Universidad de las Ciencias Informáticas
Facultas 6

Plugin diseñador de bases de datos para la herramienta de administración de bases de datos HABD.

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

Autores:

Yeneysi Pérez Castellanos
Marlon Alain Morejón Martínez

Tutores

Ing. Marianela Gutiérrez Rodríguez
Ing. Daymel Bonne Solís

Ciudad de la Habana, Junio 2012
"Año 54 de la Revolución"

" Si salgo, llego: si llego, entro: si entro, triunfo"

Fidel Castro Ruz

Declaración de autoría

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 6 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Yeneysi Pérez Castellanos

_____

Firma del autor

Marlon A Morejón Martínez

_____

Firma del autor.

Marianela Gutiérrez Rodríguez

_____

Firma del tutor

Daymel Bonne Solís

_____

Firma del tutor

Datos de contactos

Tutora:

Ing. Marianela Gutiérrez Rodríguez

Universidad de las Ciencias Informáticas, Habana, Cuba

e-mail: mgrodriguez@uci.cu

Tutor:

Ing. Daymel Bonne Solís

Universidad de las Ciencias Informáticas, Habana, Cuba

e-mail: dbonne@uci.cu

Agradecimientos

De Yeneysi

Para poder realizar esta tesis fue necesario el apoyo de algunas personas a las cuales quiero agradecer: Ante todo quiero agradecer a Dios que me dio la fuerza necesaria para luchar por mis sueños y a no detenerme jamás.

Este es uno de los textos más difíciles que me ha tocado escribir en toda mi vida, pues me resulta complicado expresar en unas cuantas líneas lo agradecida que estoy por alguien que ha hecho por mí un gran esfuerzo. Hay una persona que me enseñó a tener el carácter, la disciplina, el profesionalismo que hoy tengo. Me enseñó que las oportunidades pasan una sola vez en la vida y hay que saberlas aprovechar. A mi padre porque siempre se esforzó por hacerme la mujer que soy y porque incansablemente luchó por mi todo este tiempo. Por tus principios e integridad porque siempre me enseñaste con tus acciones a luchar por lo que se quiere. No hay palabras en este mundo que sirvan para describir todo lo que siento en este momento. Te agradezco por haberme educado así. Estoy orgullosa de ser como soy y eso te lo debo a ti. Has sido para mí un ejemplo de vida, de crecimiento, de evolución. Simplemente gracias, por ser mi luz y mi guía en todo momento.

Estas líneas que voy a escribir son para agradecer a una persona muy especial en mi vida, al ángel dulce, tierno y maravilloso que Dios me ha prestado; son mis sentimientos expresados por estas líneas solo para una hermosa mujer que un obsequio genial me ha regalado como lo es la vida. Estas palabras son para ti: Madre mía te agradezco los momentos que me acogías cuando me equivocaba. Si estaba feliz, celebrabas conmigo. Si estaba triste, no sonreías hasta que me hicieras reír. Que puedo decirte yo que no te haya dicho otras veces, es bueno que existan momentos como estos para agradecerte una vez más por hacer de mí una gran persona. Hoy quiero darte gracias por todo lo que me has brindado y darle gracias a Dios por darme una madre como tú. Madre para ti mil gracias y todo mi amor.

Quiero agradecer a una personita que llena mi vida día a día, que las cosas que hago son por él. Hermano quiero darte las gracias por estar siempre cuando más lo necesité, eres la inspiración de mi vida sin ti no hubiese tenido algo por lo que luchar. Gracias por existir y por darme apoyo siempre.

Quiero agradecer a mi abuelo querido del alma por tanto esfuerzo dedicado a sus nietos. Abuelo, gracias por tratar de hacer sentirme bien en todos los momentos de mi vida. No hay persona tan noble como tú en este mundo. Gracias por preocuparte siempre por los demás y en especial por mí.

Agradecimientos

Cuando menos lo esperamos la vida nos coloca delante de un desafío que pone a prueba nuestro coraje y nuestra voluntad de cambio. En esta vida, gracias a Dios, nada sucede como deseamos, como suponemos, ni como tenemos previsto. Quiero agradecer a una persona que me hace sonreír solo con la mirada, Ariel gracias por enseñarme a amar, te agradezco tu paciencia, sé que no fue fácil, gracias por todo el amor que me demostraste todo este tiempo.

La vida te depara momentos donde das las respuestas adecuadas. A veces aciertas, a veces no, agradezco a alguien que no solo contribuyó para que este sueño se hiciera realidad, sino que estuvo conmigo en algunos momentos determinantes de mi vida. Marlon gracias por todos los momentos, ya sean buenos o malos que pasamos juntos, tal vez y el destino se equivoco con nosotros y quiso vernos caminar en la vida como amigos. Te agradezco el haber contribuido con este trabajo.

Doy las gracias a mis tutores por su grandísimo esfuerzo. Nela no te imaginas cuanto te admiro como profesional, haces que todas las personas que te rodean se sientan orgullosos de ti .Gracias por que sin ti esto no hubiese sido posible. Bonne te agradezco la forma en que contribuiste con este trabajo, eres una persona que da el paso al frente cuando es necesario. Gracias, mil gracias a los dos por su enorme contribución.

Le agradezco a todas mis amistades porque de una manera u otra estuvieron presentes siempre a lo largo de mi carrera estudiantil. A Yadiris por haber sido mi compañera de estudio y por haber compartido juntas tantos momentos de conocimientos, a Ismel por tener ese espíritu alentador en cada cosa que hacíamos juntos, a Denis por tenerme la confianza e inspirarme a hacer siempre lo correcto. A bety por compartir tantos momentos de convivencia juntas. A todas las muchachitas del apto 101205. Al grupo 6508 y 6105 por tantos momentos de alegría.

También le agradezco a aquellos profesores que marcaron mi vida en cada año de mi carrera, Yamila, Lacoste, Tejeda, José Carlo todos hicieron que cada año aprendiera algo nuevo.

Quiero agradecer a los integrantes del proyecto PostgreSQL que me acogieron con tanto cariño y en el cual aprendí muchas cosas. Anthony, Adrian, Marcos, gracias por estar siempre dispuestos a ayudar. Edgar gracias por dirigirnos de esa manera. Ojalá y cada departamento tuviera un jefe como tú.
Cada persona mencionada en este trabajo ha contribuido de manera positiva en mi vida. No tengo palabras para expresar a cada uno lo orgullosa que me siento de haber podido contar con personas como ustedes. A todos muchas gracias y nunca cambien, pues la vida es eso, ser auténticos y ayudar a los demás.

Agradecimientos

De Marlon

Le agradezco a mi madre que siempre ha luchado para que yo sea un hombre de bien. Mama lo logramos este es nuestro sueño hecho realidad .Gracias te doy por ser la esperanza y la luz que ilumina mi vida. En cada momento de mi vida has estado presente, nadie como tu ha tenido esa confianza que siempre haz depositado en mí .Gracias, no tengo palabras para describir lo orgulloso que estoy de tener una madre como tú. Eres y serás siempre mi ejemplo a seguir.

Agradezco también a mi padre por haberme transmitido la enseñanza necesaria para ser un hombre de bien. Gracias papá por darme el apoyo necesario para no decaer en los momentos más críticos de mi vida. Por ser tu quien me educara y me guiara en el transcurso de mi vida.

Me gustaría agradecer a mi hermano Hiscli que modestamente siempre me aconsejaba cuando más me hacía falta. Gracias por ser tú un hermano ejemplar en todo momento .Te admiro y no puedo dejar de agradecer el hecho de que la vida me diera un hermano como tú.

Agradezco a mi dúo de tesis por el esfuerzo realizado para lograr juntos que este trabajo saliera adelante.

A mis tutores, pilar esencial en el trayecto de este trabajo, llegue a ustedes mi agradecimiento por el esfuerzo realizado día a día. Sin ustedes este logro no hubiese sido posible.

Agradezco a mis amigos por compartir momentos buenos y malos junto a mí. Al grupo 6509 con los cuales me divertí bastante, a los varones del edificio 102 y a todos aquellos que de una manera u otra contribuyeron en mi vida estudiantil.

Dedicatoria

De Yeneysi

A mi padre Luis Ramírez González quisiera dedicarle este presente documento quien permanentemente me apoyó con su espíritu alentador, contribuyendo incondicionalmente a lograr mis metas y objetivos propuestos y que al brindarme su ejemplo me enseñó a ser perseverante y darme la fuerza que me impulsó a conseguirlo. Por el enorme esfuerzo que siempre hizo para que yo siempre tuviera ganas de luchar.

A mi madre Lazara Castellanos González por haber vivido mis días de universidad como si fuera yo. Cada sufrimiento mío, cada alegría, cada triunfo era de ella también. A ti por toda la infinita bondad, por todo el cariño, la confianza y el amor que me transmitiste. Sin palabras este triunfo es de las dos.

A mi hermano querido Javier Ramírez González por quien me he esforzado todo este tiempo, para que se sienta orgullo de su hermana y poderle servir de ejemplo a lo largo de su vida.

A mi abuelo Luis Ramírez Medina por mantener esa ilusión por su nieta. Por saber ganarse a las personas como nadie .Este logro es para ti porque siempre me esperabas con una sonrisa al llegar a casa.

Dedicatoria

De Marlon

A mi madre María Denis Martínez Días por ser mi gran inspiración para luchar por mis objetivos y metas, solo yo sé cuánto te mereces este logro. Por depositar en mí la confianza y ver más allá cuando todo estaba gris .Por ti soy quien soy. A ti va dedicado este triunfo.

A mi padre Pedro Morejón por guiarme en mi vida y ser el que me enseñó lo bueno y malo de todas las cosas. Por ser un ejemplo a seguir y enseñarme las alternativas necesarias cuando los caminos más cerrados estaban. Este momento es para ti para que siempre te sientas orgulloso del hijo que tienes.

A mi hermano que siempre me dio apoyo y aliento para que me sintiera bien .Por ser más que hermano un amigo .Quiero dedicarte este momento para que veas en mi a alguien digno de admirar.

A demás familiares que de una forma u otra contribuyeron con este trabajo.

Resumen

Uno de los pasos cruciales en la construcción de una aplicación que maneje una base de datos, es sin duda su diseño. Para suplir las necesidades que actualmente tiene el país en este ámbito se están desarrollando en diferentes centros herramientas que ayudan a facilitar el trabajo de diseño de bases de datos por parte de los usuarios, apostando siempre por la soberanía tecnológica. La Universidad de las Ciencias Informáticas es uno de los centros inmerso en este proceso, la cual cuenta con diferentes proyectos que realizan aplicaciones con el mismo fin. Uno de ellos es el proyecto PostgreSQL, en el cual se desarrollan herramientas que facilitan el trabajo con este gestor. Desde el año 2010 se implementa la aplicación insigne de este proyecto llamada herramienta de administración de bases de datos HABD que aunque posee funcionalidades muy potentes, no cuenta con un diseñador de bases de datos para llevar a cabo la confección de los modelos de datos. En el presente trabajo se desarrolló un plugin el cual fue integrado a HABD, lo que permite realizar el diseño de las bases de datos.

Palabras claves: bases de datos, diseñador, HABD, plugin

Índice

Índice de tablas

Índice de figuras

Introducción

En la actualidad se desarrollan diferentes sistemas dada la enmarcada difusión de las nuevas tecnologías informáticas en el ámbito internacional, cada uno de estos se aplican en áreas específicas de la sociedad, y se realizan con el fin de resolver una determinada problemática. La informática en sus diferentes ramas tiene asegurado un papel protagónico en el futuro de la humanidad, dependiendo este en gran medida de los conocimientos que se alcancen.

Con el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) se ha tratado de satisfacer en Cuba las necesidades de información y conocimiento de todas las personas y ámbitos de la sociedad. Con este proceso se ha logrado un mayor desarrollo de cada una de las esferas del país aumentando sistemáticamente la calidad de cada una de sus aplicaciones.

Dada las transformaciones del mundo de hoy y sus influencias en Cuba, es urgente preparar profesionales que sean capaces de afrontar el proceso de informatización que se está llevando a cabo para el desarrollo de un gran número de aplicaciones. Actualmente el país está batallando por colocarse entre las primeras naciones en el ámbito de la industria tecnológica, buscando nuevas alternativas dentro de esta industria para alcanzar resultados sobresalientes.

Uno de los centros inmerso en el proceso de desarrollo de software libre es la Universidad de la Ciencias Informáticas (UCI), pilar esencial en dicho proceso. Con diversos proyectos importantes aporta tanto al servicio de la universidad como al propio país, materializando convenios con empresas nacionales y extranjeras. Es un centro de estudios, con el propósito de desarrollarse de manera autodidacta y de esta forma ayudar al país económicamente. Para ello despliega un amplio parque tecnológico en materia de hardware y software, permitiendo la construcción de aplicaciones informáticas robustas, eficientes y con costes de tiempos mínimos.

En estos momentos la UCI se encuentra en un proceso de cambio, migrando algunas de las aplicaciones hacia software libre. Siguiendo las políticas de migración en las que actualmente está encaminada, el centro DATEC perteneciente a la universidad tiene asociados diferentes proyectos que están trabajando en la implementación de herramientas netamente libres, dentro de los cuales se encuentra el proyecto PostgreSQL. Este proyecto tiene como misión, mejorar las prestaciones de las bases de datos, contribuir al fortalecimiento de las tecnologías cubanas, utilizando como núcleo al gestor de bases de datos PostgreSQL, desarrollando soluciones integrales y consultorías relacionadas

con la migración y la explotación de este gestor.

Este proyecto cuenta con todo lo necesario para cumplir con dichos objetivos, pero aún existen necesidades de crear herramientas para el mejoramiento del mismo, es por ello que se crea la herramienta de administración de bases de datos HABD, logrando que los productos se realicen con la mayor calidad posible, permitiéndoles a los desarrolladores ampliar en gran mayoría los servicios. Dicha herramienta posee gran flexibilidad y brinda la posibilidad de incrementar funcionalidades haciendo más factible el uso para los usuarios. El resultado de esta herramienta es un Front –End, el cual no posee aún las funcionalidades para lograr que se realice el proceso de producción de un modelo detallado de datos de una base de datos, y el modelo de datos lógico conteniendo todas las opciones de diseño lógico y físico, así como los parámetros de almacenamiento para generar un diseño en un lenguaje de definición. Esto provoca que los usuarios no puedan realizar el diseño de sus bases de datos de forma gráfica produciéndose errores en los diseños e inconsistencias en los datos, por lo que el trabajo cada vez es más difícil para los mismos y el proceso lento y engorroso.

Por la situación planteada con anterioridad surge la siguiente **interrogante**: ¿Cómo diseñar bases de datos relacionales en la herramienta de administración de base de datos HABD para PostgreSQL?

Con el desarrollo de este trabajo se pretende dar solución al problema antes citado. Para la obtención de estos resultados se plantea como **objeto de estudio:** El proceso de modelado de bases de datos relacionales en PostgreSQL, enmarcado en el **campo de acción:** Herramientas para el diseño de bases de datos en PostgreSQL.

El **objetivo general** del trabajo: Desarrollar un Plugin para la herramienta de administración de base de datos HABD, que permita el diseño de bases de datos en PostgreSQL. En correspondencia con el mismo se trazan los siguientes **objetivos específicos**:

1. Analizar los diseñadores de bases de datos existentes en el mundo.
2. Diseñar el Plugin de bases de datos para la herramienta HABD.
3. Implementar el Plugin diseñador de bases de datos para la herramienta HABD.
4. Probar el plugin de bases de datos para la herramienta HABD.

Para dar cumplimiento a los anteriores objetivos se han definido un grupo de tareas que encaminarán el transcurso de la investigación

1. Caracterización de las herramientas diseñadoras de bases de datos existentes en el mundo.

2. Caracterización de los diseñadores de bases de datos embebidos en las herramientas de administración de bases de datos.

3. Definición del modelo de dominio.

4. Elaboración de las historias de usuarios.

5. Elaboración del plan de iteraciones.

6. Confección del modelo de diseño.

7. Implementación de las historias de usuarios identificadas.

8. Integración del plugin diseñador de bases de datos en la herramienta HABD.

9. Definición de las estrategias y técnicas de prueba, que probarán la solución desarrollada.

10. Ejecución de las estrategias y técnicas definidas que probarán las funcionalidades implementadas.

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

Capítulo 1: Fundamento teórico. En el capítulo se realiza una investigación de las herramientas libres que contienen embebidos diseñadores de base de datos. También se lleva a cabo el estudio de diferentes diseñadores de bases de datos que no están embebidos en herramientas de administración. Se plasman diferentes definiciones importantes para lograr el entendimiento de la investigación. Además se evidencia la metodología de desarrollo de software y se muestran las herramientas y tecnologías a utilizar.

Capítulo 2: Descripción del sistema propuesto. En el capítulo se muestran las diferentes características del sistema propuesto. Se define el modelo de dominio para lograr un mejor entendimiento del sistema. Se identifican las historias de usuarios. Se define la arquitectura del sistema. Son planificadas las iteraciones en las cuales serán implementadas cada historia de usuario, se estima el tiempo que durará la implementación de cada historia de usuario a través de las tareas de la ingeniería y se implementan las funcionalidades identificadas.

Capítulo 3: Validación y prueba. En este capítulo se realiza un estudio para definir las pruebas que serán aplicadas al sistema para verificar su correcto funcionamiento .Además se validan las funcionalidades implementadas a través de las estrategias y técnicas definidas. Se muestran los resultados obtenidos luego de haber realizado las pruebas.

Capítulo 1: Fundamento teórico

Introducción

Los sistemas de bases de datos ocupan hoy en día un lugar muy importante dentro de los sistemas de información, muchas aplicaciones o soluciones informáticas hacen uso de ésta forma de organización de la información. El escenario actual de la tecnología de bases de datos es el resultado del perfeccionamiento que a lo largo de la historia informática ha tenido lugar en el procesamiento de datos y en la gestión de la información, es por ello que en este capítulo se analizan los conceptos fundamentales relacionados con las bases de datos y se investigan las principales herramientas de diseño de bases de datos para de esta forma ver sus tendencias actuales .También se muestran diferentes características de las herramientas y tecnologías a utilizar para el desarrollo del mismo.

1.1 Conceptos asociados a la introducción

A continuación se enunciarán diferentes conceptos los cuales son de vital importancia en el proceso de entendimiento del desarrollo del presente trabajo.

1.1.1 Sistema Gestor de Bases de Datos (SGBD)

Los SGBD proporcionan a los usuarios la gestión del acceso a la base de datos por múltiples usuarios de manera simultánea y garantizan que no sucedan inconvenientes con la integridad de los mismos. Aunque poseen muchas ventajas es importante aclarar que el trabajo con SGBD no es una tarea fácil, pues contienen programas muy complejos que proveen grandes soluciones, y requieren de gran conocimiento en el tema de administración de bases de datos para poder obtener el mayor provecho de ellos. Para ayuda de los administradores de base de datos los gestores proveen interfaces y lenguajes de consulta que simplifican el trabajo con los datos. (1)

Se pueden mencionar diferentes SGBD que existen hoy en el mundo mostrando una lista de los que son comúnmente usados distinguiendo entre los SGBD libres y comerciales:

Ejemplos de SGBD libres

- PostgreSQL
- MySQL

Ejemplos de SGBD comerciales

- Oracle
- DB2, Informix (IBM)
- dBase (dBI)

• Paradox (Borland)
• SQL-Server (MS)
• Access (MS)
• FoxPro (MS)

En el siguiente sub-epígrafe se comentará las características más relevantes del sistema gestor de bases de datos PostgreSQL, debido a que es un gestor de bases de datos objeto-relacional que soporta gran parte del estándar SQL; que cuenta con características avanzadas como consultas complejas, llaves foráneas, disparadores, vistas, control de concurrencia multiversión ,su costo de adquisición es bajo o nulo , no tiene problemas con la licencia ,se puede desarrollar negocios bajo la licencia BSD, tiene requerimientos mínimos a la hora de instalarlo y si se utiliza en Linux es inmune a los virus. Es un gestor que está enfocado a las necesidades de la universidad.

1.1.2 PostgreSQL

El Centro de Tecnologías y gestión de datos (Datec) tomó como base al gestor de bases de datos PostgreSQL para futuros desarrollos; siendo la entidad, por parte de la Universidad de las Ciencias Informáticas, que proporcionó el impulso necesario para que las empresas cubanas comenzaran a utilizar tecnologías de bases de datos libres, contribuyendo al necesario proceso de migración que debe llevar a cabo el país.

Surge entonces la propuesta de crear un nuevo paquete de PostgreSQL, el PostgreSQL Empresarial Cubano, donde estén incluidos módulos para el análisis de datos, el monitoreo, la administración, el desarrollo y la seguridad en las bases de datos, un conjunto de herramientas que faciliten la utilización y optimización del gestor, así como una guía para su configuración y personalización, acorde a las necesidades de las empresas que lo utilicen. (2)

El poder contar con dicho paquete proporcionó para el proyecto de PostgreSQL aumentar las posibilidades de explotación y generalización del gestor para soluciones empresariales en el país, favoreciendo la migración de las empresas cubanas al gestor, esto influyó positivamente en la separación de las tecnologías propietarias para poder alcanzar la soberanía tecnológica.

1.1.3 Plugin

Un plugin es un complemento que sirve para la integración de otras aplicaciones obteniéndose una función nueva, este le permite a los desarrolladores interactuar con la aplicación y así aumentar la cantidad de funcionalidades que estos puedan realizar. Un programa puede tener varios conectores.

Permite también reducir el tamaño de la aplicación a la cual se le va a integrar dicho plugin y brinda la posibilidad de separar el código fuente de la aplicación debido a cualquier incompatibilidad que exista con respecto a las licencias.

1.2 Herramientas de administración

Las herramientas de administración de bases de datos son la base para el desarrollo de todo proyecto y para la calidad de todas las aplicaciones que se realicen, proveen facilidades para la manipulación de grandes volúmenes de datos .Una herramienta de administración de bases de datos es el eje principal de un software, actúa de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan, controlan la creación, el mantenimiento y el uso de las bases de datos de una organización y de sus usuarios finales. A continuación en el siguiente apartado se comentará sobre la herramienta de administración de bases de datos HABD creada en el proyecto de PostgreSQL en el año 2010.

1.2.3 Herramienta de administración de bases de datos (HABD)

El gestor PostgreSQL necesitaba de nuevas herramientas para la administración de sus bases de datos, es por ello que se crea la herramienta HABD para solucionar los problemas a la hora de realizar las aplicaciones y los productos. Dicha herramienta brinda la posibilidad de que los desarrolladores amplíen su gama de servicios y que se le agregue nuevas funcionalidades para hacer más factible el uso para los usuarios. La herramienta HABD está basada en plugin, el resultado de ella es un Front-End, el cual va a contribuir con la integración de dichos plugin para lograr la realización satisfactoria de las aplicaciones y que el trabajo sea más rápido y menos engorroso. Esta herramienta posee gran flexibilidad y una interfaz amigable. La aplicación contribuye al intercambio entre el usuario y el gestor PostgreSQL y facilita el trabajo de estos.

1.3 Diseñadores de bases de datos

Un diseñador de bases de datos permite de una forma visual realizar el diseño de las bases de datos, en especial cuando se habla de base de datos relacionales, es beneficioso ver las relaciones entre distintos campos de forma rápida y que deje muy pequeño margen de error. También realiza el proceso detallado de modelos de datos de una base de datos, permite hacer una separación entre los modelos lógicos y físicos, pero manteniendo una total integración entre ellos. Es posible generar múltiples modelos físicos asociados a un mismo modelo lógico, para diferentes plataformas. A continuación se

muestra un estudio realizado a los diseñadores de bases de datos embebidos en herramientas de administración y a los que no están embebidos en ninguna herramienta.

1.3.1 Diseñadores de bases de datos embebidos en herramientas de administración

Diseñador de bases de datos de PgAccess

PgAccess proporciona una interfaz gráfica para PostgreSQL donde se pueden gestionar las tablas, editarlas, definir consultas, secuencias y funciones. El diseñador de esta herramienta permite abrir cualquier base de datos en un determinado host, especificando dicho host, el puerto, el nombre de usuario y contraseña, ejecutar VACUUM ,crear tablas con un asistente, renombrar y borrar tablas y recuperar información sobre las tablas. Esta herramienta de administración, debido a que es una herramienta desactualizada, sin soporte y como no se puede tomar en cuenta como herramienta de administración pues su diseñador de bases de datos no es la solución al problema planteado.

Diseñador de bases de datos de PgAdmin

La herramienta de administración PgAdmin no posee un diseñador de bases de datos gráfico que permita realizar el proceso de producción de un modelo detallado de datos de una base de datos, relacionar las tablas, especificando atributos y campos. Los usuarios que trabajen con esta herramienta no pueden realizar el diseño de sus bases de datos de forma gráfica, debido a que gráficamente no existe ninguna aplicación que las diseñe. En el anexo número 1,2 y 3 se encuentra una imagen del PgAdmin donde se evidencia como se tiene que crear una tabla en esta herramienta.

.

1.3.2 Diseñadores de bases de datos que no están embebido en herramientas de administración

Visual Paradigm

Visual Paradigm permite diseñar las bases de datos de forma dinámica, permite realizar el diseño de sus tablas y llenar las mismas con sus atributos, brinda la posibilidad de relacionar las tablas entre sí, estableciendo relaciones entre ellas, de esta forma logra la realización de diagramas los cuales te permiten una vez hecho generar un script. Esta herramienta permite establecer las relaciones de uno a mucho, uno a uno y mucho a mucho. Visual Paradigm proporciona un conjunto de productos que le facilita a las organizaciones de diseño, integrar y desplegar sus aplicaciones y sus bases de datos.

A pesar de todas las características descritas anteriormente Visual Paradigm es una herramienta que para el diseño de sus bases de datos no soporta algunos tipos de datos para PostgreSQL como son:

XML y Moneda. Además esta herramienta no realiza la herencia entre tablas, por lo que no cumple con las características que el proyecto PostgreSQL necesita para encontrar una aplicación que se pueda integrar a la herramienta de administración de bases de datos HABD.En el anexo número 4 se muestra un diagrama diseñado en Visual Paradigm.

PgDesigner

PgDesigner es un programa de código abierto para la base de datos de diseño gráfico en PostgreSQL. El código está escrito en el lenguaje Gambas, y en la actualidad sólo se ejecuta en sistema operativo Linux. (3)

Es una herramienta excelente en productividad y agilidad. Con ella se puede crear las relaciones visualmente, crear tablas con sus columnas y todo lo referente a ellas. Es el complemento o plugin que le falta al PgAdmin III para estar un tanto a la altura de SQL Server. Actualiza automáticamente las relaciones entre tablas. Este diseñador es un asistente para la construcción de puntos de vistas, impresión del diagrama de bases de datos. Permite generar el script, salvar el modelo realizado y cargarlo en caso que fuese necesario.

PgDesigner es una herramienta potente, pero no cuenta con una documentación sobre su instalación. Necesita librerías Gambas las cuales deben instalarse como paquetes adicionales para poderlo ejecutar, condición no factible para la solución del problema planteado. Además PgDesigner soporta solamente las versiones de PostgreSQL 7.x y 8.x y actualmente el gestor se encuentra en la versión 9.1.En el anexo número 5 se evidencia una imagen de este diseñador.

DBDesigner 4

DBDesigner es un sistema totalmente visual de diseño de bases de datos, que combina características y funciones profesionales con un diseño simple, muy claro y fácil de usar, a fin de ofrecerte un método efectivo para gestionar tus bases de datos. Te permite administrar la base de datos, diseñar tablas, hacer peticiones SQL manuales y mucho más, como ingeniería inversa en MySQL, Oracle, MSSQL y otras bases de datos, modelos XML. (4)

Este diseñador brinda la posibilidad de conectarse con el "backend" de la base de datos y exporta e importa el script SQL. Dispone de excelente documentación, facilitando el trabajo para los usuarios. Es de código abierto disponible para Microsoft Windows NT/XP/Vista/7 y Linux KDE/Gnome, distribuido con licencia GPL. Es importante destacar que la herramienta DBDesigner posee grandes

funcionalidades desarrolladas y optimizadas para el código abierto de MySQL, no así para el gestor PostgreSQL en el cual se trabaja actualmente. En el anexo número 6 se evidencia una imagen de un modelo realizado en este diseñador.

SQL Designer

SQL Designer es un diseñador que permite a los usuarios realizar el diseño de sus bases de datos de manera exitosa. Brinda la posibilidad de crear una estructura bien definida de las bases de datos, definir índices y claves foráneas. Permite importar esquemas directamente desde bases de datos existentes. Exporta los modelos en un formato XML o los guarda directamente a una base de datos para su futura edición.

Es muy sencillo ya que su interfaz permite cambiar su idioma entre varios disponibles incluido el español, además permite crear los objetos, editarlos, establecer sus relaciones y arrastrarlos, todo esto desde la barra de tareas, esto hace que la herramienta resulte muy fácil e intuitiva de usar y que en poco tiempo uno logre dominarla.SQL Designer es una potente y práctica herramienta que como la podemos instalar en nuestro servidor, podemos acceder a ella desde cualquier equipo con conexión a Internet. (5)

Esta herramienta es un diseñador que contiene características muy buenas para el diseño de las bases de datos pero esta desarrollado en PHP, MySQL, y Java script por lo que para instalarlo necesitamos un servidor Web con soporte para PHP y MySQL, y una de las características principales con las cuales debe contar el plugin es que tenga las mínimas dependencia hacia otras tecnologías.

ERWIN

Es una herramienta de diseño de bases de datos .Brinda productividad en diseño, generación y mantenimiento en aplicaciones. Desde un modelo lógico de los requerimientos de información, hasta el modelo físico perfeccionado para las características específica de la base de datos diseñada. Más que una herramienta de dibujo ERWIN automatiza el proceso de diseño de una manera inteligente. (6)

Esta herramienta permite diseñar las bases de datos de alto desempeño .Separa los modelos físicos y lógicos. Permite realizar ingeniería inversa y hacia delante. Es sin duda una herramienta muy potente y muy usable logrando dejar satisfecho al usuario con la realización de sus diseños de bases de datos pero es una herramienta muy costosa y privativa, por lo cual no se puede hacer uso de ella para diseñar las bases de datos en la herramienta de administración de bases de datos HABD.

ER/Studio

Es una herramienta de modelado de datos fácil de usar y multinivel para el diseño de bases de datos a nivel físico y lógico. Direcciona las necesidades diarias de los administradores de bases de datos, desarrolladores y arquitectos de datos que construyen y mantienen aplicaciones de bases de datos grandes y complejas. ER/Studio está equipado para crear y manejar diseños de bases de datos funcionales y confiables. Ofrece fuertes capacidades de diseño lógico, sincronización bidireccional de los diseños físicos y lógicos, construcción automática de bases de datos, documentación y fácil creación de reportes. (7)

Este diseñador permite generar otros objetos de bases de datos como son: vistas ,procedimientos almacenados ,reglas y tipos de datos de usuarios. Brinda la posibilidad a los usuarios de cambiar el estilo de las líneas ,los colores y los tipos de letras. Pero esta herramienta es privativa y trabaja sobre los diferentes sistemas operativos de windows solamente.

El estudio hecho anteriormente arrojó como conclusión que ninguna de las herramientas analizadas brindan la solución a la interrogante planteada ya que no pueden integrarse como plugin a la herramienta de administración de bases de datos HABD. Es por ello que se decide realizar una herramienta que cumpla las expectativas del proyecto y que ayude a solucionar el problema planteado anteriormente A pesar de no haber seleccionado ninguno de los diseñadores analizados se tomaron en cuenta características importantes para la realización del sistema como pueden ser: comprobar que no existan atributos de igual nombre entre dos tablas y la forma de insertar datos visualmente en una tabla.

1.4 Metodologías de Desarrollo de Software

El proceso de desarrollar software no es una tarea fácil, se debe contar con un proceso bien detallado y para esto se necesita aplicar una metodología que sea capaz de llevar a cabo el control total del producto. Las Metodologías de Desarrollo de Software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto de software. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo software, pero los requisitos de un software a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del software. (8)

Las metodologías se dividen en dos grupos, tradicionales (pesadas) y ágiles (ligeras). Las tradicionales, se centran en la definición detallada de los procesos y tareas a realizar, herramientas a utilizar, y requiere una extensa documentación, pretendiendo prever todo de antemano. En las ágiles es más importante lograr que un producto de software se realice con la calidad requerida que hacer una buena documentación, en este tipo de metodología el cliente está presente en todo momento y colabora con el proyecto como un miembro más.

Dentro de las metodologías de desarrollo, se decidió trabajar con la Metodología Extreme Programming (XP) debido a que posee un grupo de ventajas que logran que los trabajos salgan con la mayor calidad posible. Dentro de las ventajas se puede mencionar, que es una metodología que satisface al cliente a través de la entrega temprana y continua de las aplicaciones, son aceptados los requisitos que puedan tener algún cambio durante el desarrollo del proyecto. Su principal asunción es que con una corta planificación, un poco de codificación y unas pocas pruebas se puede decidir si se está siguiendo un camino acertado o equivocado, evitando así tener que virar hacia atrás demasiado tarde. A continuación se hará una breve caracterización de la metodología XP para tener un conocimiento más amplio de la misma.

1.4.1 Metodología Extreme Programming (XP)

XP es una de las llamadas metodologías ágiles de desarrollo de software más exitosas de los tiempos recientes. La metodología propuesta en XP está diseñada para entregar el software que los clientes necesitan en el momento en que lo necesitan. XP alienta a los desarrolladores a responder a los requerimientos cambiantes de los clientes, aún en fases tardías del ciclo de vida del desarrollo. (9)

Esta metodología ofrece muchas ventajas, con ella en un proyecto se hace más eficaz el trabajo porque se ahorra tiempo en documentación y los miembros del equipo se enfocan en poner todo sus esfuerzos en la realización del producto, los cuales son más fiables y robustos contra los fallos gracias al diseño de las pruebas de forma previa a la codificación. También posibilita que existan la menor cantidad de errores, ya que los requisitos obtenidos pueden ser modificados con gran facilidad. En esta metodología cualquier miembro del proyecto puede realizar el rol de desarrollador, brinda la posibilidad de que el código siempre se mejore y se simplifique usando sistemas tipo CVS (Controlador de versiones) para evitar la duplicación de trabajo. Además se utilizará esta metodología porque el proyecto de PostgreSQL cumple con cuatro de los 5 criterios que se debe tener en cuenta en un proyecto para aplicar una metodología ágil. Es una metodología para fortalecer el trabajo de un

proyecto en equipo, donde el cliente forma parte de este proceso y de esta manera el proyecto se enfoca en las necesidades del mismo para que se sienta satisfecho.

1.5 Tecnologías y herramientas a utilizar

Para el desarrollo del plugin diseñador de bases de datos que será parte del conjunto de plugin que serán integrados en la herramienta de administración HABD del gestor PostgreSQL es necesario tener en cuenta las herramientas con las que se va a realizar el diseño y la implementación del mismo. A continuación se mencionan las que se utilizarán para la realización de la aplicación:

1.5.1Herramientas CASE (Computer-Aided Software Engineering) para el modelado

Las herramientas de modelado son un grupo de programas que utilizan las personas que intervienen en el desarrollo de un software, para agilizar y facilitar su trabajo, dichas herramientas proveen de métodos, técnicas y utilidades que ayudan al perfeccionamiento del desarrollo de sistemas de información, de forma total o parcialmente. Las mismas ofrecen muchos beneficios para todos los involucrados en un proyecto, permiten aplicar la metodología de análisis y diseño orientados a objetos y abstraerse del código fuente, en un nivel donde la arquitectura y el diseño se tornan más obvios y más fáciles de entender y modificar. Estas herramientas son de gran ayuda en todos los aspectos del proceso de desarrollo de cualquier software. En la actualidad, posibilitan la creación y modificación de diagramas con gran facilidad, además de automatizar varias actividades como generación de código para el desarrollo de software, lo cual mejora considerablemente la calidad, el rendimiento, la utilidad y fiabilidad de las herramientas CASE.

Visual Paradigm para UML 6.4

Visual Paradigm para UML es una herramienta multiplataforma que permite proporcionar un ambiente de modelado visual para satisfacer la tecnología del software de hoy y necesidades de comunicación, es muy fácil de usar y presenta un ambiente gráfico y agradable para el usuario, soporta el ciclo de vida completo de cualquier metodología.

Posibilita además la construcción rápida de aplicaciones con una mayor calidad y a menor costo. Es una herramienta colaborativa, por lo que soporta múltiples usuarios trabajando sobre el mismo proyecto; facilita el modelado de base de datos, requerimientos, los procesos de negocio, la interoperabilidad, genera la documentación del proyecto automáticamente en varios formatos como

web o PDF y de código base para lenguajes como Java, C# y PHP así como la integración con otras herramientas de desarrollo. Posee además una interfaz gráfica muy amigable y fácil de usar por el usuario. (10)

Diseña y desarrolla productos que eliminan la complejidad, mejoran la productividad, y comprimen el software en los plazos de desarrollo. Se decide utilizar esta herramienta porque tiene la capacidad de ejecutarse sobre diferentes sistemas operativos lo que le confiere la característica de ser multiplataforma, está disponible para Windows, Linux, MacOS X, etc. Es fácil de instalar y actualizar y compatible entre ediciones. Integra diferentes funcionalidades para el desarrollo de aplicaciones. Además la Universidad De Las Ciencias Informáticas (UCI), cuenta con su licencia y la utiliza mucho en proyectos productivos para realizar aplicaciones que contribuyen al desarrollo de la universidad.

1.5.2 Lenguaje de Modelado

El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño. Algunas organizaciones los usan en combinación con una metodología de desarrollo de software para avanzar de una especificación inicial a un plan de implementación y para comunicar dicho plan a todo un equipo de desarrolladores.

Lenguaje Unificado de Modelado (Unified Modeling Language o UML)

El Lenguaje Unificado de Modelado (UML) es un lenguaje para especificar, visualizar construir y documentar los artefactos de los sistemas software, así como para el modelado del negocio y otros sistemas no software. (11)

UML, por sus siglas en inglés, (Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad, sirve para el modelado de sistemas con tecnología orientada a objeto que permite especificar, visualizar, construir y documentar ,está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el mismo es un lenguaje, cuenta con reglas para combinar tales elementos.

Se hará uso de este lenguaje para el desarrollo de este trabajo debido a que ofrece un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación,

esquemas de bases de datos y componentes de software reutilizables. Pero además es tan bueno que incluso el Grupo de Manejo de Objeto (OMG) lo utiliza como estándar de modelado.

1.5.3 Entorno de desarrollo

En el momento de escoger un lenguaje de programación, se debe tener en cuenta específicamente lo perseguido. La versatilidad de un lenguaje está estrechamente relacionada con la complejidad que posea. Según la complejidad en el aprendizaje de cierto lenguaje así será el espectro de tareas que puede resolver, mientras más complejo más amplio.

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que son interpretadas por las computadoras, diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, es un modo práctico para que los seres humanos puedan dar instrucciones a una computadora. También la palabra programación se define como el proceso de creación de un programa de computadora, mediante la aplicación de procedimientos lógicos. (12)

Un lenguaje es diseñado para describir un conjunto de acciones que un equipo debe ejecutar. Está formado por un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Permiten principalmente el desarrollo de software, además de que es utilizado para controlar el comportamiento físico y lógico de una máquina.

C++

Aunque en un principio C++ se plantea como una mejora de C (’C con clases’), en la actualidad es un lenguaje independiente, aunque se conserva la compatibilidad con C. Es un lenguaje fuertemente tipado, soporta multitarea mediante clases y permite modularidad. (13)

Se trata simplemente del sucesor de un lenguaje de programación hecho por programadores (de alto nivel) para programadores, lo que se traduce en un diseño pragmático al que se le han ido añadiendo todos los elementos que la práctica aconsejaba como necesarios, con independencia de su belleza. Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales. Es un lenguaje procedural (orientado a algoritmos) y orientado a objetos. Se decide para la realización de este trabajo escoger este lenguaje de programación, debido a que es un lenguaje que permite que los niveles de velocidad de ejecución de los programas sean más altos, el consumo de memoria es más pequeño que otros

lenguajes, permite que los programadores realicen el código en un alto y bajo nivel y PostgreSQL tiene sus librerías programadas en este lenguaje.

1.5.4 Entorno de desarrollo integrado (IDE)

Un Entorno de Desarrollo Integrado o IDE (por sus siglas en inglés), es una herramienta que permite a los desarrolladores de software escribir sus programas en uno o más lenguajes. Consiste básicamente en una plataforma en la que se integran un editor de código, un compilador, un depurador y una interfaz gráfica de usuario.

Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Los IDE pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Se hará referencia al IDE QT Creator siendo este más manejable para los desarrolladores en el trabajo con las aplicaciones.

QT Creator

Es un nuevo entorno de desarrollo integrado ligero, el cual fue lanzado por Nokia. Esta plataforma cruzada soporta los sistemas operativos incluyendo Linux, Mac Os X y Windows. (14)

Qt Creator es un entorno multi-plataforma de desarrollo integrado (IDE) adaptados a las necesidades de los desarrolladores de Qt. Es también la primera versión de Qt bajo las tres licencias: LGPL, privativas y GPL. También posee una GUI (Interfaz Gráfica de Usuario) y un resaltado y completado de código .Se decide utilizar este IDE debido a que fue diseñado para hacer que el desarrollo en C++ sea más rápido y fácil ya que posee un editor de código C++.

1.5.5 Framework

Un framework en el desarrollo de software es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado, típicamente puede incluir soportes de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

En general, con el término framework, se hace referencia una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta. (15)

Los framework son la piedra angular de la moderna ingeniería del software. Un framework no tiene funcionalidades de una aplicación específica, sino que las aplicaciones se construyen sobre ellos. A continuación se enuncian diferentes características del framework QT 4.7.

QT 4.7

Qt es un framework de desarrollo de aplicaciones multiplataforma. Viene acompañado de un conjunto de herramientas para facilitar su uso. Incluye una colección de contenedores genéricos. (16)

Se utiliza ampliamente para el desarrollo de aplicaciones de software con una interfaz gráfica de usuario (GUI), para el desarrollo de programas no GUI, tales como herramientas de línea de comandos y las consolas de los servidores. Funciona en todas las principales plataformas y tiene un amplio apoyo. El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL. En este trabajo se hace uso de este framework ya que posee disponibilidad del código fuente, rendimiento de C++, compatibilidad multiplataforma con un solo código fuente, excelente documentación y una arquitectura lista para plugin.

1.5.6 Controlador de versiones

Un sistema de control de versiones es un es un software que administra el acceso a un conjunto de ficheros, y mantiene un historial de cambios realizados. Por supuesto, sólo el seguimiento de las distintas versiones de un usuario (o grupo de usuarios) y los archivos-director no es muy interesante en sí mismo. Un controlador de versiones es útil porque permite explorar los cambios que dio lugar a cada una de esas versiones y facilita el retiro arbitrario de la misma.

Subversion

Subversion es un sistema de control de versiones libre/código abierto. Maneja ficheros y directorios con el paso del tiempo. Árbol de ficheros en un repositorio central. Puede acceder a su repositorio a través de redes, lo que le permite ser utilizados por personas de diferentes computadoras. En cierto nivel, la posibilidad de que varias personas puedan modificar y gestionar el mismo conjunto de datos de sus respectivas ubicaciones fomenta la colaboración. (17)

Se puede progresar más rápidamente sin un único conducto a través del cual todas las modificaciones se producen. Los archivos versionados no tienen cada uno un número de revisión independiente, en cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos

los archivos del repositorio en un instante determinado. Se decide utilizar este controlador de versiones porque brinda la posibilidad de gestionar las modificaciones durante el desarrollo. Permite que varias personas trabajen sobre los mismos ficheros, implementa un sistema virtual de fichero versionado que sigue los cambios en todos los árboles de directorios. Además expresa las diferencias entre ficheros usando un algoritmo de diferenciación binario, que funciona exactamente igual, tanto en ficheros de textos como en ficheros binarios.

1.6 Conclusiones del capítulo

En el capítulo se realizó un estudio de los diseñadores de bases de datos, analizando las principales desventajas de cada uno de ellos .También se tuvieron en cuenta algunas de las características principales de estos diseñadores para implementar una herramienta práctica y sencilla que permita diseñar de forma gráfica las bases de datos. Fue seleccionada la metodología Extreme Programing, la cual guiará el proceso de desarrollo del software. Se realizó un estudio de las herramientas y tecnologías existentes con el propósito de seleccionar las más adecuadas a la solución expuesta; partiendo del anterior análisis se determinó utilizar como herramienta CASE Visual Paradigm 6.4 con el lenguaje de modelado UML, el framework QT 4.7, el IDE de desarrollo QT Creator, empleando el lenguaje de programación C++, utilizando el controlador de versiones Subversión.

Capítulo 2: Descripción de la solución

Introducción

En el capítulo se describen las principales características que tendrá el diseñador. Se muestran los requisitos, se definen las historias de usuario y se realiza un diagrama de clases del diseño que se utiliza como complemento de la metodología definida para lograr un mejor entendimiento del componente. Además se planifican las iteraciones en las cuales serán implementadas cada historia de usuario y se realiza una estimación de la duración de cada tarea de la ingeniería para de esta forma lograr una implementación correcta y con éxito.

2.1. Identificación del problema

El diseño de las bases de datos es importante para la realización de cualquier proyecto que requiera de una base de datos, esto hace que las aplicaciones tengan mejor calidad y sean lo más eficaz posible. El proyecto de PostgreSQL es uno de los proyectos de la Universidad de las Ciencias Informáticas que se enfoca en el trabajo con las bases de datos teniendo siempre en cuenta los elementos necesarios para la realización de las mismas. En este proyecto se creó en el año 2011 una herramienta de administración llamada HABD con el objetivo de lograr la realización de diversas operaciones sobre las bases de datos, ella posee una interfaz amigable para los usuarios, su resultado es un Front-End y está basada en una arquitectura de plugins, pero aun no tiene las funcionalidades necesarias para poder realizar los diseños de sus bases de datos. El proyecto de PostgreSQL necesita para que esta herramienta pueda realizar todo tipo de operaciones un diseñador de bases de datos para integrarlo a la misma, de este modo se puede realizar los modelos de las bases de datos y la ejecución y generación de los script. El presente trabajo de diploma arrojará la solución a dicho problema con la creación de un plugin para diseñar las bases de datos.

2.2. Modelo de dominio

El modelo de dominio es una visualización de los conceptos en el dominio del mundo real, para la realización del mismo se necesita definir las clases conceptuales, las cuales son las más significativas y brindan un mejor entendimiento del sistemas. Este modelo es un diagrama con los objetos que existen (reales) relacionados con el proyecto que se va a realizar y las relaciones que hay entre ellos. El Modelo de Dominio ayuda a comprender los conceptos que utilizan los usuarios, los conceptos con los que trabajan y con los que deberá trabajar la aplicación. A pesar de que este modelo no forma parte

de la metodología escogida, se decide realizar debido a que brinda un mejor entendimiento para la implementación del plugin.

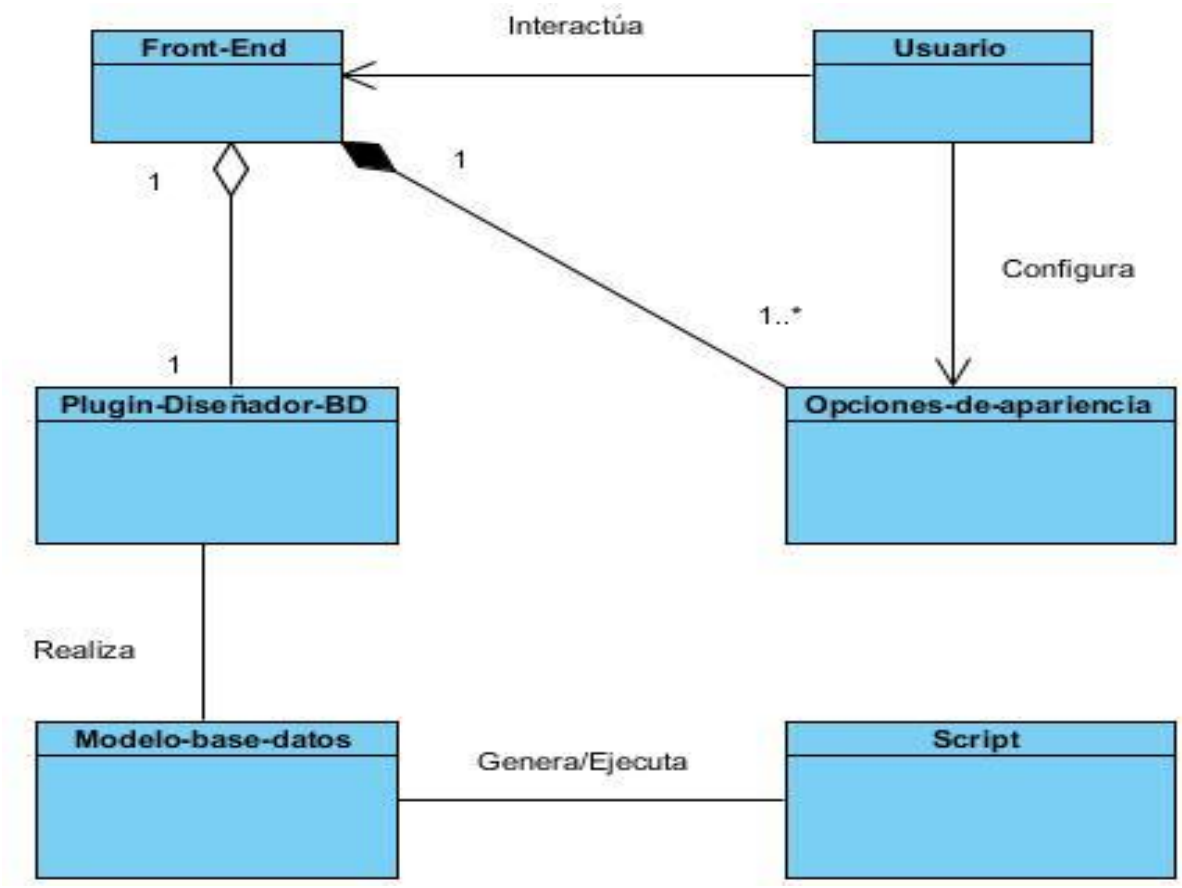


Figura 1: Modelo de dominio del sistema

Usuario: Persona que interactúa con la herramienta.

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

Plugin-Diseñador-BD: Es el módulo encargado de diseñar los modelos de las bases de datos.

Script: Modelo generado por el diseñador de bases de datos.

2.3. Propuesta del componente a desarrollar

Diseñador de bases de datos para HABD

El diseñador de bases de datos de la herramienta HABD brinda la posibilidad de realizar gráficamente el proceso completo de cada uno de los modelos que contienen los datos y tablas para una base de datos, esta herramienta consiste en realizar la gestión de las tablas dando la posibilidad de eliminarlas, crearlas, visualizarlas y modificarlas. Permite además insertar eliminar, modificar y visualizar los atributos de las tablas, trabajar la herencia, especificando restricciones necesarias. Para lograr un buen diseño en los modelos se decidió que cuando un usuario decida crear un atributo o una tabla no permitirle poner espacio y si comienza con un dígito se le notificará al usuario una alerta de lo que está escribiendo y de esta manera las bases de datos se conforman, sin errores ni inconsistencias. También genera el script de la base de datos para que el usuario tenga la posibilidad de cargarlo en otro lugar .Maneja de forma dinámica el trabajo con los esquemas y tablespace. Una vez creado el modelo, el diseñador accede a exportarlo en formato mdl y cargarlo en caso de que sea necesario.

2.4. Historias de usuarios

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. (18)

Se definieron para el desarrollo del diseñador de bases de datos 9 historias de usuarios, de las cuales se muestran en las siguientes tablas las más significativas.

Tabla 1. Historia de usuario: Gestionar tabla

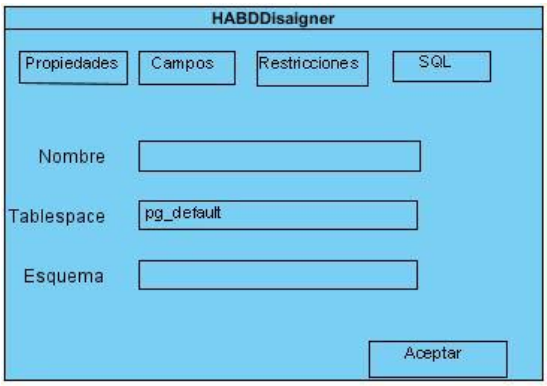
Historia de Usuario	
Número: 1	**Nombre de la Historia de Usuario:** Gestionar Tablas.
Cantidad de modificaciones a la Historia de Usuario: 0	
Usuario: Usuario	**Iteración asignada:** 1
Prioridad en negocio: Muy alta	**Puntos estimados:** 2 semana
Riesgo en desarrollo: Alto	**Puntos reales:** 2 semanas.
Descripción: Permitirá la creación, eliminación, modificación y visualización de la tabla con sus atributos y restricciones.	
Observaciones: De no establecer el nombre de la tabla la aplicación emitirá un mensaje indicando la realización de esta acción.	
Prototipo de interfaces: 	

Tabla 2. Historia de usuario: Gestionar atributos

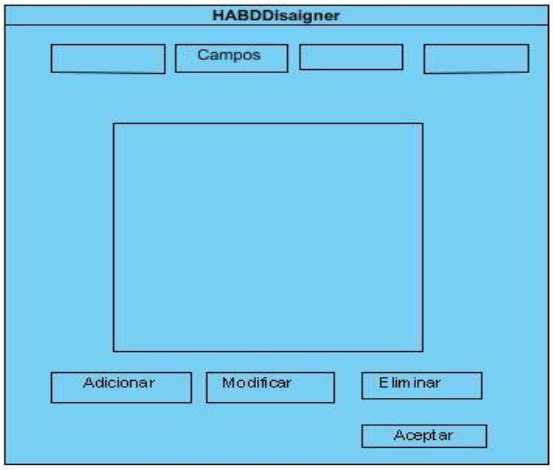
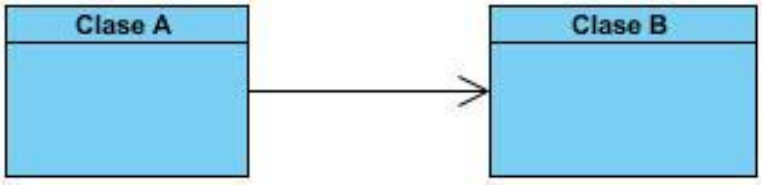
<table>
<tr><td colspan="3" align="right">Historia de Usuario</td></tr>
<tr><td>Número: 5</td><td colspan="2">Nombre de la Historia de Usuario: Gestionar Atributos.</td></tr>
<tr><td colspan="3">Cantidad de modificaciones a la Historia de Usuario: 0</td></tr>
<tr><td colspan="2">Usuario: Usuario</td><td>Iteración asignada: 1</td></tr>
<tr><td colspan="2">Prioridad en negocio: Alta</td><td>Puntos estimados: 2 semana</td></tr>
<tr><td colspan="2">Riesgo en desarrollo: Alto</td><td>Puntos reales: 2 semanas.</td></tr>
<tr><td colspan="3">Descripción: Permite crear, eliminar, modificar y visualizar los atributos con sus propiedades.</td></tr>
<tr><td colspan="3">Observaciones:</td></tr>
<tr><td colspan="3">Prototipo de interfaces:
 
</td></tr>
</table>

Tabla 3. Historia de usuario: Establecer relación

<table>
<tr><td colspan="2" align="right">Historia de Usuario</td></tr>
<tr><td>Número: 9</td><td>Nombre de la Historia de Usuario: Establecer relación.</td></tr>
<tr><td colspan="2">Cantidad de modificaciones a la Historia de Usuario: 0</td></tr>
<tr><td>Usuario: Usuario</td><td>Iteración asignada: 1</td></tr>
<tr><td>Prioridad en negocio: Muy Alta</td><td>Puntos estimados: 2 semanas</td></tr>
<tr><td>Riesgo en desarrollo: Muy Alto</td><td>Puntos reales: 2 semanas.</td></tr>
<tr><td colspan="2">Descripción: Establece la relación entre dos tablas pasando los atributos de una de las tablas hacia la otra especificando el tipo de relación. Además se puede eliminar la relación establecida, actualizándose los cambios ocasionados en las tablas correspondientes a la relación eliminada.</td></tr>
<tr><td colspan="2">Observaciones:</td></tr>
<tr><td colspan="2">Prototipo de interfaces:
 
</td></tr>
</table>

2.5 Lista de reserva del producto

En XP la lista de reserva del producto representa todos los requerimientos, tanto funcionales como no funcionales ordenados todos según la prioridad con la estimación de cada uno de los requerimientos para su implementación por semanas y el rol que realizó la estimación del requerimiento.

Tabla 4. Lista de reserva del producto

Ítem *	Descripción	Estimación	Estimado por
Prioridad: Muy Alta			
1	Crear Tabla	0.5	Analistas
2	Eliminar tabla	0.5	Analistas
3	Modificar tabla	0.5	Analistas
4	Visualizar tabla	0.5	Analistas
5	Crear atributos	0.5	Analistas
6	Eliminar atributos	0.5	Analistas
7	Modificar atributos	0.5	Analistas
8	Visualizar atributos	0.5	Analistas
9	Implementar actualización de tabla	0.6	Analistas Analistas
10	Implementar actualización de atributos	0.6	Analistas Analistas
11	Diseñar relación en el visual	0.6	Analistas
Prioridad: Alta			
12	Mostrar SQL generado	0.5 0.5	Analistas
13	Implementar generarScript	0.5 0.5	Analistas
14	Implementar conexión	0.5	Analistas
15 **16**	Implementar llamada al método generarScript	0.5	Analistas

17	Guardar datos del modelo	1	Analistas
18	Cargar modelo	1	Analistas
Prioridad: Media			
19	Crear tablespace	0.2	Analistas
20	Eliminar tablespace	0.2	Analistas
21	Modificar tablespace	0.2	Analistas
22	Visualizar tablespace	0.2	Analistas
23	Crear esquema	0.2	Analistas
24	Eliminar esquema	0.2	Analistas
25	Modificar esquema	0.2	Analistas
26	Visualizar esquema	0.2	Analistas
Prioridad: Baja			
1	**Usabilidad.** La aplicación podrá ser utilizada por cualquier persona con conocimientos básicos del Gestor PostgreSQL interesado en realizar el diseño de una base de datos.		
2	**Rendimiento.** La herramienta que se propone debe ser rápida y eficaz realizando el diseño de sus bases de datos con calidad.		

3	**Portabilidad.** Será un sistema multiplataforma, lo que permitirá poder disponer del mismo en cualquier sistema operativo.		
4	**Software.** Para poder hacer uso de este plugin se debe tener instalado las librerías de QT, el compilador de C++ y PostgreSQL.		
5	**Apariencia o interfaz externa.** La aplicación al integrarse con la herramienta HABD tendrá una interfaz amigable y fácil de interactuar para hacer más factible el uso para los usuarios.		
6	**Disponibilidad.** Se podrá hacer uso de la aplicación siempre que estén instaladas todas las librerías necesarias.		

7	**Restricciones de diseño e implementación:** El diseñador de bases de datos debe estar listo para el despliegue en un tiempo aproximado de diez meses y para el desarrollo del mismo se usará el lenguaje de programación C++.		

2.6. Tareas de la ingeniería

Una vez descritas las historias de usuarios, se procede a describir cada una de las tareas que se van a elaborar dentro de cada historia de usuario, cada una contendrán los objetivos que se deben cumplir, el tiempo en que tardará en realizar la tarea y el responsable asignado. (19)

Las tareas de la ingeniería se consideran como las entradas de trabajo para el equipo de programadores. Es la ficha que contiene el número identificador de la tarea, el identificador de la historia de usuario con la que está relacionada, el nombre de la tarea, la fecha de inicio, la fecha de fin, el equipo responsable y la descripción. A continuación se muestran tres ejemplos de algunas de las tareas de ingenierías asociadas a las historias de usuarios mencionadas anteriormente.

Tabla 5. Tarea de la ingeniería de la historia de usuario 1

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario: 1**
Nombre Tarea: Crear tabla	
Tipo de Tarea : Desarrollo	**Puntos Estimados: 0.5**
Fecha Inicio: 1/12/2011	**Fecha Fin: 4/12/2011**
Programador Responsable: Marlon A Morejón Martínez	
Descripción: Es la encargada de crear la tabla con su nombre, sus atributos y sus restricciones.	

Tabla 6.Tarea de la ingeniería de la historia de usuario 2

Tarea de Ingeniería	
Número Tarea: 5	**Número Historia de Usuario: 2**
Nombre Tarea: Crear atributos	
Tipo de Tarea : Desarrollo	**Puntos Estimados: 0.5**
Fecha Inicio: 9/1/2012	**Fecha Fin: 12/1/2012**
Programador Responsable: Marlon A Morejón Martínez	
Descripción: El usuario escribe el nombre y selecciona el tipo de datos y especifica si el atributo es nulo o no nulo y de esta forma crea los atributos con sus propiedades.	

Tabla 7. Tarea de la ingeniería de la historia de usuario 3

Tarea de Ingeniería	
Número Tarea: 9	**Número Historia de Usuario: 3**
Nombre Tarea: implementar la actualización de la tabla	
Tipo de Tarea: Desarrollo	**Puntos Estimados:0.6**
Fecha Inicio: 27/1/2012	**Fecha Fin: 1/2/2012**
Programador Responsable: Marlon A Morejón Martínez	
Descripción: Establece la relación entre dos tablas pasando los atributos de una de las tablas hacia la otra modificándole el nombre de los atributos.	

2.7 Plan de iteraciones

Una vez definidas las historias de usuario es necesario crear un plan donde se indiquen las historias de usuario que se crearán para cada versión del programa y las fechas en las que se publicarán estas versiones. Documento en el que se especifican las iteraciones necesarias para construir el producto software. (20)

Un plan de iteraciones es una planificación donde los desarrolladores y clientes establecen los tiempos de implementación ideales de las historias de usuario, la prioridad con la que serán implementadas y las historias que serán implementadas en cada versión del programa. Después de un plan de iteraciones tienen que estar claros estos cuatro factores: los objetivos que se deben cumplir (que son principalmente las historias que se deben desarrollar en cada versión), el tiempo que tardarán en desarrollarse y publicarse las versiones del programa, el número de personas que trabajarán en el desarrollo y cómo se evaluará la calidad del trabajo realizado.

Tabla 8. Plan de iteraciones

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
1	En esta iteración se realizarán las historias de usuarios de prioridad my alta, estas historias de usuarios se encargarán de el proceso de gestionar todas las tablas, los atributos, estableciendo las relaciones entre cada tabla.	1-2-3	6 semanas
2	El objetivo de esta iteración es que el plugin de diseñador de bases de datos logre generar y ejecutar el script, salvar y cargar los modelos realizados, ya que todas las historias de usuarios son de prioridad alta.	5-6-8-9	4 semanas
3	En esta iteración se implementarán las historias de usuarios de prioridad media, con esta iteración se logrará trabajar con los tablespace y los esquemas.	4-7-10	4 semanas

2.8. Diseño del sistema

El diseño de cualquier proyecto requiere de un buen diseño de sus clases para de esta forma realizarlo con la mejor calidad posible y así el cliente quede satisfecho. En la metodología XP el diseño de las clases se realiza a través de las tarjetas CRC, para de esta forma ayudar al refinamiento de las clases, estructurando el conjunto de las mismas.

2.8.1 Tarjetas CRC

Las tarjetas CRC son una técnica simple e informal pero efectiva que ha sido propuesta tanto para el modelado conceptual como para el diseño detallado de sistemas. Una tarjeta CRC establece 3 dimensiones las cuales identifican el rol de un objeto en análisis y/o diseño: nombre de la clase, responsabilidades y colaboraciones. (21)

Estas tarjetas sirven para diseñar el sistema en conjunto con todo el equipo. Permiten reducir el modo de pensar y apreciar la tecnología de objetos, el uso de estos diagramas puede aplicarse siempre, no como un artefacto de la metodología, pero se pueden utilizar siempre y cuando influyan en el mejoramiento de la comunicación, no sea un peso su mantenimiento, no sean extensos y se enfoquen en la información importante. A continuación se muestran algunas de las tarjetas CRC elaboradas en el desarrollo de la aplicación.

Tabla 9.Tarjeta CRC 1

Tarjeta CRC	
Clase: Tabla.	
Responsabilidades	**Colaboraciones**
Insertar tabla Modificar tabla Eliminar tabla Visualizar tabla	

Tabla 10.Tarjeta CRC 2

Tarjeta CRC	
Clase: Atributo.	
Responsabilidades	**Colaboraciones**
Insertar atributo Modificar atributo Eliminar atributo Visualizar atributo	Tabla

Tabla 11.Tarjeta CRC 3

Tarjeta CRC	
Clase: Relación	
Responsabilidades	**Colaboraciones**
Establecer la relación	Tabla

2.8.2 Diagrama de clases

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

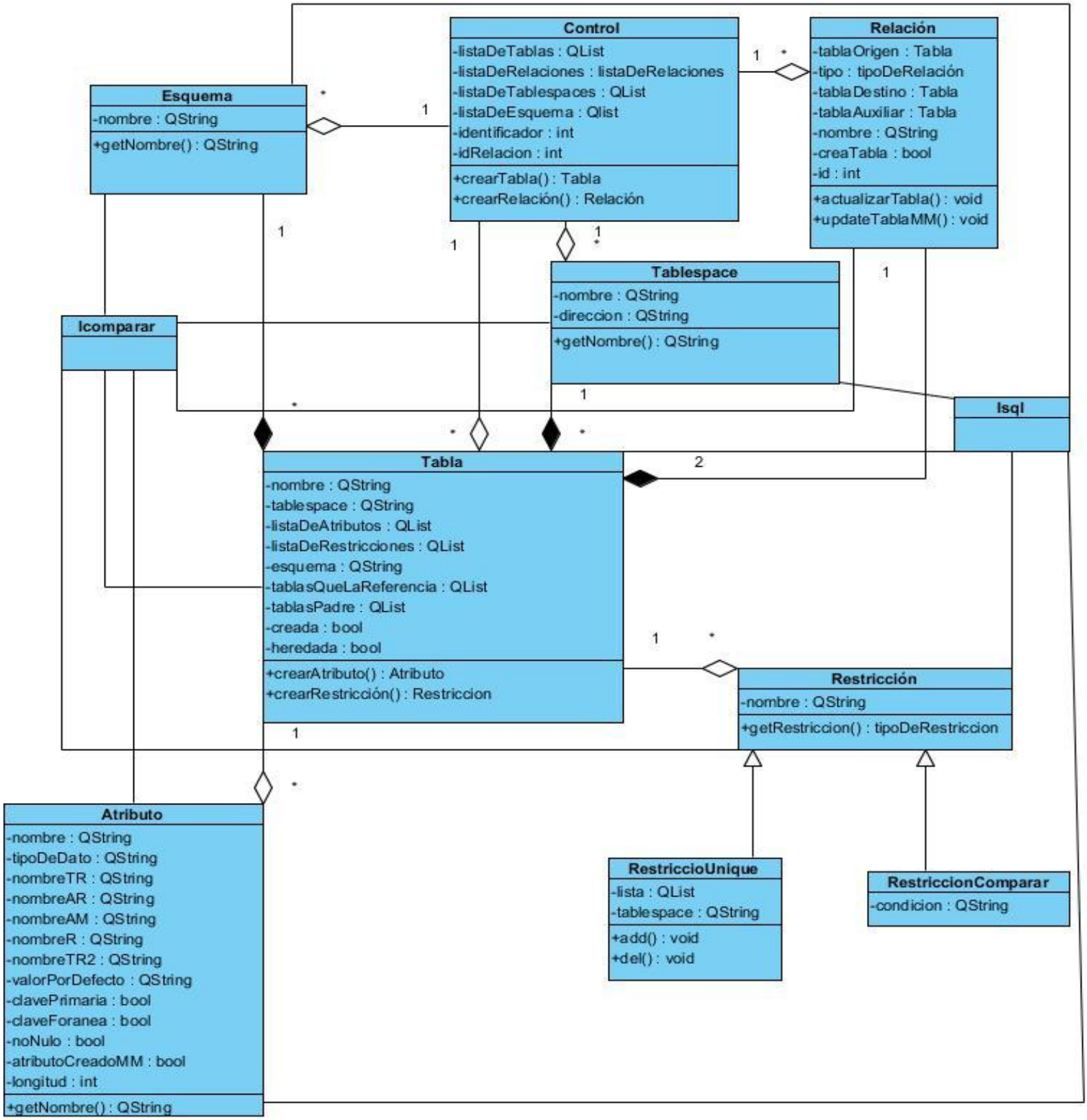


Figura 2. Diagrama de clases de la aplicación

2.9. Patrones de Arquitectura.

Actualmente se suele usar la arquitectura programación en capas la cual permite distribuir el trabajo de una aplicación por niveles. El estilo arquitectural en capas se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de la interacción con otras capas y las responsabilidad y funcionalidades que implementan. (22)

Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software .En este trabajo se hace uso del patrón programación en capas logrando que el sistema quede perfectamente organizado para de esta forma tener un orden lógico en la programación del mismo. A continuación se hace una breve explicación de cada una de las capas del sistema.

Capa de presentación(o interfaz del usuario): Esta capa contiene todas las clases visuales las cuales trae por defecto QT, además de ser con las cuales el usuario interactúa. La capa de presentación se comunica únicamente con la capa de negocio. Es una interfaz grafica que es amigable ante los ojos del usuario.

Capa de negocio: En esta capa se encuentra la lógica del negocio, ella es el puente entre la primera capa y la tercera, aquí es donde se reciben las peticiones del usuario y se envían las respuestas tras el proceso. En esta capa se establecen todas las reglas que deben cumplirse.

Capa de información: En esta capa se evidencian todas las clases que poseen alguna información necesaria, pero que ellas de por sí solas no funcionan. Estas clases contienen toda la información que la capa de negocio necesita para realizar las operaciones.

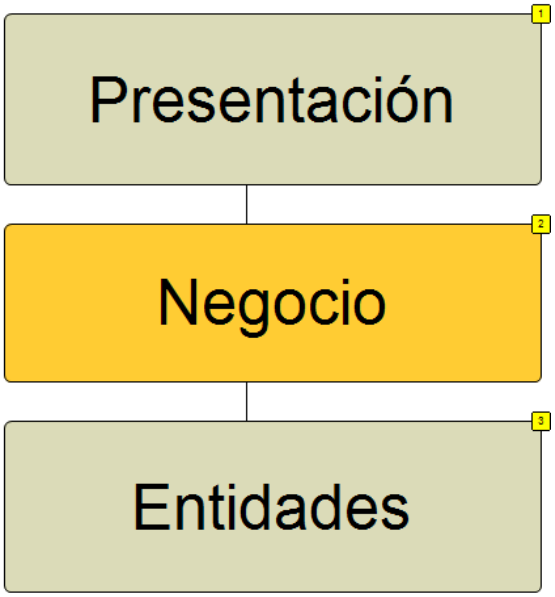


Figura 3. Modelo en capas

2.10. Patrones de Diseño.

Los patrones de diseño son los encargados de solucionar un problema en el diseño del algoritmo. Un patrón es una descripción del problema y la esencia de su solución, que se puede reutilizar en casos distintos.Cabe destacar que existen muchas familias de patrones de diseño, por lo que nos basaremos en los más usados, en función del objetivo que se ha planteado en este trabajo.

Los patrones GRASP (Patrones de Software para la asignación General de Responsabilidad) y los patrones GOF (Gang of Four).

Patrones GRASP: Acrónimo de General Responsability Assignment Software Patterns (Patrones de Software para la asignación General de Responsabilidad). Describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. (23)

Dentro de los patrones GRASP se encuentran; Experto, Creador, Bajo Acoplamiento, Alta Cohesión, Controlador.

Experto:

Nos indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce

toda la información necesaria para crearlo, es aquella clase cuya responsabilidad es llevar consigo toda la información y ser experta en el tema para no depender de ninguna otra clase.En este trabajo se evidencia este patrón en la clase Control, la cual tiene la responsabilidad de llevar en sí la información necesaria de cómo crear las tablas.

Figura 4. Patrón experto

Creador:

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos.Un ejemplo de este patrón en este trabajo se muestra en la clase Control debido a que es la encargada de crear los objetos de tipo tabla.

Figura 5. Patrón creador

Controlador:

El patrón controlador es el encargado de controlar un evento del sistema, es aquel que sirve para intermediar entre la interfaz y el algoritmo. En este trabajo se refleja este patrón en la clase Vtabla, la cual captura los datos de una tabla entrados por el usuario.

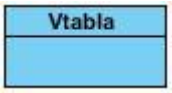


Figura 6. Patrón controlador

Alta cohesión

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. El alto nivel de cohesión, es presentado por las clases que tienen responsabilidades y colaboran con otras para llevar a cabo las tareas. Un ejemplo de este patrón se evidencia en la clase control, la cual para poder realizar la responsabilidad de generar el script necesita de otras clases que contienen métodos que son de vital importancia para poder realizar la tarea.

Figura 7: Patrón alta cohesión

GOF: Es la abreviación del grupo Gang of Four, compuesto por Erich Gamma, Richard Helm, Ralph Jhonson y John Vlisides, quienes en su publicación “Design Patterns” (década de los 90s), describen 23 patrones de diseño comúnmente utilizados y de gran aplicabilidad en problemas de diseño usando modelamiento UML. Estos patrones se agrupan en las siguientes categorías: Creacionales, estructurales y de comportamiento. (24)

Creacionales

Son los encargados de crear instancias de objetos, para de esta forma abstraer el proceso de instanciación. Dentro de los creacionales en este trabajo se hizo uso de los siguientes patrones:

Método fabricación (Factory method): Define una interfaz creando un objeto, para dejar a diferentes subclases decidir sus instancias. El Método De la fábrica deja a una clase la creación de ejemplares o copias a subclases. Este patrón se evidencia en la clase SimpleRestricciónFactoría donde crea una restricción en dependencia del tipo de restricción que se le pase por parámetro.

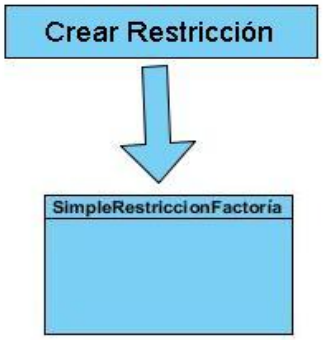


Figura 8. Patrón método fabricación

Estructurales

Definen como las clases y objetos pueden unirse para formar grandes estructuras, para de esta forma proporcionar nuevas funcionalidades. A continuación se mencionan los patrones que se ponen de manifiesto en la implementación:

Puente (Bridge): Desacopla una abstracción de su implementación de modo que los dos puedan variar por separado. Este patrón se evidencia a la hora de conectar el plugin con la herramienta de

administración HABD y se muestra en la clase Iplugin.

Comportamiento

Los patrones de comportamiento facilitan y definen la comunicación e iteración entre los objetos de un sistema y que estos estén lo menos mezclados posible.

Observador (Observe): Este patrón tiene la responsabilidad de que al cambiar el estado de un objeto todas las dependencias del mismo se actualicen. Este patrón QT lo trae por defecto, en este trabajo se emplea en todas las interfaces, ya que todas emiten señales. Utiliza el mecanismo de la interacción con las clases visuales mediante las señales y los Slot donde de forma asincrónica los objetos que están suscritos a esa señal se enteran del cambio de estado.

2.11 .Estándares de codificación

Extreme Programing (XP) promueve la programación basada en estándares, de manera que sea fácilmente entendible por todo el equipo, y que facilite la recodificación. Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código.

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

usted mismo) que necesitan entender que fue lo que se hizo en el fragmento de código. Los comentarios deben ser escritos correctamente y claros. Generalmente deben usarse comentarios de una sola línea. Reserve los comentarios de bloques para la documentación formal o para comentar porciones de código.

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.
Ejemplo: loadedInstancesLst, installedInstances.

Declaración de funciones

No debe haber espacio entre el nombre de la función y el paréntesis izquierdo, ni entre este y la lista de parámetros. Debe haber un espacio entre el paréntesis derecho y la llave de comienzo del cuerpo de la función. Las sentencias del cuerpo deben estar en la línea siguiente. La llave que cierra debe estar alineada con la línea de declaración de la función. Los nombres de las funciones se rigen por las mismas características que el de las variables.

Identificadores

Los identificadores pueden estar formados por cualesquiera de las 26 letras minúsculas o mayúsculas (A- Z, a - z), los 10 dígitos (0 - 9) y el caracter 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. Tenga en cuenta que una sentencia de asignación puede resultar en la asignación de una función o de un objeto como literal y en todos los casos como sentencia de asignación debe estar finalizada con un punto y coma.

Sentencias Compuestas

Las sentencias compuestas son aquellas sentencias que contienen una lista de sentencias encerradas

entre llaves:

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

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.

Sentencia if

La sentencia if debe ser escrita de esta manera:

```
if (condition)
{
statements
}
if (condition)
{
statements
} else
{
statements
}
if (condition)
 {
```

```
statements
}
else if (condition)
{
statements
}
else
{
statements
}
```

Estructuras repetitivas

Las estructuras repetitivas deben ser escritas de esta manera:

```
for (initialization; condition; update)
{
statements
}

while (condition)
{
statements
}

forecah (valor1, valor2)
{
statements
}
```

Sentencia switch

La sentencia switch debe ser escrita de la siguiente forma:

```
switch (expression)
{
```

case expression:
statements
case expression:
statements
default:
statements
}

Espacios en blanco

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

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

Declaraciones de clases

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

2.12 Interfaces principales de la aplicación

A continuación se muestran diferentes interfaces de la aplicación para que sean visualizadas las funciones que puede realizar el diseñador. En la figura 9 se evidencia la interfaz principal de la aplicación, la cual está compuesta por los botones, estos dan la posibilidad de guardar un modelo, crear una tabla, eliminar una tabla, cargar el modelo, crear tablespce y esquemas, además en esta interfaz se muestran los diferentes tipos de relaciones que el usuario puede usar para relacionar las tablas .En la figura 10 se evidencia un modelo realizado con tablas, atributos y relaciones. En la figura 11 se puede ver como se crea un esquema y como se visualiza en la ventana que se encuentra en la

parte derecha superior de la aplicación. En la figura 12 se muestra la creación de un esquema y como se visualiza en la ventana derecha inferior del diseñador.

En la figura 9 se muestra la ventana principal del diseñador de bases de datos.

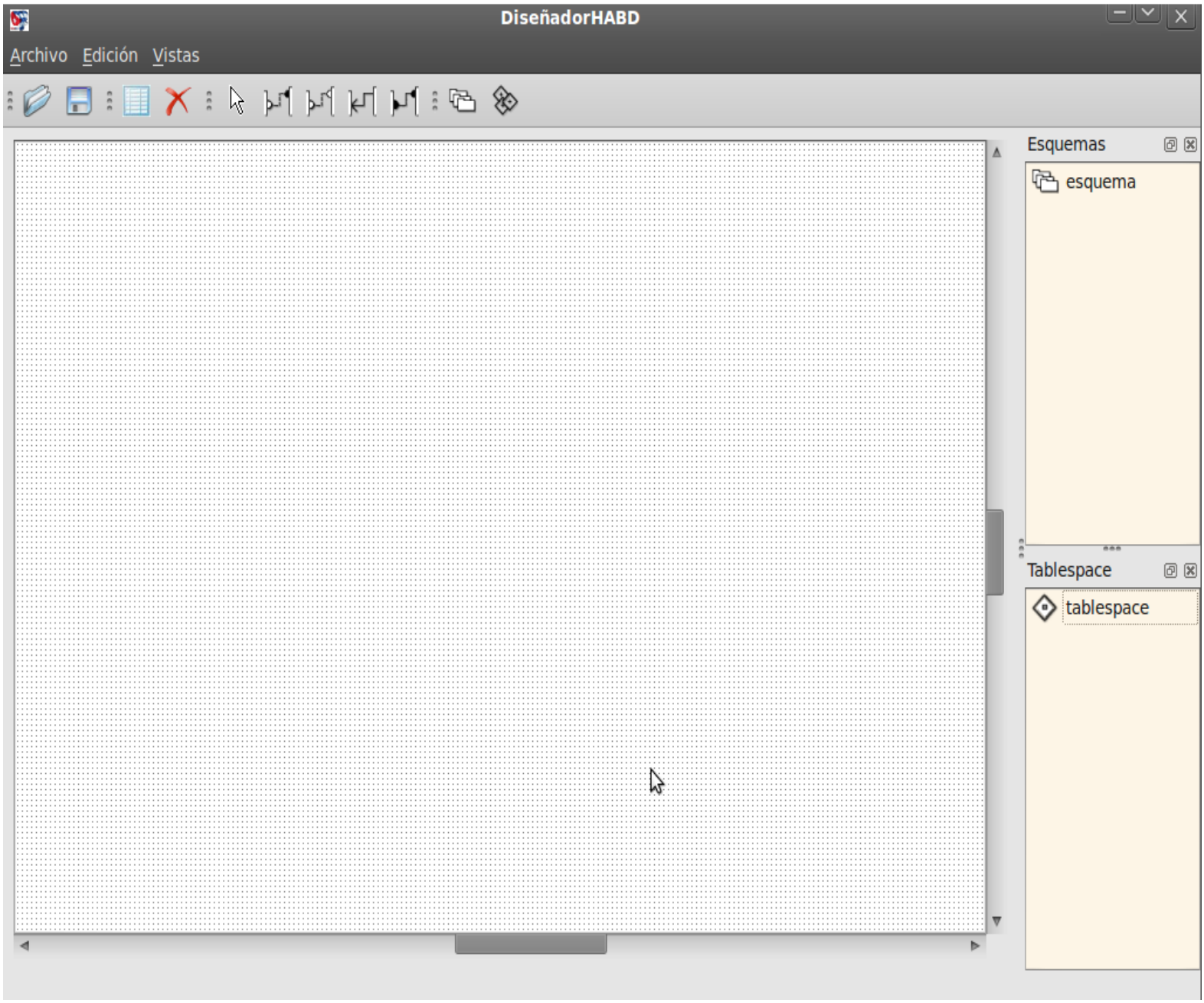


Figura 9. Portada principal del diseñador

En la figura 10 se evidencia la creación de un modelo con todas sus tablas, atributos y las relaciones entre cada tabla.

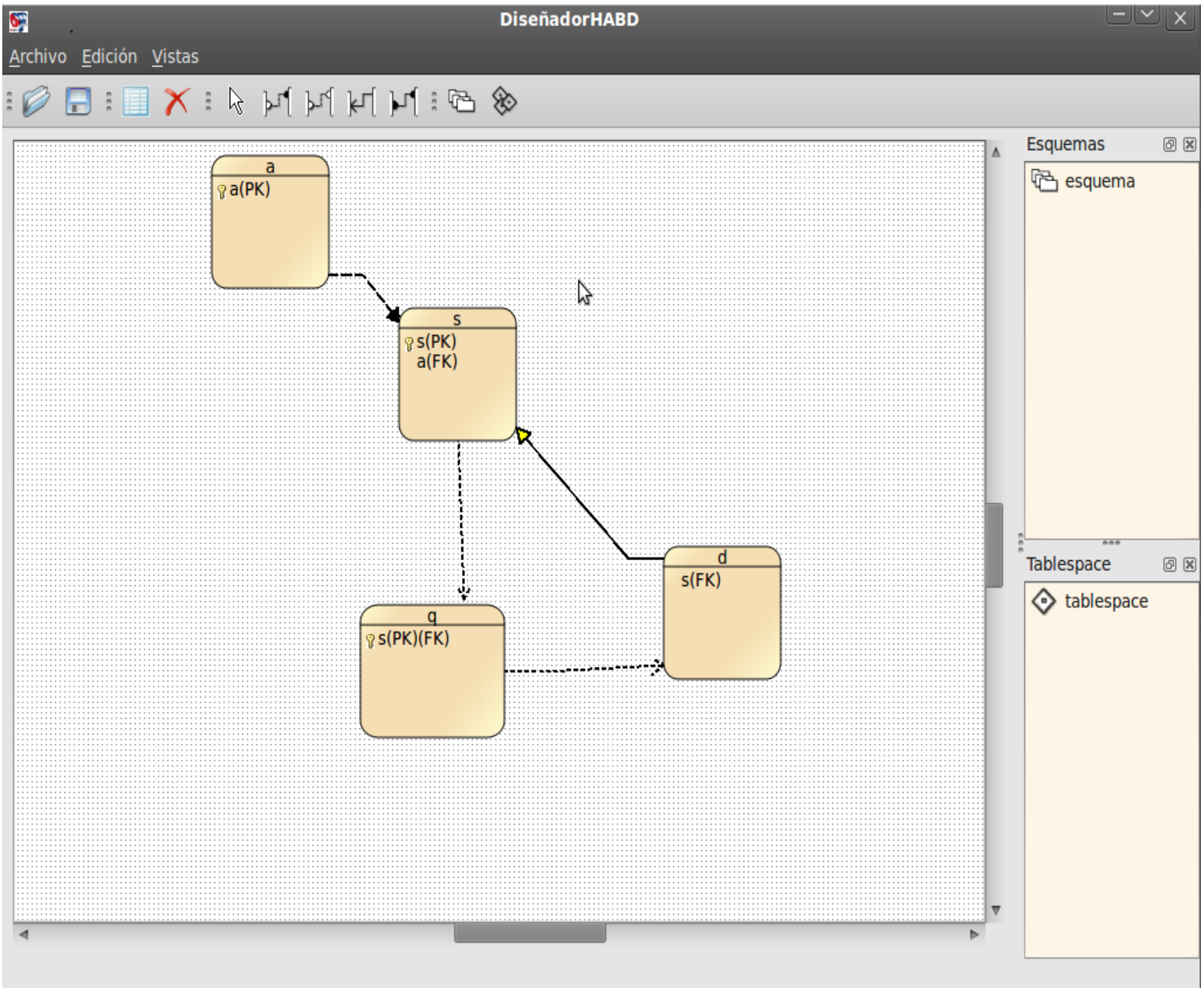


Figura 10. Modelo realizado en el diseñador

En la figura 11 se evidencia la creación de un esquema y como se visualiza en la ventana de Esquemas.

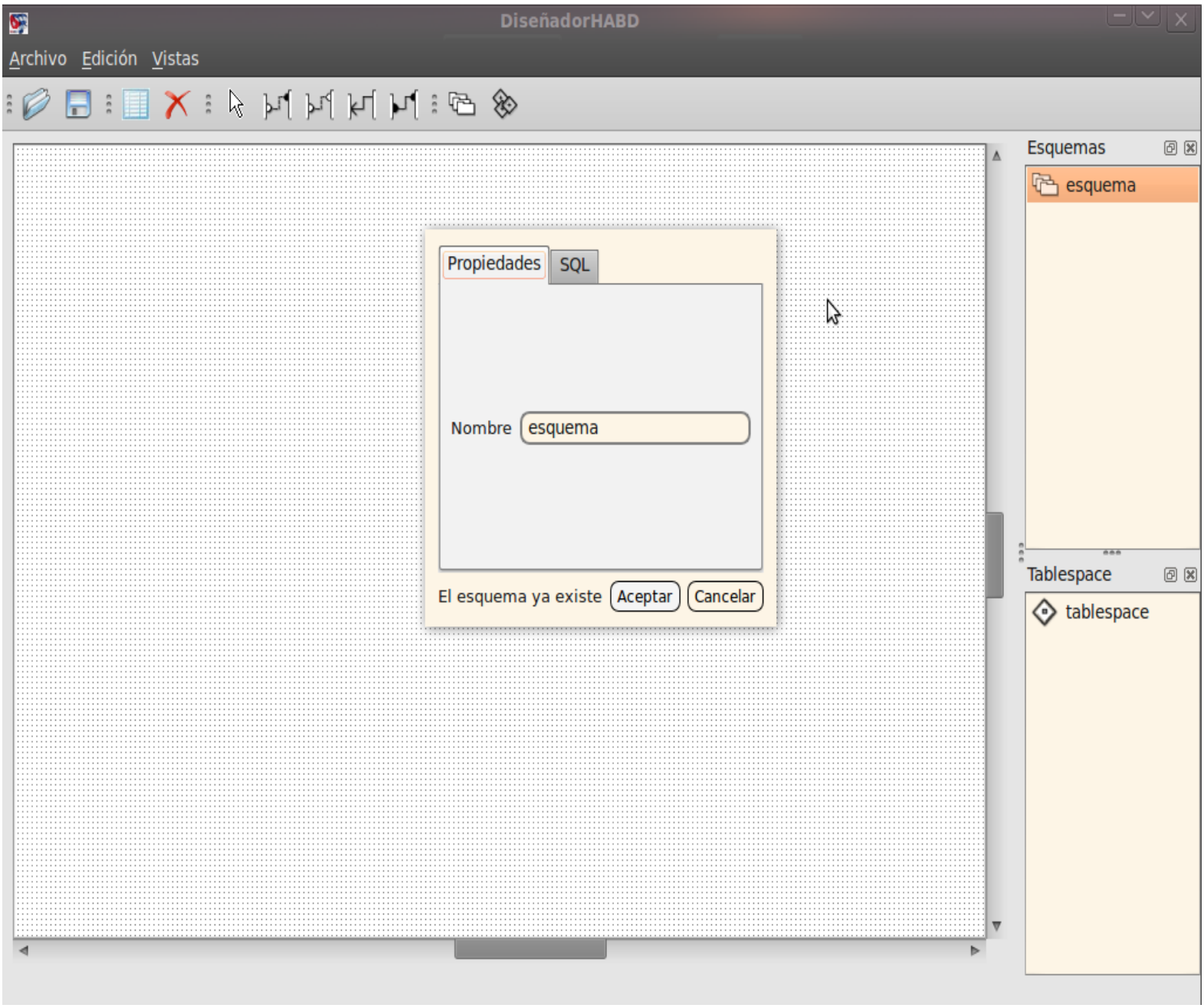


Figura 11. Esquema creado en el diseñador

En la figura 12 se evidencia la creación de un tablespace y como se visualiza en la ventana de tablespace.

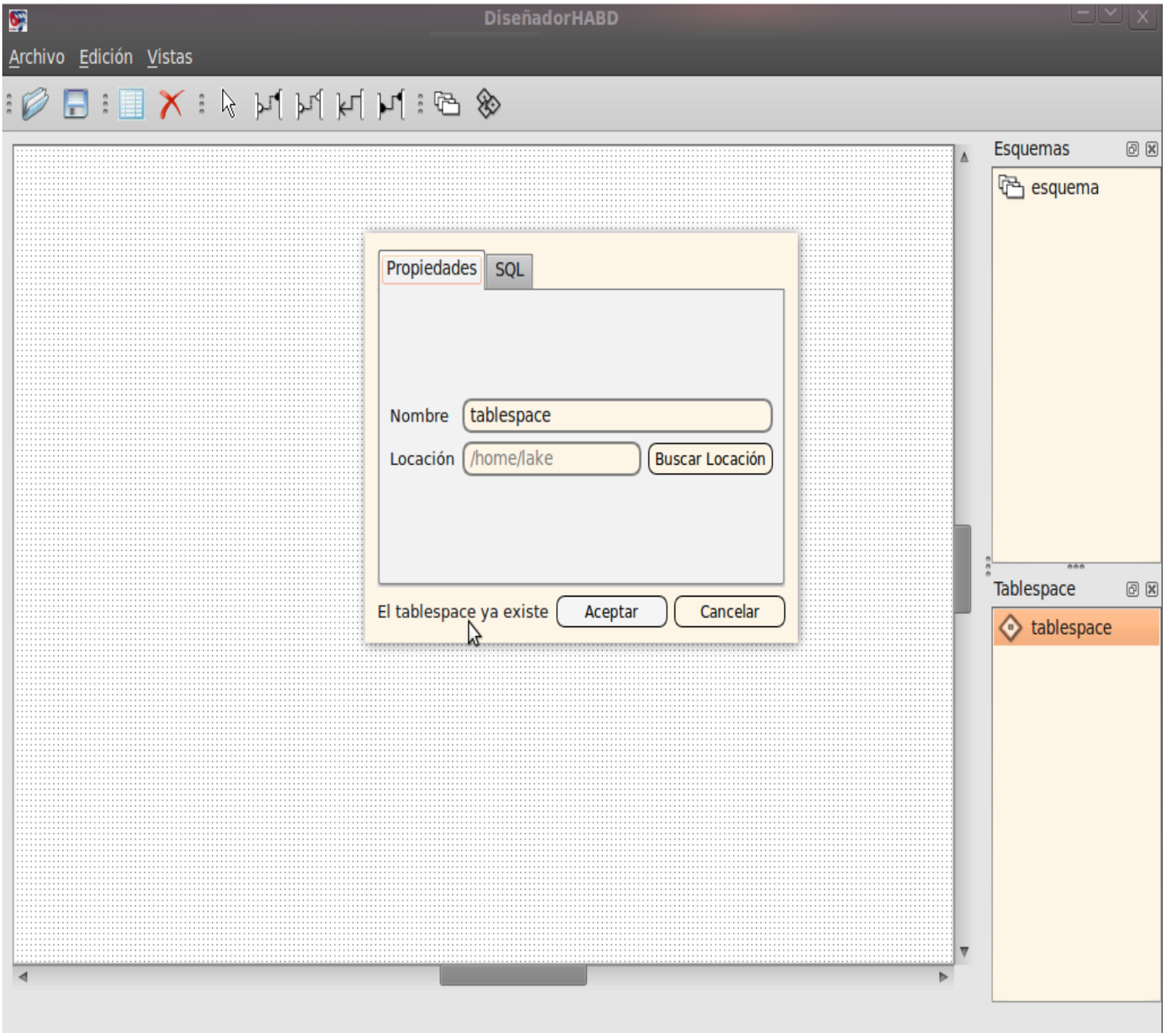


Figura 12. Tablespace creado en el diseñador

Conclusiones del capítulo

En el capítulo han sido analizados todos los elementos que describen las características y diseño del sistema. Se elaboró una propuesta con las perspectivas de solucionar el problema identificado, para eso fueron desarrolladas las fases de exploración y diseño propuestas por la metodología utilizada en las que se definieron un total de 24 tareas de la ingeniería agrupadas en 9 historias de usuarios. Se generó además el plan de iteraciones para de esta forma especificar en la iteración que se desarrollará cada historia de usuario .Se describen un total de 7 tarjetas CRC como parte del diseño en la metodológica XP. Se muestran los patrones de arquitecturas y de diseño que se emplearon.

Capítulo 3: Validación y prueba

Introducción

En XP se definen un grupo de normas para la validación de los productos, logrando que los mismos puedan ser probados para la verificación de su correcto funcionamiento. En este capítulo se analizan las estrategias de pruebas que define XP y la técnica que se utilizará para diseñar los casos de pruebas que guiarán la validación de la aplicación, además de mostrarse los resultados arrojados por estas pruebas.

3.1 Pruebas

La fase de pruebas es una de las fases fundamentales del desarrollo de una aplicación. El objetivo de cada una de las pruebas no es el de prevenir errores sino de detectarlo basándose en técnicas y estrategias empleadas en cada una de las pruebas. Hay multitud de conceptos (y palabras clave) asociadas a las pruebas, clasificarlas es difícil, pues no son mutuamente disjuntas, sino muy entrelazadas .Dentro de las estrategias de pruebas existentes podemos mencionar las pruebas de unidades, las pruebas de integración, las pruebas de sistema, las pruebas de aceptación entre muchas más.

3.1.1 Estrategias de pruebas

Uno de los pilares fundamentales en XP es el uso de las pruebas, ellas son la manera de comprobar el código que vayamos implementando. Las pruebas de software consisten en la verificación dinámica del comportamiento de un programa en un conjunto finito de casos de prueba, de forma adecuada selecciona del dominio ejecuciones generalmente infinito, contra el comportamiento esperado. (25)

Como ya se ha mencionado anteriormente existen disímiles estrategias de pruebas, el estudio de este trabajo se ha centrado específicamente en la investigación de las pruebas correspondientes a la metodología de desarrollo de software empleada en el presente trabajo de diploma .XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final. (26)

Pruebas unitarias.

Una prueba unitaria es la verificación de un módulo (unidad de código) determinado dentro de un sistema. Son llevadas a cabo por los programadores encargados de cada módulo. Aseguran que un

determinado módulo cumpla con un comportamiento esperado en forma aislada antes de ser integrado al sistema. (27)

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

Pruebas de aceptación

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

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

La prueba alfa.

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

La prueba beta.

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

Con el estudio realizado este trabajó decide llevar a cabo la estrategia de pruebas de aceptación debido a que con esta estrategia la presentación de los resultados es importante, en cambio para las unitarias no tiene mucho sentido ya que siempre se requiere un total de efectividad en los módulos más críticos. Esta estrategia en la metodología XP tienen como propósito indicarle al equipo cuando las funcionalidades de una iteración han sido completadas exitosamente. Significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente. Es casi imposible obtener una implementación libre de errores, es por ello que se debe tener bien definido un criterio de aprobación para saber cuando el software está listo para ser liberado.

3.1.2 Técnica de prueba seleccionada

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

Pruebas estructurales o Caja blanca

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

En las pruebas estructurales las pruebas se seleccionan en función del conocimiento que se tiene de la implementación del módulo. Se suelen aplicar a módulos pequeños. El probador analiza el código y deduce cuántos y qué conjuntos de valores de entrada han de probarse para que al menos se ejecute

una vez cada sentencia del código. Se pueden refinar los casos de prueba que se identifican con pruebas de caja negra. (29)

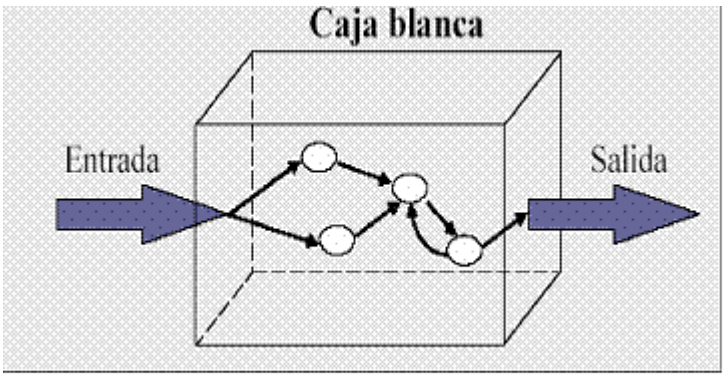


Figura 13. Técnica caja blanca

Pruebas de funcionalidad o Caja negra

También conocidas como Pruebas de Comportamiento, estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente se ve como una "Caja Negra" cuyo comportamiento sólo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas. (30)

Las pruebas de caja negra pretenden demostrar que las funciones del software son operativas y que funcionan correctamente aceptando de forma adecuada la entrada de datos y produciendo una salida correcta. Este tipo de prueba nos permite demostrarle al cliente que la aplicación puede satisfacer las necesidades del mismo.

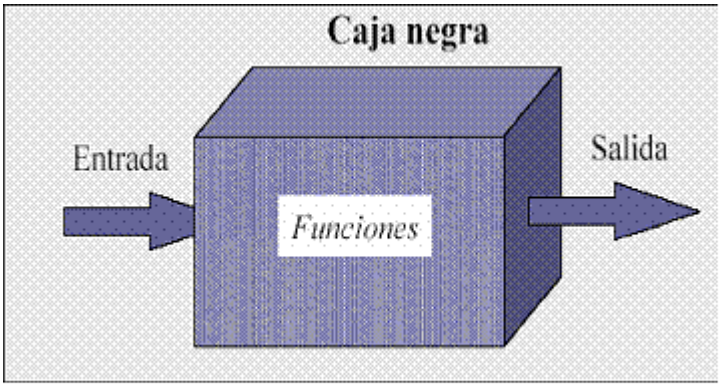


Figura 14. Técnica caja negra

Se decide aplicar la técnica de caja negra debido a que es más importante presentarle al cliente que las funcionalidades de la aplicación estén correctamente funcionando y así dejarlo satisfecho y conforme con el producto.

3.1.3 Casos de pruebas basados en historias de usuarios

En la actividad Diseño de los Casos de Prueba, usando técnicas de caja negra se desarrollan los casos de prueba. Esta actividad incluye diseñar las pruebas e identificar los datos de prueba. Para cada funcionalidad a probar, se crea una matriz que muestra su correspondencia con los casos de prueba, conociéndose de esta forma que ítem cubren los casos de pruebas. (31)

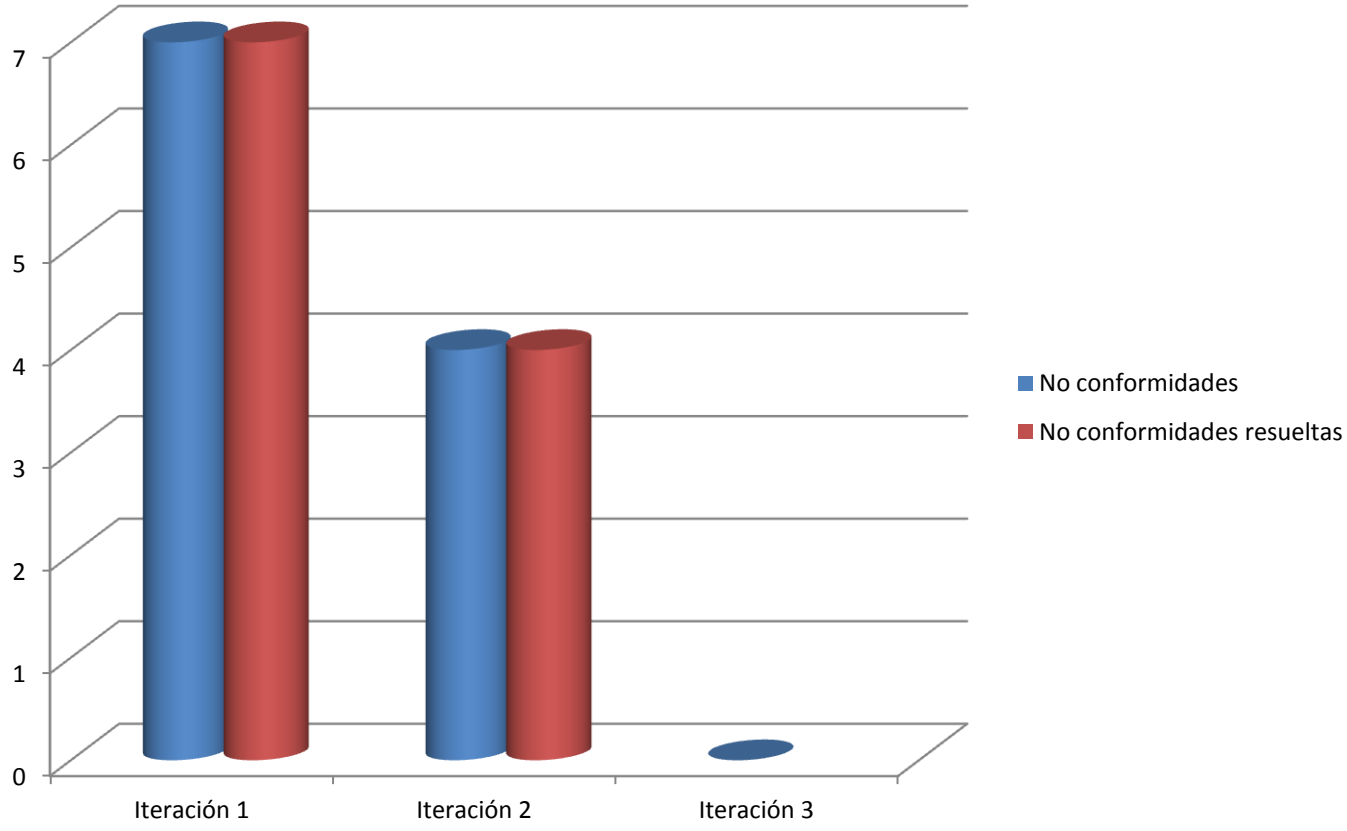
A través de los casos de pruebas podemos probar si la aplicación realmente funciona, en ellos se describen los diferentes escenarios y se indica la respuesta correcta que el sistema debe mostrar en cada escenario. Se debe dejar bien claro en cada caso de prueba el flujo central de la aplicación que no es más que los pasos a seguir para trabajar en la aplicación a la hora de probar cada historia de usuario. A continuación se muestra un ejemplo de un caso de prueba que fue utilizado para la realización de las pruebas.

Tabla 12: Caso de prueba aplicado al sistema

Escenarios	Descripción	Variable1	Respuesta del sistema	Flujo central
EC 1.1 Insertar esquema con datos correctos.	En este escenario se crea el esquema introduciendo datos correctos.	V(Esquema1)	El esquema ha sido creado correctamente.	1-El usuario accede al botón Esquemas. 2-El usuario. especifica el nombre del esquema.
EC 1.n Insertar esquemas con datos incorrectos.	En este escenario se crea el esquema introduciéndole datos incorrectos.	I(1456)	La aplicación muestra el mensaje Nombre Incorrecto.	
EC 1.n Insertar el esquema con campos vacios.	En este escenario se crea el esquema dejando de especificar el nombre del mismo.	( )	La aplicación desactiva el botón Aceptar y muestra el mensaje Entre el nombre.	

3.1.4 Presentación de los resultados de las pruebas funcionales

Las pruebas han sido aplicadas a las nueve historias de usuarios permitiendo detectar varios errores. Fueron encontradas en una primera iteración siete no conformidades las cuales fueron resueltas en diez días. Para una segunda iteración se encontraron cuatro no conformidades solucionadas completamente en un período de tres días. Finalmente para una tercera iteración no se encontraron no conformidades. A continuación se muestra una gráfica donde se recogen la cantidad de no conformidades encontradas y resueltas en cada iteración.



Gráfica 1. Resultados de las pruebas aplicadas

Conclusiones del capítulo

Las pruebas de software, permiten la verificación de la calidad de un producto. Son utilizadas para identificar posibles fallos de una aplicación básicamente es una fase en el desarrollo de software consistente en probar las aplicaciones construidas. En este capítulo se realizó un estudio de las estrategias de pruebas de la metodología XP. Fueron probadas todas las funcionalidades y fueron mostradas cada una de las no conformidades encontradas en la aplicación. Se logró presentar los resultados arrojados en cada iteración logrando obtener una aplicación que responde a todos los requisitos funcionales identificados, la cual cuenta con una alta calidad.

Conclusiones generales

- Se concluye que las herramientas estudiadas, no cumplen con las características que se necesitan para integrarse a la herramienta de bases de datos HABD, pero si se tomaron funcionalidades de cada una de ellas permitiendo realizar un Plugin capaz de diseñar bases de datos, integrándose satisfactoriamente a la herramienta HABD.
- Mediante el modelo de diseño se determinaron 27 clases del Plugin.
- Se implementaron los 26 requisitos funcionales identificados para el Plugin, se integró a la herramienta HABD permitiéndole a la misma, diseñar bases de datos en PostgreSQL.
- Se realizaron las pruebas a las nueve historias de usuario implementadas verificando el correcto funcionamiento de la aplicación partiendo del diseño propuesto.

Recomendaciones

- Implementar una funcionalidad que le permita al usuario acomodar a su gusto las líneas de las relaciones entre las tablas.
- Implementar a través del patrón memento la funcionalidad que permita al usuario deshacer los cambios hechos en la aplicación.
- Implementar la funcionalidad que permita salvar el modelo realizado en la extensión jpg.
- Se recomienda que el plugin pueda ser sacado para la web.

Referencias bibliográficas

(1. **Teodosio, Ivette Rosa.** *Exploración y deseño de la herramienta de administración de bases de datos para PostgreSQL.* La habana : s.n., 2011.

2. *Propuesta del PostgresSQL Emmpresarial Cubano.* **Vazquez Ortíz, Yudisney, y otros.** La habana : s.n., 2010.

3. **PGDesigner.** PGDesigner. [En línea] [Citado el: 5 de febrero de 2012.] http://pgdesigner.sourceforge .net/en/index.html.

4. Softonic. [En línea] Interschare, Julio de 1997. [Citado el: 6 de Feberero de 2012.] DBDesigner.softonic.com.

5. Zona Linux. [En línea] Theme Junkie. [Citado el: 7 de Marzo de 2012.]

6. **Claudi Avila Arteaga, Marina Ramirez Hernandez.** *Análisis y Diseño del sistema de Control de Servicio Social de la UAEH.* 2005.

7. **Danysoft, Equipo.** *Modelado de bases de datos.* 2006.

8. **Carrrillo Pérez, Isaías, Pérez González, Rodrigo y Rodríguez Martín, Aureliano David.** *Metodología de desarrollo de software.* 2008.

9. **Joskowics, José.** *Reglas y Practicas en eXtreme Programing.* 2008.

10. **Hermes Alexy Marañon Rodríguez, Pedro David Duharte Montero.** *Planificación y Diseño del Portal para la Comunidad Técnica Cubana de PostgreSQL.* La Habana : s.n., 2010.

11. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* Mexico : s.n., 1999. 970-17-0261-1.

12. Lenguajes de Programación. [En línea] 2009. [Citado el: 22 de Marzo de 2012.] http://www.lenguajes-de-programación.com/lenguajes-de-programación.shtml.

13. **Martínez, Juan J.** *Programación en C++.* 2003.

14. **Jiang, Liangzhong.** *Advances in intelligent and soft computing .* Melbourne , Autralia : s.n., 2011. 978-3-642-25193-1.

15. **Gutíerrez, Javier J.** *¿Que es un framework web?*

16. **Garrido, Salvador Alemany.** *Introducción a QT .Programación gráfica en c++ con QT 4.* 2009.

17. **Collins, Ben, y otros.** *Version Control with Subversion.* 2004.

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

19. **Carlos Eduardo Fuentes Paredes, Paúl Hernán Vega Silva.** *Sistema web de servicios musicales para la corporación musicológica ecuatoriana.* Quito : s.n., 2009.

20. **Lorena Guerrero, Gaby, y otros.** *Desarrollo del software.Definición general del proceso.* 2011.

21. **Sandra Casas, Hector Reinaga.** *Identificación y Modelado de Aspectos Tempranos dirigidos por Tarjetas de Responsabilidades y Colaboraciones.* 20011.

22. **de la Torre Llorente, Ceasr, y otros.** *Guía de Arquitectura N-Capas orientada al dominio con .NET.* 2010. 978-84-936696-8.

23. **Hurtado Bustamante, Diana Paola, y otros.** *Patrones Grasp.* Universidad del Valle : s.n.

24. **Hernandez, Pedro Veloso.** *Uso de Ptrones de Arquitectura.* 2007.

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

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

27. **Malforá, Dayvis, y otros.** *Testing en eXtreme Programing.* 2006.

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

29. **José M .Drake, Patricia López.** *Verificación y Validación .* 2009.

30. **Juristo, Natalia, M.Moreno, Ana y Vegas, Sira.** *Técnicas de evaluación de software.* 2006.

31. **Pérez, Beatriz.** *ProTest-Proceso de Testing Funcional.*

Bibliografía

1. **Teodosio, Ivette Rosa.** *Exploración y deseño de la herramienta de administración de bases de datos para PostgreSQL.* La habana : s.n., 2011.

2. *Propuesta del PostgresSQL Emmpresarial Cubano.* **Vazquez Ortíz, Yudisney, y otros.** La habana : s.n., 2010.

3. **PGDesigner.** PGDesigner. [En línea] [Citado el: 5 de febrero de 2012.] http://pgdesigner.sourceforge .net/en/index.html.

4. Softonic. [En línea] Interschare, Julio de 1997. [Citado el: 6 de Feberero de 2012.] DBDesigner.softonic.com.

5. Zona Linux. [En línea] Theme Junkie. [Citado el: 7 de Marzo de 2012.]

6. **Claudi Avila Arteaga, Marina Ramirez Hernandez.** *Análisis y Diseño del sistema de Control de Servicio Social de la UAEH.* 2005.

7. **Danysoft, Equipo.** *Modelado de bases de datos.* 2006.

8. **Carrrillo Pérez, Isaías, Pérez González, Rodrigo y Rodríguez Martín, Aureliano David.** *Metodología de desarrollo de software.* 2008.

9. **Joskowics, José.** *Reglas y Practicas en eXtreme Programing.* 2008.

10. **Hermes Alexy Marañon Rodríguez, Pedro David Duharte Montero.** *Planificación y Diseño del Portal para la Comunidad Técnica Cubana de PostgreSQL.* La Habana : s.n., 2010.

11. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* Mexico : s.n., 1999. 970-17-0261-1.

12. Lenguajes de Programación. [En línea] 2009. [Citado el: 22 de Marzo de 2012.] http://www.lenguajes-de-programación.com/lenguajes-de-programación.shtml.

13. **Martínez, Juan J.** *Programación en C++.* 2003.

14. **Jiang, Liangzhong.** *Advances in intelligent and soft computing .* Melbourne , Autralia : s.n., 2011. 978-3-642-25193-1.

15. **Gutíerrez, Javier J.** *¿Que es un framework web?*

16. **Garrido, Salvador Alemany.** *Introducción a QT .Programación gráfica en c++ con QT 4.* 2009.

17. **Collins, Ben, y otros.** *Version Control with Subversion.* 2004.

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

19. **Carlos Eduardo Fuentes Paredes, Paúl Hernán Vega Silva.** *Sistema web de servicios musicales para la corporación musicológica ecuatoriana.* Quito : s.n., 2009.

20. **Lorena Guerrero, Gaby, y otros.** *Desarrollo del software.Definición general del proceso.* 2011.

21. **Sandra Casas, Hector Reinaga.** *Identificación y Modelado de Aspectos Tempranos dirigidos por Tarjetas de Responsabilidades y Colaboraciones.* 20011.

22. **de la Torre Llorente, Ceasr, y otros.** *Guía de Arquitectura N-Capas orientada al dominio con .NET.* 2010. 978-84-936696-8.

23. **Hurtado Bustamante, Diana Paola, y otros.** *Patrones Grasp.* Universidad del Valle : s.n.

24. **Hernandez, Pedro Veloso.** *Uso de Ptrones de Arquitectura.* 2007.

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

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

27. **Malforá, Dayvis, y otros.** *Testing en eXtreme Programing.* 2006.

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

29. **José M .Drake, Patricia López.** *Verificación y Validación .* 2009.

30. **Juristo, Natalia, M.Moreno, Ana y Vegas, Sira.** *Técnicas de evaluación de software.* 2006.

31. **Pérez, Beatriz.** *ProTest-Proceso de Testing Funcional.*

32. **Ian Sommerville**.Ingeniería del software .séptima edición.2005. ISBN: 84-7829-074-5

33. **Ariel Erlijman, Alejandro Goyén Eros**. Problemas y soluciones en la implementación de Extreme Programing .Montevideo, 2001.

34. **Arthur Griffith** .The complete reference GCC. Estados Unidos, ISBN: 2002.0-07-222-405-3.

35. **Ian Joyner.** A Critique of C++ and Programming and Language Trends of the 1990s.1996.

36. **Beck, Kent**.Extreme Programming Explained Embrace Change. Boston .Pearson Education, 1999.

37. Presman, Roger S.Ingeniería del software un enfoque práctico .Sexta Edición .2005

38. **Schenone Marcelo Hernán**. Diseño de una Metodología ágil de Desarrollo de Software.2004

39. **Douglas Schmidt**. Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects, Volumen 2. 2000. ISBN: 0471606952

Glosario de términos

API: Es la abreviatura de Interfaz de programación de aplicaciones. Un API no es más que una serie de servicios o funciones que el Sistema Operativo ofrece al programador, es una "llave de acceso" a funciones que nos permiten hacer uso de un servicio web provisto por un tercero, dentro de una aplicación web propia, de manera segura.

Backend: Es la parte que procesa la entrada desde el Front End.

Diseño pragmático: Es un diseño en el cual la construcción se realiza a un nivel muy rápido. brindando las soluciones idóneas para solucionar un problema utilizando como materiales los que son propios de la región, o sea, los materiales que se encuentran a su alcance.

Front End: Es la parte del software que interactúa con el o los usuarios.

Host: Es el término usado en informática para referirse a las computadoras conectadas a una red, que proveen y utilizan servicios de ella.

Licencia BSD: La licencia BSD (Berkeley Software Distribution) es una licencia de software libre permisiva. Esta licencia tiene pocas restricciones y permite el uso del código fuente en software no libre.

Licencia GPL: La licencia GPL (General Public License) obliga a incluir el código fuente en su distribución, siendo imposible cambiar la licencia al programa, al distribuirlo tal cual o modificado.

Lenguaje procedural: Es un lenguaje donde el usuario especifica que datos se necesitan y como obtenerlos.

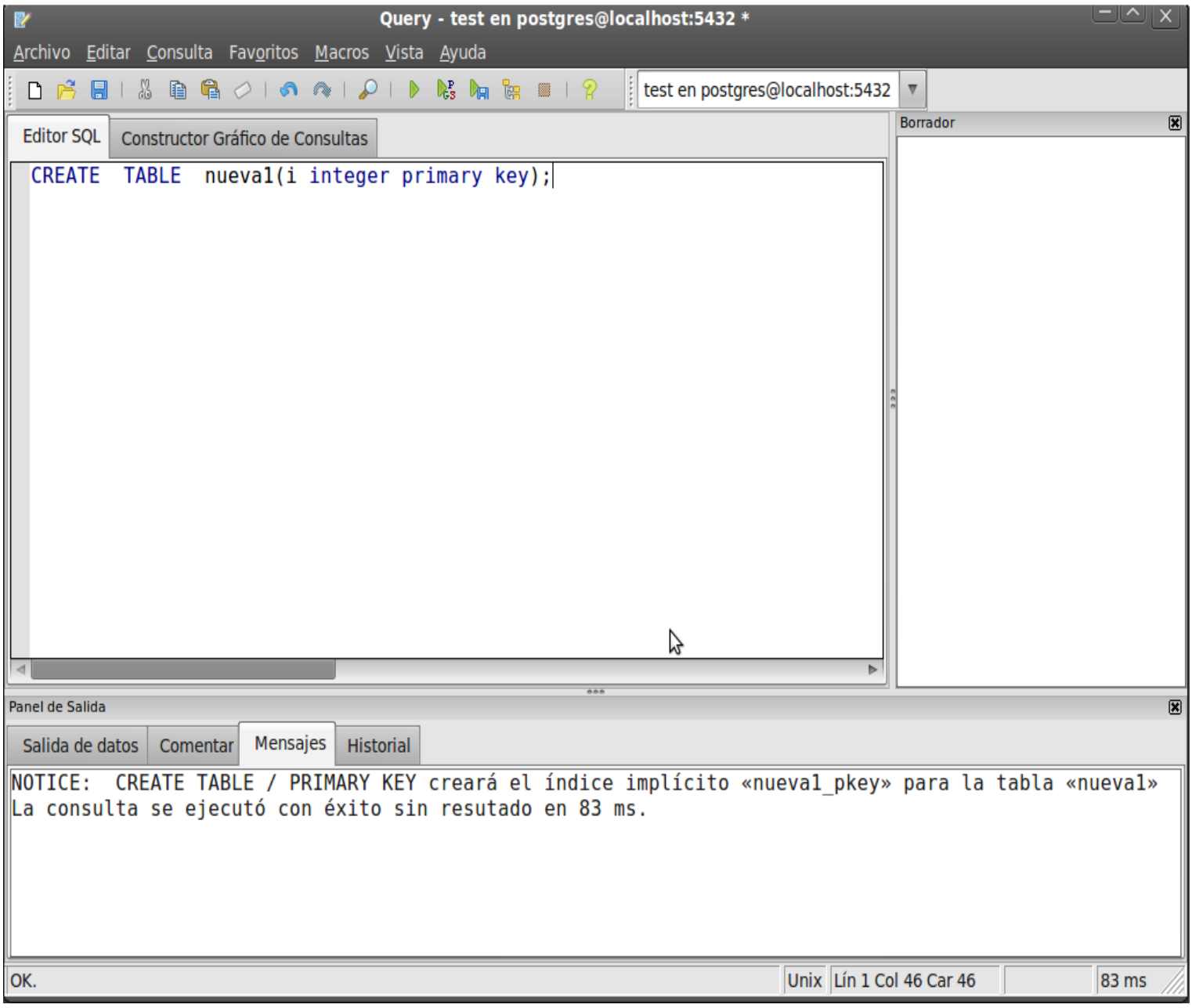
Lenguaje tipado: Es un lenguaje donde se determina de forma explícita el tipo de las variables y las operaciones que son válidas.

Script: Contiene las descripciones de las instrucciones utilizadas para crear una base de datos.

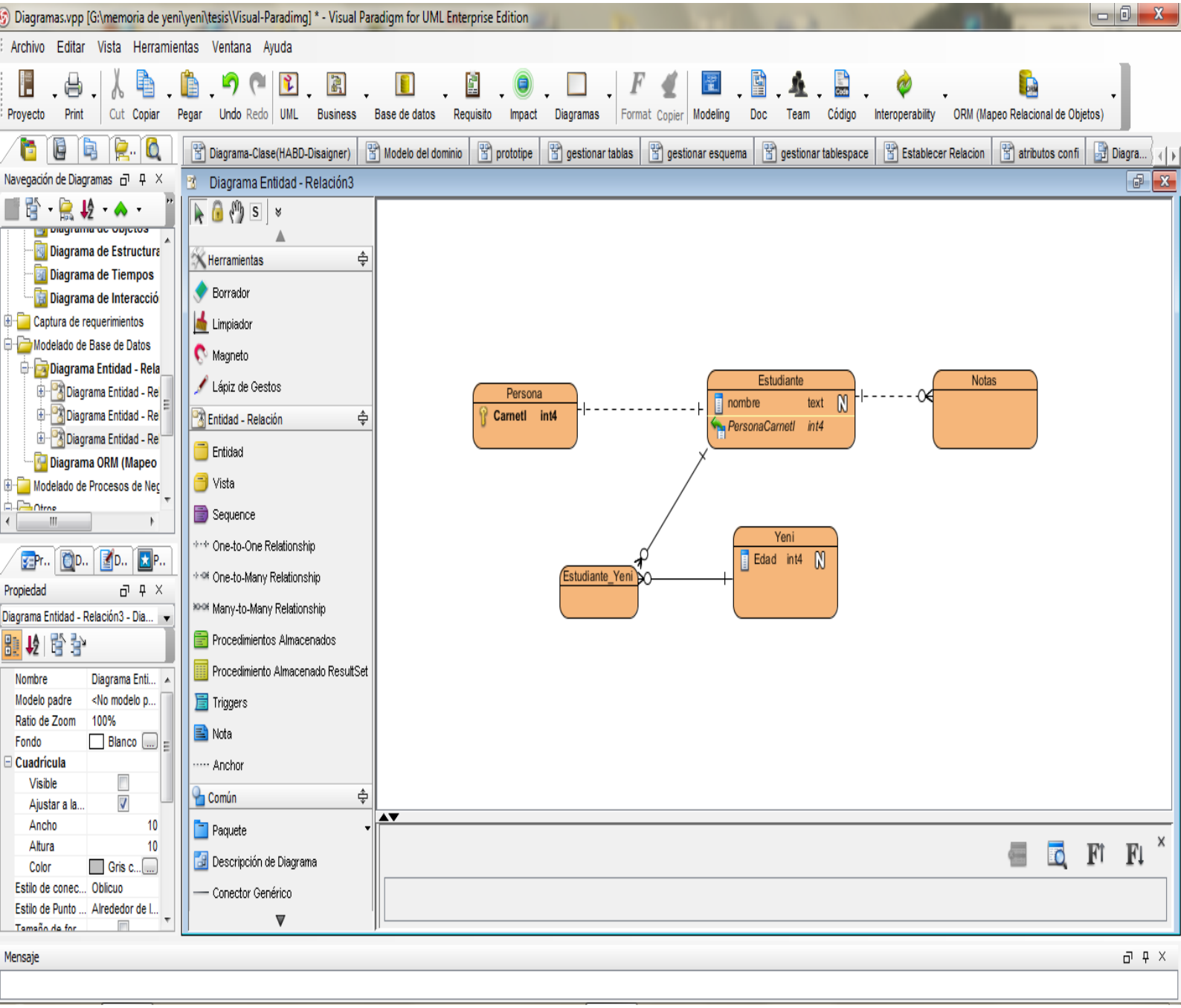
Vacuum: Es el proceso en el cual se realiza una reorganización de datos a nivel físico.

Anexos

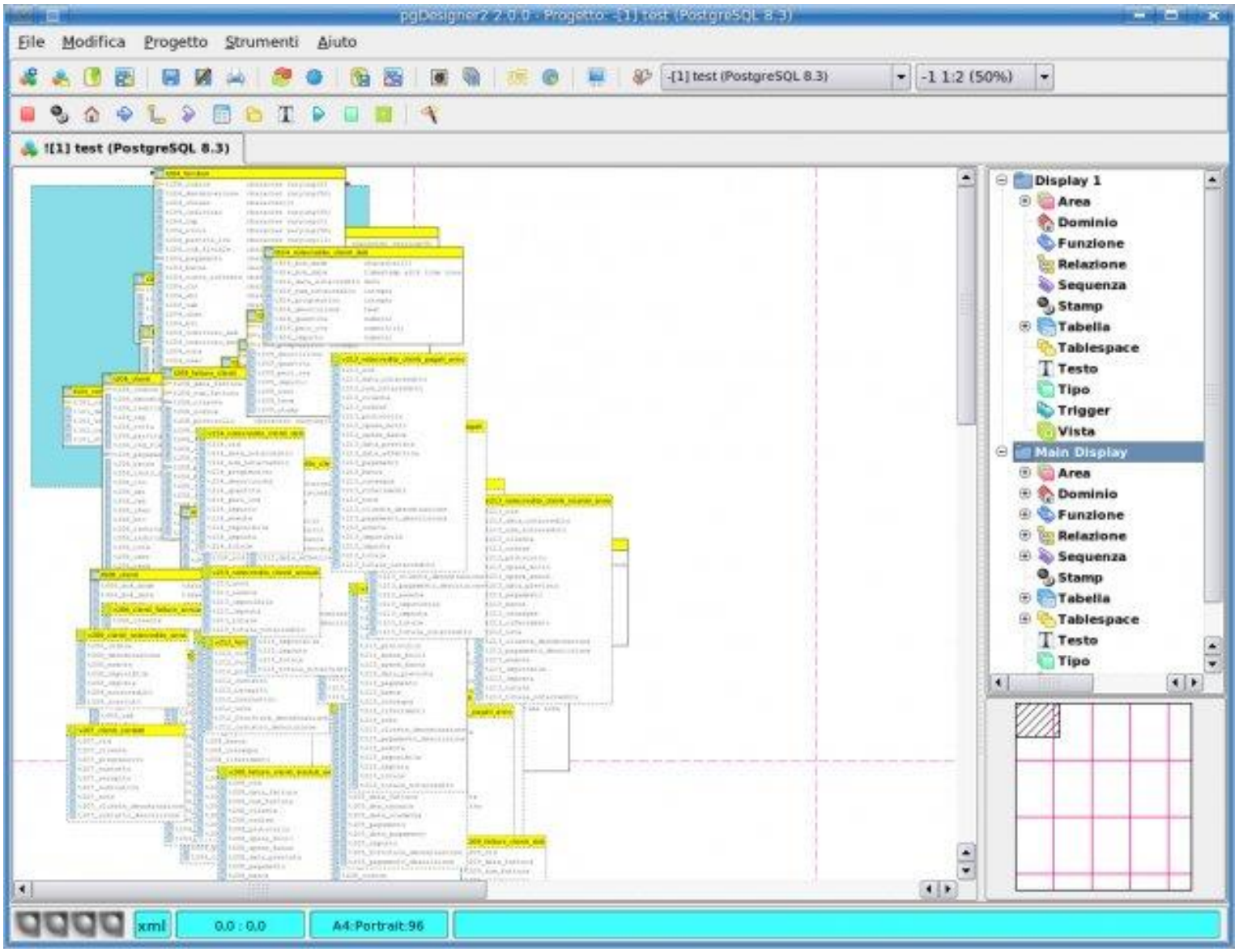
Anexo # 1: Tabla creada en PgAdmin



Anexo # 2: Modelo creado en el Visual Paradigm



Anexo # 3: Modelo creado en PGDesigner



Anexo # 4: Modelo creado en DBDesigner 4