. (19)

Se determina su empleo en el desarrollo del plugin por su rápida actualización en correspondencia con el nuevo desarrollo de técnicas del lenguaje de modelado UML 2.1, por brindar ayuda a los programadores garantizando la generación de bases de datos y código a partir de los diagramas UML para lenguajes como C++, java, php, Ada y Python. Provee la realización de ingeniería inversa, generación automática de informes en formato PDF, Word o HTML, y conjuntamente su empleo se determinó en la arquitectura del proyecto. Además permite trabajar con el sistema de control de versiones *Subversion*.

1.4.7. Controlador de versiones

Un sistema de control de versiones es un sistema centralizado que se utiliza generalmente para compartir información. Permite la gestión de archivos y directorios, y sus cambios a través del tiempo, además de la conexión de múltiples usuarios al repositorio para leer o escribir.

***Subversion* 1.6**

Subversion es un sistema de control de versiones libre, de código fuente abierto y multiplataforma empleado para la administración de proyectos y el control de versiones de ficheros electrónicos, como son el software o la documentación. Permite la realización de cambios y mantiene un historial actualizado de los mismos. (20) Se establece el empleo en el desarrollo del plugin de recuperación del estado de entidades por ser un sistema general que puede usarse para administrar cualquier conjunto de ficheros y por poseer al cliente gráfico *RapidSVN* que permite manipular los repositorios de *Subversion;* una de las alternativas distinguidas para los sistemas GNU/Linux, muy intuitivos y fáciles de utilizar además su empleo fue definido por la arquitectura de proyecto

Conclusiones Parciales

A partir de la investigación realizada acerca de las tecnologías y herramientas se asume el empleo del lenguaje C++ y la metodología XP idónea para el desarrollo del plugin de recuperación del estado de entidades pues sus características se ajustan adecuadamente a las necesidades del plugin. El lenguaje C++ determina el empleo de tecnologías como el framework de desarrollo Qt, el IDE Qt Creator. Se estableció el empleo de la herramienta CASE Visual Paradigm y del controlador de versiones Subversion.

El estudio realizado acerca de los procedimientos y herramientas para la recuperación del estado de entidades arrojó la conclusión de ser ineficaces para utilizarlos en el problema planteado. El conocimiento adquirido no aporta utilidad para la aplicación práctica en el desarrollo del plugin, sin embargo teóricamente se reconoce la existencia de herramientas capaces de corregir errores humanos de forma rápida y fácil empleo. Como consecuencia de la imposibilidad de adquirir este conocimiento para materializar las funcionalidades requeridas y el estudio de las herramientas que se tienen al alcance, no garantizan la total cobertura que se necesita para cumplir con el objetivo trazado, se decide desarrollar un plugin para la recuperación del estado de entidades empleando un enfoque diferente a lo estudiado, haciendo uso de la innovación de su equipo de desarrollo.

CAPÍTULO 2: ANÁLISIS Y DISEÑO

Introducción

La metodología XP deja en manos del equipo de desarrollo los artefactos a utilizar para la implementación, en dependencia de las capacidades de comunicación, se decide generar tantos tipos de diagramas UML como se necesiten para así facilitar el proceso de desarrollo. En el presente capítulo se expondrán las principales características del sistema, su diseño, arquitectura y los requisitos, tanto funcionales como no funcionales, a tener en cuenta en su desarrollo. Está integrado por las fases: Exploración, donde se describe la herramienta que se propone a través de las historias de usuarios, en la que se obtiene la lista de reserva del producto y la fase Diseño que recoge las tarjetas CRC (clase, responsabilidad, colaboración) así como la arquitectura de software.

2.1 Modelo de dominio

La metodología XP se inspira en la simplicidad[8], sin embargo su gran adaptabilidad permite el empleo de diagramas UML, siempre y cuando influyan en el mejoramiento de la comunicación. Es por ello que se decide realizar un modelo de dominio, pues permite a los usuarios, clientes, desarrolladores e interesados, a utilizar un vocabulario común para lograr un efectivo entendimiento del contexto en que se encuentra el sistema proporcionando una perspectiva conceptual de los objetos implícitos en él. Se realiza su descripción a través de un diagrama de clases UML. (Figura 1)

Según Ivar Jacobson, Grady Booch, y James Rumbaugh*, "un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las cosas que existen o los eventos que suceden en el entorno en el que trabaja el sistema."* (21) Es una representación visual de clases conceptuales o de objetos reales en un dominio de interés.

Diagrama de clases del modelo de dominio.

8**Simplicidad:** Característica de XP que consiste en desarrollar solo el sistema que realmente se necesita. Implica resolver en cada momento solo las necesidades actuales.

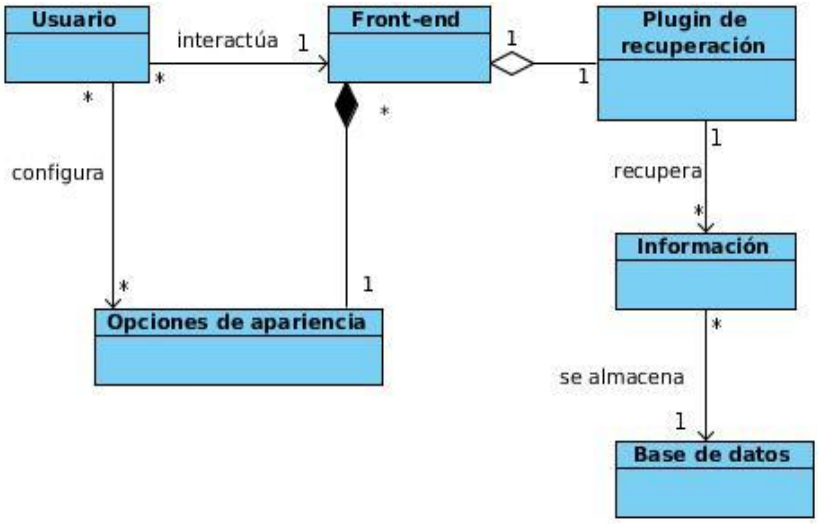


Figura 1: Modelo de Negocio

Definición de las clases del modelo de dominio.

Usuario: Persona que interactúa con el Front-end.

Front-end: Armazón o estructura que permite cargar, manipular y desinstalar cada plugin adicionado por el usuario.

Plugin de recuperación: Módulo que añade la nueva funcionalidad al Front-end de la recuperación del estado de entidades.

Opciones de apariencia: Opciones funcionales que le permite al usuario hacer cambios en la apariencia del Front-end.

Información: Conjunto de datos y conocimientos contenidos en las bases de datos.

Base de datos: Colección de datos organizados y estructurados.

2.2 Descripción del sistema propuesto

Para solucionar el problema planteado se decide desarrollar un plugin para la herramienta de administración de bases de datos HABD, que incorpora la funcionalidad automática de la recuperación del estado de una entidad que haya sido eliminada. Permite que la pérdida de datos por error de usuario sea reversible de una manera rápida en correspondencia con el procedimiento que utiliza. Su diseño pretende colaborar con la persistencia de los datos y evitar la interferencia en la confidencialidad y persistencia de la información almacenada en las bases de datos. La implementación del plugin está condicionada por el siguiente procedimiento (Figura 2):

1. Los usuarios seleccionan la tabla que necesite de la aplicación del procedimiento. Pueden seleccionar todas las tablas de la base de datos una por una.
2. Por cada tabla (Ej. de tabla con nombre T) seleccionada se crean tres tablas con los nombres (T_insert, T_update y T_delete), tres trigger[9] con los nombres (T_insert, T_update y T_delete) y tres trigger function con los nombres (T_insert, T_update y T_delete) cada uno corresponde a las acciones insertar, modificar y eliminar respectivamente.
3. Una vez aplicado el procedimiento a la tabla T, por ejemplo si se elimina un dato de dicha tabla, se dispara el trigger T_delete que invoca al trigger function T_delete, este último almacena la entidad eliminada, la fecha y la hora en la tabla T_delete, así mismo sucede con las demás tablas T_insert y T_update si insertan o modifican datos en la tabla T respectivamente. En los Anexos 1 y 2 se encuentra una descripción en la que se especifica el procedimiento.

Tiene como propósito almacenar los datos imprescindibles de las acciones insertar, modificar y eliminar que ocurran sobre las entidades. La información luego es utilizada para recuperar la entidad eliminada o actualizada que prescinda el usuario. Finalmente el usuario podrá verificar la entidad recuperada en la tabla original de la base de datos, manteniendo la integridad de los demás datos.

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

[9] **Trigger:** Es una clase que se dispara antes o después de un evento o acción. Implementa una interfaz que dispone de un método en el cual se implementa la tarea que debe ser llevada a cabo, y un método error, en el cual se implementa la tarea que debe ser llevada a cabo en caso de error.

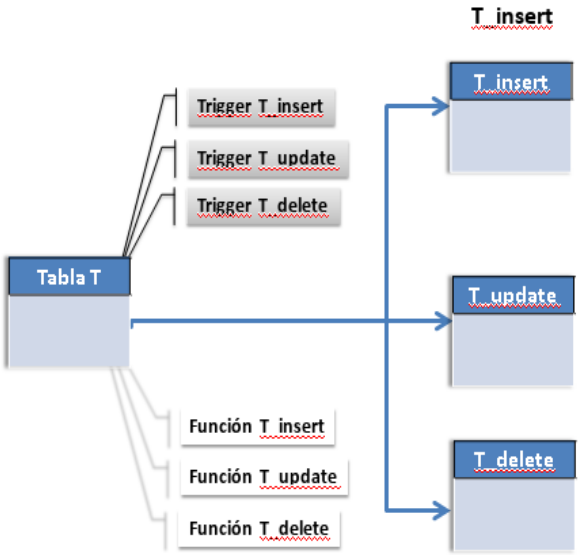


Figura 2: Procedimiento

2.3 Historias de usuario

Las historias de usuario son la técnica utilizada en XP para especificar y administrar los requisitos del software de una forma eficaz. Son descripciones cortas y escritas en el lenguaje del usuario sin terminología técnica, es por ello que son lo suficientemente comprensibles y delimitadas para que los programadores puedan implementarlas en unas semanas. Permiten responder rápidamente a los requisitos cambiantes pues su tratamiento es dinámico y flexible. (22)

Para el presente trabajo de diploma se obtienen un total de 5 HU que son implementadas en 3 iteraciones. A continuación se muestra la HU de mayor trascendencia “Realizar recuperación del estado perdido de la entidad seleccionada” el resto se encuentra en el artefacto “Historias de Usuario” presente en el Expediente de Proyecto.

Historia de Usuario	
Número: 2	**Nombre de la Historia de Usuario:** Realizar recuperación del estado perdido de la entidad seleccionada.
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Alexis Camué Hernández	**Iteración asignada:** 1

Prioridad en negocio: Muy Alta **Puntos estimados:** 1,6

Riesgo en desarrollo: Muy Alta **Puntos reales:** 1,6

Descripción: El usuario accede a la interfaz principal, la cual permite seleccionar mediante una filtración de entidades, o del reporte filtrado o no la entidad o las entidades con cambios en su estado que necesite recuperar, luego se ejecuta la recuperación del estado perdido de las mismas.

Observaciones:

Prototipo de interfaces:

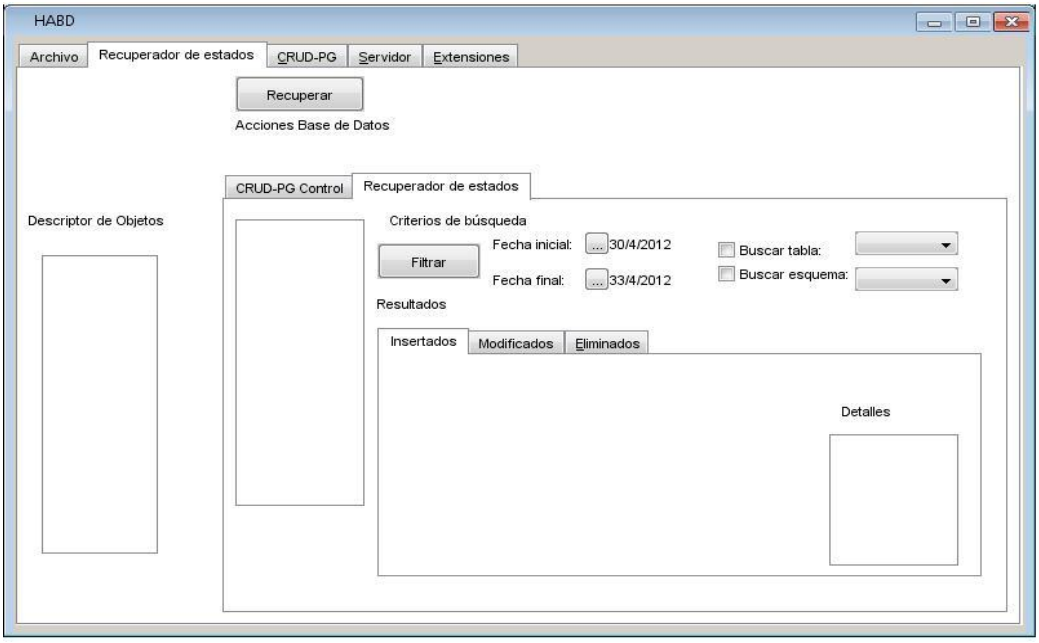


Tabla 1: Historia de usuario “Realizar recuperación del estado perdido de la entidad seleccionada”

Las HU describen las funcionalidades que debe realizar el plugin de recuperación del estado de entidades. Proveen información acerca de la prioridad de la funcionalidad a implementar, así como el tiempo estimado de duración de dicha implementación. A continuación las HU son relacionadas en el artefacto Lista de reserva del producto en la cual se referencia información relevante acerca de las mismas.

2.4 Lista de reserva del producto

La lista de reserva del producto es una tabla que contiene los requisitos funcionales que debe cumplir

el plugin que se desea realizar, ordenados según la prioridad en el negocio (Muy Alta, Alta, Media y Baja). Se indica de cada uno de ellos la estimación, su implementación por semanas y el rol que lo estimó. Contiene por último los requisitos no funcionales que requiere el sistema a desarrollar.

Ítem *	Descripción	Estimación	Estimado por
Prioridad: Muy Alta			
1	Aplicar procedimiento a la tabla seleccionada.	1,4	Analista
2	Realizar recuperación del estado perdido de la entidad seleccionada.	1,6	Analista
Prioridad: Alta			
3	Mostrar reportes de las acciones registradas en las tablas creadas.	1,6	Analista
4	Eliminar registros almacenados en las tablas creadas.	1,4	Analista
Prioridad: Media			
5	Desaplicar procedimiento a la tabla seleccionada.	1	Analista
Prioridad: Baja			

Requisitos No Funcionales			
1	**Apariencia o interfaz externa:** Interfaz intuitiva, amigable, organizada, con una navegabilidad flexible y de fácil comprensión. Diseñada para verse en cualquier resolución igual o inferior a 1024x768.		Analista

2	**Software:** Sistema Operativo: Multiplataforma. Librerías Qt Servidor de Bases de datos: SGBD PostgreSQL 9.1.	Analista
3	**Hardware:** Se necesita 100 MB de memoria RAM mínimo, 1GB de espacio libre en el disco duro para su instalación y el micro a 300 MHz.	Analista
4	**Rendimiento:** El tiempo de respuesta dependerá de la cantidad de entidades a recuperar y de la cantidad de datos contenidos en las bases de datos.	Analista
5	**Portabilidad:** El plugin es instalado en la herramienta de administración de bases de datos HABD la cual se podrá disponer en cualquier sistema operativo.	Analista
6	**Integridad:** El plugin mediante la recuperación tendrá la capacidad de hacer cambios en los datos, recuperando el estado de las entidades, los demás datos almacenados mantendrán su integridad.	Analista
7	**Disponibilidad**: Se podrá hacer uso de la aplicación siempre que esté instalada y existan entidades para recuperar su estado.	Analista
8	**Restricciones del diseño y la implementación:** Para el diseño e implementación de la aplicación se utiliza la metodología XP, haciendo uso del Visual Paradigm 8.0 para el	Analista

	modelado. Se utiliza como lenguaje de programación C++, como gestor de BD PostgreSQL 9.1 y como IDE de desarrollo Qt en la versión 4.7.

Tabla 2: Lista de reserva del producto

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

2.5 Plan de iteraciones

La metodología XP aporta mayor valor a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. El plan de iteraciones es empleado para la planificación, donde el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. (23)

Cada historia debe ser implementada en un periodo de una a tres semanas. Aquellas historias que requieran más tiempo se sub-dividen tratando de que resulten atómicas y puedan realizarse dentro del plazo máximo.

Iteración	Descripción de la iteración	Orden de la HU a implementar	Duración total
1	Tiene como objetivo desarrollar las HU con prioridad muy alta las cuales forman la base del sistema a desarrollar. Son las encargadas de realizar el proceso de la creación de tablas, el desarrollo de los triggers para registrar las acciones y finalmente la realización de la recuperación.	HU 1 HU 2	3 semanas
2	El objetivo fundamental de la iteración es implementar las HU de prioridad alta encargadas de mostrar los reportes de la información almacenada y eliminar los datos almacenados en las tablas creadas según necesite el usuario.	HU 3 HU4	3 semanas
3	El objetivo fundamental de la iteración es implementar la HU con prioridad media. Es la encargada de desaplicar el procedimiento a las tablas seleccionadas, elimina las tablas creadas, los trigger y funciones.	HU 5	1 semana

Tabla 3: Plan de iteraciones

2.6 Modelo de diseño

En la fase de diseño, XP propone mejorar la comunicación entre todos los integrantes del equipo, al crear una visión global y común de lo que se quiere desarrollar para lograr un diseño sencillo pues *"Kent Beck dice que en cualquier momento el diseño adecuado para el software es aquel que: supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos",* (23) además Pressman asegura: *"el diseño es el lugar donde se fomentará la calidad del software."* (24)

El diseño en XP aspira a:

- ✓ Mantener el código tan claro y simple como sea posible.
- ✓ Refactorizar el código, de modo que se puedan hacer mejoras cuando se necesite.

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

2.6.1 Diagrama de clases

El diagrama de clases proporciona una abstracción de la implementación del sistema y es utilizado como entrada fundamental de sus actividades. Admite una mejor compresión de los atributos, relaciones y métodos que contienen las clases, de este modo se obtiene una representación significativa de lo que se va a construir, de forma tal que satisfaga todos los requisitos funcionales. (21) A continuación se muestra el diagrama de clases del diseño para el plugin de recuperación del estado de entidades. (Figura 3)

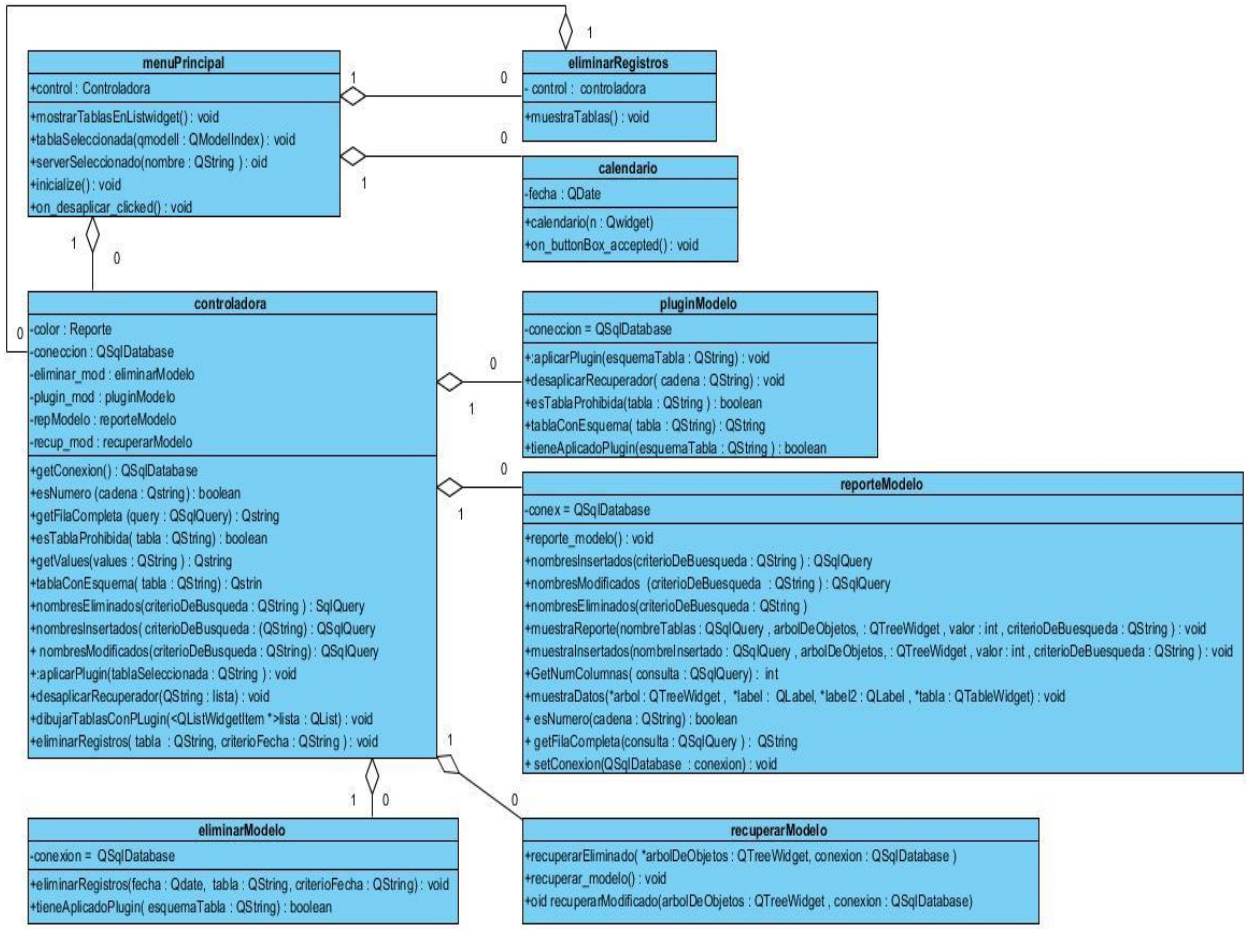


Figura 3: Diagrama de clases

El diagrama de clases muestra las relaciones, atributos y principales métodos que presenta el sistema. La clase *menuPrincipal* agrega objetos de la clase visual *eliminarRegistros, calendario y controladora,* la cual a su vez contiene clases de tipo *pluginModelo, recuperarModelo, reporteModelo y eliminarModelo. La clase* visual *menuPrincipal* representa *los datos de* las clases *pluginModelo, recuperarModelo y reporteModelo*, mientras que *eliminarRegistros* representa a *eliminarModelo.* La clase *calendario* muestra una interfaz que contiene un componente calendario*.*

La clase *pluginModelo* es *la* encargada de aplicar y desaplicar el procedimiento, dígase procedimiento a todo el proceso de creación de tablas y trigger, el cual pone en marcha el almacenamiento de los datos eliminados y la información de las acciones correspondientes. La clase *reporteModelo es la* encargada de mostrar un reporte de todos los datos que han sido modificados, borrados e insertados y la clase *recuperarModelo es* la responsable de acceder a los datos almacenados por el procedimiento, se encarga de recuperar su estado, devolviéndolos a su tabla original tal y como se encontraban. La clase *eliminarModelo* es responsable de borrar los datos almacenados innecesariamente según el

criterio del usuario, de esta forma se evita colapsar la base de datos de información, lo que podría influir en su rendimiento.

2.6.2 Tarjetas Clase, Responsabilidad y Colaboración

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

A continuación se muestran las tarjetas CRC generadas para el diseño del plugin de recuperación de entidades.

Tarjetas CRC

Tarjeta CRC	
Clase: Tabla	
Responsabilidades	Colaboraciones
Mostrar reportes. Eliminar datos almacenados.	Recuperación

Tabla 4: Tarjeta CRC “Tabla”

Tarjeta CRC	
Clase: Recuperación	
Responsabilidades	Colaboraciones
_	Tabla

Tabla 5: Tarjeta CRC “Recuperación”

Tarjeta CRC	
Clase: Procedimiento	
Responsabilidades	Colaboraciones

Aplicar. Desaplicar.	Tabla

Tabla 6: Tarjeta CRC “Procedimiento”

Los datos de una aplicación, la interfaz de usuario y la lógica de negocio se deben separar en tres componentes distintos y estructurar arquitectónicamente, para ello se utiliza los patrones de arquitectura.

2.7 Patrones de arquitectura

El diseño en XP aspira a:

- ✓ Conocer adecuadamente los patrones, no solo las soluciones sino también apreciar cuándo se usan y se evoluciona hacia ellos.
- ✓ Saber cómo comunicar el diseño a las personas que necesitan entenderlo, con el uso de código, diagramas y sobre todo conversación.

Un estilo arquitectónico es una transformación impuesta al diseño de todo un sistema que tiene como objetivo establecer una estructura para todos los componentes del mismo. Describe un conjunto de conectores y restricciones que integran dichos componentes. La arquitectura de software define un conjunto de estilos arquitectónicos para sintetizar estructuras de soluciones y son fundamentales para la toma de decisiones del diseño. Dentro de los estilos arquitectónicos de Llamada y Retorno, se encuentra el patrón Modelo-Vista-Controlador (MVC). (25)

El empleo del patrón arquitectónico MVC permite ventajas como la separación del Modelo de la Vista, es decir, separar los datos de su representación visual, pues es más sencillo agregar múltiples representaciones de los mismos datos o información; facilita agregar nuevos tipos de datos según sea requerido por la aplicación. El empleo del patrón provee las ventajas de crear independencia de funcionamiento, facilitar el mantenimiento en caso de errores y ofrecer maneras más sencillas de probar el correcto funcionamiento del sistema.

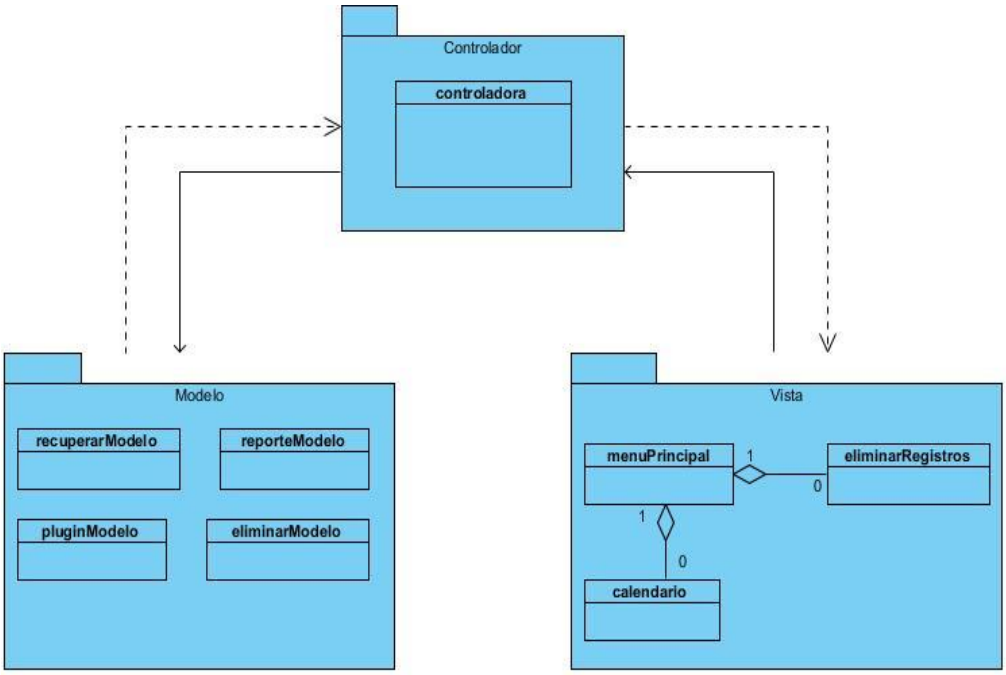


Figura 4: Patrón Modelo - Vista - Controlador

Sus tres componentes son:

Modelo: Contiene todo el código relacionado con el acceso a datos. Es primordial que el código sea lo más genérico posible y que además se pueda reutilizar en otras situaciones y proyectos. Nunca se incluirá lógica en el modelo, solamente consultas a la base de datos y validaciones de entrada de datos.

Vista: Contiene el código que representa la parte que será visualizada en pantalla por el usuario.

Controlador: Es el punto de entrada de la aplicación, se mantiene a la escucha de todas las peticiones, ejecuta la lógica de la aplicación y muestra la vista apropiada para cada caso.

El patrón MVC fue puesto en práctica en el diseño del plugin de recuperación del estado de entidades de la siguiente forma:

El paquete Vista está compuesto por la clase *menuPrincipal*, *eliminarRegistros y calendario*, estas proveen los componentes necesarios para realizar sus operaciones, de forma tal que generan una representación visual del Modelo y muestran los datos al usuario, excepto *calendario*, que muestra un componente calendario. Las vistas para interactuar con el paquete Modelo y llevar a cabo sus responsabilidades, envían eventos y peticiones a la clase *controladora* que se encuentra en el paquete Controlador, éste a la escucha, maneja los eventos de tal forma que invoca la clase del paquete Modelo correspondiente según la petición y proporciona significado a las órdenes del usuario actuando

sobre los datos representados por el Modelo, en el que se encuentran las clases encargadas de representar los datos del programa, manejarlos y controlar todas sus transformaciones.

El Modelo contiene las clases que acceden mediante consultas a las bases de datos, almacenan, examinan y eliminan la información que maneja el plugin de recuperación del estado de entidades. La clase *controladora* interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado, de esta forma en respuesta a una petición de la vista le notifica, el resultado que envía el Modelo.

Un ejemplo del flujo de datos es cuando el usuario accede a la vista *eliminarRegistros,* para borrar los datos almacenados innecesariamente, la vista envía la petición a la *controladora*, la cual accede a la clase del paquete Modelo *eliminarModelo*, encargada de acceder a la base de datos y eliminar los datos almacenados según los criterios introducidos por el usuario. La clase modelo, devuelve los datos a la *controladora,* en este caso un valor positivo, el cual informa a la vista *eliminarRegistros* del resultado, ésta devuelve al usuario un mensaje de confirmación.

2.8 Patrones de diseño

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

Patrones GRASP

GRASP acrónimo que significa *General Responsability Assignment Software Patterns* en español significa Patrones de Software para la Asignación General de Responsabilidad. Las patrones GRASP constituyen la base del cómo se diseñará el sistema. Describen los principios fundamentales de diseño de objetos y la asignación de responsabilidades, expresados como patrones. (17) De los diferentes patrones que ofrece GRASP se han tenido en cuenta para la modelación del plugin de recuperación del estado de entidades los siguientes:

Patrón Experto: Asigna una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad, favorece la robustez y el fácil mantenimiento del sistema. (26) El problema que soluciona el patrón es el siguiente:

¿Cómo asignar responsabilidades de la forma más eficiente?

La clase *controladora* para cumplir con su responsabilidad requiere información distribuida en varias clases de objetos, pues exige la colaboración de cuatro clases. Siempre que la información se encuentre esparcida en varios objetos, estos habrán de interactuar a través de mensajes para compartir el trabajo. Debido a ello el patrón se evidencia en la clase *controladora*, pues contiene toda la información necesaria para obtener datos de las clases *recuperarModelo, reporteModelo, pluginModelo y eliminarModelo.* En la Figura 5 se muestra una representación del acceso de la clase *controladora* a los datos de la clase *recuperarModelo*. (Figura 5)

Figura 5: Ejemplo del uso del patrón experto

El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y cohesivas, que son fáciles de comprender y de mantener, así se brinda soporte a una alta cohesión.

Patrón Creador*:* Guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Se asigna la responsabilidad a una clase de crear cuando contiene, agrega, compone, almacena o usa otra clase, lo que brinda una alta posibilidad de reutilizar la clase creadora. (26) El problema que soluciona el patrón es el siguiente:

¿Qué clase es la responsable de crear una nueva instancia de la clase controladora?

La clase *menuPrincipal* contiene objetos de las clases *controladora* y *eliminarRegistros*; por ello, el patrón Creador sugiere que *menuPrincipal* es idónea para asumir la responsabilidad de crear las instancias de las clases que agrega. (Figura 6)

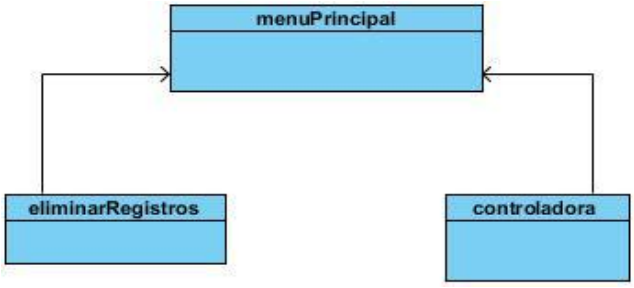


Figura 6: Ejemplo del uso del patrón creador

Se brinda soporte a un bajo acoplamiento, lo cual supone menos dependencia respecto al mantenimiento y mejores oportunidades de reutilización.

Patrón Bajo Acoplamiento: Asigna una responsabilidad de manera que la dependencia entre elementos permanezca baja. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con débil acoplamiento no depende de "muchas otras"; solo de las mínimas necesarias. *(26)* El problema que soluciona el patrón es el siguiente*:*

¿Cómo dar soporte a una dependencia escasa y un aumento de la reutilización en el diseño del plugin?

En el diseño desarrollado existen las mínimas relaciones necesarias entre las clases, se disgregan las responsabilidades más complejas en cada una de ellas, por lo que poseen la información necesaria para cumplir con su responsabilidad, de esta forma son delegadas las funciones a cada clase responsable. El patrón bajo acoplamiento se evidencia de tal forma que soporta el diseño de clases más independientes, lo que reduce el impacto del cambio. No se puede considerar de manera aislada a otros patrones como el Experto o el de Alta Cohesión, sino que necesita incluirse como uno de los diferentes principios de diseño que influyen en una elección al asignar una responsabilidad.

Controlador: Asigna la responsabilidad de administrar un mensaje de evento del sistema a una clase que representa el sistema global, dispositivo, subsistema o representa un escenario de caso de uso en el que tiene lugar el evento del sistema. *(26)* El problema que soluciona el patrón es el siguiente:

¿Qué clase es la responsable de gestionar los eventos del sistema?

La clase controladora representa el sistema global, es un objeto de interfaz no destinada al usuario que se encarga de manejar los eventos del sistema que recibe a través de las interfaces. Delega a otros objetos el trabajo que se necesita hacer; coordina o controla la actividad, de esta manera no realiza mucho trabajo por sí misma. Los beneficios del patrón controlador se deben a un mayor potencial de los componentes reutilizables, el hecho de delegar a un controlador la responsabilidad de la operación de un sistema entre las clases del dominio soporta la reutilización de la lógica para manejar los procesos afines del negocio en aplicaciones futuras.

Las clases que funcionan como controladoras definen los métodos para las operaciones de la aplicación y se encargan del manejo de eventos. La clase *controladora* es la responsable de manejar los eventos y obtener características de las clases que contiene. (Figura 7)

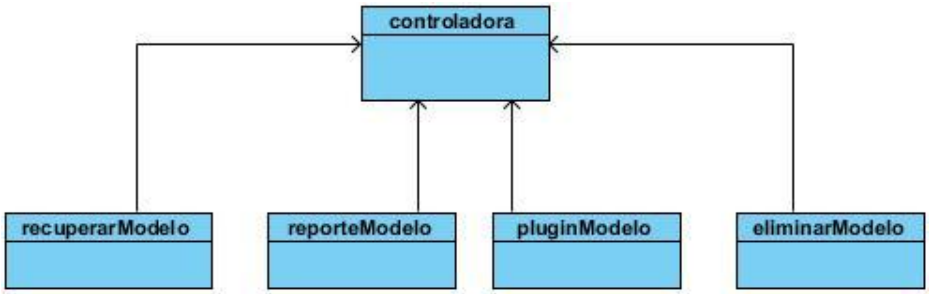


Figura 7: Ejemplo del uso del patrón controlador

Alta Cohesión: Asigna una responsabilidad de modo que la cohesión sea alta. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase, además una alta cohesión garantiza que clases con responsabilidades estrechamente relacionadas no realicen un trabajo enorme. *(26)* Es cuando los elementos de una clase trabajan juntos para proporcionar algún comportamiento bien delimitado. El problema que soluciona el patrón es el siguiente:

¿Cómo mantener la complejidad manejable en el diseño del plugin?

Los algoritmos que satisfacen las principales funcionalidades del plugin se encuentran distribuidos en diferentes clases, es decir, no se centralizan las responsabilidades, cada clase contiene solo la información necesaria para cumplir con su responsabilidad. No existen clases responsables de áreas funcionales muy heterogéneas, ni responsabilidad exclusiva de una tarea compleja dentro de un área funcional. De esta forma se evita que exista baja cohesión sobrecargando de responsabilidades a una misma clase, incrementándose la claridad y comprensión del diseño.

En la clase *pluginModelo* (Figura 8) se evidencia una alta cohesión pues posee solo la información relacionada con la aplicación y desaplicación del procedimiento, funciones con complejidad moderada y potentemente relacionadas, lo que implica que sea una clase fácil de entender, reutilizar y mantener.

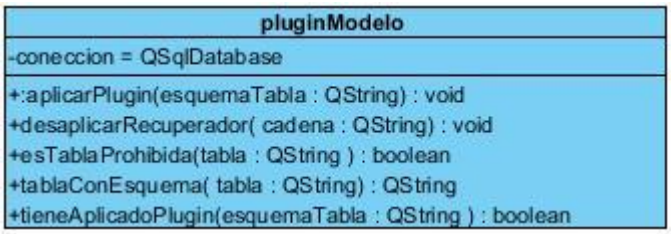


Figura 8: Ejemplo del uso del patrón alta cohesión

Patrones GOF

El libro *Patrones de Diseño*, escrito por **GOF** (*Gang of Four*, "Pandilla de los Cuatro"), recopila una serie de patrones de diseño, agrupados en tres categorías: de creación, de estructura y de comportamiento.

Patrón de Comportamiento

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

En la programación de Interfaces Gráficas de Usuario (GUI) se necesita que los cambios o eventos producidos sobre un objeto sean comunicados al resto de ellos. Qt posee el mecanismo de *Signals y Slots*, el cual posibilita entre estos comunicación segura, flexible y orientada a objetos. Básicamente, al producirse un evento sobre un objeto se emite una señal (Signal), que permite que se ejecute una determinada función (Slot) en respuesta a la señal emitida, debido a ello el patrón observador se encuentra inherente en Qt. A continuación se muestra un fragmento de código del plugin de recuperación del estado de entidades, donde se emplea el patrón Observador al ser declaradas una serie de funciones para responder a un conjunto de señales. (Figura 9)

```
private slots:
    void slt_serverSeleccionado(QString);
    void tabla_seleccionada(QModelIndex qmodelI);
    void on_botonBorrarRegistro_clicked();
    void on_checkBox_2_clicked();
    void on_checkBox_clicked();
    void on_filtrar_clicked();
    void on_listWidget_clicked(QModelIndex index);
    void on_pushButton_3_clicked();
    void on_pushButton_4_clicked();
    void on_pushButton_clicked();
    void on_eliminarRegistro_clicked();
    void on_desaplicar_clicked();
    void on_tabWidget_currentChanged(int index);
    void on_treeWidget_clicked(QModelIndex index);
    void on_treeWidget_2_clicked(QModelIndex index);
    void on_treeWidget_3_clicked(QModelIndex index);
    void on_toolButton_clicked();
    void on_toolButton_2_clicked();
```

Figura 9: Ejemplo del patrón observador

Conclusiones Parciales

La solución al problema planteado constituyó una autónoma innovación, enmarcado por el procedimiento utilizado para efectuar la implementación, aportando al estudio de la recuperación del estado de entidades una nueva visión, un nuevo campo para indagar en nuevos procedimientos que al aplicarlos logren una mejor optimización. Guía al estudio a nuevos procedimientos que logren desprenderse de las necesidades de almacenamiento de las acciones y datos que va almacenado. El empleo de los patrones diseño y el patrón de arquitectura Modelo-Vista-Controlador facilita la

obtención de clases apropiadamente diseñadas, guiadas por el diagrama de clases y la prioridad de las cinco HU identificadas en cada iteración del desarrollo del plugin. Tener en cuenta para el desarrollo los requisitos no funcionales permite la creación de un sistema con calidad.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

La implementación en el proceso de desarrollo de un software adquiere suma relevancia debido a que le provee funcionalidad al producto que se desarrolla. Del mismo modo, son significativas las pruebas que se le realizan al mismo para validar su correcto funcionamiento. En el presente capítulo se desarrolla la descripción de la implementación del sistema y se especifican las pruebas a las que fue sometido el plugin de recuperación del estado de entidades en cada una de las iteraciones, proceso que guía la identificación y corrección de fallos cometidos en las HU, así como su verificación y materialización lo que contribuye a elevar la calidad de los productos desarrollados y la seguridad en los programadores para efectuar modificaciones.

3.1 Implementación del sistema

La implementación del sistema está constituida en primer lugar por el procesamiento de las HU especificadas en el capítulo anterior, para lograr una realización exitosa de su implementación. Se describen los estándares de codificación al que está sujeto el programador, con el objetivo de lograr uniformidad en la implementación de cada plugin que integra la herramienta de administración de bases de datos HABD. Por último se evidencia la aplicación del patrón MVC en Qt y las interfaces diseñadas para el plugin de recuperación del estado de entidades.

3.1.1 Tareas de ingeniería

Las HU se deben desarrollar con la calidad requerida, para ello se lleva a cabo una descomposición de las mismas. Como resultado se generan las tareas de programación (taskcard) o tareas de ingeniería (TI), las cuales representan una característica del sistema. Cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores para ser implementadas durante una iteración y es del uso estricto de ellos. (23)

Para el desarrollo del plugin de recuperación del estado de entidades, se identificaron 12 tareas de ingenierías pertenecientes a las 5 HU documentadas en el primer capítulo. A continuación se evidencian las TI correspondientes a la HU “Realizar recuperación del estado perdido de la entidad seleccionada”, donde se muestra el encargado de programarla y una descripción breve de lo que requiere. El resto de las tareas de ingeniería se encuentran en el artefacto correspondiente “Tareas de Ingeniería” presente en el Expediente de Proyecto.

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** *2*

Nombre Tarea: Seleccionar acción a recuperar

Tipo de Tarea : Desarrollo **Puntos Estimados:** 0,8

Fecha Inicio: 13/16/2012 **Fecha Fin:** 16/2/2012

Programador Responsable: Alexis Camué Hernández

Descripción: Permite seleccionar de las entidades borradas o actualizadas cuáles de ellas se necesitan recuperar.

Tabla 7: Tarea de ingeniería "Seleccionar acción a recuperar"

Tarea de Ingeniería	
Número Tarea: 5	**Número Historia de Usuario:** 2

Nombre Tarea: Ejecutar recuperación

Tipo de Tarea : Desarrollo **Puntos Estimados: 0**,8

Fecha Inicio: 20/2/2012 **Fecha Fin:** 23/2/2012

Programador Responsable: Alexis Camué Hernández

Descripción: Permite recuperar el estado de la entidad seleccionada.

Tabla 8: Tarea de ingeniería "Ejecutar recuperación"

3.1.2 Aplicación del patrón Modelo-Vista-Controlador en Qt

EL entorno integrado de desarrollo Qt Creator posee una estructura particular para organizar las clases, sin embargo la aplicación del patrón arquitectónico Modelo - Vista - Controlador en el diseño del plugin de recuperación del estado de entidades, provocó una reestructuración de las clases que se ubican en el árbol de carpetas del proyecto, quedando organizadas de la siguiente forma:

En la carpeta Headers se fraccionaron los ficheros *.h, se ubicaron según al componente del patrón MVC al que pertenecía, por lo que se obtuvo las carpetas modelo, vista y controlador. En la carpeta modelo se encuentran las cabeceras de las clases encargadas del acceso de la base de datos, los ficheros son eliminarModelo.h, pluginModelo.h, recuperarModelo.h y reporteModelo. En la carpeta controlador se localiza el fichero controladora.h y en la visual se encuentran los ficheros eliminarRegistros.h, calendario.h y menuPrincipal.h.

Para la implementación de los métodos declarados en las cabeceras se realizó de la misma forma, en la carpeta Sources se fragmentaron los ficheros *.cpp, obteniéndose la estructuración antes mencionada. En la carpeta controlador se localiza el fichero controlador.cpp, en la carpeta visual se encuentran menuPrincipal.cpp, calendario.cpp y eliminarRegistros.cpp. En la capeta modelo se hallan eliminarModelo.cpp, pluginModelo.cpp, recuperarModelo.cpp y reporteModelo.cpp. En la figura se muestra la conformación descrita que adoptó el árbol de carpeta del proyecto perteneciente al plugin de recuperación del estado de entidades. (Figura 10)

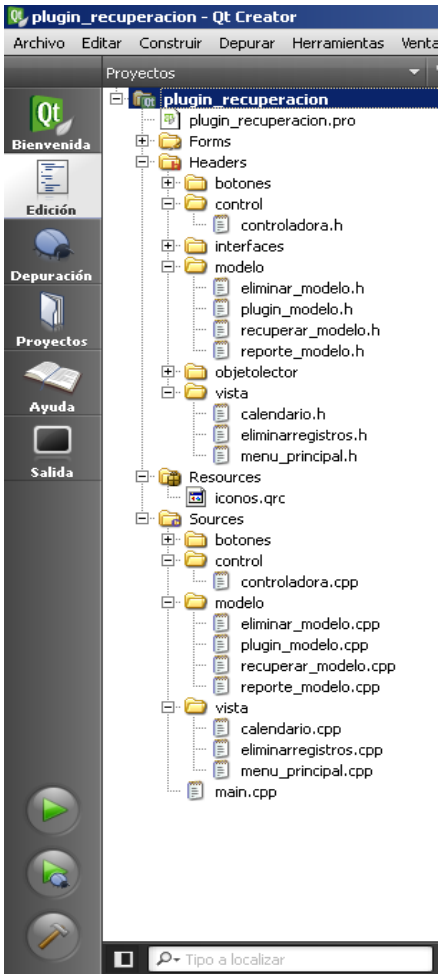


Figura 10: Aplicación del patrón Modelo-Vista-Controlador en Qt

3.1.3 Estándar de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. XP enfatiza la comunicación de los programadores a través del código y plantea que estos pueden realizar cambios en cualquier parte del código en cualquier momento, por lo

cual es indispensable que se sigan ciertos estándares de programación manteniéndolo legible para los miembros del equipo, de forma tal que se faciliten los cambios. (23)

Emplear técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es suma relevancia para lograr la calidad del software, pues lo convierte en un sistema fácil de comprender y mantener, además mantener estándares de codificación entre los programadores es esencial para la programación en pares y para la propiedad colectiva del código.

La arquitectura basada en plugin que posee la herramienta de administración de bases de datos HABD, implica la implementación de componentes autónomos y de la fusión de programadores que conforman el desarrollo de la herramienta en general. Debido a ello la arquitectura del proyecto definió exclusivamente un estándar de código para contrarrestar las consecuencias de dicha peculiar característica. El elemental hecho de seguir al pie de la letra el estándar de código definido es imprescindible, pues de esa forma se logra la homogeneidad y limpieza de los componentes desarrollados como si hubiesen sido implementados por un único programador.

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

A continuación se referencian los fragmentos de mayor significado del estándar de codificación definido por la arquitectura del proyecto y empleado para el desarrollo del plugin de recuperación del estado de entidades, evidenciado con ejemplos del código fuente correspondiente. Para mayor compresión consultar el artefacto “Estándar de Código” presente en el Expediente de Proyecto.

Estándar de código

Identación

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

Longitud de la Línea

Evitar líneas con más de 80 caracteres. Cuando una sentencia sobrepase una línea simple del editor, esta deber ser separada en otras. 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 5 espacios.

Comentarios

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

Generalmente deben usarse comentarios de una sola línea. Reserve los comentarios de bloques para la documentación formal o para comentar porciones de código. (Figura 13)

Declaración de variables

Cada variable debe de ser declarada en una línea y comentada. También deben aparecer ordenadas alfabéticamente. 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. (Figura 11)

```
private:

    eliminarModelo eliminarMod;    //objeto de la clase eliminarModelo
    QColor color;                  //objeto de tipo color
    QSqlDatabase coneccion;       //objeto de tipo conexión
    pluginModelo pluginMod;      //objeto de la clase plugin Modelo
    reporteModelo repModelo;    //objeto de la clase reporteModelo
    recuperarModelo recupMod; //objeto de la clase recueprarModelo

};
```

Figura 11: Declaración de variables

Declaración de funciones

No debe haber espacio entre el nombre de la función y el paréntesis izquierdo, ni entre éste 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. (Figura 12)

```
bool controladora::esNumero(QString cadena)
{
    return repModelo.esNumero(cadena);
}
bool controladora::esTablaProhibida(QString tabla)
{
    plugin_mod.setConexion(coneccion);
    return plugin_mod.esTablaProhibida(tabla);
}
```

Figura 12: Declaración de funciones

Identificadores

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

Sentencias

Sentencias Simples

Cada línea debe contener como máximo una sentencia. Debe poner un punto y coma ";" al final de cada sentencia simple. Se debe tener 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.

Sentencias Compuestas

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

Las sentencias encerradas deben ser identificadas a 4 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 son usadas para listar todas las sentencias, aunque se trate de una sola, cuando son parte de una estructura de control como if o for. Esto facilita agregar nuevas sentencias sin la introducción accidental de errores.

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. (Figura 14)

Sentencia if (Figura 13)

La sentencia if debe ser escrita de esta manera:

if (condition)

{

statements

}

```
//Desaplica el plugin, dada una tabla seleccionada.
void menu_principal::on_desaplicar_clicked()
{
    QString tablaSeleccionada = ui->listWidget->currentItem()->text();

    if(control.tieneAplicadoPlugin(control.tablaConEsquema(tablaSeleccionada)))
      {
            ui->label->setText("Recuperador desaplicado satisfactoriamente"
                               "en la tabla: "+tablaSeleccionada);

            control.desaplicarRecuperador(tablaSeleccionada);
            mostrarTablasEnListwidget();

            botonAplicar->setEnabled(false);
            botonDesaplicar->setEnabled(false);
            actualizarReporte(criterioBusqueda);
        }else
        {
            QMessageBox::critical(this,"Error","A la tabla: "+tablaSeleccionada+
                                  " no se le puede desaplicar el Recuperador.");
        }
}
```

Figura 13: Etiquetas "Sentencia if"

Estructuras repetitivas

Las estructuras repetitivas deben ser escritas de esta manera (Figura 14):

for (initialization; condition; update)

{

statements

}

```
QString reporte_modelo::getFilaCompleta(QSqlQuery query)
{
    QString app("");
    int n=GetNumColumnas(query);
    for(int i=0;i<=n-3;i++)
    {
        if(!esNumero(query.value(i).toString()))
        {
        app.append("'"+query.value(i).toString()+"'  ");
        } else
        {
        app.append(query.value(i).toString()+"  ");
        }
    }
    app.resize(app.size()-1);
    return  app;
}
```

Figura 14: Estructura repetitiva “for”

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. Esto ayuda a distinguir entre palabras reservadas y llamadas a funciones.

Para cualquier operador binario excepto el punto “.”, el paréntesis que abre “(” y el corchete que abre “[” todos deben ser separados por un espacio entre operandos y operador. No se debe utilizar el espacio para separar un operador unario de su operando excepto cuando ese operador es una palabra como typeof.

Cada punto y coma “;” en una sentencia de control debe ser seguido por un espacio.

Debe dejarse un espacio luego de cada coma “,”.

Declaraciones de clases

Solo debe existir un fichero con más de una clase declarada.

Ejemplo correcto:

Fichero A.h

class A

{

}

3.1.4 Interfaces de la aplicación

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

El plugin integrado en HABD consta de una Interfaz Principal (Figura 15) que le permite al usuario aplicar y desaplicar el procedimiento, mostrar reportes y recuperar el estado de las entidades que seleccione. La interfaz de HABD está compuesta por pestañas, una de ellas es el plugin de recuperación del estado de entidades una vez que se halla instalado. En la parte izquierda inferior de la pestaña del plugin, se cargan las tablas de la base de datos a la que se ha conectado el usuario, funcionalidad correspondiente al plugin Descriptor de Objetos, que le permite al usuario seleccionar las tablas una por una.

Una vez seleccionada una tabla, el plugin permite aplicar el procedimiento, funcionalidad que puede ser activada mediante el botón “Aplicar” que se encuentra en la parte superior izquierda. A continuación se encuentra el botón “Desaplicar”, que activa la funcionalidad desaplicar el procedimiento. Ambos botones son identificados con la etiqueta informativa “Gestionar Recuperador” y una imagen que representa su funcionalidad. Las tablas que le son aplicadas el procedimiento aparecerán listadas, al lado del Descriptor de Objetos.

Ubicados en la parte superior izquierda de la interfaz, consecutivos a los botones “Gestionar Recuperador”, se encuentran los botones “Mostrar Reporte”, “Recuperar estado” y “Eliminar Registros”, en ese mismo orden y cada uno representa con una imagen la funcionalidad que permite. Son identificados con la etiqueta informativa, “Acciones Base de Datos”.

Mediante el botón “Mostrar Reporte” el usuario puede verificar un reporte de las acciones ocurridas sobre las entidades, que aparecen en la parte inferior central de la interfaz. Permite que sean

mostrados el reporte de todas las modificaciones, inserciones y eliminaciones que han sido realizadas en la base de datos desde que se instaló el plugin.

El botón “Recuperar estado” permite como lo dice la palabra recuperar el estado de una o varias entidades. La funcionalidad puede ser empleada seleccionando una entidad directamente del reporte, que puede ser filtrado mediante los criterios de búsqueda ubicados en el medio de la interfaz o se pueden filtrar las entidades sin un previo reporte. En los criterios de búsqueda se encuentra la opción de seleccionar por esquema, tablas y el rango de fecha, la cual asume la fecha actual por defecto.

Los botones identificados con fecha inicial y final, ubicados debajo de la etiqueta informativa Criterios de búsqueda permiten al usuario acceder a la interfaz (Figura 16) que provee un calendario para seleccionar un rango de fecha válido. Los botones ubicados en la parte inferior derecha de la interfaz permiten aceptar y cancelar la operación respectivamente.

Una vez seleccionados los criterios al dar clic en el botón “Filtrar” ubicado en la parte izquierda central se relacionan en la parte inferior los resultados de la búsqueda clasificados en insertados, modificados y eliminados. Luego permite que se puedan seleccionar las entidades, para recuperar el usuario da clic en el botón “Recuperar”.

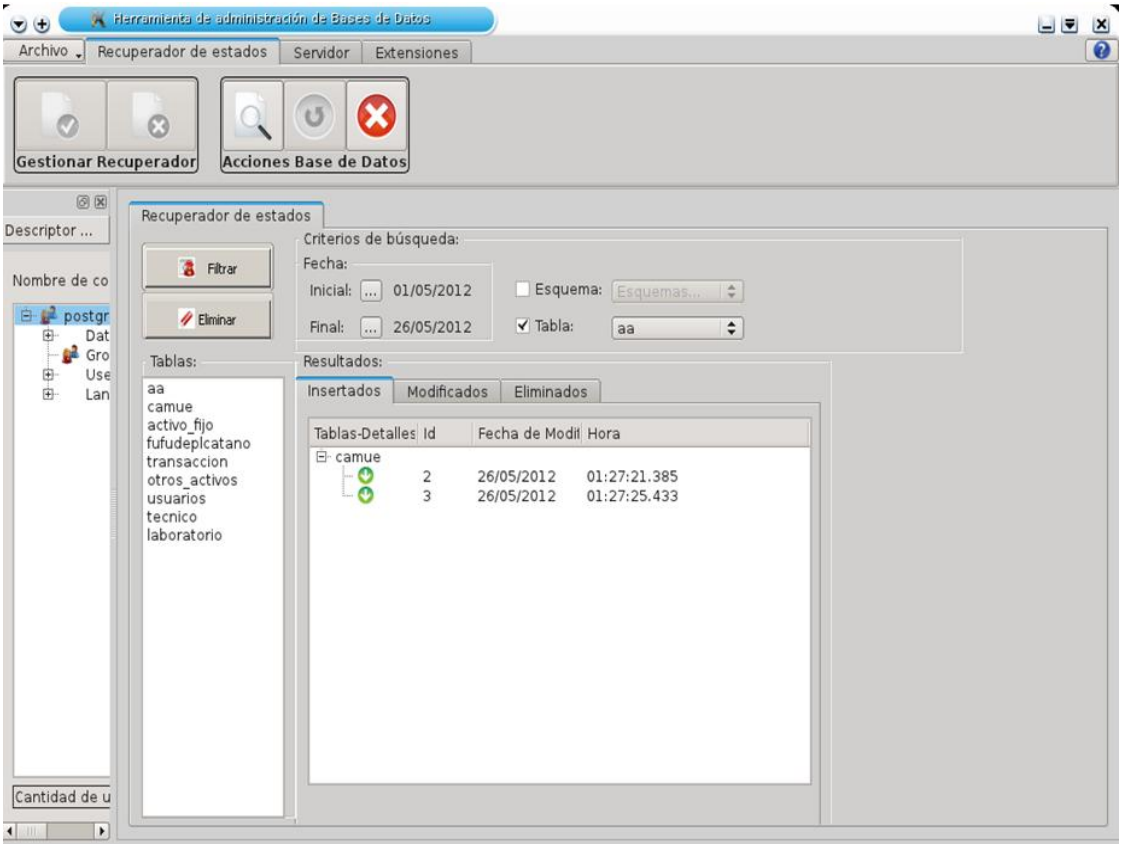


Figura 15: Interfaz Principal

Figura 16: Interfaz “Calendario”

El usuario mediante la interfaz “Eliminar Registros” puede eliminar los datos de las tablas creadas que considere su almacenamiento innecesario, Mediante el botón “Eliminar Registros” el usuario puede acceder a la interfaz que permite dicha funcionalidad (Figura 17). La interfaz lista las tablas a las que se le aplicó el procedimiento, ubicadas en la parte superior izquierda. El usuario puede seleccionar y añadirlas a la lista ubicada en la parte superior derecha mediante el botón “Añadir”,

que aparece en el centro de las dos listas, debajo se encuentra el botón “Quitar” en caso de que el usuario se equivoque de elección.

En la parte central el usuario puede seleccionar la fecha de almacenamiento de los datos que desee eliminar según estime conveniente. En la parte inferior aparecen seleccionadas las tres tablas que se crean para las acciones insertar, modificar y eliminar, en caso de que el usuario considere los datos de las tres tablas innecesarios. Si desea conservar la información de alguna de esas tablas, desmarca la que estime. Finalmente el usuario puede aceptar o cancelar mediante los botones ubicados en la parte inferior de la interfaz. Mediante el botón “Eliminar” ubicado debajo del botón “Filtrar” de la interfaz principal, el usuario puede eliminar directamente del reporte el registro que considere innecesario.

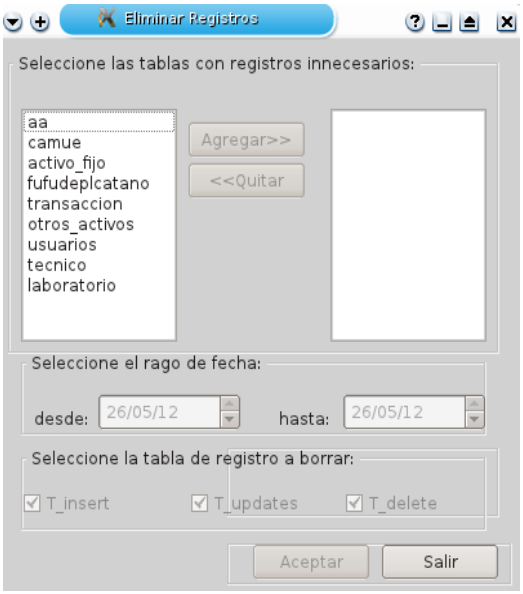


Figura 17: Interfaz “Eliminar Registros”

3.2 Pruebas

La metodología XP propone el proceso de pruebas como uno de sus soportes imprescindibles en el que los desarrolladores prueban tanto como sea posible. Mediante esta filosofía se reduce el número de errores no detectados permitiendo una mejor calidad en los productos desarrollados y la seguridad de los programadores para lidiar satisfactoriamente con la introducción de algún cambio o modificación en el sistema. (27)

La metodología XP divide las pruebas en dos grupos:

- ✓ Estrategia de pruebas unitarias.

- ✓ Estrategia de pruebas de aceptación.

Pruebas Unitarias

Las pruebas unitarias se realizan para controlar el funcionamiento de pequeñas porciones de código como subprogramas (en la programación estructurada) o métodos (en Programación Orientada a Objetos). Generalmente son realizadas por los mismos programadores pues al conocer con mayor detalle el código, se les simplifica la tarea de elaborar conjuntos de datos de prueba para testearlo. (24)

Las pruebas unitarias suelen realizarse mediante los métodos de pruebas caja blanca o de caja negra aunque el tipo de prueba a la cual se someterá a cada uno de los módulos dependerá de su complejidad. Para probar los componentes implementados como unidades individuales, se realiza el método de pruebas de caja negra, la cuales comprueban el comportamiento de la unidad observable externamente.

Pruebas de Aceptación

Las pruebas de aceptación validan que el sistema cumple con el funcionamiento esperado y permite al usuario que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento. (24) Se considera que las historias de usuario y por extensión las pruebas asociadas, juegan el papel de especificaciones del sistema. El cliente es quien crea y utiliza las pruebas para comprobar y confirmar que las historias de usuario cumplan su cometido, aunque son preparadas por el equipo de desarrollo. Una historia de usuario puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento y se considera incompleta hasta que no ha pasado por sus pruebas de aceptación.

Las pruebas de aceptación suelen realizarse mediante las técnicas de prueba de caja negra que demuestran la conformidad con los requisitos, el cual define las verificaciones a realizar y los casos de prueba asociados.

Las pruebas de aceptación se clasifican en dos tipos:

- ✓ Pruebas Alfa.
- ✓ Pruebas Beta.

Las pruebas Alfa se realizan por un cliente en un entorno controlado por el equipo de desarrollo en el cual se crea un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente.

Las pruebas Beta se realizan en las instalaciones propias de los clientes.

Las pruebas de aceptación son parte integral del creciente desarrollo experimentado por XP y apoyan todas las historias de usuario definidas por el propio cliente. Son más relevantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente. (27) Son realizadas por el propio cliente en compañía de uno de los representantes del equipo de desarrollo y se orientan a las funcionalidades del sistema.

Una propuesta metodológica para guiar la generación de pruebas de aceptación por parte del cliente debe ser lo suficientemente sencilla para que cualquier persona sin experiencia en ingeniería del software y en pruebas pueda ponerla en práctica. Un desarrollo mediante programación extrema está compuesto por una serie de iteraciones cortas. Cada iteración concluye ejecutando un conjunto de pruebas de aceptación que permitan al cliente comprobar si está satisfecho con el resultado, pues en XP el cliente es la clave en los procesos de desarrollo. (27)

Por lo anteriormente expuesto se propone realizar pruebas de aceptación de tipo Alfa utilizando la técnica de prueba caja negra, pues comprueba que cada función es operativa, estudia la especificación del software, las funciones que debe realizar, las entradas y las salidas y plantea la realización del diseño de los casos de pruebas.

3.2.1 Método seleccionado. Prueba de Caja Negra

Las pruebas de caja negra se centran en los requisitos funcionales y se llevan a cabo sobre la interfaz del software. Tienen como objetivo demostrar que las funciones del software son operativas, que las entradas acepten de forma adecuada y se produzca un resultado correcto, teniendo en cuenta que la integridad de la información externa se mantenga. Durante las iteraciones, las HU seleccionadas, serán traducidas a pruebas. En ellas se especifican, desde la perspectiva del cliente, los escenarios para probar que una HU ha sido implementada correctamente. El objetivo final de las pruebas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable.

Métodos de las pruebas de Caja Negra:

✓ *Técnica de prueba basada en gráficos***:** En la técnica se debe entender los objetos que se modelan en el software y las relaciones que conectan a estos, tales como objetos de datos, objetos de programa como módulos o colecciones de sentencias del lenguaje de programación. (24)

✓ *Partición equivalente:* Divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. En otras palabras, este método intenta dividir el dominio de entrada de un programa en un número finito de clases de equivalencia, de tal modo que se pueda

asumir razonablemente que una prueba realizada con un valor representativo de cada clase es equivalente, a una prueba realizada con cualquier otro valor de dicha clase. (24)

✓ *Análisis de Valores Límites:* El análisis de valores límite (AVL) 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. (24)

Para verificar que el sistema desarrollado cumple con las funciones específicas para las que se ha creado se utiliza las pruebas de comportamiento o de caja negra y empleando la técnica de partición equivalente. Al comenzar el proceso de realización de pruebas lo primero que se concibe es el diseño de los casos de pruebas, que se definen según las funcionalidades descritas en las HU. Se parte de las descripciones efectuadas en las HU como apoyo para las revisiones.

Un caso de prueba específica una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse. Se pueden realizar muchos casos de prueba para determinar que una HU es completamente satisfactoria. Con el propósito de comprobar que todas las HU de una aplicación son revisadas, debe haber al menos un caso de prueba para cada una de ellas.

Los casos de prueba según la metodología XP deben cumplir las siguientes características:

- ✓ Los casos de prueba (CP) deben escribirse para realizar las pruebas desde el punto de vista del usuario.
- ✓ Brindar un feedback[10] rápido y concreto de cómo se está desarrollando la iteración y el proyecto.
- ✓ Los casos de prueba exitosos de una iteración deben repetirse con éxito en las siguientes iteraciones.
- ✓ Un error aunque sea en un solo paso de un caso hace que se considere que falló el caso entero.

3.2.2 Casos de pruebas basados en HU

Para examinar las funcionalidades se diseñaron un total de 5 casos de pruebas correspondientes a las 5 HU descritas en la fase de Exploración. A continuación se muestran el caso de prueba para la HU “Aplicar procedimiento a la tabla seleccionada” con la descripción de la variables correspondientes. Se

[10] **Feedback**: Característica de XP basada en el desarrollo incremental de pequeñas partes, con entregas y pruebas frecuentes y continuas, proporciona un flujo de retro-información valioso para detectar los problemas o desviaciones.

muestra una tabla donde se especifican las no conformidades detectadas por el CP y la solución a cada una de ellas. Finalmente se disponen los resultados de las pruebas en general para cada iteración.

Escenario	Descripción	Variable 1	Respuesta del sistema	Flujo central
EC 1.1 Aplicar procedimiento con datos correctos.	Para aplicar el procedimiento con datos correctos el usuario selecciona una tabla correspondiente a la base de datos a la que se ha conectado (Plugin Descriptor de Objetos) ubicadas en la parte central izquierda de la interfaz.	V(Estudiante)	El sistema lanza un mensaje "Recuperador aplicado satisfactoriamente a la tabla Estudiante".	El usuario selecciona la pestaña Recuperador de estados luego selecciona una tabla a la cual desee aplicar el procedimiento, de las tablas de la base de datos a la que se ha conectado, las cuales están ubicadas en la parte izquierda de la interfaz (Plugin Descriptor de Objetos) luego da clic sobre el botón "Aplicar".
EC 1.2 Aplicar procedimiento con datos vacíos.	Para aplicar el procedimiento con datos vacíos el usuario no selecciona ninguna tabla.	I( )	El sistema deshabilita el botón “Aplicar”.	
EC 1.3 Aplicar el procedimiento con datos incorrectos.	Para aplicar el procedimiento con datos incorrectos el usuario selecciona una tabla del Descriptor de Objetos, a la cual se le haya aplicado el procedimiento.	I(Estudiante)	El sistema deshabilita el botón “Aplicar”.	

	satisfactoriamente con anterioridad.			
	Para aplicar el procedimiento con datos incorrectos el usuario selecciona una de las tablaT_Insert, T_Delete y T_Update.	I(Estudiante_Insert)	El sistema deshabilita el botón "Aplicar".	

Tabla 9: CP_HU “Aplicar procedimiento a la tabla seleccionada”

Para lograr un efectivo entendimiento del CP es necesario conocer el significado de las variables a la que se hacen referencias. A continuación se muestra la tabla donde se describe la variable asociada al CP descrito anteriormente.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
V1	*Tablas*	*ListWidget*	*No*	*Lista todas las tablas que contiene la base de datos a la que se ha conectado el usuario.*

Tabla 10: Descripción de las variables

Para llevar a cabo un proceso satisfactorio de pruebas, se deben realizar hasta tres iteraciones. En la primera iteración se efectúan todo los CP, las no conformidades detectadas por ellos, son aceptadas y resueltas por el equipo de desarrollo en un plazo de un día. Luego para verificar la adecuada rectificación y solución de los problemas revelados, se procede a una segunda iteración en la que se pueden descubrir nuevas no conformidades y detectar una ineficiente solución a ellas. En la tercera iteración se deben haber corregido todas las no conformidades, y luego se aborta el proceso de pruebas. En la siguiente tabla se muestran las no conformidades detectadas, en correspondencia al CP definido anteriormente, la cual pertenece a la primera iteración de pruebas.

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detección	Clasificación	Estado NC	Respuesta del Equipo Desarrollo
Aplicación	1	Falta ortográfica: Sin tilde la palabra Éxito en el mensaje de confirmación. Anexo 3	Aplicar procedimiento a la tabla seleccionada.	Prueba	NS	PD 20/5/2012 RE 20/5/2012	El equipo de desarrollo corrigió la falta ortográfica.
Aplicación	2	El mensaje de confirmación no es correcto. Anexo 4	Aplicar procedimiento a la tabla seleccionada.	Prueba	NS	PD 20/5/2012 RE 21/5/2012	El equipo de desarrollo corrigió el mensaje de confirmación.
Aplicación	3	En el mensaje de confirmación cambiar el botón OK por Aceptar. Anexo 4	Aplicar procedimiento a la tabla seleccionada.	Prueba	R	PD 20/5/2012 RE 21/5/2012	El equipo de desarrollo cambió el botón.
Aplicación	4	En el escenario 1.3 el botón Aplicar se habilitó. Anexo 5	Aplicar procedimiento a la tabla seleccionada.	Prueba	S	PD 20/5/2012 RE 21/5/2012	El equipo de desarrollo arregló la funcionalidad.

Tabla 11: No conformidades encontradas y resueltas

Resultados generales de las pruebas realizadas

Los casos de pruebas diseñados detectaron en la primera iteración 14 no conformidades, de ellas 5 no significativas, 5 recomendaciones y 4 significativas. En la segunda iteración se corrigieron 4 no conformidades, 2 significativas, una no significativa y una recomendación. En la tercera iteración las pruebas efectuaron su objetivo al ser verificada la satisfacción del cliente, logrando un producto con calidad con todos sus requisitos cumplidos. (Figura 18)

Para certificar los resultados que arrojaron las pruebas y obtener la aprobación del cliente en cuanto al plugin desarrollado, se conformó una Carta de Aceptación en el proyecto PostgreSQL, en la cual se ratifica la validez y viabilidad del plugin como resultado del objetivo trazado. La carta de Aceptación contiene además la aprobación del Jefe de Departamento del proyecto PostgreSQL. (Anexo 6)

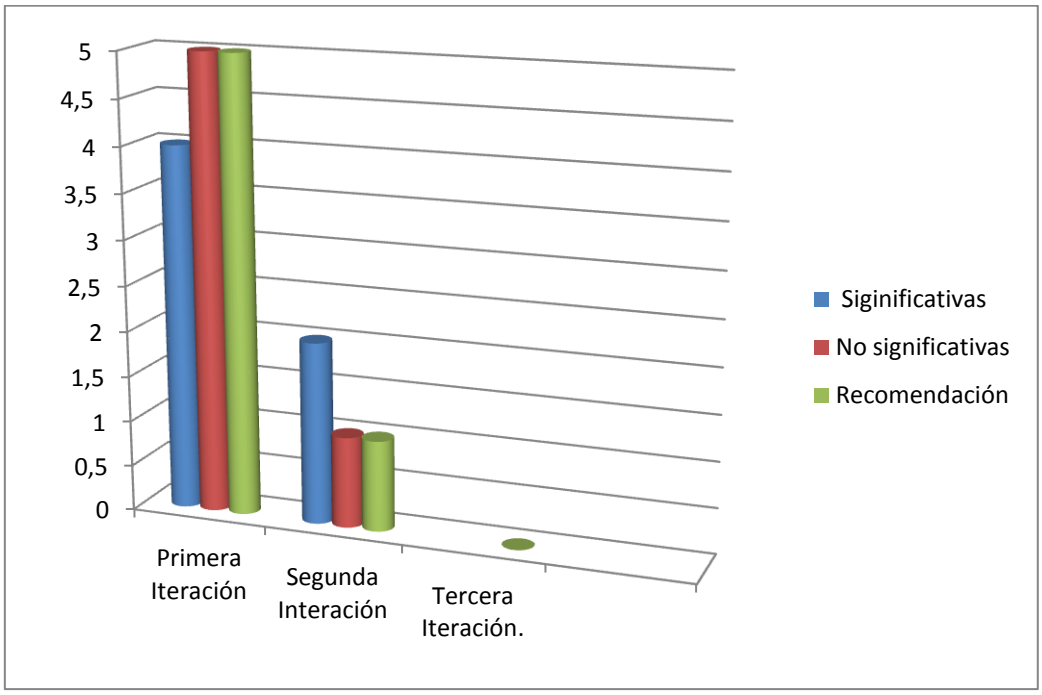


Figura 18: Resultados de las pruebas

Conclusiones Parciales

Con el empleo de la metodología XP, combinado con aportes arquitectónicos, la programación en parejas y las entregas a corto plazo, se obtuvo una mayor eficiencia del equipo de desarrollo y a su vez un producto funcional. Mediante la implementación del 100% de la 5 HU identificadas, empleando las tecnologías y herramientas idóneas para su desarrollo, se creó el plugin de recuperación del estado de entidades que pretende ser una solución para la corrección de errores de usuario que provoquen la eliminación involuntaria de entidades en una base de datos. El resultado de la pruebas arrojó las conclusiones que el método y técnica empleados mediante los 5 casos de pruebas diseñados, es capacitado para este tipo de aplicación, pues se obtuvieron 18 no conformidades, que al ser corregidos por el equipo de desarrollo se adquirió un producto con un nivel mayor de calidad. Se demostró en la tercera iteración que el plugin cumple con las necesidades del cliente, garantizando su correcto funcionamiento.

Conclusiones

Con el desarrollo finalizado del plugin y lista su integración, se establece una alternativa para que la pérdida de datos por error de usuario sea reversible de la manera más rápida posible en correspondencia con el procedimiento que utiliza. Su desarrollo estuvo enmarcado en los objetivos del presente trabajo, los cuales fueron cumplidos en un orden consecuente, guiados por la metodología XP.

El estudio de los procedimientos y herramientas para la recuperación del estado de entidades enmarcado en el SGBD PostgreSQL, destacó la inexistencia de una funcionalidad automática que lo permita, sin embargo queda fuera del alcance de la investigación el estudio de tecnologías pertenecientes al gestor privativo Oracle, que aportan herramientas para corregir errores de usuario en bases de datos. Mediante la investigación acerca de las tecnologías y herramientas, se seleccionaron las idóneas para el desarrollo del plugin, así como el framework Qt, el IDE Qt Creator, la herramienta CASE Visual Paradigm, el controlador de versiones *Subversion* y se marcó la relevancia del empleo del lenguaje de programación C++ y la metodología XP.

El diseño del plugin pretendió colaborar con la persistencia y recuperación de los datos y se proyectó para evitar la interferencia en la confidencialidad y persistencia de la información almacenada en las bases de datos. Se identificaron las HU para guiar la implementación, se concretó el diseño y arquitectura del plugin, al establecerse el empleo de los patrones de diseño, el patrón arquitectónico MVC, el diagrama de clases y la definición de las tarjetas CRC.

La implementación fue guiada por las TI identificadas en cada HU, se destacó de forma general por el empleo de patrones de arquitectura, el estándar de código definido, la aplicación del patrón MVC en Qt y la interfaces visuales definidas para el plugin. Las pruebas realizadas mediante los 5 CP diseñados por cada HU demostraron el cumplimiento de las características especificadas por el cliente, una de las premisas imprescindibles en la metodología XP.

Recomendaciones

La investigación sugiere estudiar la posibilidad de perfeccionar el procedimiento que se emplea en la implementación de modo que permita una mayor optimización en la recuperación del estado de entidades, así como incorporar una funcionalidad que posibilite gestionar la eliminación de los registros automáticamente, es decir, configurar el tiempo en que se almacenarán los datos en las tablas creadas, perfeccionando de esta forma la que se emplea en el plugin.

.

Referencias Bibliográficas

1. **Reyes, Ing. Yunior Mesa.** Comunidad Técnica Cubana de PostgreSQL. [En línea] 2.0, diciembre de 2010. [Citado el: 10 de 0ctubre de 2011.] http://tallertematico2010.fordes.co.cu/public/site/142.pdf.
2. **C.J.Date.***Introducción a los Sistemas de Bases de Datos.* [trad.] Sergio Luis María Ruiz Faudón. Séptima. México : PEARSON EDUCACIÓN, 2001. 968-444-419-2.
3. **tools, PgAdmin PostgreSQL.** PgAdmin PostgreSQL tools. [En línea] [Citado el: 2011 de 11 de 10.] http://pgadmin.org/.
4. **Medina, José Manuel Alarcón.***Administración SGBD PostgreSQL.* Generalitat Valenciana : s.n., 2006.
5. PostgreSQl-es. [En línea] [Citado el: 3 de 10 de 2012.] http://www.postgresql.org.es/node/238.
6. PostgreSQ. [En línea] 30 de 11 de 2011. http://www.postgresql.org/docs/9.0/static/continuous-archiving.html..
7. **Vegas, Jesús.** ORACLE: Backup y Recuperación . [En línea] Universidad de Valladolid , Mayo de 1998. [Citado el: 2011 de 11 de 10.] http://www.infor.uva.es/~jvegas/cursos/bd/oraback/oraback.html.
8. **Hodak, William.***Alta Disponibilidad de Oracle Database 11g.* 2007.
9. **Méndez, Alejandra Virrueta.***Metodología de desarrollo de software.* Institución Tecnológico Superior de Apatzingán. Michoacán : s.n., 2010.
10. **Wesley, Addison.***Una explicación de la programación extrema. Aceptar el cambio.* 2000.
11. **Morales, Ing. Glennis Tamayo, Rodríguez, Ing. Marianela Gutiérrez y Ing. Yunior Mesa Reyes, Ing. Glennis Tamayo Morales, Ing. Yudisney Vazquez Ortíz.***Selección de la Metodología de Desarrollo para el Proyecto de PostgreSQL Empresarial.* 2011.
12. **programación, Lenguajes de.** Lenguajes de programación. [En línea] 2009. [Citado el: 2011 de 11 de 11.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.
13. **Jalón, Javier García de.***Aprenda c++ como si estuviera en primero.* Escuela Superior de Ingenieros Industriales. Navarra : s.n., 1998.
14. **Meyer, Lisandro Damián Nicanor Pérez.***Introducción al desarrollo multiplataforma con Qt 4.* 2007.
15. Qt.Nokia. Qt: cross-platform rich client development framework. [En línea] 2008. [Citado el: 2012 de 1 de 12.] http://qt.nokia.com/products/.
16. QT creator Geeks & Linux. [En línea] [Citado el: 2011 de 11 de 10.] http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/..
17. **Cornejo, José Enrique González.** Qué es UML? [En línea] http://www.docirs.cl/uml.htm.30..
18. **Larman, Craig.***UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : s.n., 1999. págs. 4 - 15.
19. **Garzón, Jesus María.***Herramientas Case El mejor soporte para el proceso de desarrollo de software.* Instituto Nacional de Estadística e Informática. 1999. Colección Cultura Informática.
20. Visual Paradigm International. [En línea] [Citado el: 2011 de 11 de 9.] http://www.visual-paradigm.com.
21. **Serradilla, Juan Luis.***Control de Versiones con TortoiseSVN.* Universidad de Murcia.

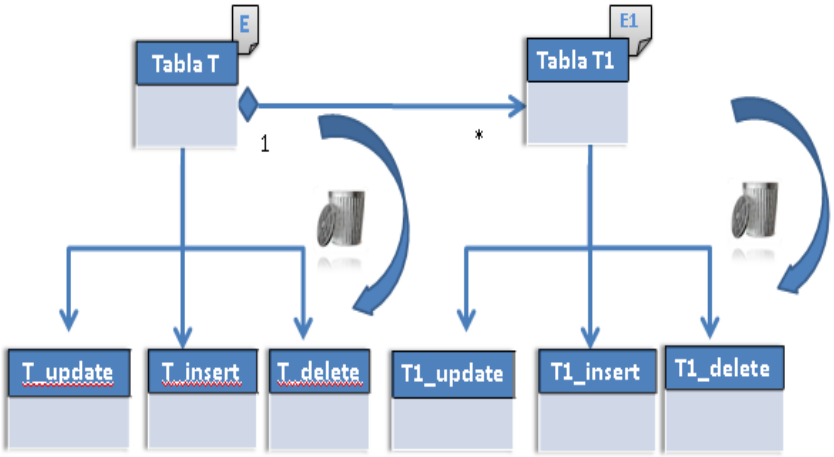
2007.
22. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.***Proceso Unificadado de Desarrollo de Software.* Primera Edición en Español. s.l. : Person Educación, S.A . pág. 112.
23. **Letelier Patricio José H Canós Sánchez, Emilio A..***Mejorando la gestión de historias de usuario en eXtreme Programming.* Departamento de Sistemas Informáticos y Computación Universidad Politécnica. Valencia : s.n.
24. **Letelier, Patricio.***Metodologías ágiles para el desarrollo de software:eXtreme Programming (XP).* Valencia : s.n.
25. **Pressman, Roger S.***Ingeniería del Software: Un enfoque práctico.* Séptima Edición.
26. **Mestras, Juan Pavón.***Estructura de las Aplicaciones Orientadas a Objetos El patrón Modelo-Vista-Controlador (MVC).* Universidad Complutense Madrid. 2009.
27. **Astudillo, Marcello Visconti y Hernán.***Fundamentos de Ingeniería de Software.* Universidad Técnica Federico Santa María.
28. **J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres.***Pruebas del Sistema en ProgramaciónExtrema.* Universidad de Sevilla.

Bibliografía

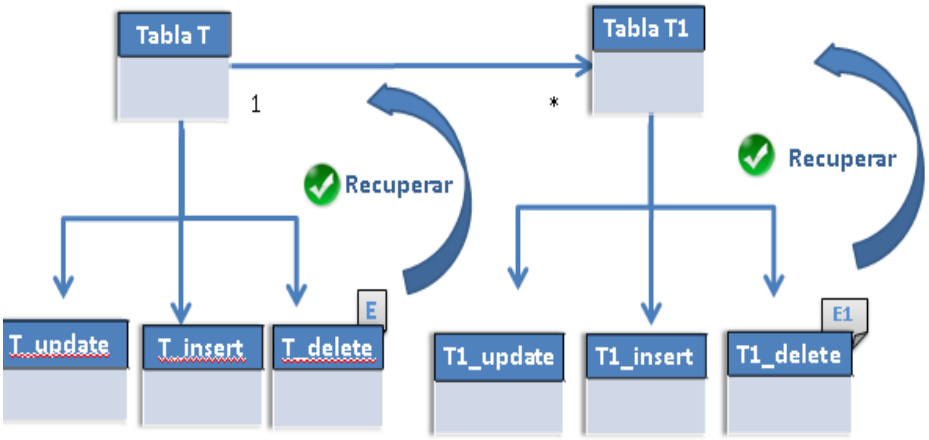
Astudillo, Marcello Visconti y Hernán.*Fundamentos de Ingeniería de Software.* Universidad Técnica Federico Santa María.
Azcárate, Ernesto Quiñones.*PostgreSQLComo funciona una Base de Datos por dentro*
Cornejo, José Enrique González. Qué es UML? [En línea] http://www.docirs.cl/uml.htm.30..
Garzón, Jesus María.*Herramientas Case El mejor soporte para el proceso de desarrollo de software.* Instituto Nacional de Estadística e Informática. 1999. Colección Cultura Informática. Visual Paradigm International. [En línea] [Citado el: 2011 de 11 de 9.] http://www.visual-paradigm.com.
Hodak, William.*Alta Disponibilidad de Oracle Database 11g.* 2007.
Gil, Fidel, Albrigo, Javier y Rosario, Javier Do.*Sistema Gestión de Base de Datos SGBD / DBMS.* Universidad de Carabobo. Valencia : s.n., 2005.
Ginestà, Marc Gibert y Mora, Oscar Pérez.*Bases de datos en PostgreSQL.*
Jacobson, Ivar, Booch, Grady y Rumbaugh, James.*Proceso Unificadado de Desarrollo de Software.* Primera Edición en Español. s.l. : Person Educación, S.A . pág. 112.
Jalón, Javier García de.*Aprenda c++ como si estuviera en primero.* Escuela Superior de Ingenieros Industriales. Navarra : s.n., 1998.
J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres.*Pruebas del Sistema en ProgramaciónExtrema.* Universidad de Sevilla.
Larman, Craig.*UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : s.n., 1999. págs. 4 - 15.
Larman, Craig.*UML y Patrones.Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Segunda Edición.
Letelier Patricio José H Canós Sánchez, Emilio A.*Mejorando la gestión de historias de usuario en eXtreme Programming.* Departamento de Sistemas Informáticos y Computación Universidad Politécnica. Valencia : s.n.
Letelier, Patricio.*Metodologías ágiles para el desarrollo de software:eXtreme Programming (XP).* Valencia : s.n.
León, Anthony R. Sotolongo.*Compendio de consultas útiles al catálogo de postgreSQL.* Universidad de Ciencias Informáticas (UCI). Habana : s.n., 2011.
León, Eduardo.*Tutorial Visual Paradigm for UML.*
Looser, Julián, y otros.OSGART: A Pragmatic Approach to MR. [En línea] [Citado el: 11 de 11 de 2012.] http://ismar06.org/data/1b-HITL.pdf.
Loza, Patricia A., R.Cargnelutti, Pablo y Agüero, Paula Sosa.*Auditoría de Base de Datos.*
Martínez, Prof. Ivette C.*Patrones, Clase 9:Diseño con Patrones.* Universidad Simón Bolívar.
Medina, José Manuel Alarcón.*Administración SGBD PostgreSQL.* Generalitat Valenciana : s.n., 2006.
Méndez, Alejandra Virrueta.*Metodología de desarrollo de software.* Institución Tecnológico Superior de Apatzingán. Michoacán : s.n., 2010.
Mestras, Juan Pavón.*Estructura de las Aplicaciones Orientadas a Objetos El patrón Modelo-Vista-Controlador (MVC).* Universidad Complutense Madrid. 2009.
Meyer, Lisandro Damián Nicanor Pérez.*Introducción al desarrollo multiplataforma con Qt 4.* 2007.
Montaldo, Diego Fernando.*Patrones de Diseño de Arquitecturas de Software Enterprise".* Universidad de Buenos Aires. 2005. Tesis de Grado en Ingeniería en Informática.

Morales, Ing. Glennis Tamayo, Rodríguez, Ing. Marianela Gutiérrez y Reyes Ing. Yunior Mesa, Ing. Yudisney Vazquez Ortíz.*Slección de la Metodología de Desarrollo para el Proyecto de PostgreSQL Empresarial.* 2011.
PostgreSQl-es. [En línea] [Citado el: 3 de 10 de 2012.] http://www.postgresql.org.es/node/238.
PostgreSQ. [En línea] 30 de 11 de 2011. http://www.postgresql.org/docs/9.0/static/continuous-archiving.html.
Pressman, Roger S.*Ingeniería del Software: Un enfoque práctico.* Séptima Edición.
programación, Lenguajes de. Lenguajes de programación. [En línea] 2009. [Citado el: 2011 de 11 de 11.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.
Qt.Nokia. Qt: cross-platform rich client development framework. [En línea] 2008. [Citado el: 2012 de 1 de 12.] http://qt.nokia.com/products/.
QT creator Geeks & Linux. [En línea] [Citado el: 2011 de 11 de 10.] http://www.glatelier.org/2009/05/qt-creator-desarrollando-aplicaciones-rapidamente/..
Reyes, Ing. Yunior Mesa. Comunidad Técnica Cubana de PostgreSQL. [En línea] 2.0, diciembre de 2010. [Citado el: 10 de 0ctubre de 2011.] http://tallertematico2010.fordes.co.cu/public/site/142.pdf.
Sánchez, Jorge. Principios sobre Bases de Datos Relacionales. [En línea] 2004. Universidad Tecnológica Nacional. Córdoba : s.n., 2008.
Serradilla, Juan Luis.*Control de Versiones con TortoiseSVN.* Universidad de Murcia. 2007.
tools, PgAdmin PostgreSQL. PgAdmin PostgreSQL tools. [En línea] [Citado el: 2011 de 11 de 10.] http://pgadmin.org/.
Torres, Patricio Letelier.*Desarrollo de Software Orientado a Objeto usando UML.* Universidad Politécnica de Valencia. España : s.n.
Vegas, Jesús. ORACLE: Backup y Recuperación . [En línea] Universidad de Valladolid , Mayo de 1998. [Citado el: 2011 de 11 de 10.] http://www.infor.uva.es/~jvegas/cursos/bd/oraback/oraback.html.
Wesley *Aceptar el cambio.* 2000.

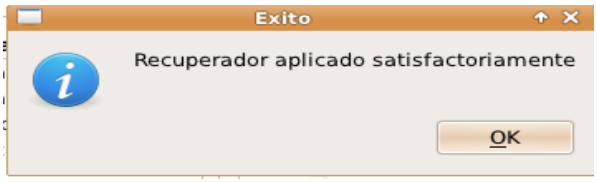
Anexos



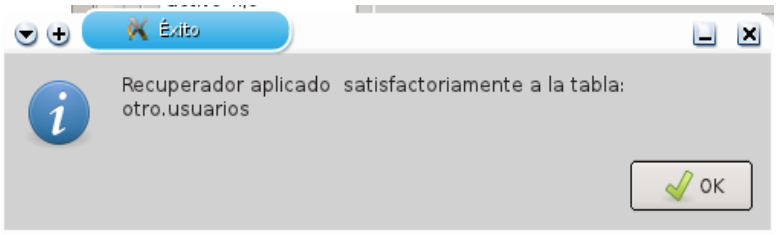
Anexo 1



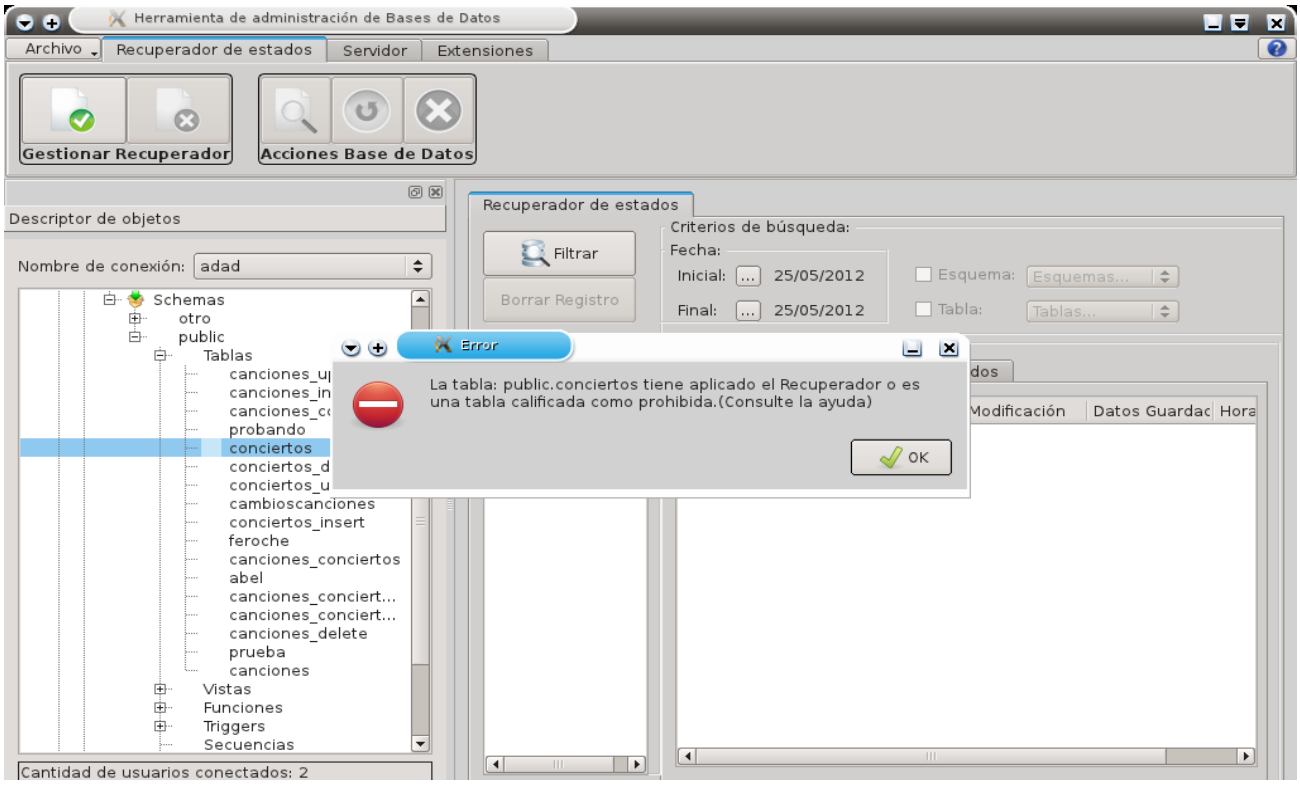
Anexo 2



Anexo 3



Anexo 4



Anexo 5

Acta de Aceptación

Proyecto: PostgreSQL

Categoría de las pruebas: Revisión a la aplicación.

Fecha de conciliación: 5 de junio de 2012.

Firma de los involucrados en el proceso:

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

Observaciones del proceso:

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

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

Ing. Flavio Enrique Roche Rodríquez

(Nombre y apellidos)
Cliente
(PostgreSQL)

Ing. Edgar Roja Ricardo

(Nombre y apellidos)
Jefe de Departamento
(PostgreSQL)

Referencia:

Anexo 6

Glosario de Términos

BD: Base de Datos.

DATEC: Centro de Tecnología de Gestión de Datos.

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

GRASP: *General Responsability Assignment Software Patterns (*Patrones de Software para la asignación General de Responsabilidades).

HU: Historia de Usuario.

IDE: Entorno Integrado de Desarrollo.

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

PITR: Point in Time Recovery.

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

SGBD: Sistema Gestor de Bases de Datos.

Tarjeta CRC: Tarjeta Clase, Responsabilidad y Colaboración.

TI: Tarea de ingeniería.

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

XP: eXtreme Programming.