GRFPOSTGRES – HERRAMIENTA GRÁFICA PARA EL MOTOR DE BASE DE DATOS POSTGRES

MARTHA NUBIA CARRILLO OBANDO
JAVIER ANDRES SANTACRUZ SALCEDO

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
PASTO - COLOMBIA
2005

GRFPOSTGRES – HERRAMIENTA GRÁFICA PARA EL MOTOR DE BASE DE DATOS POSTGRES

MARTHA NUBIA CARRILLO OBANDO
JAVIER ANDRES SANTACRUZ SALCEDO

TESIS DE GRADO PRESENTADA COMO REQUISITO PARCIAL PARA OPTAR POR EL TITULO DE INGENIERO DE SISTEMAS

DIRECTOR DEL PROYECTO
ING. NELSON ANTONIO JARAMILLO

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS
PASTO - COLOMBIA
2005

APROBACIÓN

El trabajo de tesis desarrollado bajo el nombre de "GRFPOSTGRES – HERRAMIENTA GRAFICA PARA EL MOTOR DE BASE DE DATOS POSTGRES", presentado como requisito parcial para optar el título de Ingeniero de Sistemas, fue APROBADO por su Jurado Calificador.

Atentamente,

_____

PHD(C) Ricardo Timarán Pereira
Jurado de Tesis

_____

Ing. de Sistemas Manuel Bolaños
Jurado de Tesis

San Juan de Pasto, Noviembre del 2005

A Dios y a mis Padres: Alberto Carrillo y Martha Obando, en especial a mi madre, por sus esfuerzos, que por encima de las dificultades garantizaron una buena educación y un mejor futuro para mí.

MARTHA NUBIA CARRILLO OBANDO

A todos aquellos que tienen el talento y el conocimiento pero además tienen los principios y nunca los olvidan, porque su camino ha sido, es y será siempre muy difícil, pero no se rinden ante todas las dificultades y siempre manifiestan esperanza.

A todos aquellos que desarrollan con todo el convencimiento una idea y que no encuentran apoyo debido a que los otros consideran que su esfuerzo inútil y se dedican a criticar y piensan que nunca tendrá éxito, porque para ellos es suficiente cumplir con su conciencia.

A todos aquellos que al final terminarán cediendo, porque son muchos más los otros, pero que siempre serán dueños de la verdad de cómo son las cosas y que tarde o temprano la revelaran a todos.

Que nunca se acaben.

JAVIER ANDRÉS SANTACRUZ SALCEDO

AGRADECIMIENTOS

La Autora Martha Nubia Carrillo Obando agradece a:

A Dios por las oportunidades tan grandes que me ha regalado en mi vida, en especial esta de poder concluir una etapa en mi carrera que espero no termine aquí.

A mi Padre Alberto Carrillo por todo lo que me dio durante el tiempo que estuvo conmigo; su apoyo, sus esfuerzos, su ejemplo y sobre todo su cariño incondicional.

Agradezco especialmente a mi madre Martha Obando, por su cariño, apoyo y esfuerzo sobre todo en este proyecto de mi vida, mi carrera, en la cual deposito en mí, la confianza de lograrlo.

A mi hermana Deisy Carrillo por ser como es, y siempre estar junto a mí.

Agradezco a Leandro Burbano, por estar a mi lado durante estos últimos años, convirtiéndose en un soporte, de comprensión y de amor.

A todos mis amigos durante mi carrera en especial a Lorena Estrella, Katty Medina, Fabio Salazar y Javier Santacruz, por los momentos vividos y por su amistad.

Al cuerpo Docente, a la Facultad de Ingeniería y la Universidad de Nariño en especial al Ingeniero Nelson Jaramillo, por los conocimientos brindados.

El Autor Javier Andrés Santacruz Salcedo agradece a:

A mis hermanos, Juan Carlos y Patricia, por su comprensión, a mi padre Gerardo por su apoyo a lo largo de mi vida y especialmente a mi madre Socorro, por su fuerza, su dedicación, su consejo y ayuda incondicional, a todos, por la familia que somos.

A los pocos amigos de esta etapa de mi vida, Katty Medina, Lorena Estrella, Fabio Salazar y Martha Carrillo, por siempre estar ahí y por su compañerismo.

A los profesores del programa de Ingeniería de Sistemas que tienen la vocación para enseñar, porque logran que los estudiantes desarrollen sus capacidades y que puedan creer.

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

GLOSARIO

BASE DE DATOS: Colección de datos clasificados y estructurados que son guardados en uno o varios ficheros pero referenciados como si de un único fichero se tratara.

JAVA: Lenguaje de programación orientado a objetos desarrollado por Sun Microsystems en 1995, el cual permite operar independientemente de la plataforma y del sistema operativo que se esté utilizando.

JDBC: (Java Data Base Connectivity) brinda una interfaz estándar con el servidor de base de datos. Provee de un API que puede utilizarse sin importar qué base de datos se esté usando. Además, está soportado por una cantidad de controladores JDBC que le permiten conectarse con los diferentes sistemas propietarios.

INTERFAZ: Medio de comunicación entre el usuario y una aplicación.

OBJETO: Encapsulación genérica de datos y de los procedimientos para manipularlos. Es una entidad que tiene unos atributos particulares, las propiedades, y unas formas de operar sobre ellos, los métodos.

POSTGRES: Sistema Manejador de Bases de datos Objeto – Relacional.

PROGRAMACIÓN ORIENTADA A OBJETOS: Programación que utiliza objetos, ligados mediante mensajes, para la solución de problemas. Puede considerarse como una extensión natural de la programación estructurada en un intento de potenciar los conceptos de modularidad y reutilización del código.

SMBD: Sistema Manejador de Bases de Datos.

TABLA: Es una colección de datos presentada en forma de una matriz bidimensional, donde las filas son los registros y las columnas son los campos.

RESUMEN

El manejador de Base de Datos Postgres, a pesar de ser un potente y confiable motor, las herramientas existentes para su manejo, son básicas y en algunos casos limitadas, dejando al programador gran parte de la responsabilidad de las aplicaciones, programación y seguridad.

La Herramienta Gráfica GRF-Postgres es una propuesta para ayudar al usuario final en la realización de proyectos computacionales relacionadas con Bases de Datos Postgres de forma rápida y fácil.

La Herramienta se desarrolló usando el Lenguaje Unificado de Modelado UML, utilizando el lenguaje de programación JAVA, bajo el sistema operativo Linux y utilizando el motor de Base de Datos Postgres. Cabe aclarar que la Herramienta, por ser programada en el lenguaje JAVA, es portable a diferentes plataformas.

La Herramienta Gráfica GRF-POSTGRES, diseñada para el trabajo con Bases de Datos Postgres, esta conformada por tres módulos independientes pero complementarios:

Editor de Bases de Datos. Permite al usuario final crear, modificar, consultar y eliminar estructuras de Base de Datos como tablas, llaves primarias y foráneas, incluso Bases de Datos completas, respetando relaciones y restricciones especificadas por el usuario, estas Bases de Datos pueden encontrarse en el equipo local en un equipo remoto al cual se tiene acceso.

Editor de Formularios. Construye a partir de tablas, una interfaz sencilla que permita al usuario final, realizar operaciones básicas de inserción, modificación, eliminación y consulta de registros, también localizar el registro anterior, siguiente, primero y último.

Editor de Reportes. Partiendo de ciertos campos de tablas, elabora un reporte, el cual queda incorporado a una vista gráfica que permite imprimir o guardar como archivo de extensión rtf.

La Herramienta trabaja sobre estructuras de Bases de Datos Postgres, que pueden ser creadas usando la Herramienta o importadas desde el motor con sentencias SQL estándar.

El código fuente programado en el lenguaje JAVA, de los reportes y las formas generados por la Herramienta, será suministrado como archivos independientes,

para ser reutilizados en aplicaciones de mayor complejidad, fácilmente portables y compatibles con otras plataformas.

Los reportes y formularios quedan también en formato de la herramienta para poder ser leídos por la misma en posteriores oportunidades.

ABSTRACT

The Driver of Database Postgres, in spite of being a potent and reliable motor, it has basic tools for its handling, and in some limited cases it leaves to the programmer great part of the responsibility of the applications, programming and security.

The graphic tool GRF-Postgres is a proposal to help the final user developing computer projects related with databases Postgres in a quick and easy way.

The tool was developed using the Unified Language of Modeling UML, with the JAVA programming language under the operating system Linux and using the database motor Postgres. It is necessary to make people understand that the tool, to be programmed in the language JAVA, is portable to different platforms.

The graphic tool GRF-Postgres, designed for the work with database Postgres, is conformed by three independent, but complementary modules:

Editor of Database. It allows the final user to create, to modify, to consult and to eliminate database structures like charts, primary and foreign keys, even complete database respecting relationships and restrictions specified by the user, these databases can be in the local computer or in a remote computer with access.

Editor of Forms (Applications). It builds from charts, a simple interface that allows to the final user to carry out basic operations of insert, modification, elimination and consultation of registrations, also to locate the previous, following, first and last registration.

Editor of Reports. From certain fields of charts, it elaborates a report, which is incorporated to a graphic view that allows to print or to keep like extension file rtf.

The tool works on structure of database Postgres which can be created using the tool or imported from the motor with sentences standard SQL.

The code source programmed in the JAVA language from the reports and the ways generated by the tool will be given as independent files, to be reused in applications of more complexity, easily portable and compatible with other platforms.

The reports and forms (or applications) are also in format of the tool to be able to be read by the same one in later opportunities.

INTRODUCCIÓN

Actualmente la tendencia del software libre permite explorar nuevos horizontes relacionados con el desarrollo de software bajo la licencia de código abierto y es ahí donde el programador necesita nuevas herramientas que le ayuden de forma eficiente y eficaz con su trabajo, lamentablemente estas son escasas, están en desarrollo o no existe una que se ajuste a lo necesitado. Por otro lado mientras el mercado de software exige cada vez más aplicaciones en tiempo récord, trabajar en software libre tiende a volverse dispendioso.

Un caso particular de este hecho es el del manejador de Base de Datos Postgres, que a pesar de ser un potente y confiable motor, las herramientas existentes para su manejo, son básicas y en algunos casos limitadas, dejando al programador gran parte de la responsabilidad de las aplicaciones, programación y seguridad.

La Herramienta Gráfica GRF-Postgres es una propuesta para ayudar al usuario final en gran parte con la realización de proyectos computacionales relacionadas con Bases de Datos Postgres de forma rápida y eficiente, en lo que corresponde a la creación, modificación, eliminación y consulta de estructuras de Bases de Datos Postgres, la generación de formas sobre tablas, con operaciones básicas de inserción, modificación, eliminación y consulta, permitiendo avanzar, retroceder, ir al primero y último registro; la generación de reportes con ciertos campos de tablas, totalizando o subtotalizando por atributos específicos de forma horizontal o vertical.

Para el desarrollo de esta herramienta se siguió el Lenguaje Unificado de Modelado UML, la cual es orientada a objetos, permitiendo de esta forma que se ajuste al tipo de software gráfico que se requiere y al lenguaje de programación escogido JAVA.

1. TEMA

1.1 TÍTULO

GRF-Postgres – Herramienta Gráfica para el motor de Base de Datos Postgres.

1.2 MODALIDAD

El presente proyecto de trabajo de grado corresponde a la modalidad estipulada como: TRABAJO DE INVESTIGACIÓN.

1.3 LÍNEA DE INVESTIGACIÓN

GRF-Postgres pertenece a la línea de investigación definida por el programa de Ingeniería de Sistemas de la Universidad de Nariño, como: SOFTWARE Y MANEJO DE INFORMACIÓN.

1.4 ALCANCE Y DELIMITACIONES

La Herramienta se desarrolló usando el Lenguaje Unificado de Modelado UML, programada: con el lenguaje de programación JAVA, bajo el sistema operativo Linux y utilizando el motor de Base de Datos Postgres. Cabe aclarar que la Herramienta, por ser programada en el lenguaje JAVA, es portable a diferentes plataformas.

La Herramienta Gráfica GRF-POSTGRES, diseñada para el trabajo con Bases de Datos Postgres, esta conformada por tres módulos independientes pero complementarios:

- **Editor de Bases de Datos.** Permite al usuario final crear, modificar, consultar y eliminar estructuras de Base de Datos como tablas, llaves primarias y foráneas, incluso Bases de Datos completas, respetando relaciones y restricciones especificadas por el usuario; estas estructuras pueden ser importadas y exportadas a través de scripts constituidos de sentencias SQL estándar.

  El trabajo sobre estas bases de datos puede realizarse de forma local al equipo o remota a través de una Red, con los permisos correspondientes.

- **Editor de Formularios.** Construye a partir de tablas, una interfaz sencilla en lenguaje Java, que permita al usuario final, realizar operaciones básicas de inserción, modificación, eliminación y consulta de registros, también localizar el registro anterior, siguiente, primero y último. Estas interfaces contienen elementos de uso común por parte de los programadores, como son: etiquetas, cajas de texto, cajas de combos, lista, etc. y otras de manejo avanzado como son el objeto Data.

  Los elementos que contienen los formularios (generados o no), pueden ser manipulados fácilmente por medio de una interfaz de diseño que contiene una ventana de propiedades. Por otra parte, se suministra al usuario una vista código Java del formulario, en el cual, se puede complementar la programación del código, este código fuente Java, puede ser compilado y ejecutado a través de la herramienta.

- **Editor de Reportes.** Debido a la magnitud del proyecto, este módulo se desarrollo en sus fases de análisis y diseño, incluso se llego a programar algunas de las interfaces que se utilizarían. El módulo como tal, esta planeado para permitir que con ciertos campos de tablas se elabore un reporte, que puede ser complementado totalizando, subtotalizando, y realizando sumatorias por atributos, con algunas opciones de formato (columnas, tablas y registro) y presentación de la información, además de etiquetas, imágenes, pie de página, número de página y fecha.

  Se pretende que los reportes puedan ser visualizados con opciones de vista previa, vista preliminar (Zoom), además de una vista de diseño, la cual permite realizar modificaciones básicas sobre el reporte. También se quiere implementar la opción de guardarlos ya sea como programas Java o como formato *.rtf o pueden ser enviados directamente a la impresora.

La Herramienta trabaja sobre estructuras de Bases de Datos Postgres, que pueden ser creadas usando la Herramienta o importadas desde el motor con sentencias SQL estándar.

El código fuente programado en el lenguaje JAVA, de los reportes y los formularios generados por la Herramienta, serán suministrados como archivos independientes, los cuales pueden ser reutilizados en aplicaciones de mayor complejidad, fácilmente portables y compatibles con otras plataformas.

Los formularios pueden ser guardados también en formato de la herramienta para poder ser leídos por la misma en posteriores oportunidades, esta característica también se implementaría en reportes una vez este implementado el módulo respectivo.

2. DESCRIPCIÓN DEL PROBLEMA

2.1 PLANTEAMIENTO

En el mercado se ofrecen una cantidad considerable de motores y manejadores de Bases de Datos, con diferentes funcionalidades, en diversas plataformas y con distintos requerimientos, pero en la mayoría de casos, su uso por parte de la pequeña y mediana empresa no es posible, porque la adquisición legal de los mismos representa un alto costo económico.

Postgres es un robusto motor de bases de datos, licenciado como software libre, es decir, se puede copiar, distribuir y modificar libremente; incluye el código fuente y su costo es inferior con respecto a otros motores. Convirtiéndose en una buena alternativa en motores de Bases de Datos, pero las herramientas que sirven de interfaz entre el usuario final y el motor, requieren que el usuario posea conocimientos del lenguaje SQL; estas herramientas tampoco facilitan la manipulación de estructuras de Base de Datos, ni la generación de formas y reportes a partir de la misma.

Para programar aplicaciones relacionadas con el motor Postgres, el usuario puede utilizar diferentes lenguajes, que poseen sentencias de conexión con el motor y la Base de Datos respectiva, implicando una programación un tanto extensa y complicada, pero muchos usuarios desean crear una aplicación sencilla, que no requiera profundizar en conocimientos y sin tener que programar algoritmos difíciles relacionados con el motor, esto motiva a abandonar la idea de utilizar Postgres.

2.2 FORMULACIÓN

¿Es posible por medio de un ambiente gráfico para Postgres facilitar el desarrollo de aplicaciones y manipulación de estructuras de este motor?

2.3 SISTEMATIZACIÓN

¿Se puede generar y manipular estructuras de Bases de Datos en este motor desde una aplicación grafica?

¿Es posible crear aplicaciones que realicen operaciones básicas, de forma rápida y eficiente sobre una Base de Datos realizada en el motor Postgres?

¿Se lograría la generación de reportes con información perteneciente a Bases de Datos Postgres, con elementos necesarios y de fácil manejo?

¿Es admisible la construcción de una herramienta que ayude a generar soluciones computacionales básicas, con los elementos disponibles bajo la licencia de software libre?

3. OBJETIVOS

3.1 OBJETIVO GENERAL

Analizar, diseñar y desarrollar una herramienta gráfica que facilite la manipulación de estructuras de Base de Datos y el desarrollo de aplicaciones bajo el motor Postgres.

3.2 OBJETIVOS ESPECÍFICOS

- 🕮 Generar y Manipular estructuras de Bases de Datos Postgres desde una aplicación gráfica.

- 🕮 Analizar, diseñar e implementar la generación rápida y eficiente de aplicaciones con operaciones básicas sobre tablas de una Bases de Datos Postgres.

- 🕮 Analizar y diseñar los elementos necesarios para construir reportes a partir de la información contenida en una Base de Datos Postgres.

- 🕮 Desarrollar la Herramienta usando el lenguaje unificado de modelado UML, el lenguaje de programación JAVA, el Sistema Operativo Linux y el motor de Bases de Datos Postgres.

- 🕮 Realizar los respectivos procesos de implementación, prueba y evaluación de la Herramienta.

- 🕮 Profundizar conocimientos relacionados con el análisis y diseño orientado a objetos y el lenguaje unificado de modelado UML.

4. JUSTIFICACIÓN

Como ya se describió en el planteamiento del problema, se hace evidente la necesidad de una herramienta que ayude a los usuarios finales con la manipulación de estructuras y facilite la generación de aplicaciones relacionadas con Bases de Datos Postgres.

Una herramienta gráfica, de fácil manejo y compresión, que permita a los usuarios con algunos conocimientos del motor Postgres y el Sistema Operativo Linux u otro, manipular estructuras de Bases de Datos Postgres de forma eficiente, generar formas con operaciones básicas sobre tablas, y reportes con la información contenida en la Base de Datos; esta herramienta motivará a los usuarios finales a trabajar de una forma dinámica con el motor Postgres.

Las herramientas gráficas existentes en la actualidad, y que trabajan sobre el motor Postgres, si bien ayudan a manipular estructuras, a construir formas y algunos reportes básicos sobre Bases de Datos, presentan inconvenientes en cuanto a la limitación de resultados porque solo trabajan sobre una tabla o los productos finales no poseen características de reutilización en aplicaciones complejas, sin considerar que el usuario debe poseer cierto grado de conocimiento en programación y del lenguaje SQL.

GRF-Postgres al ser licenciado como software libre, igual que el motor Postgres, se convierte en una alternativa para generar proyectos computacionales razonables, necesarios en la pequeña y mediana empresa que no cuenta, con recursos económicos suficientes para adquirir legalmente una herramienta, ni un motor de Bases de Datos de alto costo, para desarrollar dichos proyectos, sin considerar costos de capacitación, soporte y licenciamiento.

GRF-Postgres se desarrolló usando el lenguaje JAVA, permitiéndole entonces ser portable a otras plataformas, puede importar y exportar Bases de Datos, entrega al usuario el código fuente de las formas y reportes en JAVA como archivos independientes o como archivos de la herramienta, que pueden ser modificados, si el usuario conoce el lenguaje, reutilizados en aplicaciones complejas o usados en otras plataformas.

5. ANTECEDENTES

El motor de base de datos Postgres, se encuentra dentro de la categoría de los sistemas de administración de Bases de Datos relacionales orientadas a objetos (OR-DBMS), puesto que soporta un modelo de datos que consiste en una colección de relaciones, con características adicionales básicas como son clases, herencia, tipos, funciones, restricciones (constraints), Disparadores (triggers), Reglas (rules) y la integridad referencial. Es por ello que este motor de base de datos Postgres se posiciona como uno de los DBMS más robustos y completos del mercado, posibilitando un uso mayor al pertenecer al software libre.[1]

Sin embargo las herramientas existentes que permiten trabajar sobre este motor son limitadas, entre ellas podemos citar:

5.1 PGACCESS

UNA HERRAMIENTA GRÁFICA DE USUARIO PARA LA GESTIÓN DE POSTGRESQL[1]

PgAccess es un interface Tcl/Tk para PostgreSQL [1]. Como su nombre mismo lo dice: se trata de una herramienta de gestión únicamente, más no de diseño ni mucho menos de desarrollo.

Características de PgAccess
PgAccess para Windows: ventana completa, constructor de tablas, visor de tablas (query)[1].

Tablas:

- Apertura de tablas para visualización, con un máximo de 200 registros (limitante en visión de registros).

[1] MOREA, Lucas. Documento COMO para el RDBMS-SQL-Base de datos para Linux (Sistema de base de datos objeto-relacional Postgresql) Herramientas de Gestión de PostgreSQL (Online). Disponible en Internet: URL: http://linux.dsi.internet2.edu/docs/LuCaS/Postgresql-es/web/navegable/Howto/PostgreSQL-COMO-10.html, (Citado 13 de marzo del 2004).

- Reajuste del tamaño de la tabla, desplazando la línea vertical de la rejilla (mejor en el espacio de la tabla que en la cabecera de la tabla)[2]. (Visualización)Retorno carro en el texto en las celdas. Dibujo salvado para cada tabla.

- Capacidades de ordenación de registros (introduzca manualmente los campos a ordenar)[2].

- Edición en el sitio[2].

- Asistente al generador de tablas[2].

- Edición de campos[2].

Las funciones sobre las tablas son limitadas tratándose en la mayoría de visualización con efecto únicamente dentro de la herramienta, además posee un limitante en la visualización de registros a solo 200 registros, cuando Postgres posee ilimitado número de posibles registros (Se conocen bases de datos de hasta 60 Gigas)

Consulta:

- Define, edita y almacena "consultas definidas por el usuario"[2].

- Almacena consultas como vistas[2].

- Ejecución de consultas[2].

- Visualización de resultados de consultas de tipo select[2].

- Borrado y renombrado de consultas.

Las consultas solo pueden verse y ejecutarse dentro de la herramienta, ligando todo el trabajo a la misma y al equipo en donde se este trabajando, no permite una exportación de las consultas, tampoco pueden ser tratadas como reportes con todos los diseños que esto implica, no permite la exportación fuera de la herramienta del trabajo que se esta realizando sobre esta base de datos, ni la independencia de los formatos de inserción de datos. En síntesis el gran problema de PgAccess es el ligamento que realiza de los elementos trabajados a la herramienta y al equipo, a tal grado que guarda todos estos elementos en tablas con datos encriptados indescifrables en los cuales la única forma de acceder a ellos es abriéndolos en la herramienta.

[2] MOREA, Lucas. et al.

Además se necesita la librería de interface de PostgreSQL a Tcl libpgtcl, colocada en línea como un módulo que se puede cargar en Tcl/Tk. La librería libpgtcl y la fuente está localizada en el directorio /src/interfaces/libpgtcl del PostgreSQL. Específicamente, se necesita una librería libpgtcl que se pueda cargar desde Tcl/Tk. Esto es técnicamente diferente de un fichero objeto que se obtiene de PostgreSQL ordinario, porque libpgtcl es una colección de ficheros objeto. Bajo Linux, esto se llama libpgtcl.so. Puede descargarse desde la dirección anterior una versión ya compilada para sistemas Linux i386[3].

5.2 ISQL o PSQL

También llamada PSQL, interfaz texto fácil de utilizar, diseñada para terminales de línea de comando de formato carácter, incluida en la distribución de Postgres. Es muy similar al ISQL de Sybase o al SQL*Plus de Oracle. Puede utilizarse en los scripts de la shell[4].
Esta interfaz necesita aprender comandos de sintaxis estricta con modificadores para obtener diferentes funciones, por ejemplo para obtener ayuda con los comandos se teclea \h.

El comando psql proporciona acceso a una terminal interactiva de PostgreSQL. Suponiendo que se quiera crear una base de datos llamada demodb, los comandos serían:

```
$ createdb demodb
$ psql demodb
```

y el prompt resultante, donde se puede introducir sentencias de SQL, sería:

```
demodb=#
```
[4]

Tras ejecutar psql y obtener el prompt anterior, se puede poner:
\h para obtener ayuda sobre SQL
\? para obtener ayuda sobre comandos específicos de psql
\g para ejecutar consultas
\q para salir de psql[4]

[3] MOREA, Lucas. et al.
[4] EFEBER.NET. Clientes: psql en modo texto, PgAccess en modo gráfico (Online). Disponible en Internet. URL: http://www.efaber.net/formacion/fp/curso_acs/index.html#clientes (Citado 13 de Marzo del 2004)

No deja de ser una interfaz de texto en la cual la ayuda para un usuario estándar sin conocimientos en SQL es casi nula, amenos que este haya realizado un curso completo de comandos o se siente durante horas a tratar de comprenderlos y aprenderlos.

6. MARCO TEORICO

6.1 EL MOTOR DE BASE DE DATOS POSTGRES

En los últimos años, el software de bases de datos ha experimentado un auge extraordinario, a raíz de la progresiva informatización de casi la totalidad de las empresas de hoy día. No es extraño pues, que existan multitud de gestores de bases de datos, programas que permiten manejar la información de modo sencillo. De este modo están Oracle™, Microsoft SQL Server™, Borland Interbase™ entre otras. Las soluciones software que se citaron son comerciales. Como siempre, en el mundo del software libre, siempre que se necesita algo, tarde o temprano se implementa. De esta forma se tiene MySQL™, gestor muy usado en la Web (combinado con PHP y Apache) o PostgreSQL™, que será el gestor que se tratará (Existe una interesante comparativa de rendimiento entre PostgreSQL y MySQL (ver tabla 1). Viene a decir que PostgreSQL es superior en BBDD "a lo grande", mientras que para conjuntos de datos pequeños o medianos MySQL funciona mejor.)[5]

Como se ha comentado, PostgreSQL es software libre. Concretamente está liberado bajo la licencia BSD, lo que significa que cualquiera puede disponer de su código fuente, modificarlo a voluntad y redistribuirlo libremente (Es más, la licencia BSD permite redistribuir el código modificado o no como software cerrado, en contraposición a la licencia GPL que fuerza a que las modificaciones sean publicadas también bajo la GPL), PostgreSQL además de ser libre es gratuito y se puede descargar libremente de su página Web para multitud de plataformas[5].

Tabla 1. Comparativa entre PostgreSQL y MySQL

	PostgresSQL	MySQL
Descripción	Postgres intenta ser un sistema de base de datos de mayor nivel que MySQL, a la altura de Oracle, Sybase o Interbase	Su principal objetivo de diseño fue la VELOCIDAD. Se sacrificaron algunas características esenciales en sistemas más “serios” con este fin.

[5] ROBLES, Tomás J; TURIENZO, Raúl. Introducción a PostgreSQL (online), Disponible en Internet: URL: http://programacion.com/bbdd/tutoriales/PostgreSQL/, programación en castellano (URL: http://programacion.com), 11 de marzo del 2003 (citado el 13 de marzo del 2004).

	PostgresSQL	MySQL
Licencia	BSD	GPL a partir de la versión 3.23.19
Ventajas	Por su arquitectura de diseño, escala muy bien al aumentar el número de CPUs y la cantidad de RAM. Soporta transacciones y desde la versión 7.0, claves ajenas (con comprobaciones de integridad referencial). Tiene mejor soporte para tigres y procedimientos en el servidor. Soporta un subconjunto de SQL92 MAYOR que el que soporta MySQL. Además, tiene ciertas características orientadas a objetos.	Mayor rendimiento. Mayor velocidad tanto al conectar con el servidor como al servir selects y demás. Mejores utilidades de administración (backup, recuperación de errores, etc.). Aunque se cuelgue, no suele perder información ni corromper los datos. Mejor integración con PHP. No hay limites en tamaño de los registros, Mejor control de acceso, en el sentido de qué los usuarios tienen acceso a que tablas y con qué permisos. MySQL se comporta mejor que Postgres a la hora de modificar o añadir campos a una tabla “en caliente”
Desventajas	Consume más recursos y carga más el sistema. Limite del tamaño de cada fila de las tablas a 8k (se puede ampliar a 32k recompilando, pero con un coste añadido en el rendimiento). Es de 2 a 3 veces más lenta que MySQL. Menos funciones en PHP.	No soporta transacciones, “roll - backs”, ni subselects. No considera las claves ajenas. Ignora la integridad referencial, dejándola en manos del programador de la aplicación.
Tamaño máximo BD	Ilimitado, actualmente se conoce que existen BD con un tamaño 60GB	Ilimitado
Tamaño máximo de Tablas	64 TB	8 TB
Conexiones simultaneas	32	101
Columnas en una tabla	1600	3398
Tamaño de Query	16777216	1048574

MELO, Diego Samir. Webs dinámicas utilizando PHP. En: II CONGRESO INTERNACIONAL SOFTWARE LIBRE GNU/LINUX. (2º: 2003: Manizales). Ventana Informática – Edición Especial II Congreso Internacional Software Libre GNU/Linux: Departamento de Publicaciones Universidad de Manizales, 2003. p 59.

6.2 LENGUAJE UNIFICADO DE MODELADO UML

El Lenguaje Unificado de Modelado (UML, Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema con gran cantidad de software[6].

UML proporciona una forma estándar de escribir los planos de un sistema, cubriendo tanto las cosas conceptuales, tales como procesos del negocio y funciones del sistema, como las cosas concretas, tales como las clases escritas en un lenguaje de programación específico, esquemas de base de datos y componentes software reutilizables[6].

Los lenguajes de modelado orientados a objetos aparecieron, en algún momento, entre la mitad de los sesenta y finales de los ochenta cuando los metodologistas, enfrentados a los nuevos lenguajes de programación orientados a objetos y a unas aplicaciones cada vez más complejas, empezaron a experimentar con enfoques alternativos al análisis y al diseño[6].

De estas experiencias, comenzaron a aparecer nuevas generaciones de métodos, entre los que destacaron de manera muy clara unos pocos métodos, en especial el método Booch, el método OOSE (Object-Oriented Software Engineering, Ingeniería del Software Orientada a Objetos) de Jacobson y el método OMT (Object Modeling Technique, Técnica de Modelado de Objetos) de Rumbaugh. Otros métodos importantes fueron Fusion, Shlaer-Mellor y Coad-Yourdon. Cada uno de estos era un método completo, aunque todos tenían sus puntos fuertes y sus debilidades[6].

Una masa crítica de ideas comenzó a formarse en la primera mitad de los noventa, cuando Grady Booch (Rational Software Corporation), Ivar Jacobson (Objectory) y James Rumbaugh (General Electric) empezaron a adoptar ideas de cada uno de los otros dos métodos, los cuales habían sido reconocidos en conjunto como los tres principales métodos OO2. Del trabajo en conjunto durante muchos años surgió lo que se conoce ahora como UML, el cual ha recibido apoyo de muchas más corporaciones y reconocimiento de otras como la OMG (Object Mangament Group)[7].

UML es apropiado para modelar sistemas de información en empresas hasta aplicaciones distribuidas basadas en la Web, e incluso para sistemas empotrados de tiempo real muy exigentes. Es un lenguaje muy expresivo, que cubre todas las vistas necesarias para desarrollar y luego desplegar tales sistemas.

---

[6] BOOCH, G., et al. El Lenguaje Unificado de Modelado: El libro introductorio a UML escrito por sus creadores. 1 ed. Madrid: Addison Wesley, 1999. p. XIX, XXI, p. 11 - 12

[7] BOOCH, G., Op cit., p. XXII – XXIV, p.11 - 15

UML es solo un lenguaje y por tanto es tan sólo una parte de un método de desarrollo del software. UML es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, interactivo e incremental[7].

UML es un lenguaje que proporciona un vocabulario y las reglas para combinar palabras de ese vocabulario con el objetivo de posibilitar la comunicación. Un lenguaje de modelado es un lenguaje cuyo vocabulario y reglas se centran en la representación conceptual y física de un sistema[7].

El modelado proporciona una compresión de un sistema. Nunca es suficiente un único modelo. Más bien, para comprender cualquier cosa, a menudo se necesitan múltiples modelos conectados entre sí, excepto en los sistemas más triviales[7].

Para sistemas con gran cantidad de software, se requiere un lenguaje que cubra las diferentes vistas de la arquitectura de un sistema mientras evoluciona a través del ciclo de vida del desarrollo del software[7].

El vocabulario y las reglas e un lenguaje como UML indican cómo crear y leer modelos bien formados, pero no dice qué modelos se deben crear ni cuando se deberían crear. Esta es la tarea del proceso del desarrollo del software (Ver metodología)[8].

Para comprender UML, se necesita adquirir un modelo conceptual del lenguaje, y requiere aprender tres elementos principales: los bloques de construcción de UML, las reglas que dictan cómo se puede combinar estos bloques básicos y algunos mecanismos comunes que se aplican a través de UML[8].

- 🕮 **Bloques de construcción de UML.** El vocabulario de UML incluye tres clases de bloques de construcción: Los elementos que son abstracciones, que son ciudadanos de la primera clase en un modelo; las relaciones que ligan estos elementos entre sí; los diagramas que agrupan colecciones interesantes de elementos[8].

1. **Elementos.** Incluyen los elementos estructurales, elementos de comportamiento, elementos de agrupación y elementos de anotación. Estos elementos son los bloques básicos de construcción orientados a objetos de UML. Se utilizan para escribir modelos bien formados[8].

2. **Relaciones.** Hay cuatro tipos de relaciones: Dependencia, asociación, generalización y realización. Estas relaciones son los bloques básicos de

[7] BOOCH, G., Op cit., p. XXII – XXIV, p.11 - 15
[8] Ibid., p. 12, p.15-23

construcción para relaciones de UML. Se utilizan para escribir modelos bien definidos[9].

3. **Diagramas.** Un diagrama es la representación gráfica de un conjunto de elementos, visualizando la mayoría de veces como un grafo conexo de nodos (elementos) y arcos (relaciones). Un diagrama se dibuja para visualizar un sistema desde diferentes perspectivas, de forma que un diagrama es una proyección de un sistema[9].

UML incluye nueve diagramas: Diagramas de clases, Diagrama de objetos, Diagrama de casos de uso, Diagrama de secuencia, Diagrama de colaboración, Diagrama de estados, Diagrama de actividades, Diagrama de componentes y diagramas de despliegue[9].

- **Reglas de UML.** Los bloques de construcción de UML, no pueden simplemente combinarse de cualquier manera. Como cualquier lenguaje, UML tiene un número de reglas que especifican a qué debe parecerse un modelo bien formado el cual debe ser semánticamente autoconsistente y está en armonía con todos sus modelos relacionados[9].

UML tiene reglas semánticas para:

- Nombres: Como llamar a los elementos, relaciones y diagramas.
- Alcance: El contexto que da un significado específico a un nombre.
- Visibilidad: Cómo se puede ver y utilizar esos nombres por otros.
- Integridad: Cómo se relacionan apropiada y consistentemente unos elementos con otros.
- Ejecución: Qué significa ejecutar o simular un modelo dinámico[9].

- **Mecanismos comunes en UML.** En UML se aplican cuatro mecanismos comunes a través de todo el lenguaje de forma consistente[9]:

1. **Especificaciones.** UML es algo más que un lenguaje gráfico. Más bien, detrás de cada elemento de su notación gráfica hay una especificación que proporciona una explicación textual de la sintaxis y semántica de ese bloque de construcción[10].

[9] BOOCH, G., Op cit., p.15 - 25.
[10] Ibid., p. 22-25.

Las especificaciones de UML proporcionan una base semántica que incluye a todos los elementos de todos los modelos de un sistema, y cada elemento está relacionado con otros de manera consistente. Los diagramas de UML son así simples proyecciones visuales de esa base, y cada diagrama revela un aspecto específico interesante del sistema[11].

2. **Adornos.** La mayoría de los elementos de UML tienen una única y clara notación gráfica que proporciona una representación visual de los aspectos más importantes del elemento[11].

   La especificación de una clase puede incluir otros detalles, tales como si es abstracta o la visibilidad de sus atributos y operaciones. Muchos de estos detalles se pueden incluir como adornos gráficos o textuales en la notación rectangular de una clase[11].

3. **Divisiones comunes.** Al modelar sistemas orientados a objetos, el mundo puede dividirse, al menos, en un par de formas[11].

   En primer lugar, esta la división entre clase y objeto. Una clase es una abstracción; un objeto es una manifestación concreta de esa abstracción[11].

   En segundo lugar, tenemos la separación entre interfaz e implementación. Una interfaz declara un contrato, y una implementación representa una realización concreta de este contrato, responsable de hacer efectiva de forma fidedigna la semántica completa de la interfaz[11].

4. **Mecanismos de extensibilidad.** UML proporciona un lenguaje estándar para escribir planos de software; pero no es posible que un lenguaje cerrado sea siempre suficiente para expresar todos los matices posibles de todos los modelos en todos los dominios y en todos los momentos. Por esta razón, UML es abierto-cerrado, siendo posible extender el lenguaje de manera controlada[11].

Los mecanismos de extensión de UML incluyen:

- Estereotipos.
- Valores etiquetados.
- Restricciones[11].

[11] BOOCH, G., Op cit., p. 22-25

6.3 JAVA

Según Joyanes Aguilar, et al. [12], java desde su lanzamiento se ha convertido en un lenguaje de programación tanto de propósito general como de Internet.

Java, desarrollado por Sun Microsystems en 1995, es un magnífico y completo lenguaje de programación orientado a objetos diseñado para distribuir contenidos a través de una red. Una de sus principales características es que permite operar de forma independiente de la plataforma y del sistema operativo que se esté utilizando[12].

> La idea de Java, es poner una capa sobre cualquier plataforma de hardware y sobre cualquier sistema operativo que permite que cualquier aplicación desarrollada en Java quede ligada únicamente a Java, independizada por lo tanto de la plataforma. Esta concepción queda recogida en el concepto de máquina virtual JVM (Java Virtual Machine), un software que interpreta instrucciones para cualquier máquina sobre la que esté corriendo y que permite, una vez instalado, que una misma aplicación pueda funcionar en un PC o en un Mac sin tener que tocarla. Hoy en día cualquier sistema operativo moderno (Windows, Machintonsh, Linux, Unís, Solaris, etc.) cuenta con una JVM. Así lo que hace Java en combinación con esta <<máquina>> es funcionar como hardware y como sistema operativo virtual, emulando en software una CPU universal. Al instalar Java esta actuando como una capa de abstracción entre un programa y el sistema operativo, otorgando una total independencia de lo que haya por debajo; es decir, cualquier aplicación funcionará en cualquier máquina e incluso cualquier dispositivo[12].

Java es un descendiente de C++ que a su vez es descendiente de C. Muchas características de Java se han heredado de estos dos lenguajes. De C, Java ha heredado su sintaxis y de C++, las características de programación orientada a objetos[12].

El diseño original de Java fue concebido por James Gosling, Patrick Naughton, Chris Warth, Ed Frank y Mike Sheridan, ingenieros y desarrolladores de Sun Microsystems en 1991, que tardaron 18 meses en terminar la primera versión de trabajo. Este lenguaje se llamo inicialmente <<Oak>>, y se le cambio el nombre por Java en 1995[12].

Sun Microsystems es propietaria de Java: sin embargo, numerosos fabricantes contribuyen a la mejora y desarrollo de las especificaciones del estándar. Sun proporciona las licencias de esta tecnología pero ejerce siempre un cierto control

[12] JOYANES, Luís, et al. Java 2: Manual de Programación. 1 ed. Madrid: McGraw-Hill, 2001. p. xiii

sobre las implementaciones que se hacen de la misma, con el objetivo de mantener la independencia de plataforma[13].

En síntesis par Joyanes Aguilar[13], et al., el significado de Java tal y como se le conoce en la actualidad es el de un lenguaje de programación y un entorno para ejecución de programas escritos en el lenguaje Java. Al contrario que los compiladores tradicionales, que convierten el código fuente en instrucciones a nivel de máquina, el compilador Java traduce el código fuente Java en instrucciones que son interpretadas por la Máquina Virtual Java (JVM, Java Virtual Machine). A diferencia de los lenguajes C y C++ en los que está inspirado, Java es un lenguaje interpretado.

Aunque hoy en día Java es por excelencia el lenguaje de programación para Internet y la Web en particular, Java no comenzó como proyecto Internet y por esta circunstancia es idóneo para tareas de programación de propósito general y, de hecho, muchas de las herramientas Java están escritas en Java[13].

Java ha conseguido una enorme popularidad. Su rápida difusión e implantación en el mundo de la programación en Internet y fuera de línea ha sido posible gracias a sus importantes características. Los creadores de Java escribieron un artículo, ya clásico, en el que definían al lenguaje como sencillo, orientado a objetos, distribuido, interpretado, robusto, seguro, arquitectura neutra, alto rendimiento, multihilo y dinámico[13].

La programación orientada a objetos (POO) es la base de Java y constituye una nueva forma de organización del conocimiento en la que las entidades centrales son los objetos[13].

> En un objeto se unen una serie de datos con una relación lógica entre ellos, a los que se denomina variables de instancia, con las rutinas necesarias para manipularlos, a las que se denomina métodos. Los objetos se comunican unos con otros mediante interfaces bien definidas a través de paso de mensajes; en POO los mensajes asociados con métodos, de forma que cuando un objeto recibe un mensaje, ejecuta el método asociado.

Cuando se escribe un programa utilizando programación orientada a objetos, no se definen verdaderos objetos, sino clases; una clase es como una plantilla para construir varios objetos con características similares. Los objetos se crean cuando se define una variable de su clase[13].

En las clases pueden existir unos métodos especiales denominados constructores que se llaman siempre que se crea un objeto de esa clase y cuya misión es iniciar

[13] JOYANES, Op cit., p. 11.

el objeto. Los destructores son otros métodos especiales que pueden existir en las clases y cuya misión es realizar cualquier tarea final que corresponda realizar en el momento de destruir el objeto. Las propiedades fundamentales de los objetos son: [14]

- **El encapsulamiento.** Que consiste en la combinación de los objetos y las operaciones que se puedan ejecutar sobre esos datos en un objeto, impidiendo usos indebidos al forzar que el acceso a los datos se efectúe siempre a través de los métodos del objeto[14].

  En Java la base del encapsulamiento es la clase, donde se define la estructura y el comportamiento que serán compartidos por el grupo de objetos pertenecientes a la misma. Para hacer referencia a los componentes accesibles de un objeto será necesario especificar su sintaxis[14].

- **La herencia.** Es la capacidad para crear nuevas clases (descendientes) que se construyen sobre otras existentes, permitiendo que estas les trasmitan sus propiedades. En programación orientada a objetos, la reutilización de código se efectúa creando una subclase que constituye una restricción o extensión de la clase base, de la cual hereda sus propiedades[14].

- **El polimorfismo.** Consigue que un mismo mensaje pueda actuar sobre diferentes tipos de objetos y comportarse de modo distinto. El polimorfismo adquiere su máxima expresión en la derivación o extensión de clases; es decir, cuando se obtienen nuevas clases a partir de una ya existente mediante la propiedad de derivación de clase o herencia.

Los programas en Java se dividen en dos grandes categorías: aplicaciones y applets[14].

Las aplicaciones son programas autónomos independientes (standalone), tal como cualquier programa escrito utilizando lenguajes de alto nivel, como C++, C, Ada, etc.; las aplicaciones se pueden ejecutar en cualquier computadora con un intérprete de Java y son ideales para el desarrollo del software[14].

Los applets son un tipo especial de programas Java que se pueden ejecutar directamente en un navegador Web compatible Java; los applets son adecuados para desarrollar proyectos Web[14].

Los applets son programas que están incrustados, <<empotrados>> (embedded) en otro lenguaje; así cuando se utiliza Java en una página Web, el código Java se empotra dentro del código HTML. Por el contrario una aplicación es un programa

[14] JOYANES, Op cit., 12 - 13.

Java que no está incrustado en HTML ni en ningún otro lenguaje y puede ser ejecutado de modo autónomo[15].

6.4 SOFTWARE LIBRE

Para entender los beneficios del Software Libre, se empieza explicando que el software son los programas que permiten que el computador desarrolle funciones y procedimientos, como es la manipulación de información que el hombre realiza pero en forma digital, hay software que permite llevar el manejo financiero de un empresa, suite ofimáticas, juegos, graficadores, visores de dibujo, vídeo, música (mp3), virus, antivirus, etc. [16]

En términos generales, cuando alguien necesita adquirir un software específico existen cuatro alternativas del tipo de software:

- Software Comercial.
- Software Shareware.
- Software Freeware.
- Software de Licencia Pública General (Software Libre) [16].

6.4.1 El Software Comercial. Es aquel por el que se debe pagar para utilizarlo; más que el software, se está comprando la licencia para poder emplearlo y se deben comprar tantas licencias como equipos a utilizar dicho software. Ejemplos de software comercial son los productos de Microsoft, como el sistema operativo Windows y familia (9x, Me, XP), Sistema Manejador de Bases de Datos, programas para servicios de Internet y sus productos para el manejo de oficina (Suite Ofimática), para el caso Office[16].

En esta región el software comercial más utilizado es el programa de contabilidad Apolo. Si una empresa compra estos productos: Sistemas operativos Windows XP, la suite de Office y el Apolo debería invertir casi $2’800.000, un valor mayor al de un computador estándar[16].

Otra de las desventajas del software comercial, es que no se puede modificar el programa (el código fuente no está disponible) y solo el fabricante puede realizar modificaciones; los clientes dan sugerencias de nuevas características que desean agregar a los programas y los fabricantes deciden si las hacen (en caso de considerarlas convenientes) o no[16].

---

[15] JOYANES, Op cit., p. 12 - 13.

[16] GOMEZ, Julio Cesar. Beneficios del software Libre. En: II CONGRESO INTERNACIONAL SOFTWARE LIBRE GNU/LINUX. (2º: 2003: Manizales). Ventana Informática – Edición Especial II Congreso Internacional Software Libre GNU/Linux: Departamento de Publicaciones Universidad de Manizales, 2003. p 14.

Cuando salen nuevas versiones se debe pagar más dinero si se desea actualizar la versión del programa adquirido[17].

6.4.2 El Software Shareware. Son los programas que se pueden utilizar libremente sin ningún problema de licencia por un tiempo determinado. Normalmente utilizan este sistema las casas productoras de software para que los usuarios conozcan sus productos y si les gusta lo compren. Estas versiones de Shareware la mayoría de las veces no viene con todas las funciones activas[17].

Algunos de los sitios donde se puede adquirir este tipo de software son: http://www.tucows.com y www.download.com.[17]

Uno de los programas Shareware más usados es el ws_ftp, que nos permite realizar transferencias de archivos por medio de ftp (file transfer protocolo) otro es el conocido winzip, que ayuda a comprimir / descomprimir en distintos formatos de compresión de archivos. Winzip tiene un costo de US $29 y el periodo de prueba es por 21 días[17].

El software Shareware tiene los mismos intereses que el software comercial, con la diferencia de que este se puede probar antes de adquirirlo, en caso que el software no cumpla con las expectativas del usuario final, este se debe eliminar para no tener problemas de licenciamiento con los fabricantes. De igual forma que el software comercial, el Shareware no permite modificaciones adicionales de características nuevas por parte del usuario del programa[17].

6.4.3 Software Freeware. Son programas que se pueden utilizar si ningún problema de licenciamiento, no hay que comprar licencias, pero se sigue limitado a que las modificaciones en el software sólo pueden ser realizadas por el fabricante del producto[17].

Uno de los programas Freeware más utilizado es la suite ofimática StarOffice de la Sun Microsystems que le esta haciendo una fuerte competencia a la suite ofimática de Microsoft. Actualmente en el Eje Cafetero, ya hay muchas empresas, universidades y escuelas que están utilizando esta herramienta Ofimática, con la ventaja de no tener problemas de licenciamiento[17].

6.4.4 Software de Licencia pública general (Software Libre). Las tecnologías digitales de la información contribuyen al mundo haciendo que sea más fácil copiar y modificar la información. Las computadoras prometen hacer testo más fácil para todos[17].

[17] GOMEZ, Op cit., p. 16 – 17.

No todo el mundo quiere que esto sea más fácil. El sistema del copyright permite que los programas de software tengan “propietarios”, la mayor parte de los cuales pretenden privar al resto del mundo del beneficio potencial del software. Los propietarios desearían ser los únicos que pueden copiar y modificar el software que se utiliza (http://www.gnu.org/philosophy/why-free.es.html) [18].

La fundación de Software Libre (http://www.gnu.org), lucha porque todos los programas sean libres de utilizar y modificar de acuerdo con las necesidades del usuario y no estén atados a un fabricante con sus derechos de autor. Con el software libre además del programa, viene con el código fuente, el cual se puede modificar y adaptar a nuestras necesidades[18].

Con la distribución del código fuente lo que sé esta logrando verdaderamente es la transferencia del conocimiento, puesto que se utiliza el aplicativo y al analizar y depurar el código se aprende cómo se pueden implementar y realizar esos procedimientos e interfaces. El software libre ayuda al desarrollo de los países subdesarrollados, debido a que se tiene la posibilidad de adquirir el conocimiento y hasta en un futuro cercano ser parte activa de esta comunidad[18].

El software libre contribuye con la reducción de los gastos de las instituciones y empresas, puesto que no hay necesidades de comprar licencias. Una licencia de un sistema operativo para servidor de una empresa de Software comercial tiene un valor aproximado de $2’500.000. El programa servidor Web, - este programa atiende las solicitudes de página Web- más usado del mundo es el Apache (http://www.apache.org), el cual es usado en el 70% de los servidores Web de Internet y el cual también es licencia libre y con el código fuente abierto[18].

El sistema operativo GNU/Linux cada vez tienen más adeptos, inicialmente se utilizaba como servidor de una red, pero desde hace unos tres años se ha trabajado para ser también utilizado como computador cliente (usuario final), y los avances han sido muy grandes[18].

En países tan desarrollados como Francia y Alemania, por ley, las entidades gubernamentales tienen que trabajar sólo con software libre. Se cree que en los países latinoamericanos para el 2003 de cada tres computadores uno va a trabajar con GNU/Linux[18].

[18] GOMEZ, Op cit., p. 16 – 17.

Con el gran auge del software libre las empresas de software comercial, han rebajado el precio de sus programas y los deben rebajar más o correr el peligro de desaparecer, inclusive se habla ya de pagar por demanda, es decir, pagar por el tiempo que lo utiliza, puesto que no es justo que si alguien lo utiliza poco, pague lo mismo que alguien que lo usa mucho más[19].

El futuro del negocio del software es cobrar por el soporte, ya existen muchas empresas de software comercial que regalan sus programas y cobran es por el soporte. La gran ventaja del software libre es que el soporte lo puede encontrar con muchas empresas y a precios muy razonables[19]

6.4.5 Conclusiones de beneficios del software libre. Entre otras:

- 🕮 Transferencia del conocimiento para el desarrollo de los países subdesarrollados.
- 🕮 Contribuye con la reducción de gastos.
- 🕮 También hay soporte y cursos de entrenamiento y certificación.
- 🕮 El software comercial ha rebajado el precio de sus programas y los debe rebajar aun más.[19]

[19] Ibid., p. 17 –18.

7. METODOLOGÍA

Para el proceso de desarrollo de esta herramienta computacional se usó el Lenguaje Unificado de Modelado UML, basándose en la metodología del Proceso Unificado de Desarrollo del Software[20], expuesta en el libro que lleva el mismo nombre y complementada en el libro UML y Patrones[21].

UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar los componentes del proceso de desarrollo de un software, dispone de un conjunto de notaciones y diagramas estándar, para modelar sistemas orientados a objetos, y describe el significado de los diagramas y símbolos. Estos diagramas son diversos y buscan representar las diferentes vistas del sistema.

El modelado de sistemas orientado a objetos se basa en el desarrollo de software modelando situaciones del mundo real, mediante la abstracción del mismo se pueden generar modelos simplificados con los cuales se puede interactuar fácilmente, otras características son: la uniformidad en la representación de los modelos, la independencia entre los componentes, la flexibilidad ante modificaciones o ampliaciones, la estabilidad para aislar componentes que permanecen inalterables y la reusabilidad.

UML fue el apropiado para utilizarlo en el proceso de desarrollo de esta Herramienta, no solo por ser gráfico, sino porque de acuerdo al objetivo de la misma, las formas y reportes entregados, pueden utilizarse en una aplicación completa, es decir, se puede reutilizar sin dificultad y eficientemente.

Además, el manipular estructuras de Bases de Datos y generar pequeñas aplicaciones a partir de estas, es una situación propia para modelar con objetos, puesto los componentes de la Bases de Datos pueden clasificarse, y cada uno posee características y procedimientos que realizan, tratándose Postgres de un DBMS Objeto - Relacional.

El proceso de desarrollo de la Herramienta, coherente con el modelado orientado a objetos, se basó en tres aspectos:

[20] JACOBSON, Ivar, et al. El proceso unificado de desarrollo del software. Madrid: Pearson Educación S.A., 2000, 464 p.
[21] LARMAN, Craig. UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. Madrid: Pearson Educación S.A., 2003. 624 p.

- 🕮 Dirigido por los requisitos de los usuarios. Por lo tanto se hizo un estudio para conocer que necesitan y desean los usuarios.

- 🕮 Obtener una vista completa del diseño del sistema. Luego de obtener una vista general de los requisitos que los usuarios presentan se bosqueja el sistema de acuerdo con los mismos.

- 🕮 Dividir el sistema en partes más pequeñas. Las cuales deben desarrollarse de forma que cubran los aspectos importantes de las funcionalidades de la Herramienta.

Es importante aclarar que se pretende obtener la primera versión del software, y que como ventaja adicional del modelado orientado a objetos, a partir de esta versión se pueden generar nuevas versiones con otras funciones importantes.

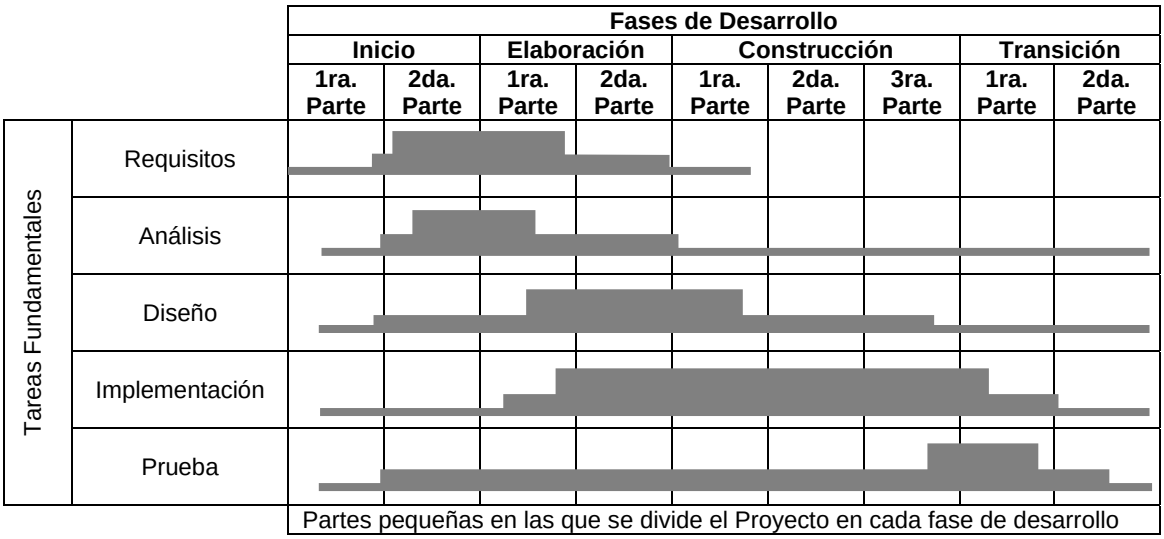
Para el desarrollo de esta primera versión del software, existieron cuatro fases importantes:

La fase de inicio. Donde se describe el producto final, y como será el plan de desarrollo.
La fase de elaboración. En la cual se detalla que aspectos debe cubrir el software y se crean los primeros modelos del sistema.

La fase de construcción. Donde se desarrolla el software y terminan de definirse los modelos.

La fase de transición. Donde se realizan las pruebas convenientes y se corrigen algunos defectos.

Un esquema que representa lo que se realizó es el siguiente:



Toda la estructura de cómo se va a ser el proceso de desarrollo del software de la Herramienta, es conocida como el proceso de desarrollo unificado de software que usa el lenguaje unificado de modelado UML.

8. DESARROLLO DEL PROYECTO

Siguiendo la metodología propuesta, se desarrollaron los flujos de trabajo (Requisitos, análisis, diseño, implementación y prueba) durante las fases de inicio, elaboración, construcción y transición; tomando hasta dos interacciones en cada una de las fases.

8.1 FLUJO DE TRABAJO DE REQUISITOS

Durante las diferentes fases (inicio, elaboración, construcción y transición) e interacciones que se realizaron en cada una de ellas.

8.1.1 Captura de Requisitos. Se realizo la captura de requisitos, en la cual se determinó lo que los posibles usuarios de la herramienta esperaban de ella, se realizo una investigación sobre algunas propuestas existentes en el mercado (Ver antecedentes), y se comparo con las herramientas que existen para otros tipos de bases de datos. De esta forma se identificaron 56 requisitos candidatos para ser implementados y cubiertos por la herramienta.

8.1.2 Funciones del Sistema. Los requisitos candidatos capturados se clasificaron en las siguientes funciones del sistema (Ver tabla 2).

Tabla 2. Tabla de Funciones del Sistema

Ref. No.	Función	Descripción	Categoría
R.1	Crear una Base de Datos	Crear una base de datos partiendo del nombre de la misma, usuario y maquina.	Evidente
R.1.1	Crear Tabla	Crear una tabla con su nombre y los campos que componen su estructura.	Evidente
R.1.1.1	Crear Campo	Crear los campos que componen la estructura de una tabla, cada campo tiene un nombre, un tipo y un tamaño.	Evidente
R.1.1.2	Crear Llaves Primaria	Asignar a uno o varios campos de la tabla como pertenecientes a una llave primaria y asignarle un nombre.	Evidente
R.1.1.3	Crear Llaves Foráneas	Asignar a uno o varios campos de la tabla como perteneciente a una llave foránea la cual esta ligada con otra tabla en la misma BD.	Evidente

Ref. No.	Función	Descripción	Categoría
R.1.1.4	Crear Restricciones de Campo	Para validar los datos que se insertan a un campo de una tabla, estos deben cumplir con ciertas condiciones, como tamaño y valor por defecto, además de no permitir campos nulos.	Evidente
R.2	Modificar una Base de Datos	Modificar la estructura de una base de datos existente	Oculta
R.2.1	Modificar tabla	Sirve para modificar los atributos de una tabla, como su nombre y los respectivos campos que componen su estructura	Evidente
R.2.1.1	Modificar campo	Modificar los campos que componen la estructura de una tabla, como su tipo y tamaño.	Evidente
R.2.1.2	Modificar Llaves Primarias	Consiste en reformar la asignación de la llave primaria de uno o más campos de la tabla, o cambiar su nombre.	Evidente
R.2.1.3	Modificar Llaves Foráneas	Permite modificar atributos de la llave foránea, como campos, nombre y tabla foránea	Evidente
R.2.1.4	Modificar restricciones de Campo	Sirve para cambiar las restricciones creadas para los campos de una tabla	Evidente
R.3	Eliminar una Base de Datos	Para eliminar la base de datos, con la totalidad de sus tablas y registros contenidos	Evidente
R.3.1	Eliminar Tabla	Eliminar la tabla completa incluyendo su estructura y sus registros, advirtiendo la perdida de los mismos.	Evidente
R.3.1.1	Eliminar Campos	Eliminar un campo existente dentro de una tabla	Evidente
R.3.1.2	Eliminar Llaves Primarias	Para eliminar la llave primaria de una tabla existente	Evidente
R.3.1.3	Eliminar Llaves Foráneas	Sirve para eliminar las llave foránea de una tabla existente	Evidente
R.3.1.4	Eliminar Restricciones sobre un Campo	Permite eliminar las restricciones hechas en los campos de una tabla.	Evidente

Ref. No.	Función	Descripción	Categoría
R.3.2	Cerrar Una Base de Datos	Cerrar una base de datos dentro de la herramienta, sin eliminarla dentro del motor	Evidente
R.4	Crear y diseñar formularios gráficos	Generar clases y objetos necesarios para la creación y diseño un formulario enlazado a una tabla de una base de datos	Evidente
R.4.1	Generar formularios de tablas	Generar el código y los objetos necesarios para que el formulario creado permita trabajar con los registros de la tabla con la cual se encuentra enlazado.	Evidente
R.4.2	Área de trabajo de formulario	Generar un espacio de trabajo gráfico donde se pueda modificar y examinar el código y diseño de un formulario por parte del usuario	Oculta
R.4.3	Visor de código	Generar un visor que permita examinar y modificar el código de un formulario	Oculta
R.4.4	Creación de objetos adicionales	Generar clases de objetos con funciones especificas compatibles con la herramienta para usar en el diseño de formularios	Evidente
R.4.5	Insertar objeto al diseñar formulario	Generar una barra de herramientas donde el usuario pueda seleccionar un tipo de objeto y luego dibujar e insertar el objeto en el diseño del formulario.	Evidente
R.4.6	Visor de propiedades	Generar el visor de propiedades para modificar las características de diseño de un objeto formulario seleccionado.	Evidente
R.4.7	Crear función de Inserción	Crear el objeto y código necesario para realizar la función de inserción de registros en un formulario.	Oculta
R.4.8	Crear función de modificación	Crear el objeto y código necesario para realizar la función de modificación de registros en un formulario.	Oculta
R.4.9	Crear función de eliminación	Crear el objeto y código necesario para realizar la función de eliminación de registros en un formulario.	Oculta
R.4.10	Crear función de consulta y búsqueda de datos	Crear el objeto y código necesario para implementar la función de consulta de registros con opciones de anterior, siguiente, primero, último y búsqueda de registro por posición en la tabla.	Oculta

Ref. No.	Función	Descripción	Categoría
R.4.11	Guardar archivo	Se permitirá guardar el código fuente JAVA generado y el reporte o el formulario diseñado en la herramienta	Evidente
R.4.12	Compilar y ejecutar código JAVA	Compilar y ejecutar el código generado por la herramienta para un formulario o reporte, usando el interprete y compilador de JAVA	Evidente
R.4.13	Asistentes para la creación de formularios	Construir interfaces para guiar al usuario en la definición de características relacionadas con el formulario a crear	Evidente
R.4.14	Abrir formulario	Abrir un formulario previamente guardado y vincularlo a una base de datos existente en la herramienta	Evidente
R.4.15	Cerrar formulario	Desvincular un formulario a una base de datos existente en la herramienta	Evidente
R.4.16	Modificar formularios	Modificar un formulario existente en su diseño gráfico o en su código	Evidente
R.4.17	Cortar, Copiar, Pegar y eliminar objetos del formulario	Permite realizar operaciones de cortar, copiar, pegar y eliminar sobre los objetos de diseño del formulario	Evidente
R.5	Crear y diseñar reportes gráficos	Generar objetos y clases para la creación y diseño de reportes gráficos estándar	Evidente
R.5.1	Asistente para generación de reportes	Construir interfaces necesarias para guiar al usuario en la generación de un reporte	Evidente
R.5.2	Área de diseño del reporte	Proporcionar un área de trabajo en donde el usuario pueda diseñar la estructura y formato de un reporte	Evidente
R.5.3	Insertar objeto al diseñar un reporte	Generar una barra de herramientas donde el usuario pueda seleccionar un tipo de objeto y luego dibujar e insertar el objeto en el diseño del reporte	Evidente
R.5.4	Visor de propiedades	Generar el visor de propiedades para modificar las características de diseño de un objeto reporte seleccionado.	Evidente

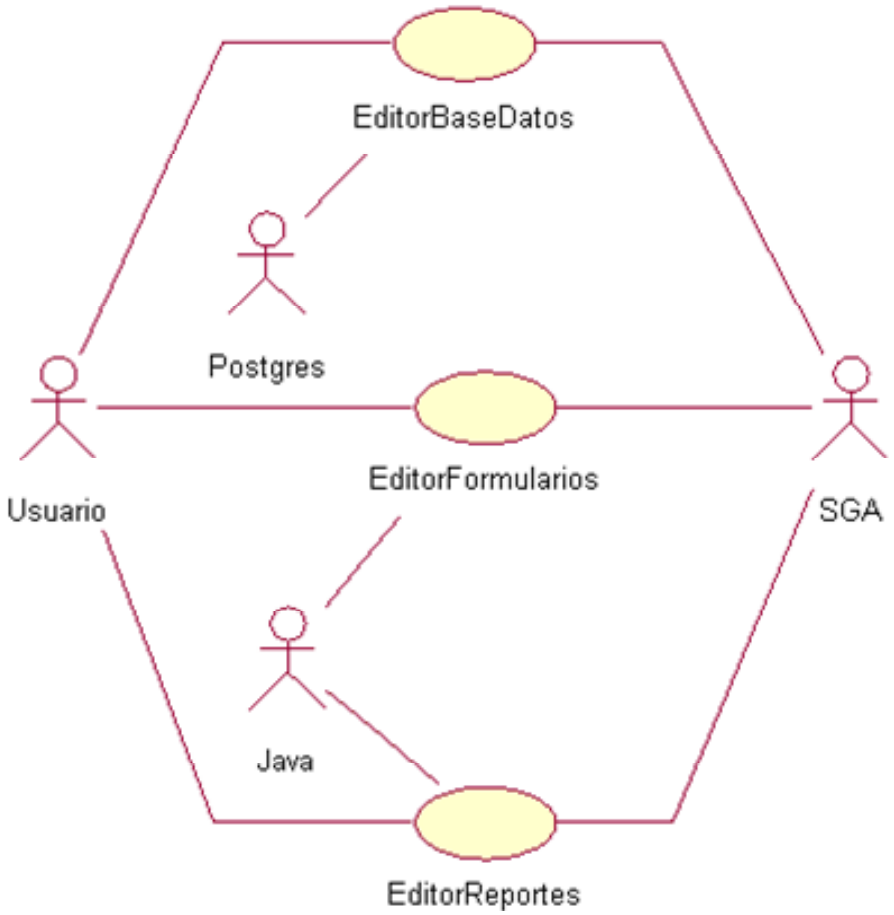
Ref. No.	Función	Descripción	Categoría
R.5.5	Generar reportes de tablas	Generar el código y los objetos necesarios para la creación de reportes vinculados a una tabla de una base de datos específica	Evidente
R.5.6	Guardar resultados de un reporte	Obtener y actualizar los datos de un reporte previamente diseñado y guardarlos en un archivo independiente en formato RTF.	Evidente
R.5.7	Generar distribución	Permitir distribuir el reporte de diferentes formas	Evidente
R.5.8	Crear una vista previa	Generar un visor por el cual pueda ser examinados el resultado del diseño y de los datos de un reporte	Oculto
R.5.9	Cerrar reportes	Desvincular el reporte de la base de datos a la cual fue asignado	Evidente
R.5.10	Modificar reportes	Modificar el diseño de la estructura de un reporte existente	Evidente
R.5.11	Creación de objetos adicionales	Generar clases de objetos con funciones especificas compatibles con la herramienta para usar en el diseño de reportes	Evidente
R.5.12	Abrir un reporte	Abrir un reporte existente previamente guardado	Evidente
R.6	Crear y cerrar conexión	Crear conexiones con el motor proporcionando un login, contraseña y nombres de base de datos y Cerrarlas posteriormente	Evidente
R.6.1	Mantener la conexión	Verificar la conexión con las bases de datos que se estén trabajando en la herramienta	Oculto
R.6.2	Capturar estructura	Capturar y actualizar la estructura de las bases de datos registradas en la herramienta	Oculto
R.7	Conexión en una red LAN	Conectar a bases de datos que se encuentran en un servidor de base de datos instalado en una red LAN	Oculto
R.8	Generar Scripts	Generar scripts con sentencias SQL estándar	Evidente
R.8.1	Ejecutar Scripts	Ejecutar scripts con sentencias SQL estándar	Evidente
R.8.2	Abrir Script	Abrir un script existente con sentencias SQL estándar	Evidente
R.8.3	Cerrar Script	Cerrar un script trabajado en la herramienta	Evidente
R.8.4	Guardar Script	Guardar un script ya existente dentro de la herramienta	Evidente
R.9	Mostrar las estructuras	Mostrar de forma rápida y fácil las estructuras capturadas y generadas en la herramienta	Evidente

Ref. No.	Función	Descripción	Categoría
R.10	Informar al usuario	Mantener al usuario informado de las operaciones que se están realizando en la herramienta	Evidente

8.1.3 Diagrama de Casos de uso. En la figura 1, se puede observar el modelo de Contexto de sistema GRFPostgres y como se relaciona con su entorno.

Figura 1. Diagrama de Casos de Uso - Modelo del Contexto de GRFPostgres

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Tabla 2	Tabla de funciones del sistema



En la figura 2 se observan los casos de uso del caso de uso Editor de Base de Datos que interactúa con el usuario y con el motor Postgres.

Figura 2A. Diagrama de Casos - Casos de uso de Editor de BD

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Tabla 2	Tabla de funciones del sistema
Figura 1	Diagrama de Casos de Uso - Modelo del Contexto de GRFPostgres

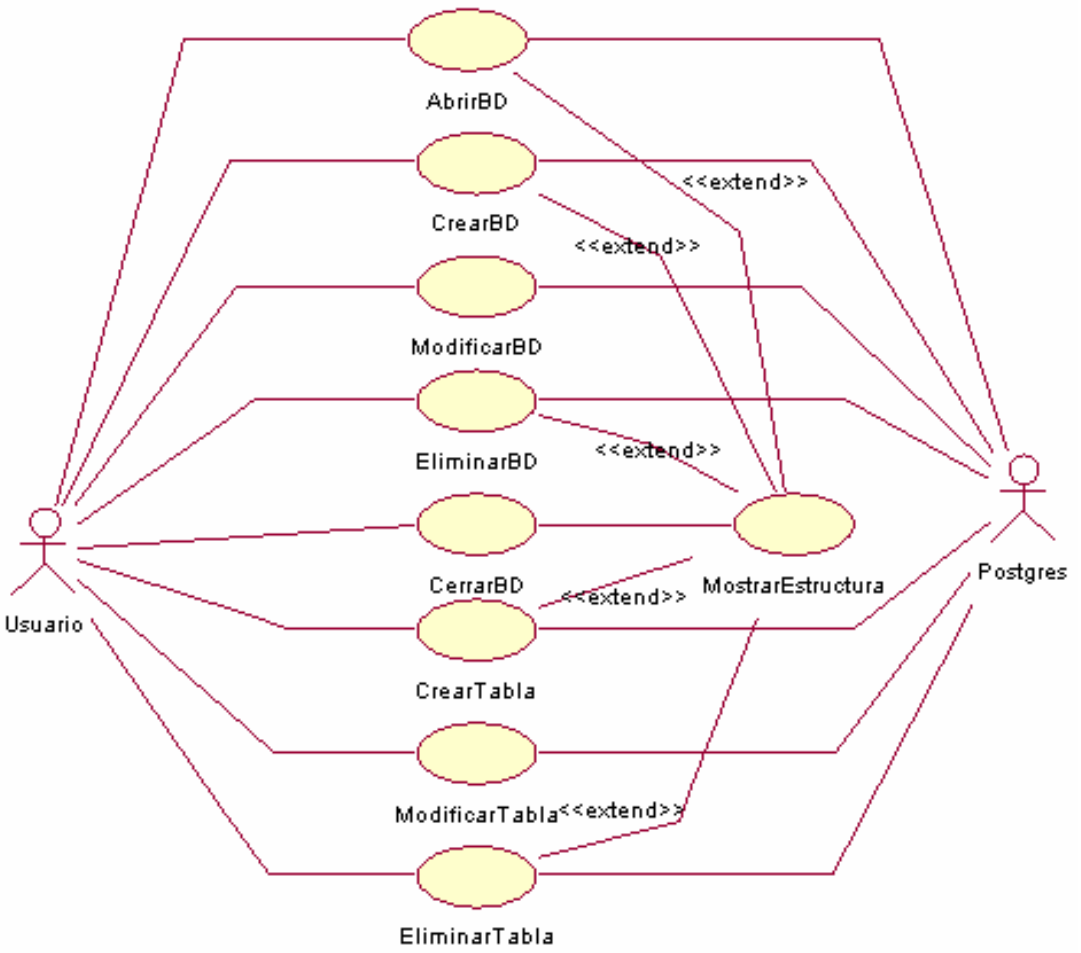
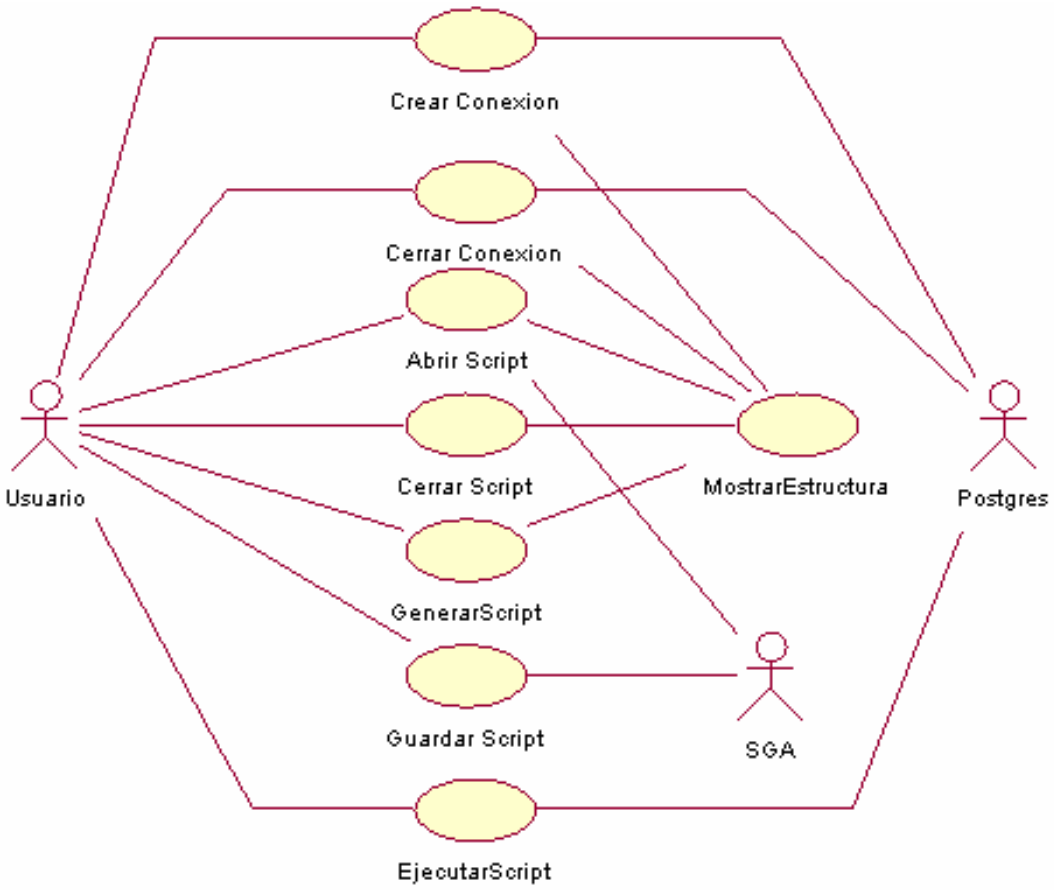


Figura 2B. Diagrama de Casos - Casos de uso de Editor de BD

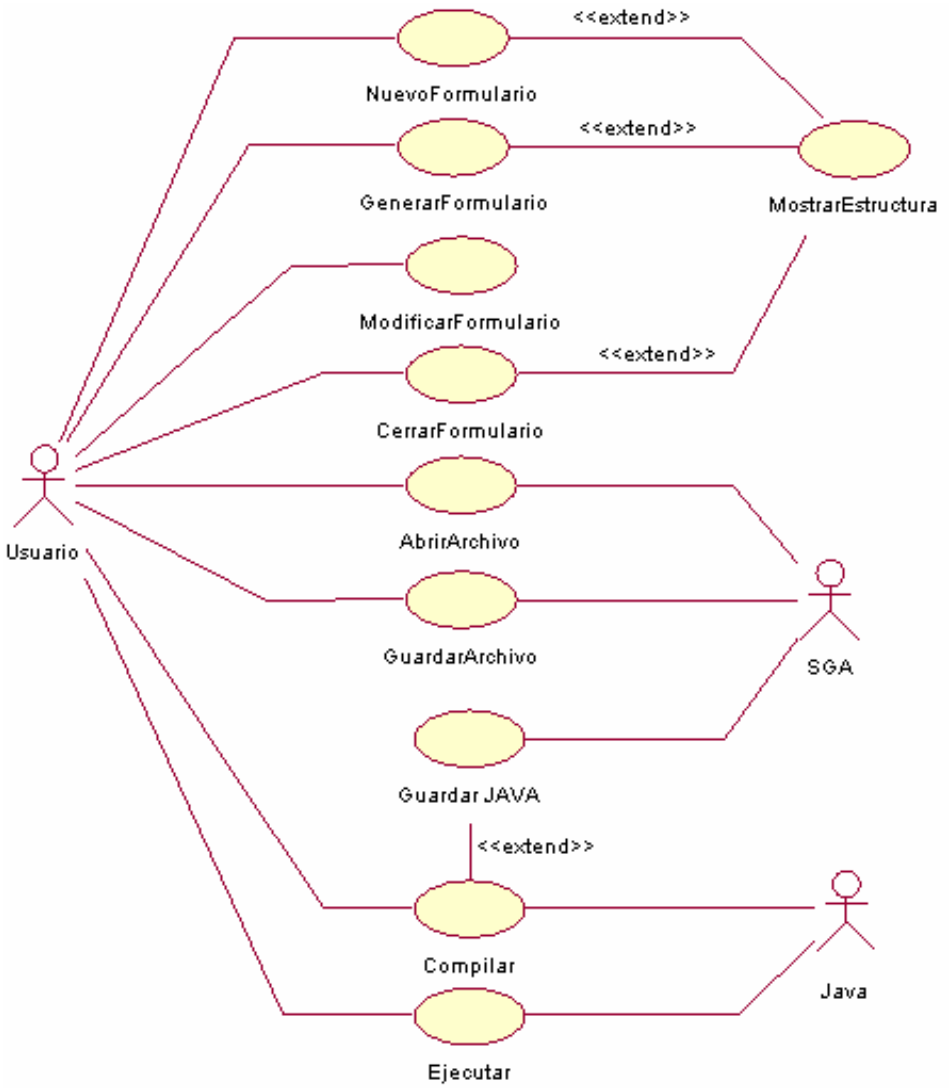
DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Tabla 2	Tabla de funciones del sistema
Figura 1	Diagrama de Casos de Uso - Modelo del Contexto de GRFPostgres



En la figura 3 se observan los casos de uso del caso de uso editor de formularios.

Figura 3. Diagrama de casos de uso - casos de uso Editor de formulario

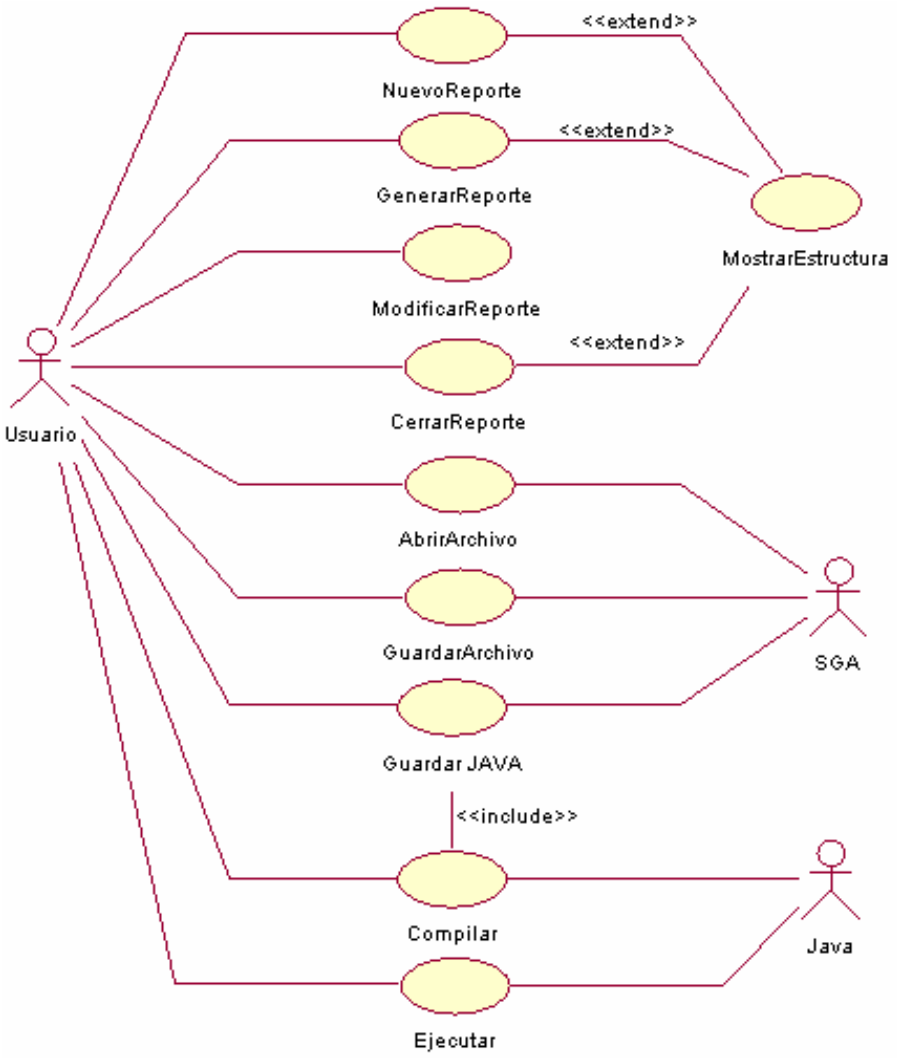
DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Tabla 2	Tabla de funciones del sistema
Figura 1	Diagrama de Casos de Uso - Modelo del Contexto de GRFPostgres



En la figura 4 se tienen los casos de uso del caso de uso del Editor de Reportes.

Figura 4. Diagrama de casos de uso - casos de uso Editor de reportes

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Tabla 2	Tabla de funciones del sistema
Figura 1	Diagrama de Casos de Uso - Modelo del Contexto de GRFPostgres



Encontrar los Actores: Se encontró que GRFPostgres interactúa tanto con usuarios avanzados como con un usuario estándar (Sin conocimientos específicos en Bases de Datos y programación), además de interactuar con el motor Postgres y con el sistema gestor de archivos del Sistema Operativo en el cual se haya implementado la herramienta.

Tabla 3. Actores que interactúan con la herramienta

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Tabla 2	Tabla de funciones del sistema
Figura 1	Diagrama de Casos de Uso - Modelo del Contexto de GRFPostgres

TIPO	PAPEL DEL ACTOR Y QUE USO LE DA A LA HERRAMIENTA
Usuario	Representa a la persona que utiliza la herramienta, con sus funciones y procedimientos. Esta persona posee conocimientos necesarios en cuanto a programación y bases de datos.
El motor (El ORDMBS Postgres)	Manejador de Base de Datos Objeto-Relacional, que permite administrar una base de datos. Postgres utiliza el lenguaje SQL como lenguaje de definición y manipulación de datos de la base de datos, junto a otras instrucciones propias del manejo del motor.
El Sistema Gestor de Archivos (SGA)	Cada sistema operativo posee un sistema gestor de archivos que se encarga de administrar como se va a almacenar los archivos en el medio físico (Disco duro, disquetes), y le asigna que características deben poseer para su lectura y escritura. Este actor proporciona todas sus funciones a la herramienta para que esta pueda almacenar archivos que produce.
Java	El código fuente generado por la herramienta puede ser compilado y ejecutado en la misma, para ello se hace uso del compilador e interprete que viene con la versión de JAVA utilizada.

Encontrar los Casos de Uso: Encontrados los requisitos candidatos y los actores que intervienen en el sistema y que interacción tienen estos con la herramienta, se determinaron y enumeraron los siguientes casos de uso:

Tabla 4. Casos de Uso encontrados que conformarán la herramienta

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Tabla 2	Tabla de funciones del sistema
Figura 1	Diagrama de Casos de Uso - Modelo del Contexto de GRFPostgres
Figura 2	Diagrama de Casos de Uso – Casos de uso Editor de Base de Datos
Figura 3	Diagrama de Casos de Uso – Casos de uso Editor de Formularios
Figura 4	Diagrama de Casos de Uso – Casos de uso Editor de Reportes

No	NOMBRE	ACTOR	REFERENCIA CRUZADA
C1	Crear Conexión	Usuario	R.6, R.6.1, R.6.2, R.7
C2	Cerrar Conexión	Usuario	R.6
C3	Abrir BD	Usuario	R.6, R.6.2
C4	Crear BD	Usuario	R.1
C5	Modificar BD	Usuario	R.2
C6	Eliminar BD	Usuario	R.3
C7	Cerrar BD	Usuario	R.3.2
C8	Crear Tabla	Usuario	R.1.1, R.1.1.1, R.1.1.2, R.1.1.3, R.1.1.4, R.2
C9	Modificar Tabla	Usuario	R.2, R.2.1, R.2.1.1, R.2.1.2, R.2.1.3, R.2.1.4
C10	Eliminar Tabla	Usuario	R.2, R.3.1, R.3.1.1, R.3.1.2, R.3.1.3, R.3.1.4
C11	Generar Script	Usuario	R.8
C12	Ejecutar Script	Usuario	R.8.1
C13	Abrir Script	Usuario	R.8.2
C14	Cerrar Script	Usuario	R.8.3
C15	Guardar Script	Usuario	R.8.4
C16	Nuevo Formulario	Usuario	R.4, R.4.1, R.4.2, R.4.3, R.4.4, R.4.5, R.4.6
C17	Generar Formulario	Usuario	R.4.1, R.4.2, R.4.3, R.4.4, R.4.5, R.4.6, R.4.7, R.4.8, R.4.9, R.4.10, R.4.13
C18	Modificar Formulario	Usuario	R.4.2, R.4.3, R.4.4, R.4.5, R.4.6, R.4.16, R.4.17

C19	Cerrar Formulario	Usuario	R.4.15
C20	Nuevo Reporte	Usuario	R.5, R.5.2, R.5.4, R.5.8
C21	Generar Reporte	Usuario	R.5.1, R.5.2, R.5.3, R.5.4, R.5.5, R.5.7, R.5.8
C22	Modificar Reporte	Usuario	R.5.2, R.5.3, R.5.4, R.5.10, R.5.11
C23	Cerrar Reporte	Usuario	R.5.9
C24	Mostrar Estructura	Usuario	R.6.2, R.9, R.10
C25	Abrir Archivo	Usuario	R.4.14, R.5.12
C26	Guardar Archivo	Usuario	R.4.11, R.5,6
C27	Guardar JAVA	Usuario	R.4.11
C28	Compilar	Usuario	R.4.12
C29	Ejecutar	Usuario	R.4.12

8.2 FLUJO DE TRABAJO DE ANÁLISIS

8.2.1 Descripción de casos de uso

CASO DE USO: CREAR CONEXIÓN

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Crear una conexión con el motor de base de datos
Resumen	El usuario inicia este caso de uso cuando desea crear una conexión proporcionando nombre de la conexión, login, ip, contraseña, puerto y un listado de base de datos que formarán parte de la conexión.
Tipo	Primario
Referencias Cruzadas	R.6, R.6.1, R.6.2, R.7

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario escribe: nombre de la conexión, login, password, ip, puerto y las bases de datos de la conexión.	2. Verifica que la conexión sea valida.
	3. Capturar las estructuras de las bases de datos de la conexión
	4. Convoca al caso de uso Mostrar Estructura

Curso alterno (subflujos) (flujos de excepción)
1. El usuario puede seleccionar una conexión anteriormente trabajada.
3. Además de capturar las estructuras captura los formularios que se trabajaron en esa conexión desde el sistema gestor de archivos

CASO DE USO: CERRAR CONEXIÓN

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Cierra una conexión establecida con el motor de base de datos
Resumen	El usuario cierra una conexión que esta registrada en la herramienta.
Tipo	Primario
Referencias Cruzadas	R.6

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario selecciona una conexión registrada y solicita cerrarla	2. Termina la conexión con el motor, y la elimina del conjunto de conexiones registradas
	3. Llama al caso de uso Mostrar Estructura

CASO DE USO: ABRIR BD

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos
INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Identificar una base de datos ya existente dentro del motor
Resumen	Este caso de uso es encargado de identificar una base de datos Postgres ya existente en una máquina.
Tipo	Primario
Referencias Cruzadas	R.6, R.6.2

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario selecciona una conexión existente en la herramienta. Solicita abrir una base de datos.	2. La herramienta verifica que la base de datos exista
	3. Captura la estructura de la base de datos
	4. Llama al caso de uso Mostrar Estructura

Curso alterno (subflujos) (flujos de excepción)
2. La base de datos especificada por el usuario no existe o no se puede establecer una conexión con la misma. 3. Se muestra un mensaje de error informando al usuario de este hecho

CASO DE USO: CREAR BD

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Crear una nueva base de datos dentro del motor con las especificaciones entregadas por el usuario
Resumen	El usuario emplea este caso de uso para crear una Base de Datos en el motor Postgres.
Tipo	Primario
Referencias Cruzadas	R.1

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario escoge la conexión por medio de la cual se creará la BD, luego solicita crear Nueva Base de Datos.	2. La herramienta lanza una interfaz que le permite al usuario especificar las características de la nueva BD a crear.
3. El usuario proporciona diferentes características de la BD a crear como: nombre, descripción y otras características avanzadas que proporciona el motor.	4. La herramienta verifica los datos suministrados. También verifica que el nombre propuesto para la Base de Datos no este utilizado por una Base de Datos ya existente en la herramienta o el motor.

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
	5. La herramienta crea la base de datos en el motor
	6. Llama al caso de uso Mostrar Estructura

Curso alterno (subflujos) (flujos de excepción)
4. Si los datos corresponden a una base de datos ya existente se informa del error por medio de un mensaje emergente y del visor de mensajes

CASO DE USO: MODIFICAR BD

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Modificar parámetros de una base de datos
Resumen	Modificar el nombre y la descripción de la base de datos
Tipo	Primario
Referencias Cruzadas	R.2

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario selecciona una base de datos a modificar	2. La herramienta lanza la interfaz de edición de base de datos
3. El usuario modifica los parámetros que desea editar	4. La herramienta verifica que los datos modificados sean validos
	5. La herramienta realiza las modificaciones a la base de datos dentro del motor
	6. Llama al caso de uso Mostrar Estructura

Curso alterno (subflujos) (flujos de excepción)
4. Los datos suministrados por el usuario no son válidos en cuyo caso se informa al usuario por medio de un mensaje de error.

CASO DE USO: ELIMINAR BD

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Eliminar una base de datos existente en el motor Postgres
Resumen	El usuario emplea este caso de uso para suprimir una Base de Datos existente en el motor. Al eliminar la Base de Datos se borran tanto la estructura como los registros que contiene.
Tipo	Primario
Referencias Cruzadas	R.3

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario selecciona la base de datos a eliminar	2. La herramienta verifica que no se trate de la base de datos principal de la conexión, si no es proporciona la interfaz correspondiente para realizar la acción.
3. El usuario confirma la eliminación tomando a riesgo la perdida de la estructura y datos de la misma	4. La herramienta realiza la operación dentro del motor
	5. La herramienta llama al caso de uso Modificar Estructura

Curso alterno (subflujos) (flujos de excepción)
3. El usuario decide no eliminar la base de datos y cancela la acción.
La herramienta no logra establecer una conexión con esta base de datos o no puede eliminarla por encontrarse protegida.

CASO DE USO: CERRAR BD

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Cerrar una base de datos registrada en la herramienta
Resumen	Permite cerrar la base de datos seleccionada de la herramienta, sin eliminarla ni modificarla dentro del motor
Tipo	Primario
Referencias Cruzadas	3.2

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario selecciona la base de datos a cerrar e inicia la acción	2. La herramienta verifica que no se trate de la base de datos principal de la conexión
	3. La herramienta quita la base de datos del conjunto de bases de datos de la conexión
	4. La herramienta llama al caso de uso Mostrar Estructura

Curso alterno (subflujos) (flujos de excepción)
2. La herramienta determina que la base de datos seleccionada para ser cerrada es la base de datos principal de la conexión, en cuyo caso informa al usuario y cancela la operación.

CASO DE USO: CREAR TABLA

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Crear una nueva tabla dentro de una base de datos existente
Resumen	Este caso de uso comienza cuando el usuario decide crear una nueva tabla, se a determinando con anterioridad una base de datos

	con la cual trabajar.
Tipo	Primario
Referencias Cruzadas	R.1.1, R.1.1.1, R.1.1.2, R.1.1.3, R.1.1.4, R.2

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario inicia este caso de uso con la opción crear tabla y seleccionando la base de datos en la cual será creada	2. La herramienta proporciona la interfaz correspondiente para realizar la acción.
3. El usuario entrega los datos de la nueva tabla, como mínimo el Nombre de la tabla.	
4. Determina la estructura de la tabla, como son: los campos que componen la misma, así como la llave primaria, las llaves foráneas y restricciones si existen.	5. La herramienta crea la nueva tabla con las especificaciones entregadas, ejecutando la sentencia de creación.
	6. Se llama a Mostrar Estructura.

Curso alterno (subflujos) (flujos de excepción)
5. No se puede crear la tabla dentro de la base de datos en el motor, ya sea por falta de permisos o por caída en la conexión con el motor, en ambos casos se informa al usuario el problema y se cancela la acción.

CASO DE USO: MODIFICAR TABLA

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Modificar una tabla ya existente en su estructura.
Resumen	Este caso de uso comienza cuando el usuario decide modificar una tabla ya existente.
Tipo	Primario
Referencias Cruzadas	R.2, R.2.1, R.2.1.1, R.2.1.2, R.2.1.3, R.2.1.4

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario elige la opción modificar tabla después de haber seleccionado una tabla	2. La herramienta proporciona la interfaz correspondiente para realizar la acción.
3. El usuario entrega los datos de la Tabla a modificar. Así como la información que desea cambiar en su estructura (campos, llave primaria o las llaves foráneas)	5. La herramienta advierte que esta operación puede ocasionar la perdida de los datos o registros actuales de la Tabla, y pide confirmación para realizar la operación de modificación.
	6. Se ejecuta las modificaciones correspondientes en el motor.
	7. Llama al caso de uso Mostrar Estructura.

Curso alterno (subflujos) (flujos de excepción)
5. El usuario cancela la operación por la posible perdida de los datos existentes. 6. No se puede realizar la operación dentro del motor, en cuyo caso se informa del error al usuario

CASO DE USO: ELIMINAR TABLA

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Eliminar una tabla ya existente dentro de una base de datos en el motor
Resumen	Este caso de uso comienza cuando el usuario decide eliminar una tabla ya existente.
Tipo	Primario
Referencias Cruzadas	R.2, R.3.1, R.3.1.1, R.3.1.2, R.3.1.3, R.3.1.4

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el usuario elige la opción “eliminar tabla”, posteriormente de haber seleccionado la tabla a eliminar.	2. Se le presenta al usuario una interfaz correspondiente con la acción.

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
3. El usuario determina el método de eliminación a utilizar con esta tabla y confirma la eliminación	4. Se elimina la tabla dentro del motor
	5. Se llama a Mostrar Estructura

Curso alterno (subflujos) (flujos de excepción)
3. El usuario decide no eliminar esta tabla, cancelando la acción a realizar. 4. La herramienta no logra establecer la conexión o ejecutar dicha sentencia de eliminación en cuyo caso se informa al usuario del error y se cancela la acción.

CASO DE USO: GENERAR SCRIPT

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Tabla 2	Tabla de funciones del sistema
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Generar Script con sentencias SQL
Resumen	En este caso de uso, por solicitud del usuario se genera un script de una base de datos o de una tabla, con las sentencias SQL necesarias para la creación de las mismas en un motor.
Tipo	Primario
Referencias Cruzadas	R.8

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario decide generar un script y selecciona una base de datos dentro de una conexión	2. La herramienta lanza una interfaz para capturar los datos necesarios en la generación de scripts
2. El usuario entrega datos como nombre del script base de datos o tabla.	3. Partiendo de los datos capturados se genera un script con las sentencias SQL necesarias.
	4. La herramienta muestra el script generado en una interfaz
	5. Llama al Caso de uso Mostrar Estructura

CASO DE USO: EJECUTAR SCRIPT

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Tabla 2	Tabla de funciones del sistema
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Ejecutar un script con sentencias SQL estándar
Resumen	El usuario selecciona el script y lo ejecuta.
Tipo	Primario
Referencias Cruzadas	R.8.1

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario elige la opción ejecutar script, previamente seleccionado.	2. La herramienta ejecuta el script sobre la base de datos y la conexión a la cual estaba enlazado
	3. Llama al caso de uso Mostrar Estructura

CASO DE USO: ABRIR SCRIPT

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Abrir un script previamente guardado
Resumen	Abre un script con sentencias SQL estándar desde el sistema gestor de Archivos
Tipo	Primario
Referencias Cruzadas	R.8.2

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario selecciona una base de datos donde será vinculado el script y llama a abrir el script	2. La herramienta lanza la interfaz correspondiente para esta acción
3. El usuario selecciona la ubicación y el script a abrir	4. La herramienta muestra el contenido de este script
5. El usuario acepta la operación	5. La herramienta crea y vincula el script a la base de datos seleccionada
	6. Muestra el script abierto en una interfaz
	7. Llama a Mostrar Estructura

CASO DE USO: CERRAR SCRIPT

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Cerrar un script registrado en la herramienta
Resumen	Cierra un script de sentencias SQL estándar que haya sido creado o abierto dentro de una base de datos registrada en la herramienta
Tipo	Primario
Referencias Cruzadas	R.8.3

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario selecciona el script a cerrar y inicia la acción	2. La herramienta desvincula el script de la base de datos
	3. Llama al caso de uso Mostrar Estructura

CASO DE USO: GUARDAR SCRIPT

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Guardar un script existente dentro de la herramienta
Resumen	Guarda un script dentro del sistema gestor de archivos, que haya sido creado o abierto dentro de una base de datos registrada en la herramienta
Tipo	Primario
Referencias Cruzadas	R.8.4

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario selecciona un script registrado en la herramienta	2. La herramienta despliega la interfaz correspondiente
3. El usuario determina la ubicación y nombre con los cuales el script será guardado	4. La herramienta guarda el script en el sistema gestor de archivos

CASO DE USO: NUEVO FORMULARIO

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 3	Diagrama de Casos de Uso – Casos de Uso Editor de Formularios

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Permite la creación de un nuevo formulario
Resumen	En este caso de uso el usuario puede diseñar desde cero su formulario. El formulario ya tiene un control enlazado a una tabla de una BD
Tipo	Primario
Referencias Cruzadas	R.4, R.4.2, R.4.3, R.4.4, R.4.5, R.4.6

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el usuario elige la opción “nuevo formulario”.	2. la herramienta despliega una interfaz para capturar los datos correspondientes al nuevo formulario

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
3. El usuario suministra la conexión, la base de datos y la tabla relacionada con el formulario, además del nombre del formulario.	4. La herramienta genera el nuevo archivo producto de esta operación, y lo muestra en una interfaz tanto el diseño como el código fuente generado, desplegándose los visores de código y propiedades respectivos.
	5. Llama a Mostrar Estructura

CASO DE USO: GENERAR FORMULARIO

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 3	Diagrama de Casos de Uso – Casos de Uso Editor de Formularios

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Generar un nuevo formulario modelo partiendo de una tabla
Resumen	Este caso de uso le sirve al usuario para generar el código fuente de un formulario modelo relacionado con una tabla de una BD existente y previamente especificada, este caso de uso interactúa con el usuario capturando las especificaciones del formulario.
Tipo	Primario
Referencias Cruzadas	R.4, R.4.1, R.4.4, R.4.5, R.4.6, R.4.7, R.4.8, R.4.9, R.4.10, R.4.1.3

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. Este caso de uso comienza cuando el usuario elige la opción “generar formulario”	2. La herramienta proporciona al usuario un asistente para la generación del formulario
3. El usuario proporciona la conexión, la BD, la tabla y los campos sobre los cuales se realizarán las operaciones sobre registros. Además, el nombre y el título del formulario.	4. La herramienta genera el código fuente y los objetos de diseño del formulario de acuerdo con los parámetros establecidos y almacena los datos del formulario en el archivo producto de la operación.

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
	5. La herramienta muestra el archivo producto de esta operación en una interfaz en la cual se puede observar su diseño y código, además de lanzar los diálogos que permiten modificar las propiedades y el código.
	6. Llama al caso de uso Mostrar Estructura

CASO DE USO: MODIFICAR FORMULARIO

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 3	Diagrama de Casos de Uso – Casos de Uso Editor de Formularios

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Modificar un formulario ya existente dentro de la herramienta ya se en diseño o programación
Resumen	El usuario inicializa este caso de uso para modificar un formulario existente ya generado, en el puede cambiar los atributos de los componentes del mismo, y adaptarlo a su gusto dentro de la herramienta con la ventaja de obtener un código JAVA generado el cual puede ser visualizado en cualquier punto de esta modificación y obtenerlo en un archivo independiente.
Tipo	Primario
Referencias Cruzadas	R.4.2, R.4.3, R.4.4, R.4.5, R.4.6, R.4.16, R.4.17

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario escoge el formulario de desea modificar.	2. La herramienta despliega la interfaz apropiada del formulario en su vista diseño y su vista código, así como los diálogos para editar propiedades y código.
3. El usuario realiza las modificaciones necesarias al objeto formulario.	4. La herramienta realiza las modificaciones dentro del código JAVA y del diseño del formulario.

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
	5. La herramienta guarda dichas modificaciones en el archivo producto de la operación.

Curso alterno (subflujos) (flujos de excepción)
3. El usuario decide dejar de modificar el formulario, guarda o lo compila para ver su ejecución.

CASO DE USO: CERRAR FORMULARIO

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 1	Diagrama de Casos de Uso – Modelo del Contexto de GRFPostgres
Figura 3	Diagrama de Casos de Uso – Casos de Uso Editor de Formularios

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Cerrar un formulario ya existente dentro de la herramienta
Resumen	El usuario inicia este caso de uso cuando desea cerrar un formulario que se genero o creo con anterioridad.
Tipo	Primario
Referencias Cruzadas	R.4.15

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario selecciona el formulario a cerrar e inicia la acción.	2. La herramienta verifica se ha modificado o si el archivo esta guardado, sino pregunta si desea guardarlo
3. El usuario confirma que desea guardarlo	4. Llama al caso de uso guardar archivo
	5. La herramienta desvincula el formulario de la base de datos a la cual esta enlazado
	6. La herramienta llama a mostrar Estructura

Curso alterno (subflujos) (flujos de excepción)
3. El usuario decide no guardar el formulario 4. Se salta al paso 5.

CASO DE USO: NUEVO REPORTE

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Crear un nuevo reporte
Resumen	Crear un reporte en blanco que ya se encuentre relacionado con una tabla de una base de datos registrada en la herramienta
Tipo	Primario
Referencias Cruzadas	R.5, R.5.2, R.5.4, R.5.8

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario solicita crear un nuevo reporte	2. La herramienta lanza la interfaz correspondiente
3. El usuario proporciona los datos del reporte (conexión, base de datos, tabla), además del nombre y titulo del reporte	4. La herramienta genera el archivo producto de esta operación
	5. Muestra el reporte en la interfaz de vista de diseño, código y previa, además del dialogo de propiedades
	6. Llama a mostrar estructura

CASO DE USO: GENERAR REPORTE

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 3	Diagrama de Casos de Uso – Casos de Uso Editor de Reportes

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Generar un reporte partiendo de una tabla existente
Resumen	El usuario emplea este caso de uso para generar el código fuente

	de un reporte modelo relacionado con una tabla de una BD registrada.
Tipo	Primario
Referencias Cruzadas	R.5.1, R.5.2, R.5.3, R.5.4, R.5.5, R.5.7, R.5.8

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario escoge la opción generar reporte.	2. la herramienta proporciona el asistente para la generación de reportes
3. El usuario proporciona la conexión, la BD, la tabla, los campos del reporte, además de datos distribución y formato.	
	4. La herramienta genera el código fuente y los objetos de diseño del reporte de acuerdo con los parámetros establecidos y almacena los datos en un archivo producto, además de mostrarlo en una interfaz que permite una vista de diseño, previa y código.
	5. Llama a Mostrar Estructura

CASO DE USO: MODIFICAR REPORTE

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 3	Diagrama de Casos de Uso – Casos de Uso Editor de Reportes

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Modificar un reporte existente.
Resumen	El usuario inicializa este caso de uso para modificar un reporte existente ya generado, en el puede cambiar los atributos de los componentes del mismo, y adaptarlo a su gusto dentro de la herramienta con la ventaja de obtener un código JAVA generado el cual puede ser visualizado en cualquier punto de esta modificación y obtenerlo en un archivo independiente, además de una vista previa del reporte.
Tipo	Primario
Referencias Cruzadas	R.5.2, R.5.3, R.5.4, R.5.10, R.5.11

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario escoge la opción modificar reporte, eligiendo el reporte a modificar.	2. La herramienta convoca al caso de uso mostrar reporte que proporciona las vistas de diseño, previa y código para que el usuario pueda realizar sus modificaciones y verlas reflejadas en el código JAVA.
3. El usuario realiza las modificaciones necesarias al objeto reporte.	4. La herramienta realiza las modificaciones dentro del código JAVA del reporte y del objeto.

CASO DE USO: CERRAR REPORTE

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 3	Diagrama de Casos de Uso – Casos de Uso Editor de Reportes

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Cerrar un reporte existente.
Resumen	El usuario inicia este caso de uso cuando desea cerrar un objeto reporte que se genero o creo con anterioridad.
Tipo	Primario
Referencias Cruzadas	R.5.9

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario selecciona el reporte que desea cerrar.	2. La herramienta verifica que el reporte se encuentra modificado o guardado, de lo contrario pregunta si desea guardarlo
3. El usuario emite una confirmación.	4. La herramienta guarda el archivo producto.
	5. Desvincula del conjunto de reportes de la base de datos a la cual se encontraba enlazado.
	6. Llama a Mostrar Estructura

CASO DE USO: MOSTRAR ESTRUCTURA

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Tabla 2	Tabla de funciones del sistema
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos
Figura 3	Diagrama de Casos de Uso – Casos de Uso Editor de Formulario
Figura 4	Diagrama de Casos de Uso – Casos de Uso Editor de Reportes

INFORMACIÓN GENERAL	
Actores	
Propósito	Mostrar una versión de las estructuras que se están manejando en el sistema como conexiones, bases de datos, tablas y de los objetos que se estén diseñando scripts, formularios y reportes
Resumen	Por medio de este caso de uso el usuario tiene una vista rápida y sencilla de las conexiones, bases de datos y tablas que se están trabajando además de los scripts, formularios y reportes; además de cómo están enlazadas estas estructuras entre ellas. Por medio de esta vista también puede acceder de forma rápida a las opciones más comunes para cada tipo de objeto con un simple clic derecho.
Tipo	Primario
Referencias Cruzadas	R.6.2, R.9, R.10

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
	1. Por medio de un árbol ubicado en la parte izquierda de la ventana principal, muestra al usuario a través de nodos los elementos registrados.
	2. Pone a disposición del usuario un menú emergente con las opciones más comunes en cada uno de los objetos
3. El usuario pide la ejecución de una opción	4. Se ejecuta la operación solicitada

CASO DE USO: ABRIR ARCHIVO

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Abrir un archivo existente
Resumen	Abrir un archivo producto anteriormente guardado que puede ser un formulario o un reporte
Tipo	Primario
Referencias Cruzadas	R.4.14, R.5.12

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario selecciona la base de datos a la cual desea enlazar el archivo producto a abrir	2. La herramienta lanza la interfaz adecuada para esta operación
3. El usuario entrega datos como nombre del archivo y su ubicación dentro del sistema gestor de archivos	4. La herramienta abre el archivo y lo vincula a la base de datos seleccionada
	5. Muestra el archivo producto en la interfaz correspondiente
	6. Llama a Mostrar Estructura

Curso alterno (subflujos) (flujos de excepción)
4. La herramienta verifica que archivo ya se encuentra abierto en cuyo caso pregunta si desea volver a la versión guardada

CASO DE USO: GUARDAR ARCHIVO

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Formularios
Figura 3	Diagrama de Casos de Uso – Casos de Uso Editor de Reportes

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Almacenar un archivo producto ya sea un formulario o un reporte
Resumen	El usuario inicia este caso de uso cuando desea guardar el código fuente de estos elementos de forma independiente a la herramienta.
Tipo	Primario
Referencias Cruzadas	R.4.11, R.5.6

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario selecciona el archivo producto a guardar	2. La herramienta lanza la interfaz adecuada para esta operación
3. El usuario entrega datos como nombre del archivo y su ubicación dentro del sistema gestor de archivos	4. La herramienta guarda el archivo producto de acuerdo con la información suministrada por el usuario

CASO DE USO: GUARDAR JAVA

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Base de Datos

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Guardar el código fuente JAVA de un archivo Producto
Resumen	Guardar este código fuente de un formulario o un reporte
Tipo	Primario
Referencias Cruzadas	R.4.11

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario selecciona el archivo producto a guardar	2. La herramienta lanza la interfaz correspondiente
3. El usuario entrega datos como nombre del archivo y ubicación en el sistema gestor de archivos	4. La herramienta guarda el código JAVA del archivo producto seleccionado de acuerdo con los datos suministrados por el usuario

CASO DE USO: COMPILAR

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Tabla 2	Tabla de funciones del sistema
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Formularios
Figura 3	Diagrama de Casos de Uso – Casos de Uso Editor de Reportes

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Compilar un archivo Java correspondiente a un formulario o reporte
Resumen	El usuario utiliza este caso de uso para obtener una versión compilada del objeto que esta diseñando ya sea un formulario o un reporte, obteniendo un archivo independiente ejecutable de este objeto.
Tipo	Primario
Referencias Cruzadas	R.4.12

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario escoge la opción compilar	2. La herramienta verifica si el objeto se encuentra guardado de lo contrario lanza el caso de uso Guardar JAVA.
	3. La herramienta envía el archivo Java al compilador de Java para que realice la compilación
	4. Se ubica el archivo .class con el mismo nombre del java y en la misma ubicación

Curso alterno (subflujos) (flujos de excepción)
3. Si el compilador Java encuentra un error en archivo java se retorna este error a la herramienta y esta lo muestra al usuario

CASO DE USO: EJECUTAR

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Tabla 2	Tabla de funciones del sistema
Figura 2	Diagrama de Casos de Uso – Casos de Uso Editor de Formularios
Figura 3	Diagrama de Casos de Uso – Casos de Uso Editor de Reportes

INFORMACIÓN GENERAL	
Actores	Usuario
Propósito	Ejecutar un archivo compilado Java (.class) de un formulario o reporte
Resumen	El usuario utiliza este caso de uso para obtener una versión ejecutable del archivo producto que esta diseñando ya sea un formulario o un reporte.
Tipo	Primario
Referencias Cruzadas	R.4.12

Curso Normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario escoge la opción ejecutar	2. La herramienta verifica la existencia del archivo .class del objeto que se esta diseñando, el cual puede ser un formulario o un reporte
	3. La herramienta partiendo de este archivo compilado lo envía a ejecutar al intérprete de java, mostrando la versión en ejecución del objeto.

Curso alterno (subflujos) (flujos de excepción)
2. Si no se encuentra compilado la herramienta informa al usuario de esta situación. 3. Si existe algún problema con el archivo ejecutable (.class) java retorna el error y la herramienta lo trasmite al usuario

8.2.2 Modelo Conceptual

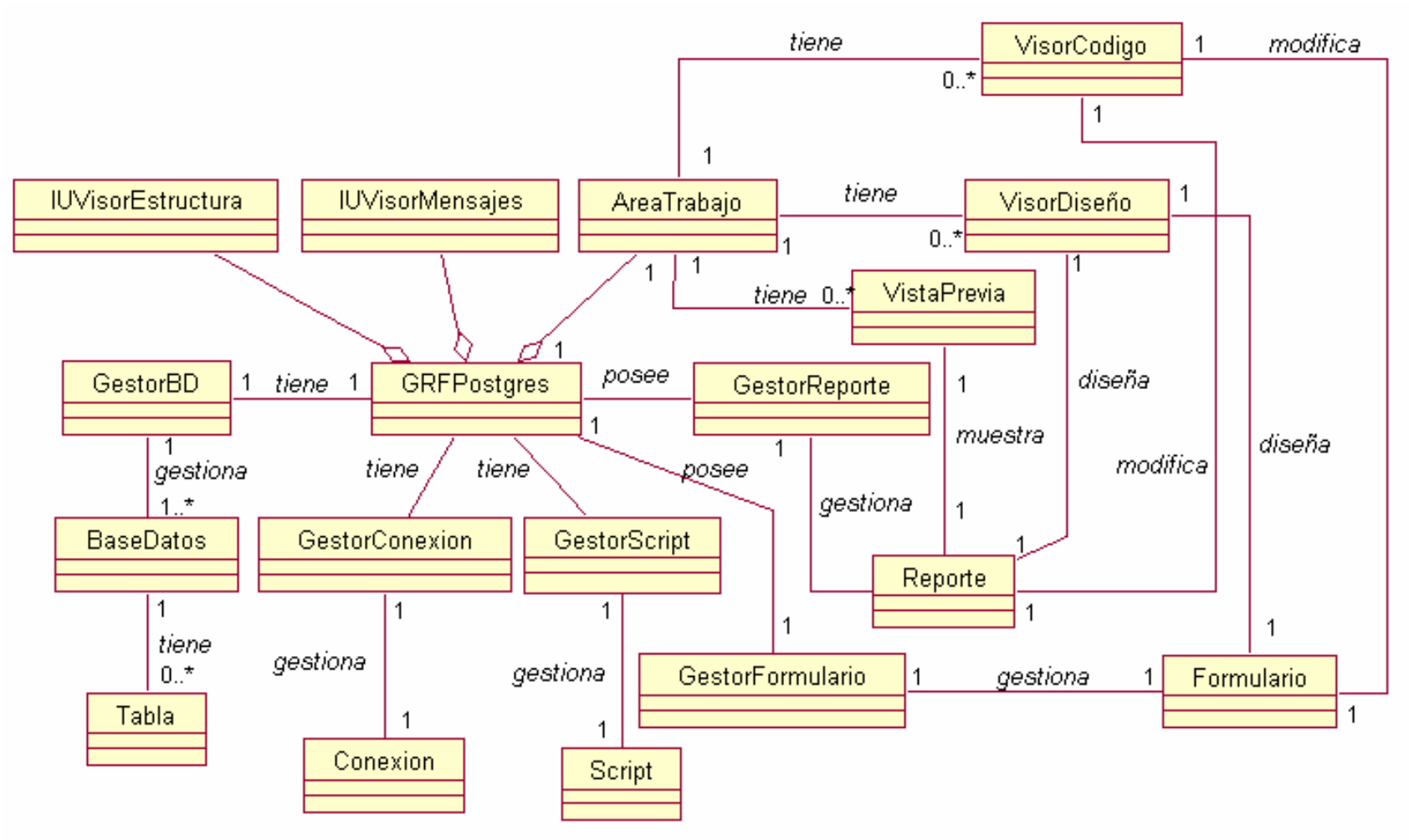
DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Tabla 2	Tabla de funciones del sistema
Figura 1	Diagrama de Casos de Uso – Modelo de Contexto GRFPostgres
Figura 2	Diagrama de Casos de Uso – Casos de uso Editor de Bases de Datos
Figura 3	Diagrama de Casos de Uso – Casos de uso Editor de Formularios
Figura 4	Diagrama de Casos de Uso – Casos de uso Editor de Reportes

Tabla 5. Listado de conceptos – Modelo Conceptual

Concepto	Descripción
VisorMensajes	Es la entidad encargada de servir como interfaz de emisión de mensajes de la herramienta
VisorEstructura	Muestra las estructuras actualmente utilizadas en el sistema (conexiones, bases de datos, tablas, scripts, formularios y reportes)
AreaTrabajo	Representa un área contenedora de diferentes elementos de trabajo dentro de la herramienta
Interfaz GRFPostgres	Interfaz principal intermediaria entre el usuario y la herramienta, establece una comunicación constante entre los mismos, a través de la cual se realizan todas las operaciones descritas.
GestorConexion	Entidad encargada de gestionar las conexiones que se estén trabajando en la herramienta.
GestorBD	Entidad encargada de gestionar todas las operaciones relacionadas con las bases de datos dentro del sistema
Gestor Script	Entidad encargada de gestionar los scripts que estén dentro de la herramienta
GestorFormulario	Responsable de las operaciones y el control sobre los objetos formularios
GestorReporte	Entidad que se encarga de gestionar el manejo y operaciones que se realicen sobre los reportes.
Conexión	Entidad que guarda y representa la conexión que se establece con el motor.
BaseDatos	Entidad encargada de contener los datos de una base de datos con la cual se este trabajando dentro del sistema
Tabla	Contiene los datos y la estructura de una tabla real en el motor

Concepto	Descripción
Script	Contiene los datos de un script con sentencias SQL estándar
Formulario	Objeto que encapsula todas las características, atributos y métodos de un formulario dentro de la herramienta
Reporte	Objeto que encapsula todas las características, atributos y métodos de reporte dentro de la herramienta
VisorDiseño	Espacio contenedor del objeto que se este actualmente diseñando el cual puede ser un formulario o un reporte
VisorCodigo	Área donde puede verse el código fuente Java del objeto a modificar, ya sea un formulario o un reporte
VistaPrevia	Espacio que permite una vista general de un reporte que se esta diseñando actualmente dentro del sistema.

Figura 5. Modelo Conceptual de GRFPostgres

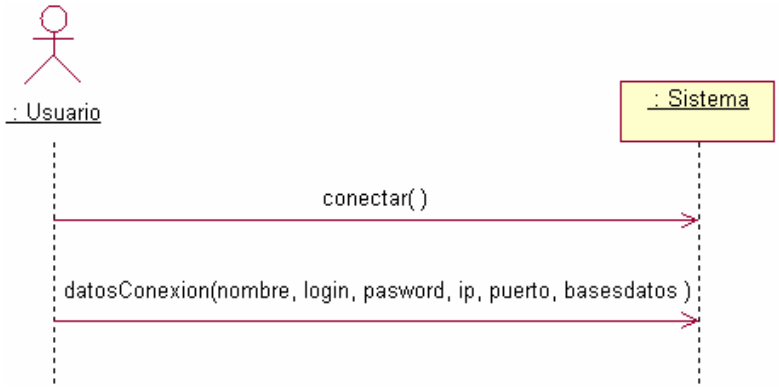


8.2.3 Diagramas de Secuencia del Sistema

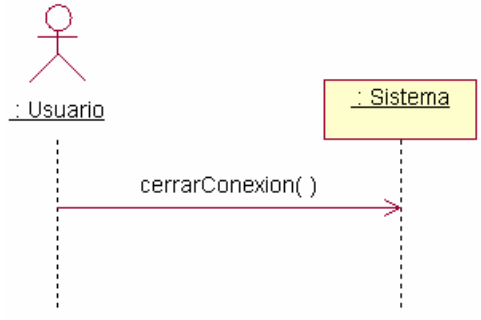
DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Tabla 2	Tabla de funciones del sistema
Figura 1	Diagrama de Casos de Uso – Modelo de Contexto GRFPostgres
Figura 2	Diagrama de Casos de Uso – Casos de uso Editor de Bases de Datos
Figura 3	Diagrama de Casos de Uso – Casos de uso Editor de Formularios
Figura 4	Diagrama de Casos de Uso – Casos de uso Editor de Reportes
Figura 5	Modelo Conceptual GRFPostgres

Figura 6. Diagramas de secuencia del sistema

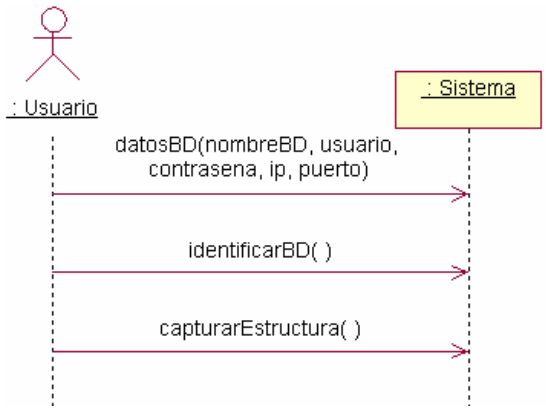
Caso de Uso. Crear Conexión



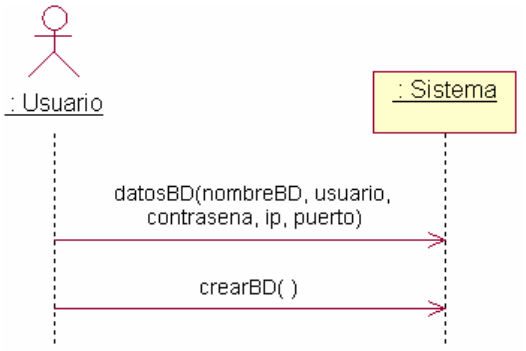
Caso de Uso. Cerrar Conexión



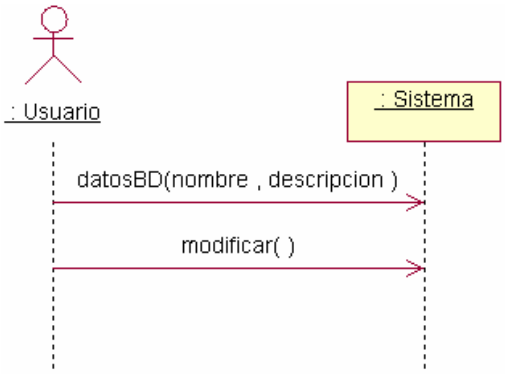
Caso de Uso. Abrir BD



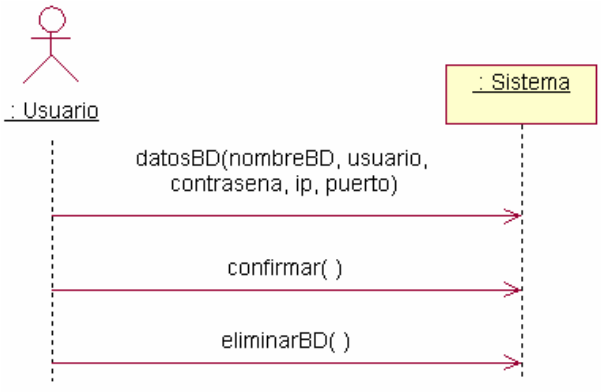
Caso de Uso. Crear BD



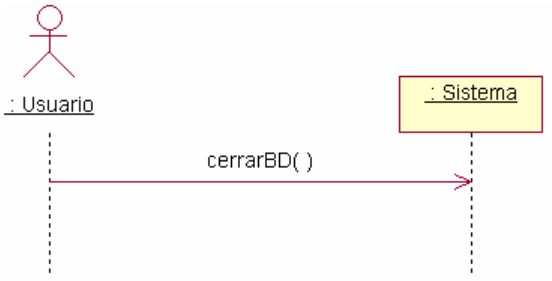
Caso de Uso. Modificar BD



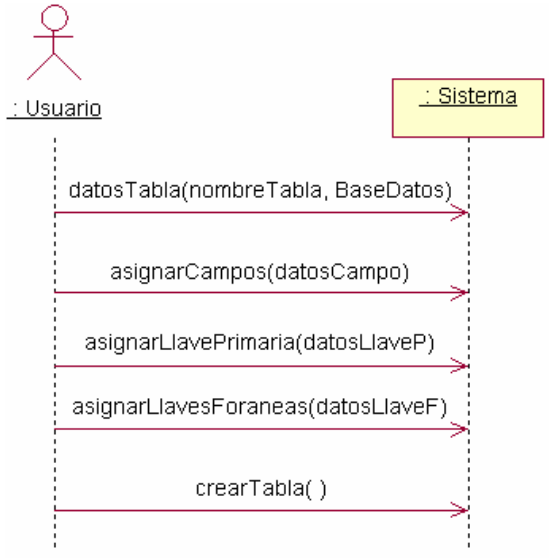
Caso de Uso. Eliminar BD



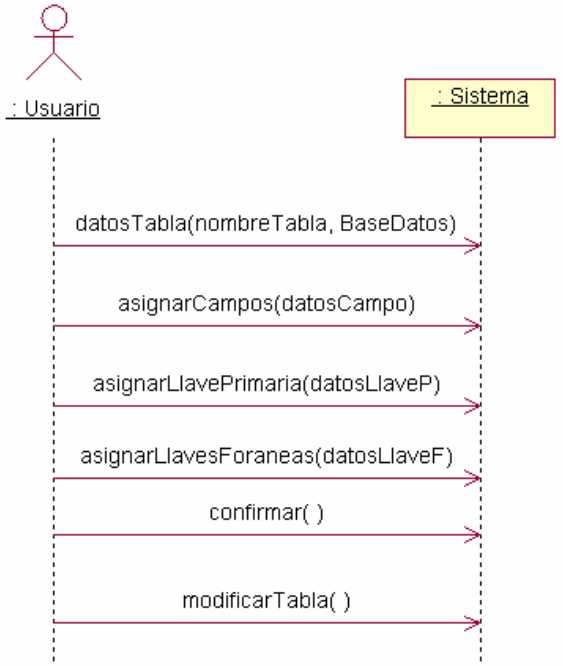
Caso de Uso. Cerrar BD



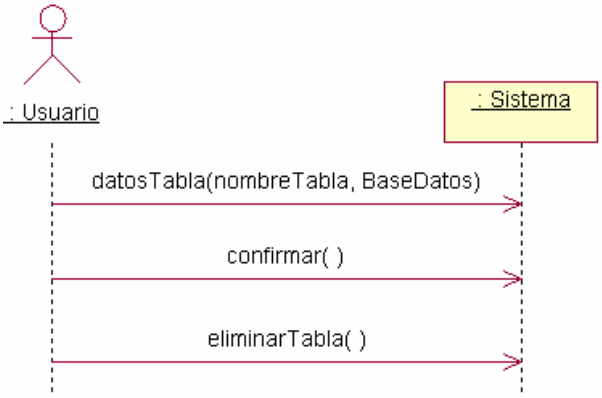
Caso de Uso. Crear Tabla



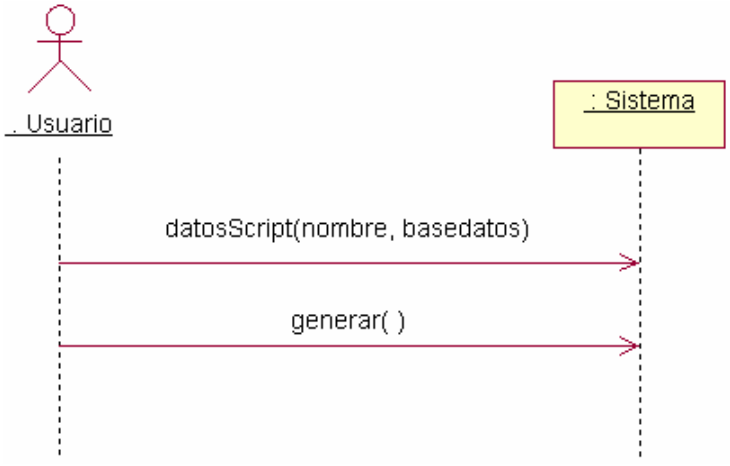
Caso de Uso. Modificar Tabla



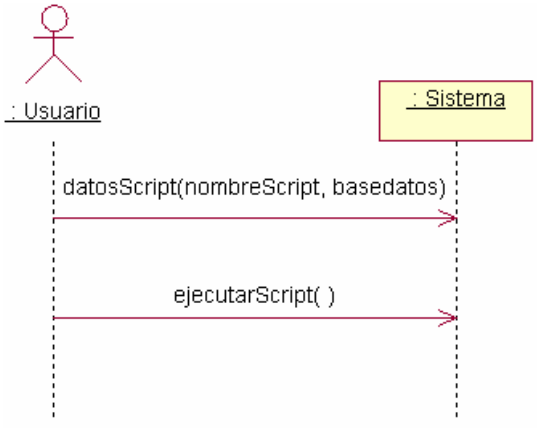
Caso de Uso. Eliminar Tabla



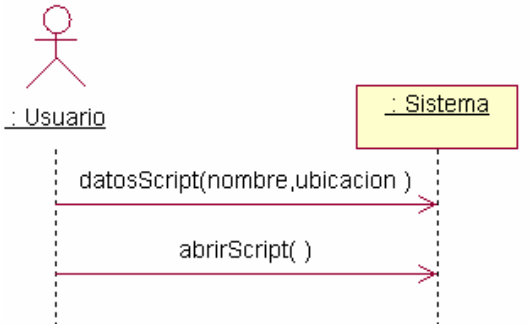
Caso de Uso. Generar Script



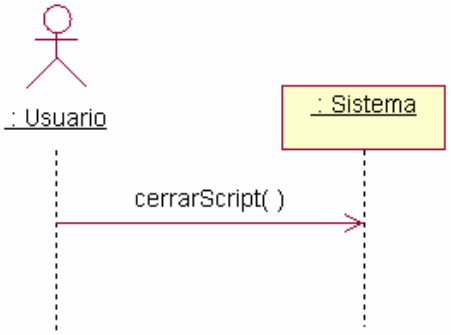
Caso de Uso. Ejecutar Script



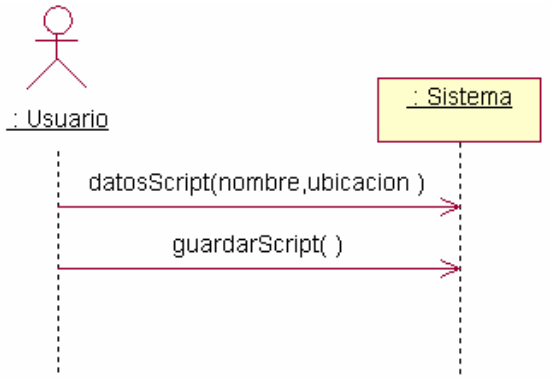
Caso de Uso. Abrir Script



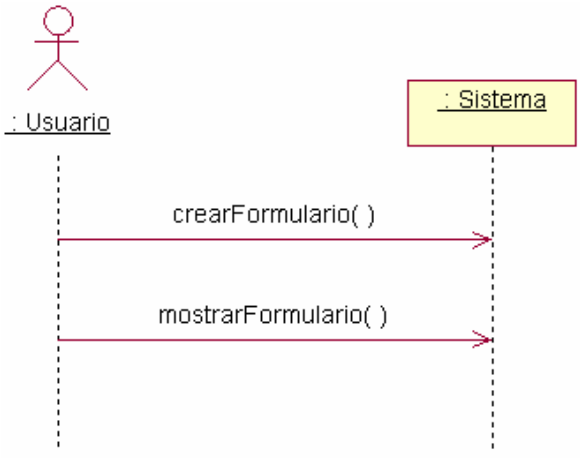
Caso de Uso. Cerrar Script



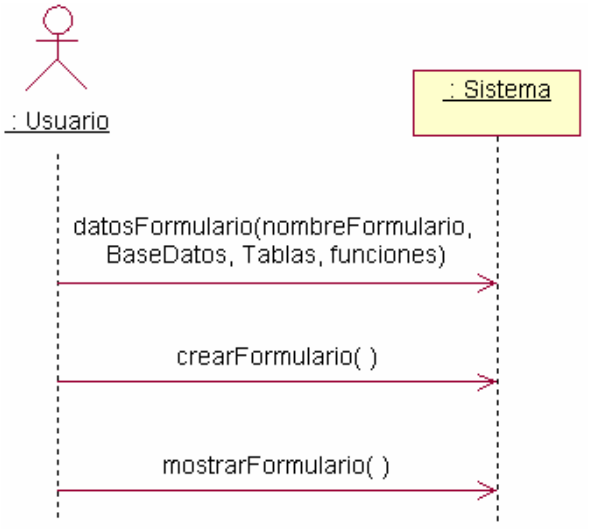
Caso de Uso. Guardar Script



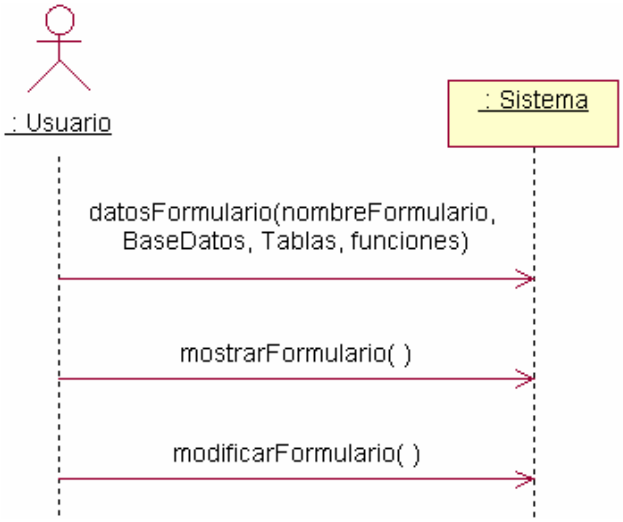
Caso de Uso. Nuevo Formulario



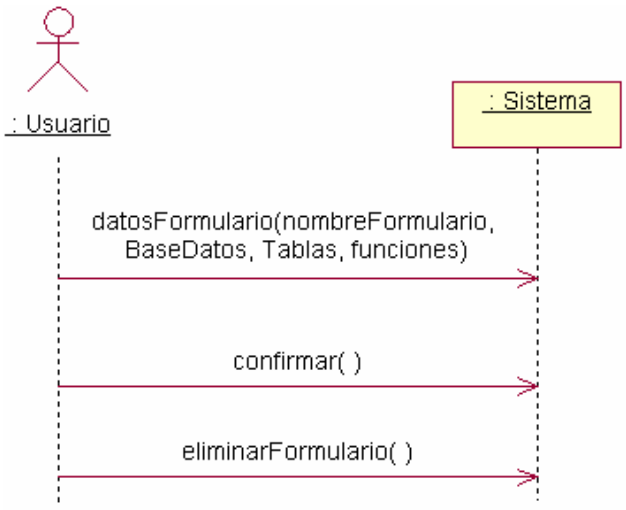
Caso de Uso. Generar Formulario



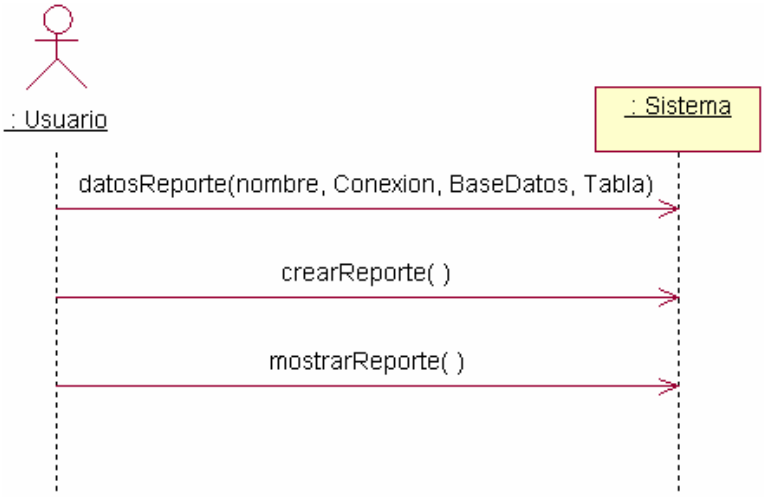
Caso de Uso. Modificar Formulario



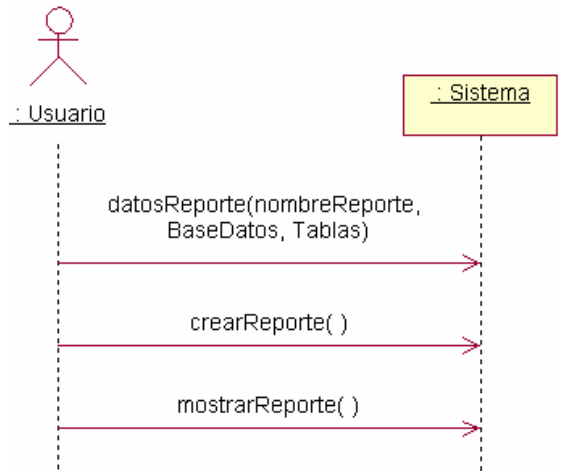
Caso de Uso. Cerrar Formulario



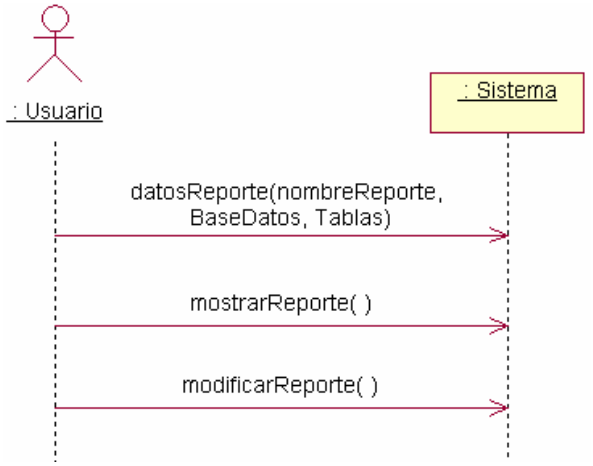
Caso de Uso. Nuevo Reporte



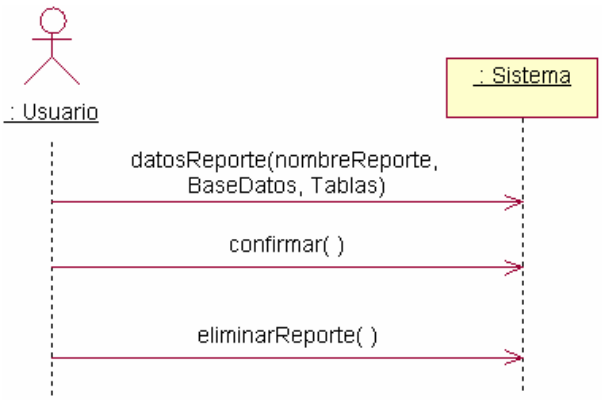
Caso de Uso. Generar Reporte



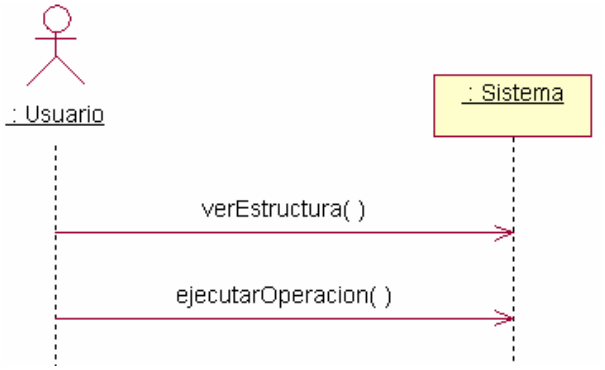
Caso de Uso. Modificar Reporte



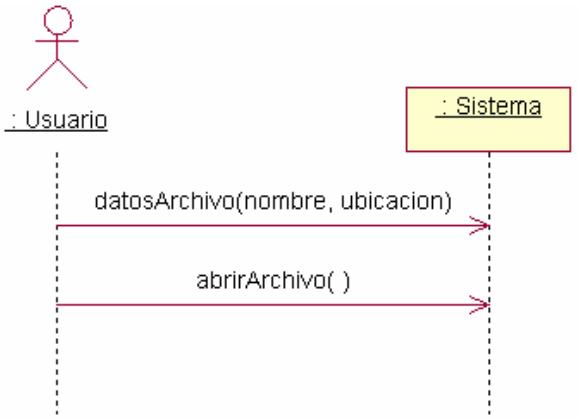
Caso de Uso. Cerrar Reporte



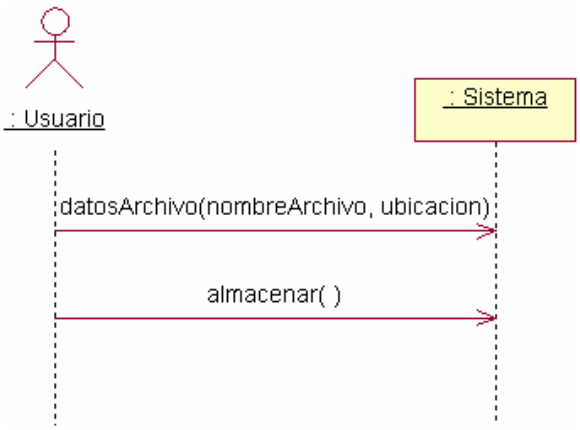
Caso de Uso. Mostrar Estructura



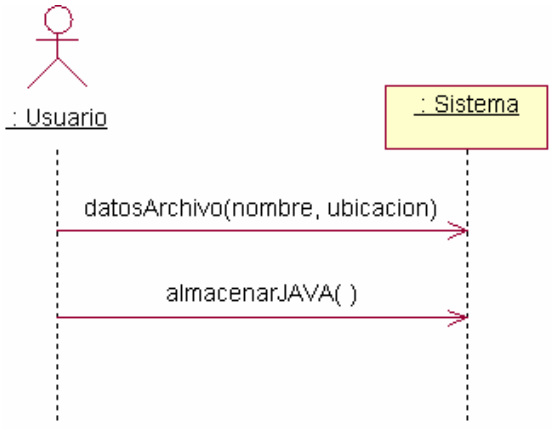
Caso de Uso. Abrir Archivo



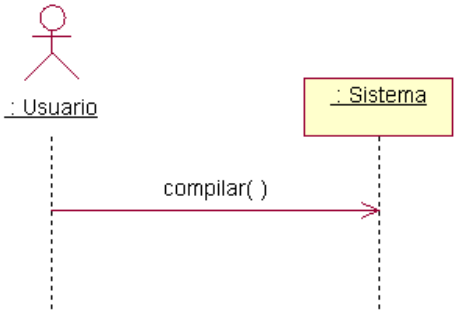
Caso de Uso. Guardar Archivo



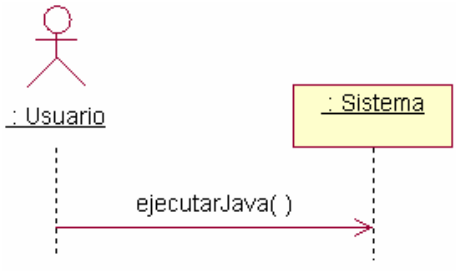
Caso de Uso. Guardar JAVA



Caso de Uso. Compilar



Caso de Uso. Ejecutar



Registro de las Operaciones del Sistema

Sistema
datosBD(nombreBD, usuario, contrasena, ip, puerto)
identificarBD()
capturarEstructura()
eliminarBD()
confirmar()
crearTabla()
datosTabla(nombreTabla, BaseDatos)
asignarLlavePrimaria(datosLlaveP)
asignarCampos(datosCampo)
asignarLlavesForaneas(datosLlaveF)
crearBD()
modificarTabla()
eliminarTabla()
crearFormulario()
datosFormulario(nombreFormulario, Conexion, BaseDatos, Tablas)
mostrarFormulario()
modificarFormulario()
eliminarFormulario()
datosReporte(nombreReporte, Conexion, BaseDatos, Tablas)
mostrarReporte()
modificarReporte()
eliminarReporte()
datosArchivo(nombreArchivo, ubicacion)
almacenar()
compilar()
ejecutarScript()
verEstructura()
ejecutarOperacion()
datosScript(nombreScript, BaseDatos)
generar()
especificacionesScript()
ejecutarJava()
crearReporte()
conectar()
datosConexion(nombre, login, password, ip, puerto, basesdatos)
cerrarConexion()
modificarBD()
cerrarBD()
abrirScript()
cerrarScript()
guardarScript()
abrirArchivo()
almacenarJAVA()

8.2.4 Contratos de operación

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Tabla 2	Tabla de funciones del sistema
Figura 1	Diagrama de Casos de Uso – Modelo de Contexto GRFPostgres
Figura 2	Diagrama de Casos de Uso – Casos de uso Editor de Bases de Datos
Figura 3	Diagrama de Casos de Uso – Casos de uso Editor de Formularios
Figura 4	Diagrama de Casos de Uso – Casos de uso Editor de Reportes
Figura 5	Modelo Conceptual GRFPostgres
Figura 6	Diagramas de Secuencia del Sistema

Tabla 6. Contratos de operación del sistema

Nombre:	datosBD(nombreBD, usuario, contraseña, ip, puerto)
Responsabilidades:	Captura los datos de una base de datos (nombre, máquina y usuario)
Tipo:	Sistema
Referencias Cruzadas:	
Notas:	Son capturados a través de una interfaz de usuario
Excepciones:	
Salida:	
Precondiciones:	
Poscondiciones:	

Nombre:	identificarBD()
Responsabilidades:	Reconocer y capturar las características de una base de datos ya existente dentro del motor
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.6, R.6.2, R.7
Notas:	
Excepciones:	Problemas de conexión con dicha base de datos
Salida:	
Precondiciones:	Se debe conocer previamente la base de datos
Poscondiciones:	Una base de datos identificada en el sistema

Nombre:	capturarEstructura()
Responsabilidades:	Capturar la estructura completa (tablas, campos, llaves primarias y foráneas) de una base de datos identificada en el sistema
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.6.2
Notas:	La estructura capturada es guardada en forma de objetos enlazados de forma lógica. Una base de datos posee un conjunto de tablas las cuales poseen campos, llaves primarias y foráneas.
Excepciones:	No se pueda establecer una conexión estable con esta base de datos
Salida:	
Precondiciones:	
Poscondiciones:	Un estructura capturada y guardada

Nombre:	eliminarBD()
Responsabilidades:	Eliminar una base de datos existente tanto en el motor como en la herramienta
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.3
Notas:	
Excepciones:	Cuando la base de datos ha eliminar se encuentra protegida o ocupada por otra aplicación
Salida:	
Precondiciones:	La base de datos existe y el usuario tiene permisos para realizar dicha operación
Poscondiciones:	Se elimina la base de datos del motor Se elimina el objeto base de datos de la herramienta Se elimina toda la estructura de dicha base de datos

Nombre:	confirmar()
Responsabilidades:	Esperar del usuario una confirmación para la realización de alguna acción critica dentro de la herramienta
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.10
Notas:	Se confirma por medio de una ventana diálogo
Excepciones:	
Salida:	
Precondiciones:	
Poscondiciones:	

Nombre:	crearTabla()
Responsabilidades:	Crear una tabla en el motor y en la herramienta
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.1.1, R.2
Notas:	
Excepciones:	El usuario no tiene permisos de creación de tablas
Salida:	
Precondiciones:	Existe una base de datos previamente registrada en el sistema en la cual se creará la base de datos
Poscondiciones:	Se crea la tabla en el motor Se crea un objeto Tabla

Nombre:	datosTabla(nombreTabla, BaseDatos)
Responsabilidades:	Capturar los datos de una tabla como son su nombre y la base de datos a la cual pertenece
Tipo:	Sistema
Referencias Cruzadas:	
Notas:	Son capturados a través de una interfaz de usuario o una selección del objeto
Excepciones:	
Salida:	
Precondiciones:	
Poscondiciones:	

Nombre:	asignarLlavePrimaria(datosLlaveP)
Responsabilidades:	Capturar la llave primaria que el usuario asigna a una tabla específica
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.1.1.2, R.2.1.2, R.3.1.2
Notas:	Lanzando una interfaz, el usuario puede determinar la llave primaria que desea asignar a la tabla seleccionada
Excepciones:	
Salida:	
Precondiciones:	Se tiene una tabla seleccionada
Poscondiciones:	

Nombre:	asignarCampos(datosCampo)
Responsabilidades:	Capturar tanto los campos como sus características, que pertenecen a una tabla específica
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.1.1.1, R.1.1.4, R.2.1.1, R.2.1.4, R.3.1.1, R.3.1.4
Notas:	Se proporciona una interfaz, en la cual el usuario entrega los datos de cada uno de los campos
Excepciones:	
Salida:	
Precondiciones:	Se ha seleccionado una tabla
Poscondiciones:	

Nombre:	asignarLlavesForaneas(datosLlaveF)
Responsabilidades:	Asigna una o varias llaves foráneas a una tabla específica
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.1.1.3, R.2.1.3, R.3.1.3
Notas:	El usuario asigna las llaves foráneas de una tabla a través de una interfaz en la cual puede determinar las características de las mismas
Excepciones:	
Salida:	
Precondiciones:	Una tabla se encuentra seleccionada previamente
Poscondiciones:	

Nombre:	crearBD()
Responsabilidades:	Crear una nueva base de datos dentro del sistema
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R1
Notas:	Se crea tanto en el motor como en la herramienta
Excepciones:	
Salida:	
Precondiciones:	Se ha identificado una base de datos previamente
Poscondiciones:	Se crea una base de datos dentro del motor Se crea un objeto base de datos en la herramienta

Nombre:	modificarTabla()
Responsabilidades:	Modificar una tabla existente dentro de una base de datos en el motor

Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.2, R.2.1
Notas:	
Excepciones:	No se puede modificar la tabla por falta de permisos
Salida:	
Precondiciones:	La tabla ya existe previamente en el motor
Poscondiciones:	Se modificar la tabla dentro del motor Se modificar el objeto tabla dentro de la herramienta

Nombre:	eliminarTabla()
Responsabilidades:	Eliminar una tabla existente tanto en el motor como en la herramienta
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.2, R.3.1
Notas:	
Excepciones:	La tabla se encuentra protegida dentro del motor
Salida:	
Precondiciones:	La tabla ya existe previamente en el motor
Poscondiciones:	Se elimina la tabla del motor Se destruye el objeto tabla de la herramienta

Nombre:	crearFormulario()
Responsabilidades:	Crear un nuevo formulario dentro de la herramienta
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.4, R.4.1, R.4.7, R.4.8, R.4.9, R.4.10, R.4.13
Notas:	Este formulario es generado por la herramienta a partir de especificaciones entregadas por el usuario
Excepciones:	
Salida:	
Precondiciones:	
Poscondiciones:	Se muestra este objeto formulario

Nombre:	datosFormulario(nombreFormulario, Conexión, BaseDatos, tabla)
Responsabilidades:	Recibir los datos de un formulario (nombre)
Tipo:	Sistema
Referencias Cruzadas:	
Notas:	Son capturados a través de una interfaz de usuario o por medio de una selección del objeto

Excepciones:	
Salida:	
Precondiciones:	
Poscondiciones:	
Nombre:	mostrarFormulario()
Responsabilidades:	Mostrar un formulario existente
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.4.2, R.4.3, R.4.4, R.4.5, R.4.6
Notas:	El formulario se muestra en dos vistas una de diseño; y una de código java del mismo
Excepciones:	
Salida:	
Precondiciones:	Existe el objeto formulario previamente
Poscondiciones:	

Nombre:	modificarFormulario()
Responsabilidades:	Modificar un formulario existente
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.4.4, R.4.15
Notas:	
Excepciones:	
Salida:	
Precondiciones:	Existe el formulario previamente
Poscondiciones:	Se reconstruye el formulario

Nombre:	eliminarFormulario()
Responsabilidades:	Eliminar un objeto formulario en la herramienta
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.4.14
Notas:	
Excepciones:	
Salida:	
Precondiciones:	El formulario ya existe previamente en la herramienta
Poscondiciones:	Se elimina el objeto formulario en la herramienta

Nombre:	datosReporte(nombreReporte, Conexión, BaseDatos, Tabla)
Responsabilidades:	Captura los datos de una base de datos (nombre, máquina y usuario)
Tipo:	Sistema

Referencias Cruzadas:	
Notas:	Son capturados a través de una interfaz de usuario o por una selección del objeto
Excepciones:	
Salida:	
Precondiciones:	
Poscondiciones:	

Nombre:	MostrarReporte()
Responsabilidades:	Mostrar un reporte existente
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.5.2, R.5.3, R.5.4, R.5.8
Notas:	Se muestra el reporte en tres vistas: una de diseño; una vista previa y una vista código
Excepciones:	
Salida:	
Precondiciones:	Existe el objeto reporte previamente
Poscondiciones:	

Nombre:	modificarReporte()
Responsabilidades:	Modificar un reporte ya existente
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.5.10
Notas:	
Excepciones:	
Salida:	
Precondiciones:	Ya existe un reporte previamente en la herramienta que modificar
Poscondiciones:	Se reconstruye el objeto reporte

Nombre:	eliminarReporte()
Responsabilidades:	Eliminar una reporte existente en la herramienta
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.5.9
Notas:	
Excepciones:	
Salida:	
Precondiciones:	Existe el reporte previamente
Poscondiciones:	Se elimina el reporte de la herramienta

Nombre:	datosArchivo(nombreArchivo, ubicacion)
Responsabilidades:	Capturar los datos de una archivo (nombre y ubicación del mismo) que será almacenado
Tipo:	Sistema
Referencias Cruzadas:	
Notas:	Se capturan a través de una ventana de diálogo
Excepciones:	
Salida:	
Precondiciones:	
Poscondiciones:	

Nombre:	almacenar()
Responsabilidades:	Almacenar un archivo ya sea un script, formulario o reporte que se este trabajando en la herramienta
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.4.11, R.5.6
Notas:	Se interactúa con el sistema manejador de archivos del sistema operativo en el que se esté trabajando, emitiendo una ventana tipo dialogo para ubicar el archivo y asignarle un nombre
Excepciones:	
Salida:	
Precondiciones:	Se tiene un objeto tipo script, formulario o reporte que guardar
Poscondiciones:	Se almacena este archivo

Nombre:	Compilar()
Responsabilidades:	Compilar un objeto reporte o un formulario (.java)
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.4.12
Notas:	
Excepciones:	Se encuentre algún error dentro del código fuente Java de este objeto a compilar
Salida:	Un archivo .class
Precondiciones:	Existe un objeto reporte o formulario seleccionado
Poscondiciones:	Se crea un objeto ejecutable .class

Nombre:	ejecutarScript()
Responsabilidades:	Ejecutar un script de sentencias SQL estándar
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.8.11
Notas:	
Excepciones:	Existe algún tipo de error en el script
Salida:	
Precondiciones:	
Poscondiciones:	

Nombre:	verEstructura()
Responsabilidades:	Mostrar las estructuras capturadas (bases de datos y tablas), además de las generadas (formularios y reportes)
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.6.1, R.9, R.10
Notas:	
Excepciones:	
Salida:	
Precondiciones:	
Poscondiciones:	

Nombre:	ejecutarOperacion()
Responsabilidades:	Ejecutar un operación designada por el usuario sobre un objeto el cual puede ser una BD, Tabla, Formulario o reporte
Tipo:	Sistema
Referencias Cruzadas:	
Notas:	
Excepciones:	
Salida:	
Precondiciones:	Un objeto seleccionado
Poscondiciones:	

Nombre:	datosScript(nombreScript, ubicacion)
Responsabilidades:	Captura los datos de una script (nombre y ubicación)
Tipo:	Sistema
Referencias Cruzadas:	
Notas:	Son capturados a través de una ventana de dialogo
Excepciones:	

Salida:	
Precondiciones:	
Poscondiciones:	

Nombre:	generar()
Responsabilidades:	Generar un script con las especificaciones previamente entregadas por el usuario
Tipo:	Sistema
Referencias Cruzadas:	R.8
Notas:	
Excepciones:	
Salida:	
Precondiciones:	Se entregaron una especificaciones de script
Poscondiciones:	

Nombre:	ejecutarJava()
Responsabilidades:	Ejecutar un objeto previamente compilado
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.4.12
Notas:	
Excepciones:	Existe algún tipo de error en el código ejecutable o no se encuentre
Salida:	
Precondiciones:	El archivo . class existe previamente
Poscondiciones:	Se lanza una versión ejecutable de dicho código

Nombre:	crearReporte()
Responsabilidades:	Crear un nuevo reporte dentro de la herramienta
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.5, R.5.1, R.5.5, R.5.7
Notas:	
Excepciones:	
Salida:	
Precondiciones:	Existe una base de datos previamente identificada
Poscondiciones:	Se crea el objeto reporte en la herramienta Se muestra el objeto reporte

Nombre:	conectar()
Responsabilidades:	Iniciar una conexión con el motor
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.6,R.6.1, R.6.2
Notas:	
Excepciones:	No se puede realizar la conexión
Salida:	
Precondiciones:	Una conexión previamente identificada
Poscondiciones:	Una conexión con el motor

Nombre:	datosConexion(nombre, login, contraseña, ip, puerto, basesdatos)
Responsabilidades:	Recibir los datos de un usuario, los cuales son: nombre o login y contraseña, además de datos de conexión como su nombre y las bases de datos a las cuales se conectará.
Tipo:	Sistema
Referencias Cruzadas:	
Notas:	Son capturados a través de una interfaz de usuario
Excepciones:	
Salida:	
Precondiciones:	
Poscondiciones:	

Nombre:	modificarBD()
Responsabilidades:	Modificar una Base de Datos existente
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.2
Notas:	
Excepciones:	No se tiene permisos para realizar la modificación
Salida:	
Precondiciones:	Una base de datos existente registrada en la herramienta
Poscondiciones:	Una base de datos modificada

Nombre:	cerrarBD()
Responsabilidades:	Cerrar una base de datos de una conexión
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.3.2
Notas:	
Excepciones:	Si es la base de datos principal de la conexión

Salida:	
Precondiciones:	Una base de datos identificada en la herramienta
Poscondiciones:	La base de datos se cierra de la conexión

Nombre:	abrirScript()
Responsabilidades:	Abrir un script existente
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.8.2
Notas:	
Excepciones:	El archivo no tiene un formato valido
Salida:	
Precondiciones:	Un script existente en el sistema gestor de archivos
Poscondiciones:	Un script abierto en la base de datos

Nombre:	cerrarScript()
Responsabilidades:	Cerrar un script abierto en la herramienta
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.8.3
Notas:	
Excepciones:	
Salida:	
Precondiciones:	Un script abierto en la herramienta
Poscondiciones:	Un script cerrado de la herramienta

Nombre:	guardarScript()
Responsabilidades:	Guardar un script existente en la herramienta
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.8.4
Notas:	
Excepciones:	
Salida:	
Precondiciones:	Un script existente en la herramienta
Poscondiciones:	El script guardado en el sistema gestor de archivos

Nombre:	abrirArchivo()
Responsabilidades:	Abrir un archivo producto
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.4.14, R.5.12
Notas:	

Excepciones:	El archivo ya se encuentra abierto en la herramienta
Salida:	
Precondiciones:	El archivo producto ya existe y no esta abierto
Poscondiciones:	El archivo producto abierto en la herramienta

Nombre:	almacenarJAVA()
Responsabilidades:	Guardar el código fuente de un archivo producto
Tipo:	Sistema
Referencias Cruzadas:	Funciones del Sistema: R.4.11
Notas:	
Excepciones:	
Salida:	
Precondiciones:	Un archivo producto existente y seleccionado en la herramienta
Poscondiciones:	Un archivo .java guardado en el sistema gestor de archivo, validar que el archivo java ya se encuentra guardado

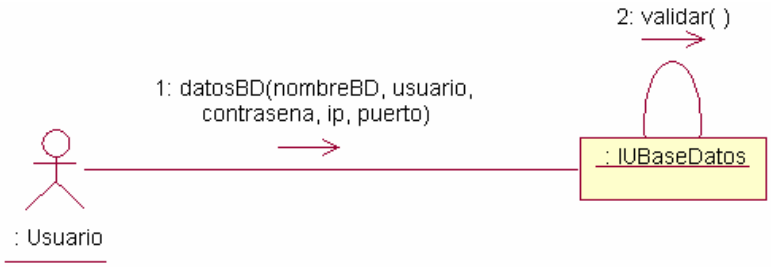
8.3 FLUJO DE TRABAJO DE DISEÑO

8.3.1 Diagramas de colaboración

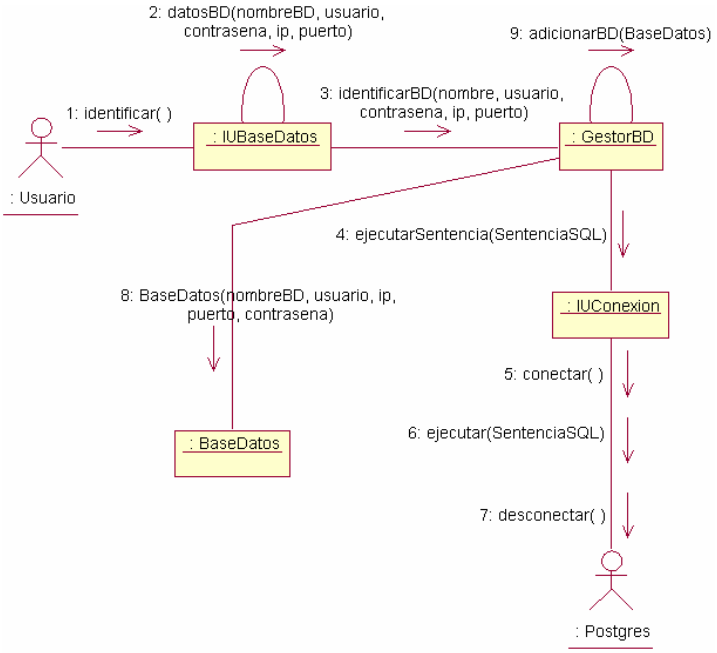
DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 5	Modelo Conceptual GRFPostgres
Tabla 6	Contratos de operación del sistema

Figura 7. Diagramas de colaboración

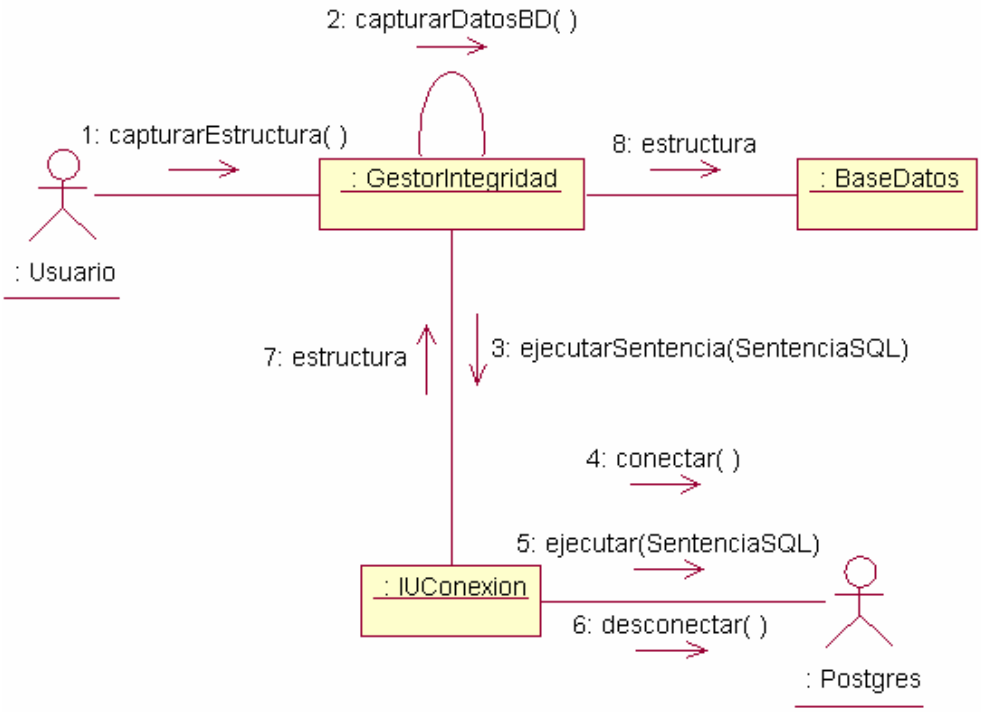
datosBD(nombreBD, usuario, contraseña, ip, puerto)



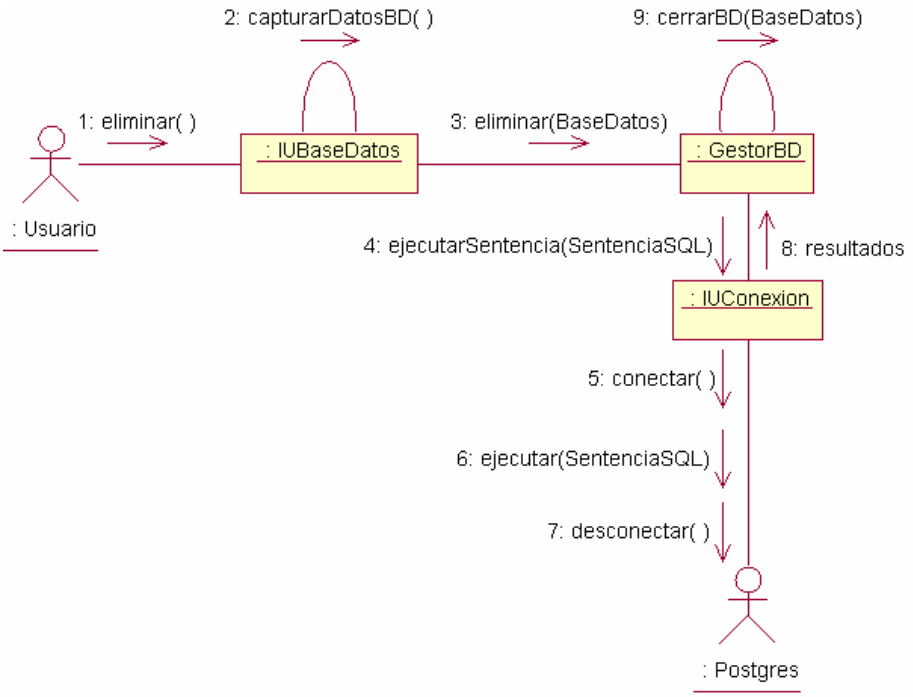
identificarBD()



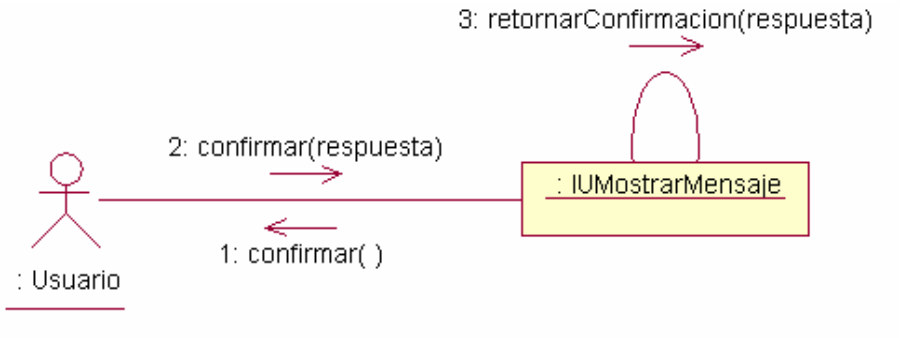
capturarEstructura()



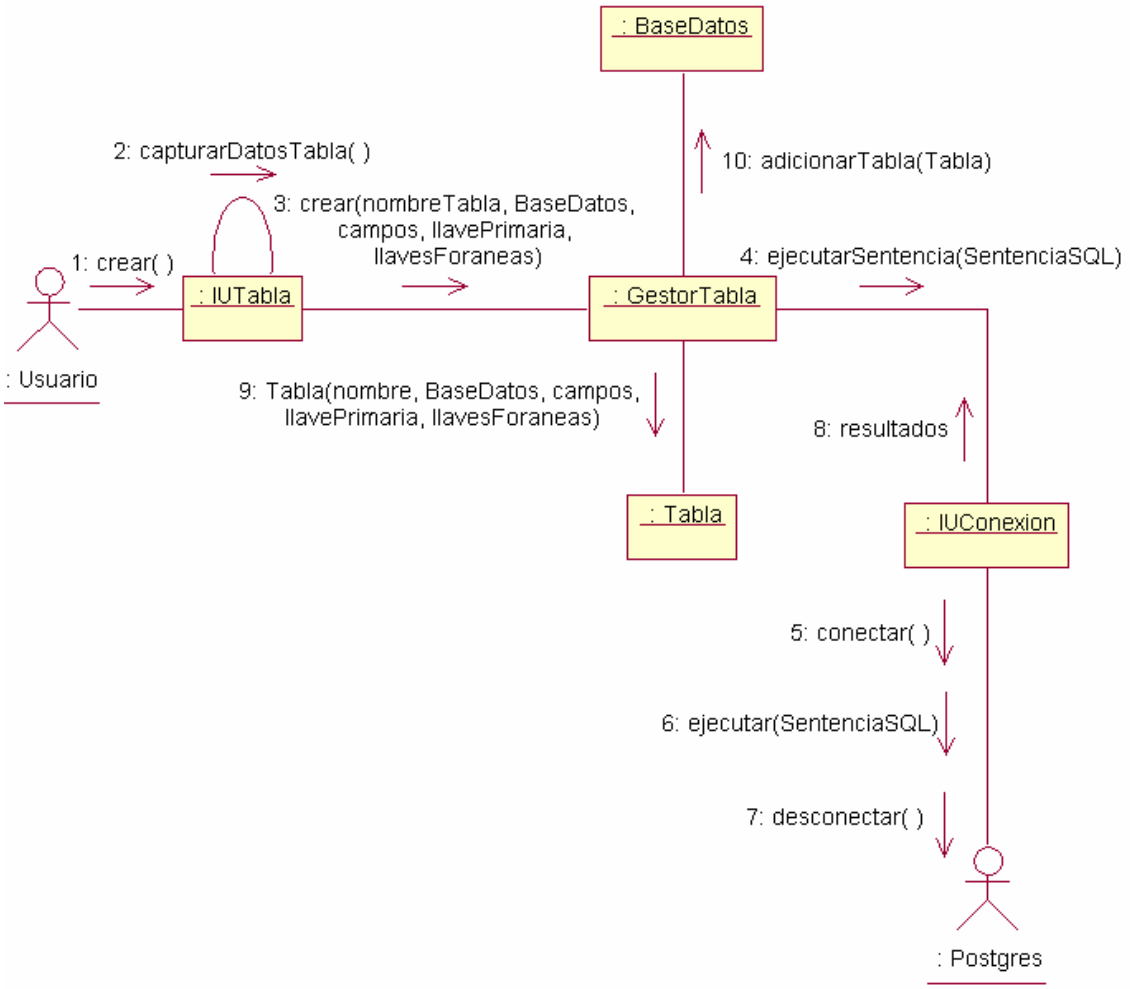
eliminarBD()



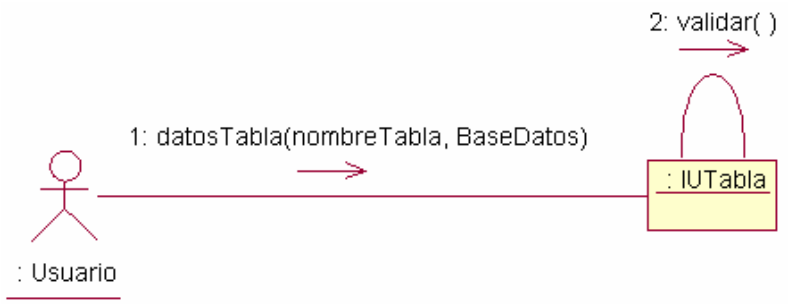
confirmar()



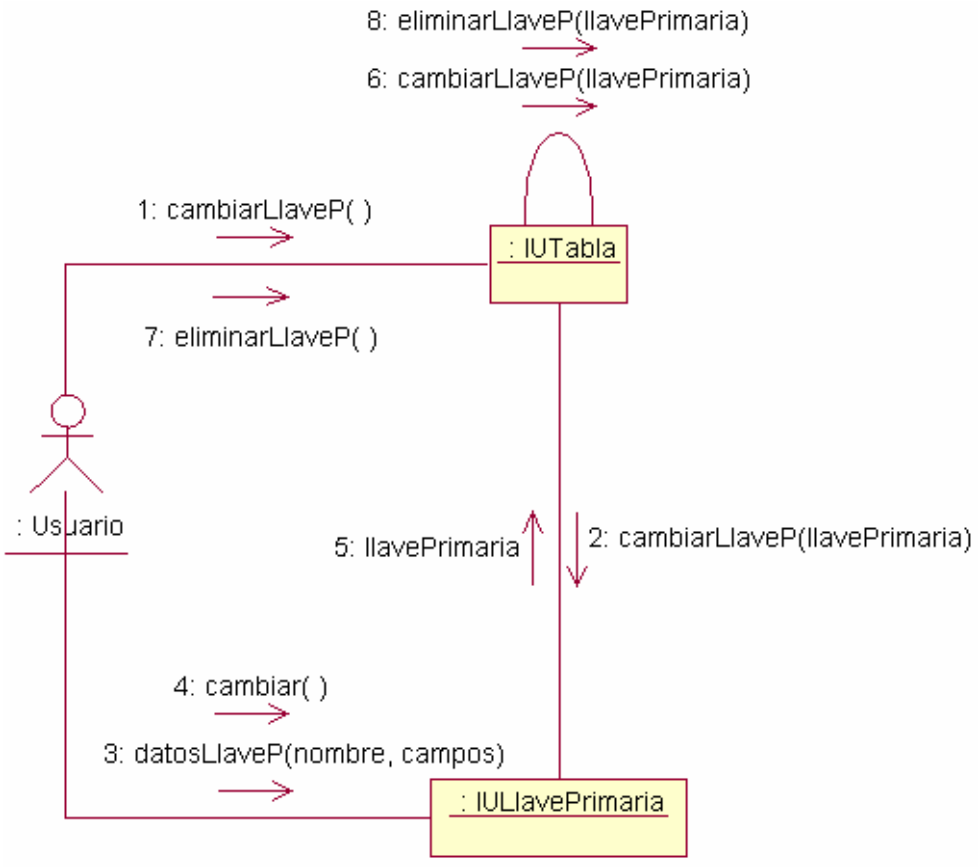
crearTabla()



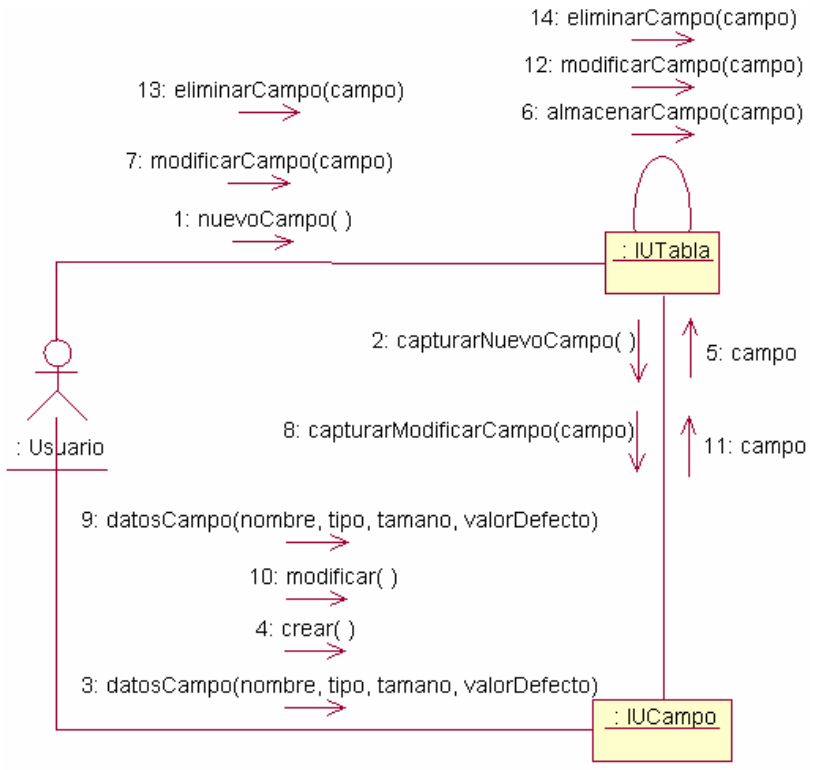
datosTabla(nombreTabla, BaseDatos)



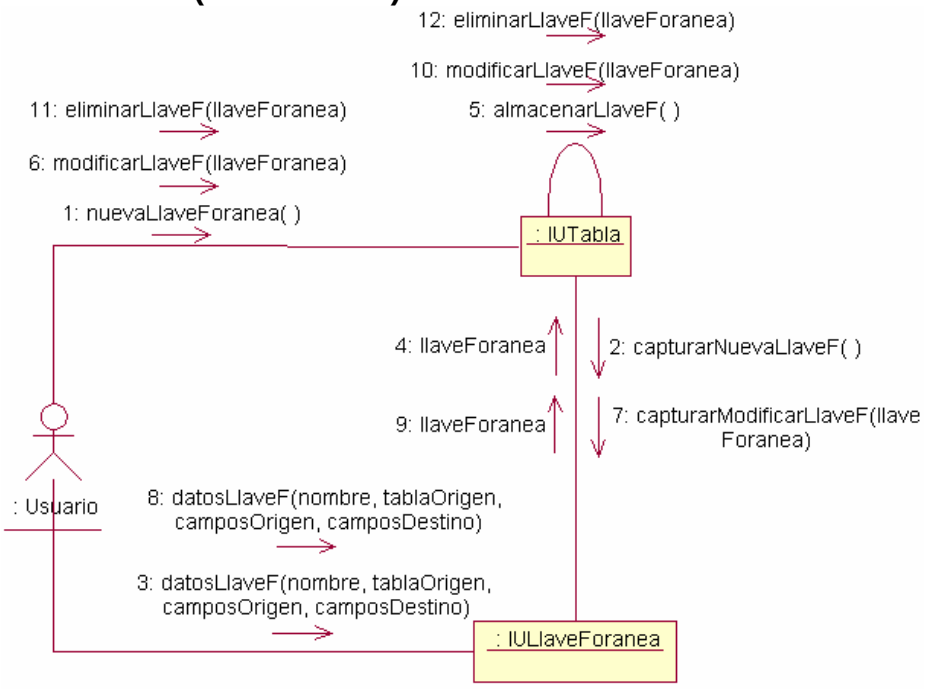
asignarLlavePrimaria(datosLlaveP)



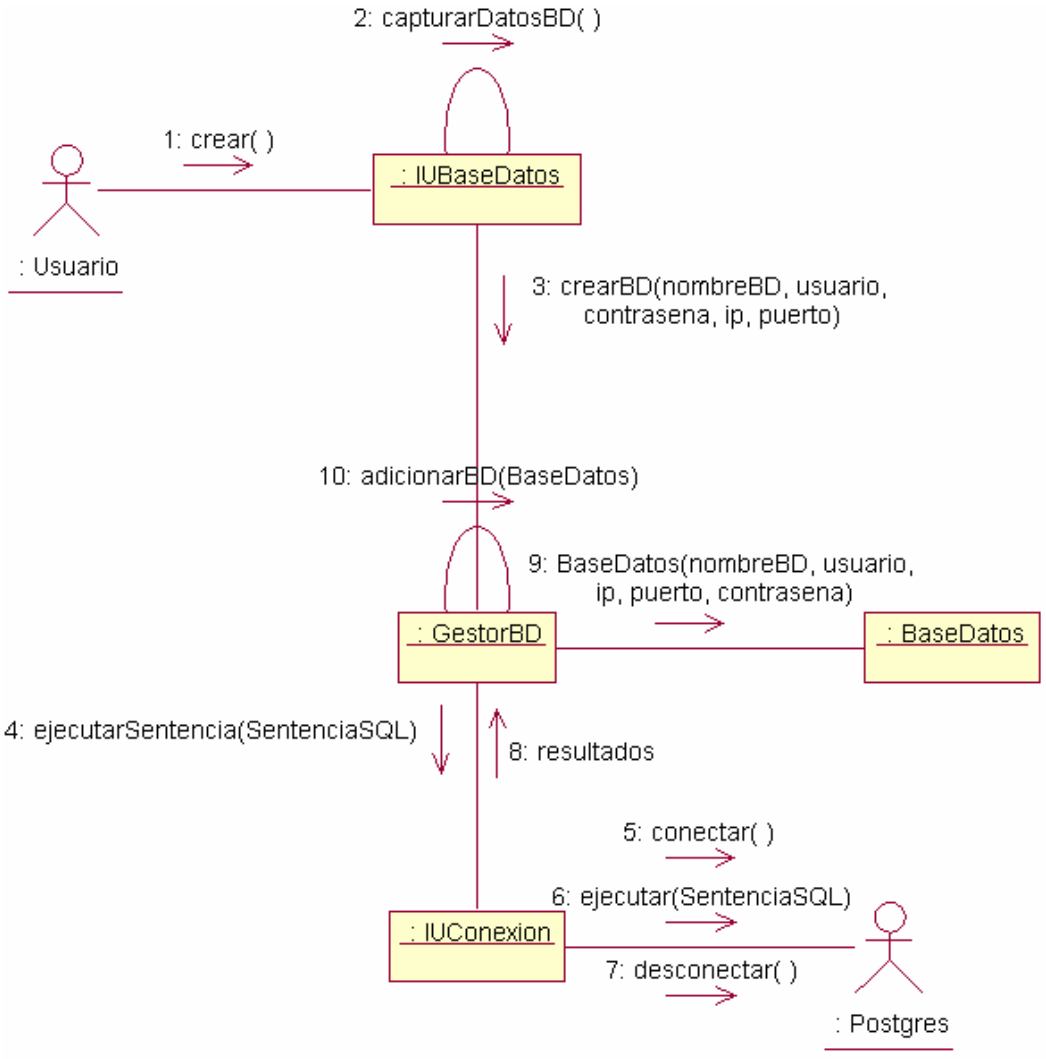
asignarCampos(datosCampo)



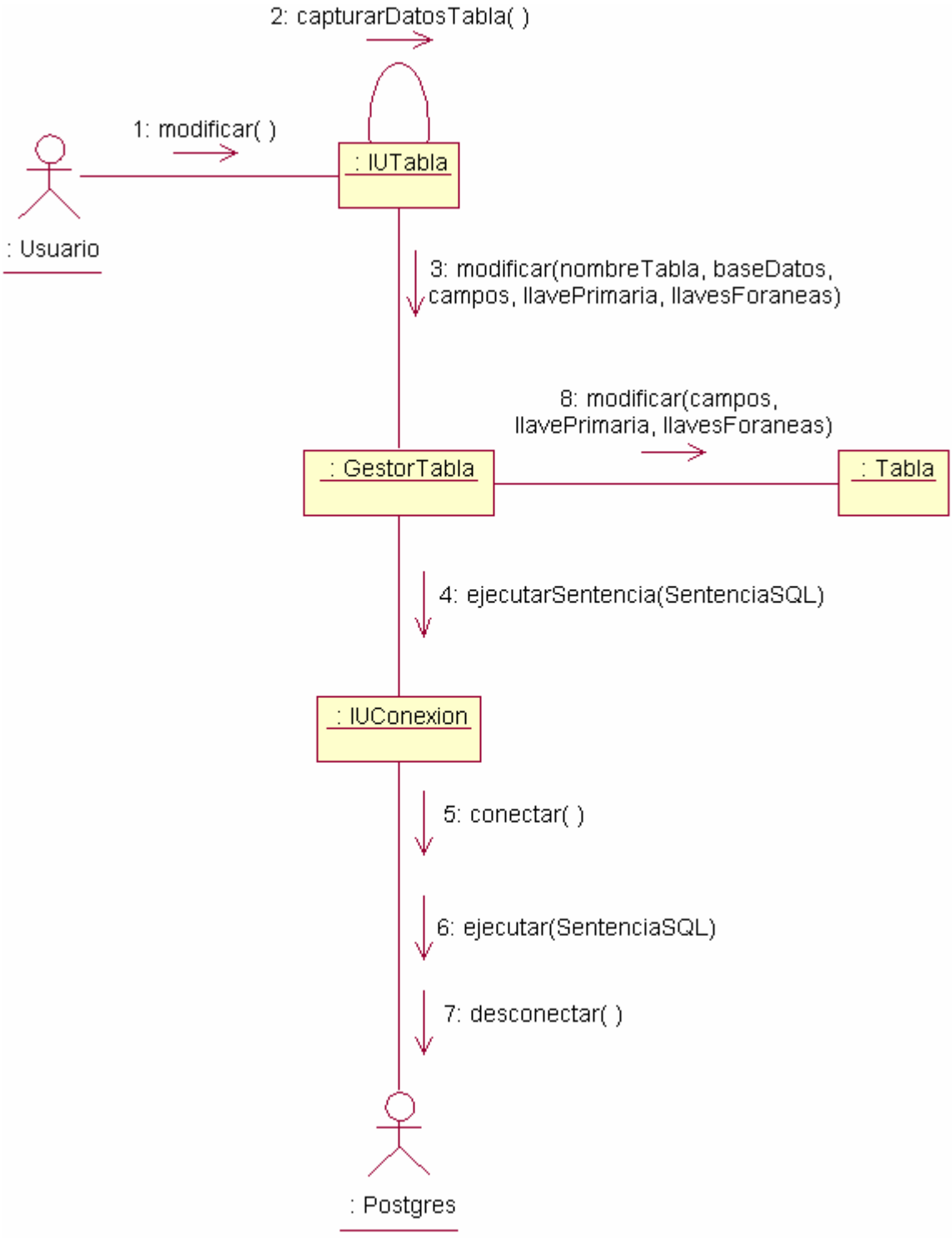
asignarLlaveForanea(datosLlaveF)



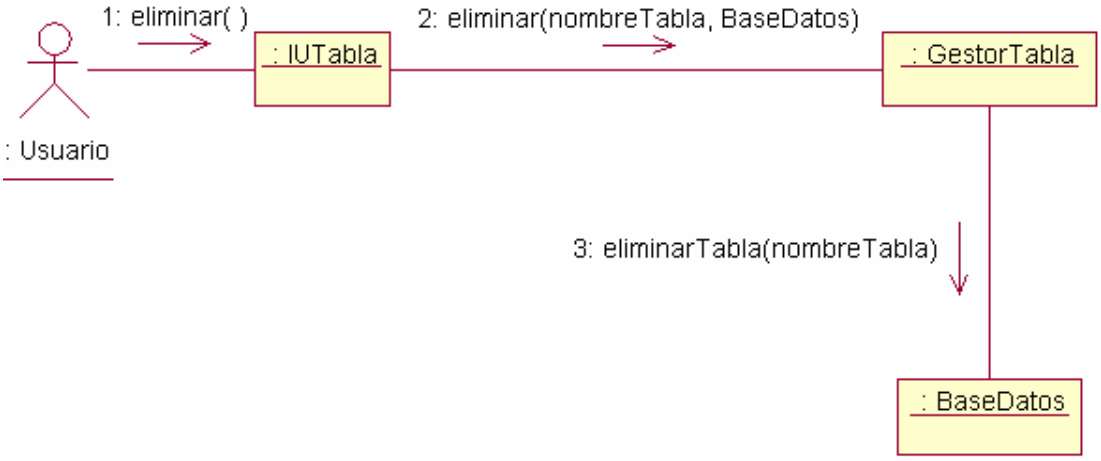
crearBD()


2: capturarDatosBD( )
1: crear( )
: IUBaseDatos
: Usuario
3: crearBD(nombreBD, usuario, contrasena, ip, puerto)
10: adicionarBD(BaseDatos)
9: BaseDatos(nombreBD, usuario, ip, puerto, contrasena)
: GestorBD
: BaseDatos
4: ejecutarSentencia(SentenciaSQL)
8: resultados
5: conectar( )
6: ejecutar(SentenciaSQL)
: IUConexion
7: desconectar( )
: Postgres

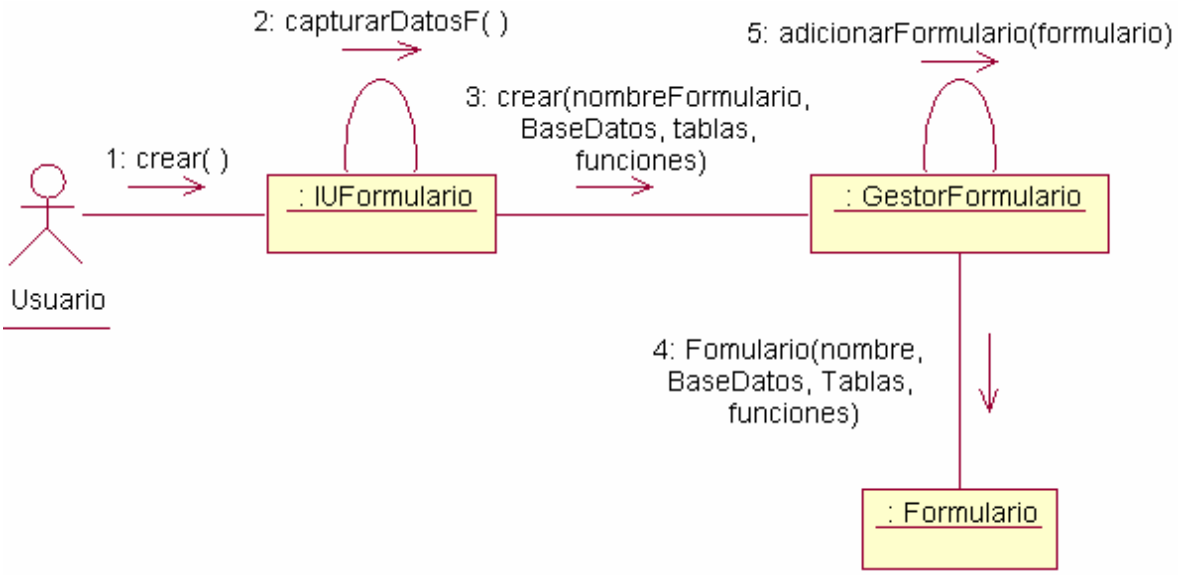
modificarTabla()



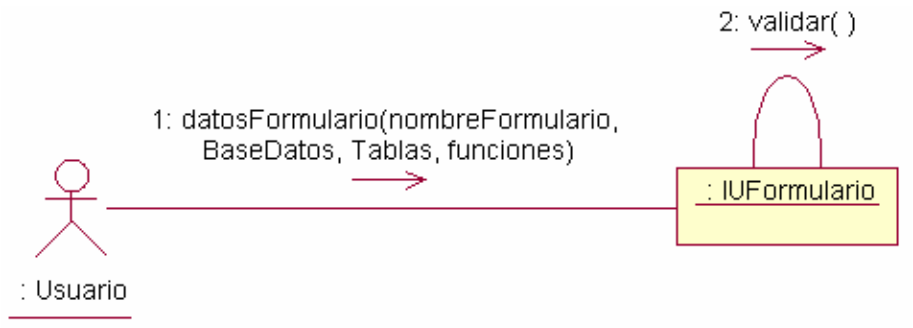
eliminarTabla()



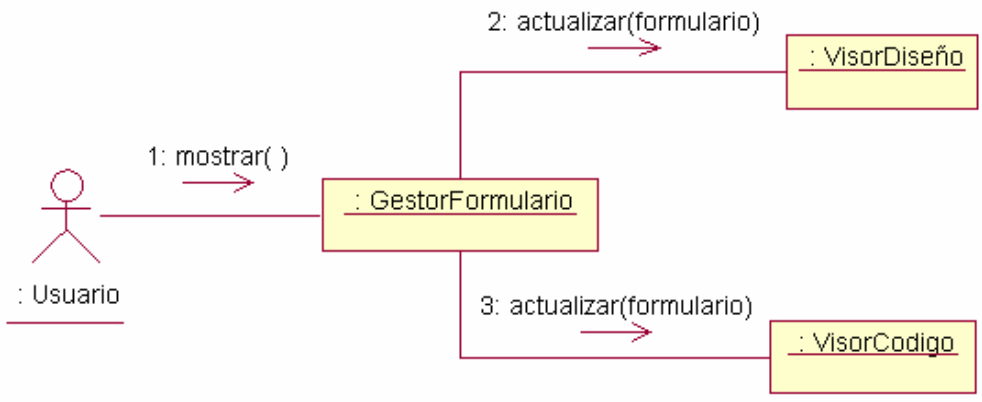
crearFormulario()



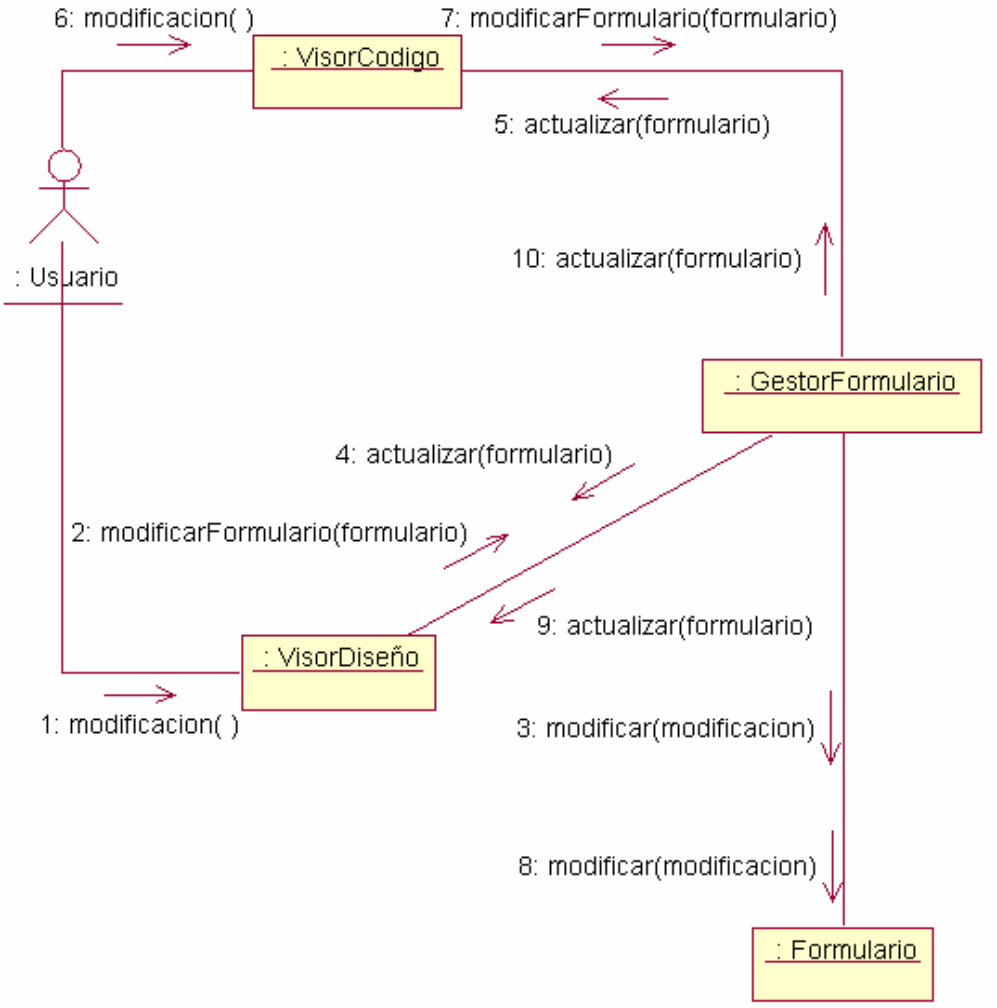
datosFormulario(nombreFormulario, BaseDatos, Tablas, funciones)



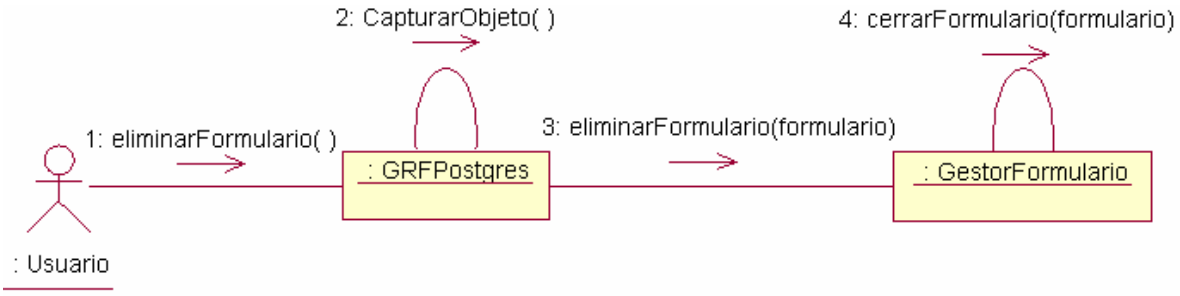
mostrarFormulario()



modificarFormulario()

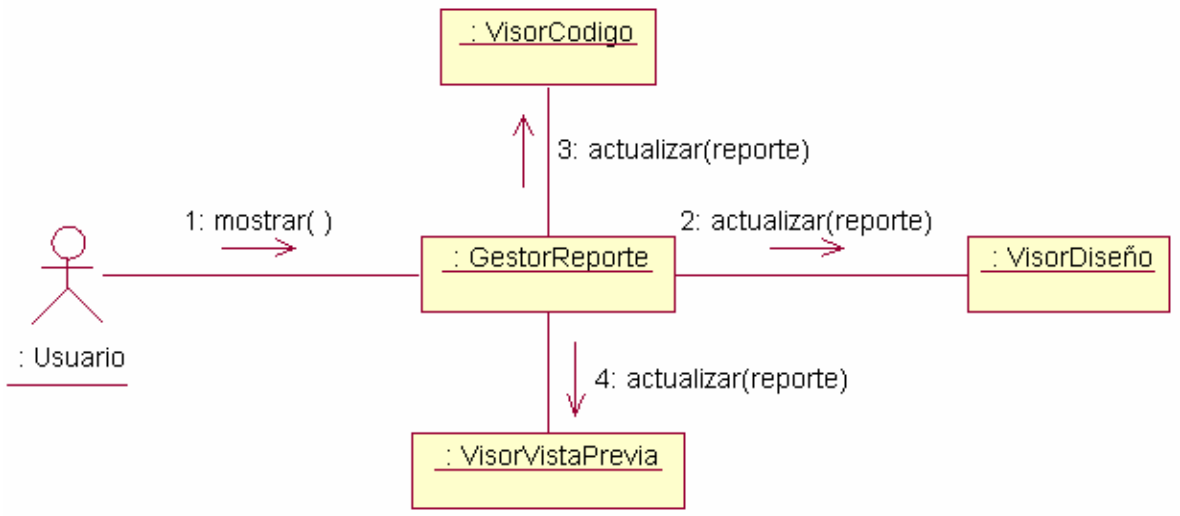


eliminarFormulario()

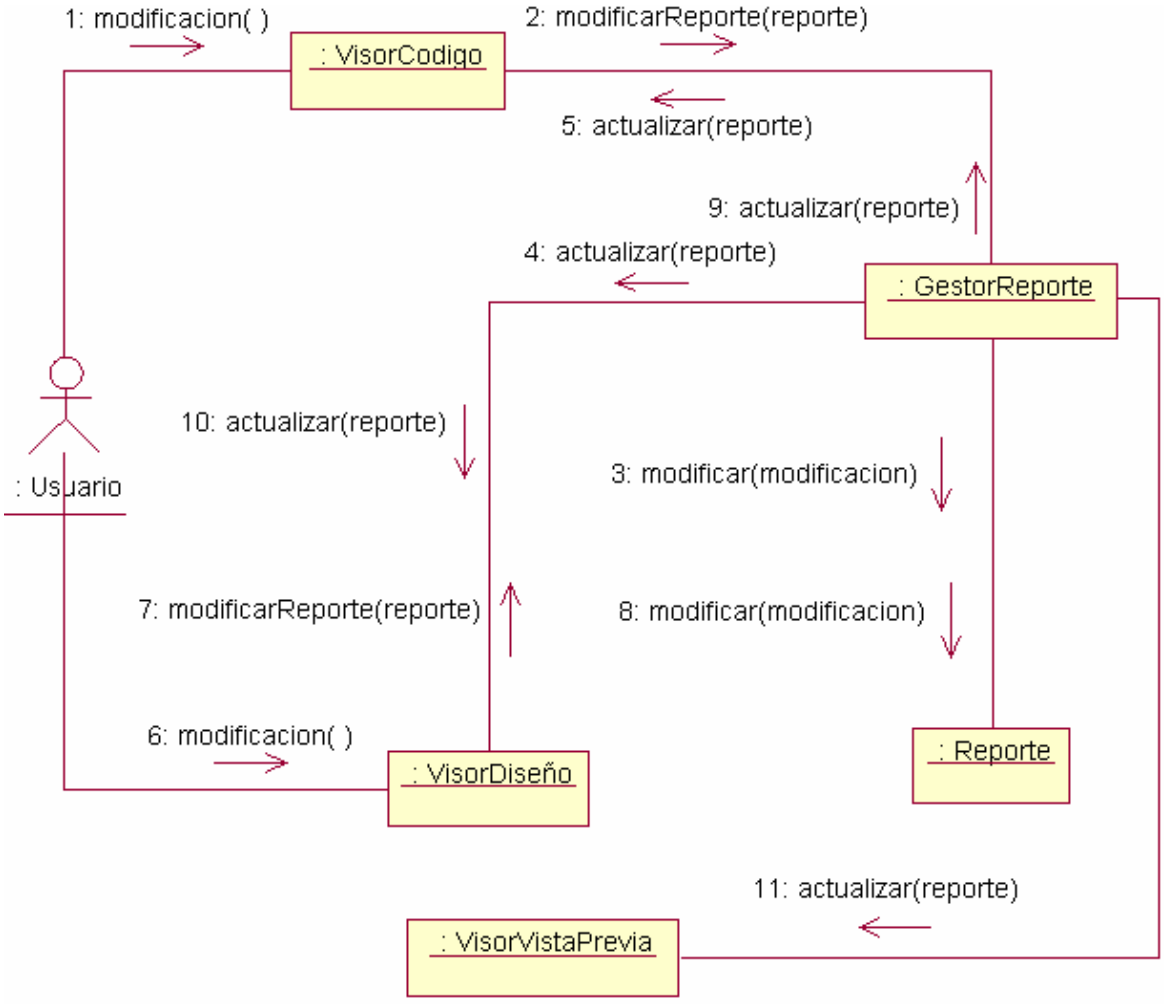


datosReporte(nombreReporte, BaseDatos, Tablas)

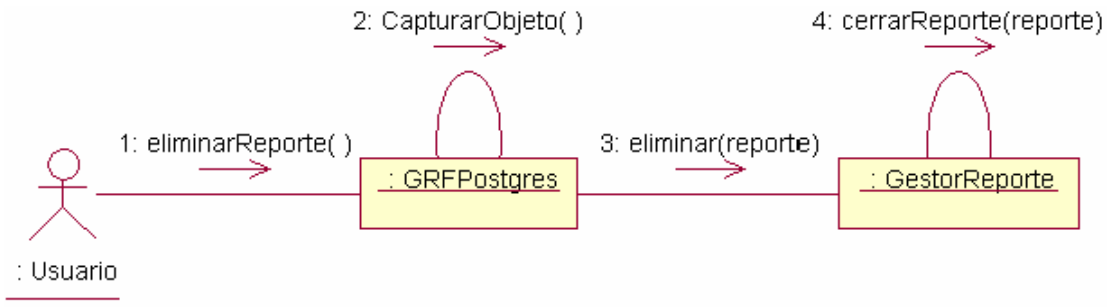
mostrarReporte()



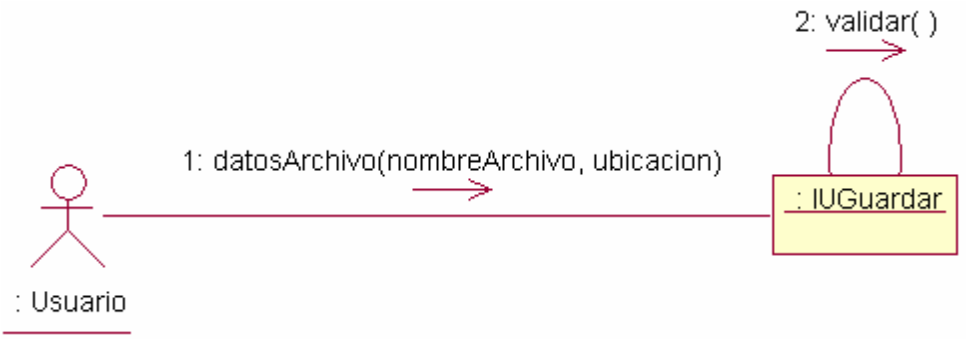
modificarReporte()



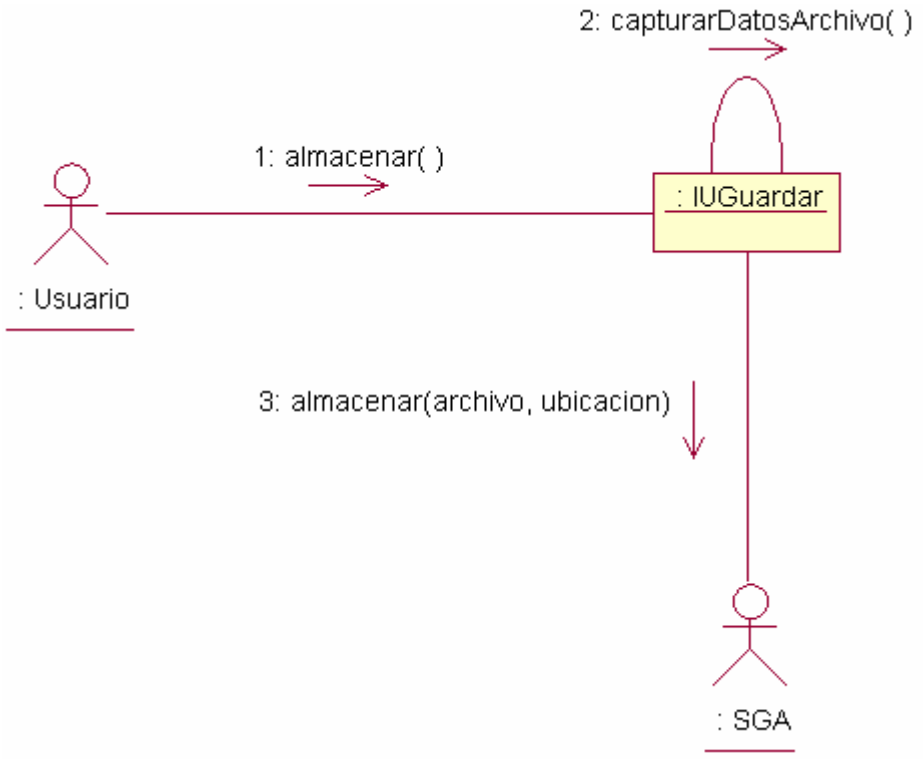
EliminarReporte()



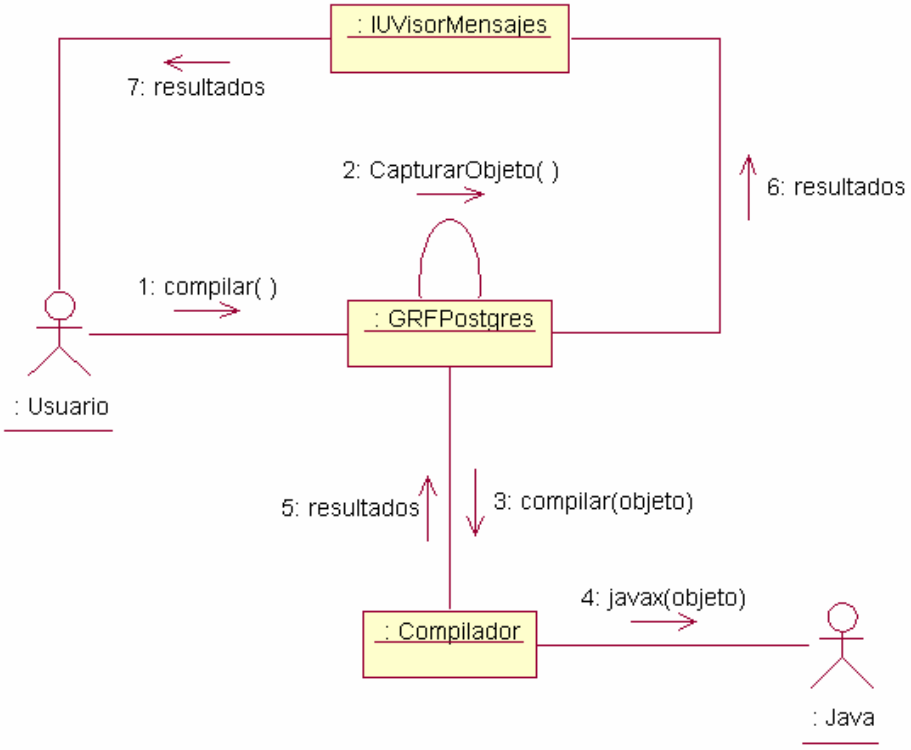
datosArchivo(nombreArchivo, ubicacion)



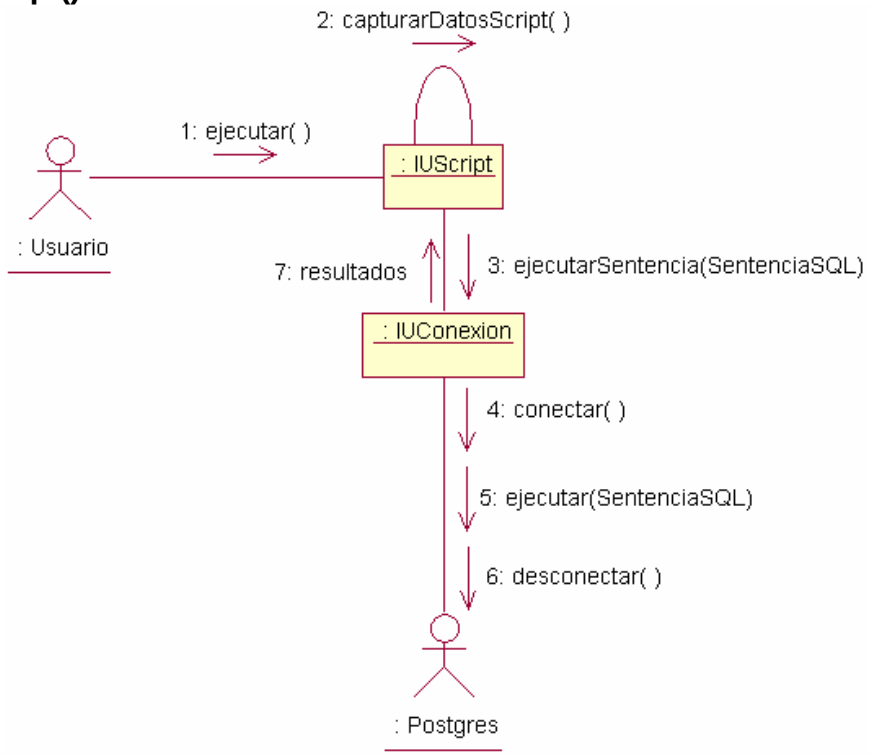
almacenar()



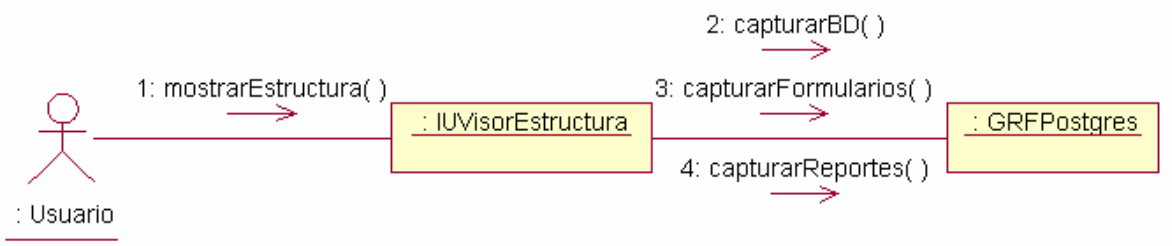
compilar()



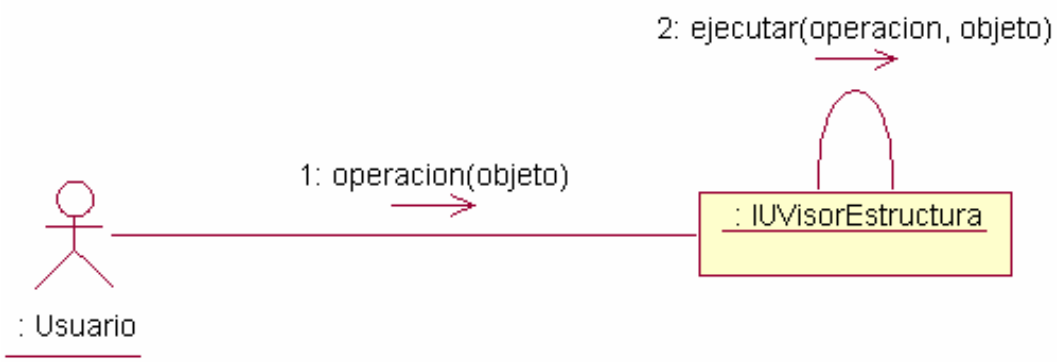
ejecutarScript()



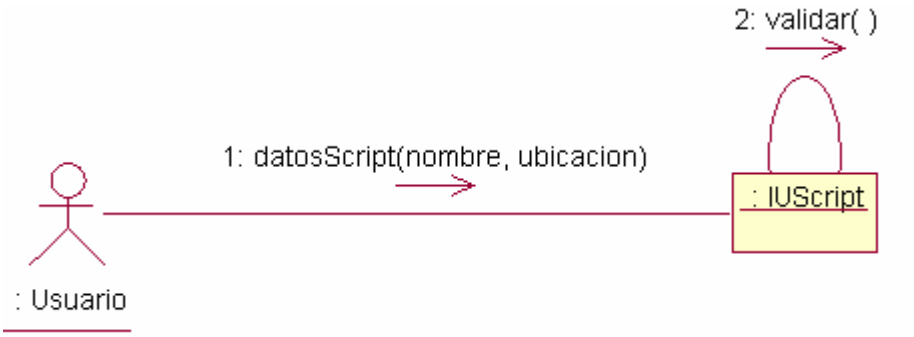
verEstructura()



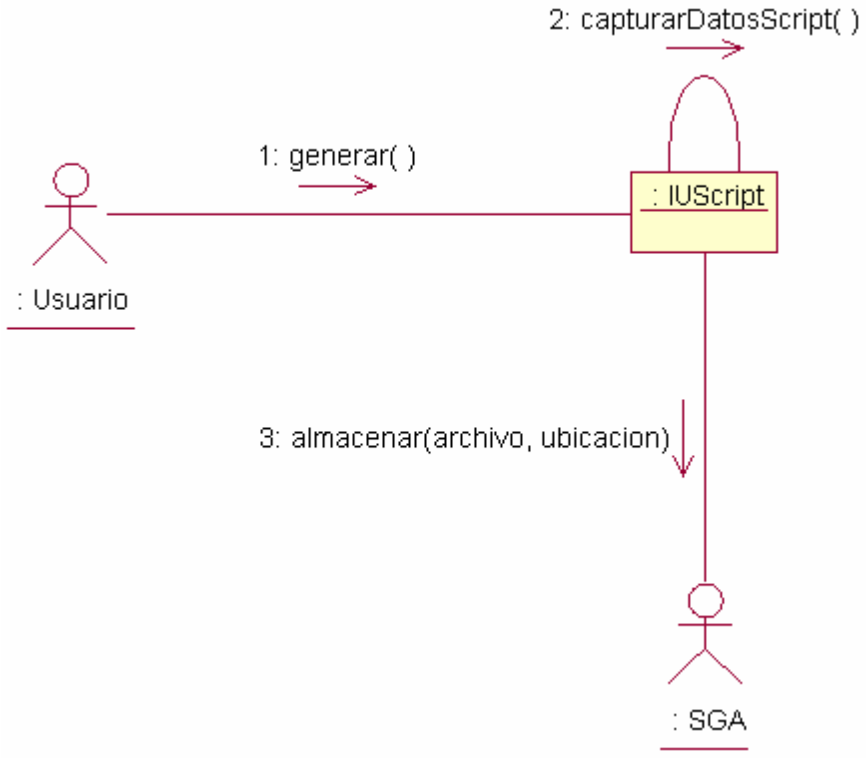
ejecutarOperacion()



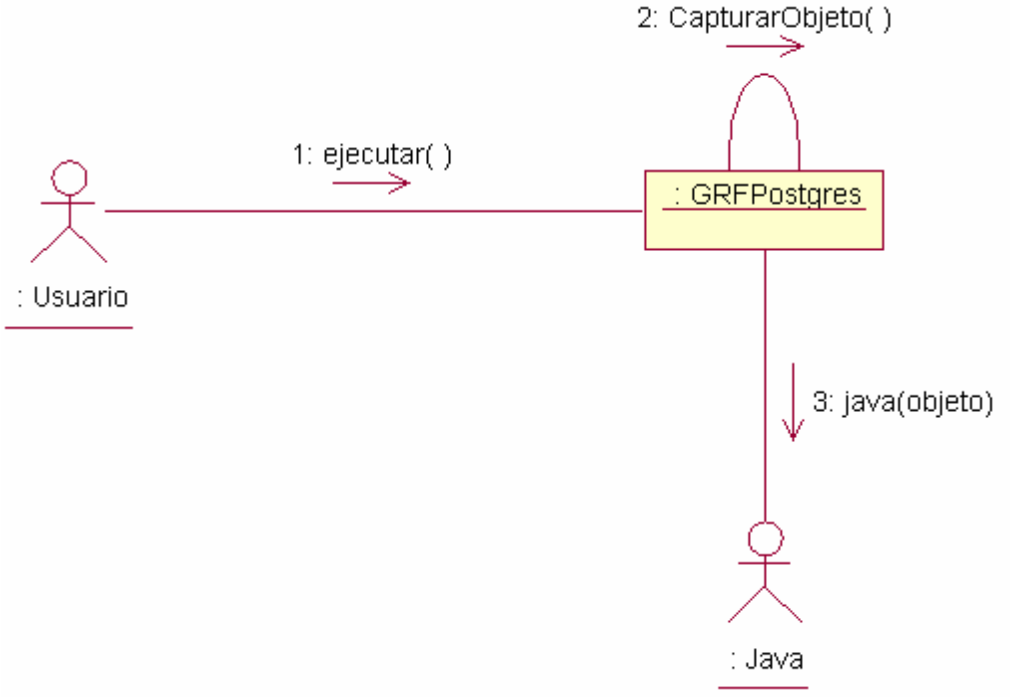
datosScript(nombreScript, ubicación)



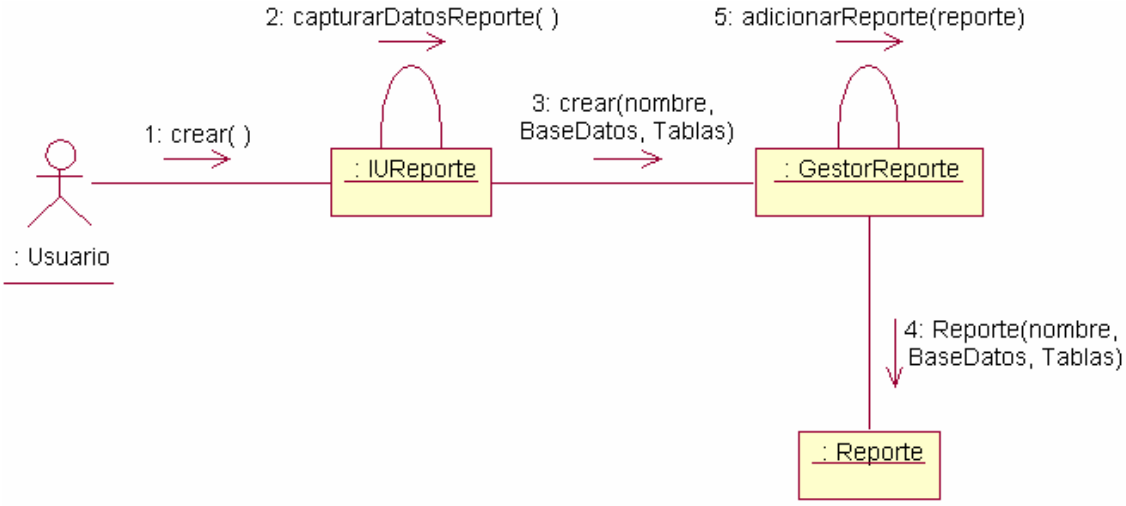
generar()



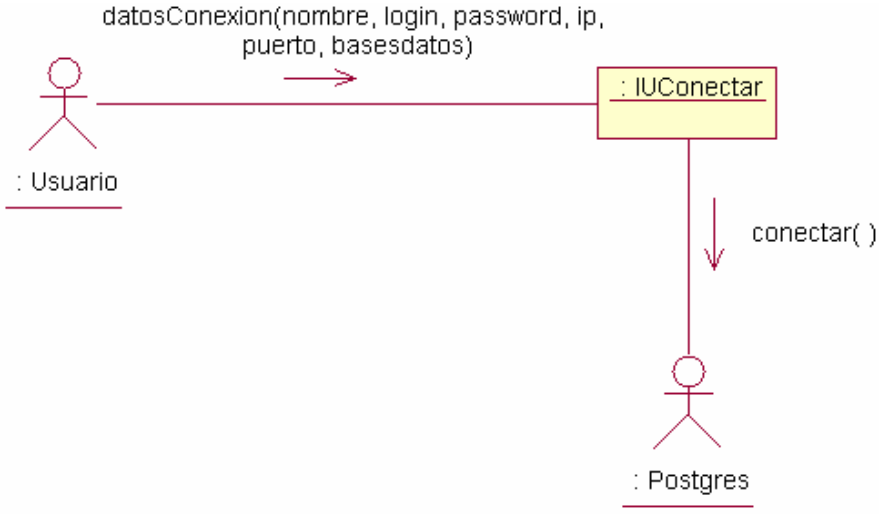
ejecutarJava()



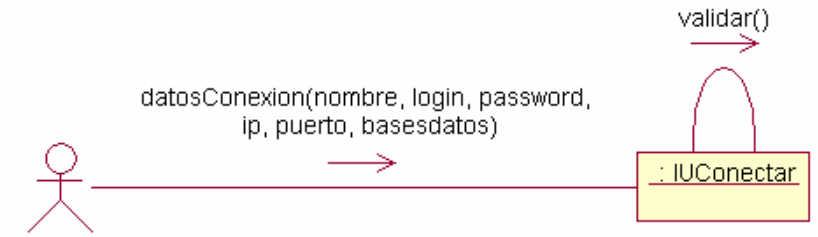
crearReporte()



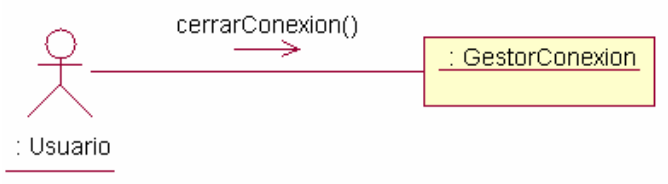
conectar()



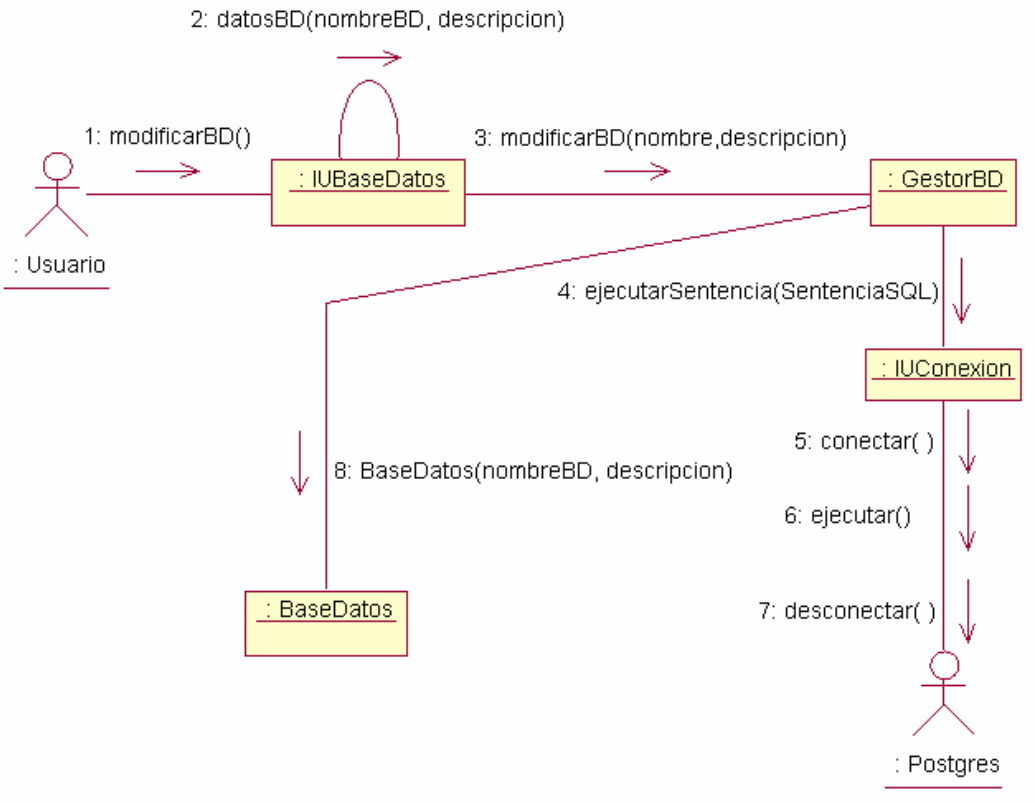
datosConexion(nombre, login, password, ip, puerto, basedatos)



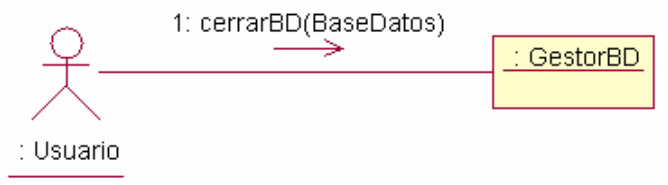
cerrarConexion()



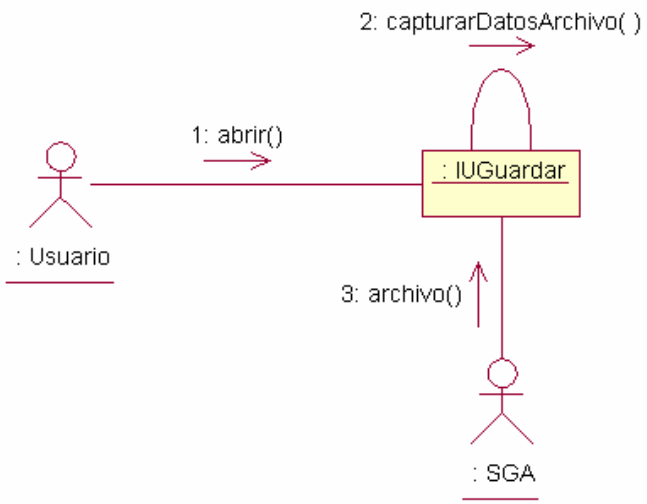
modificarBD()



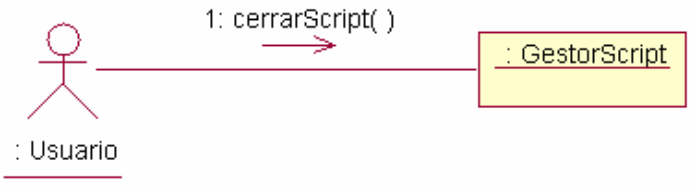
cerrarBD()



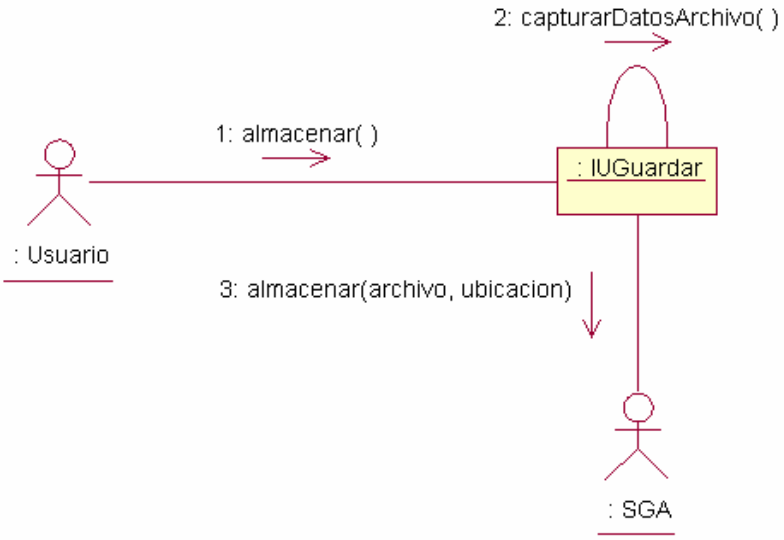
abrirScript()



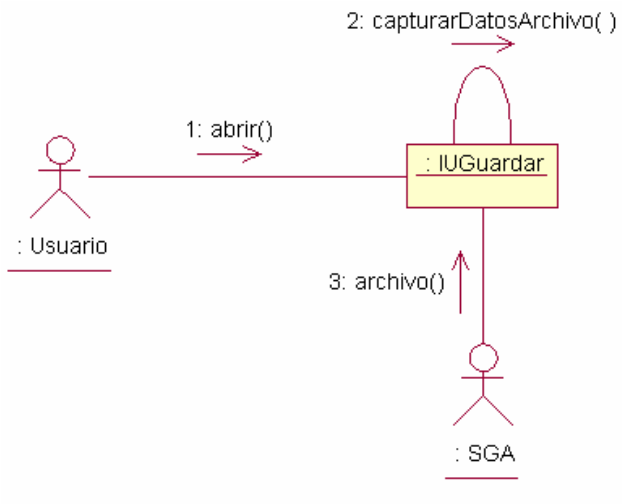
cerrarScript()



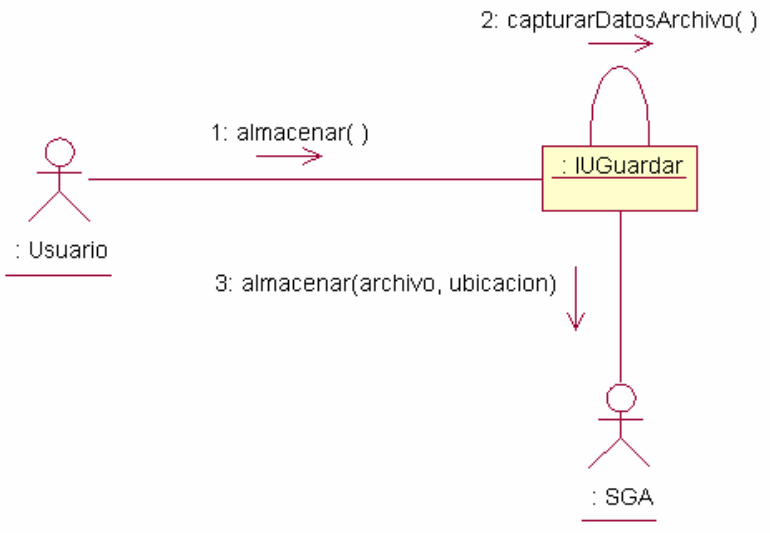
guardarScript()



abrirArchivo()



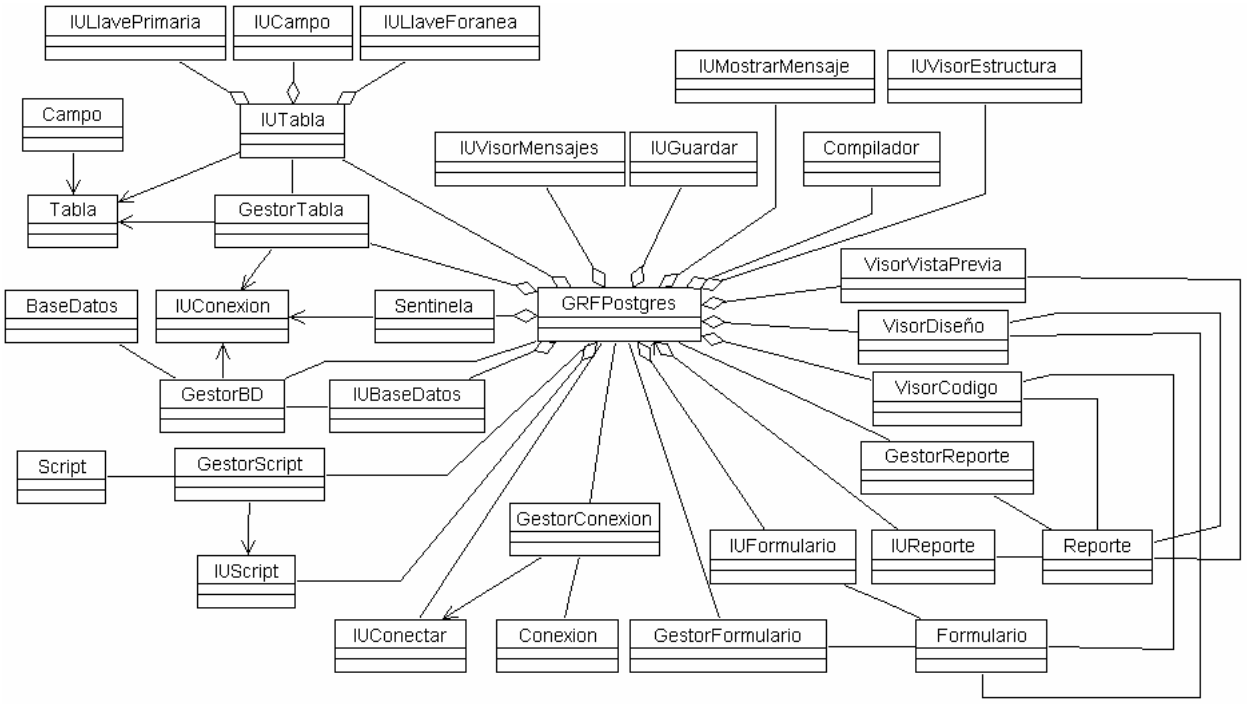
almacenarJava()



8.3.2 Diagrama de Clases

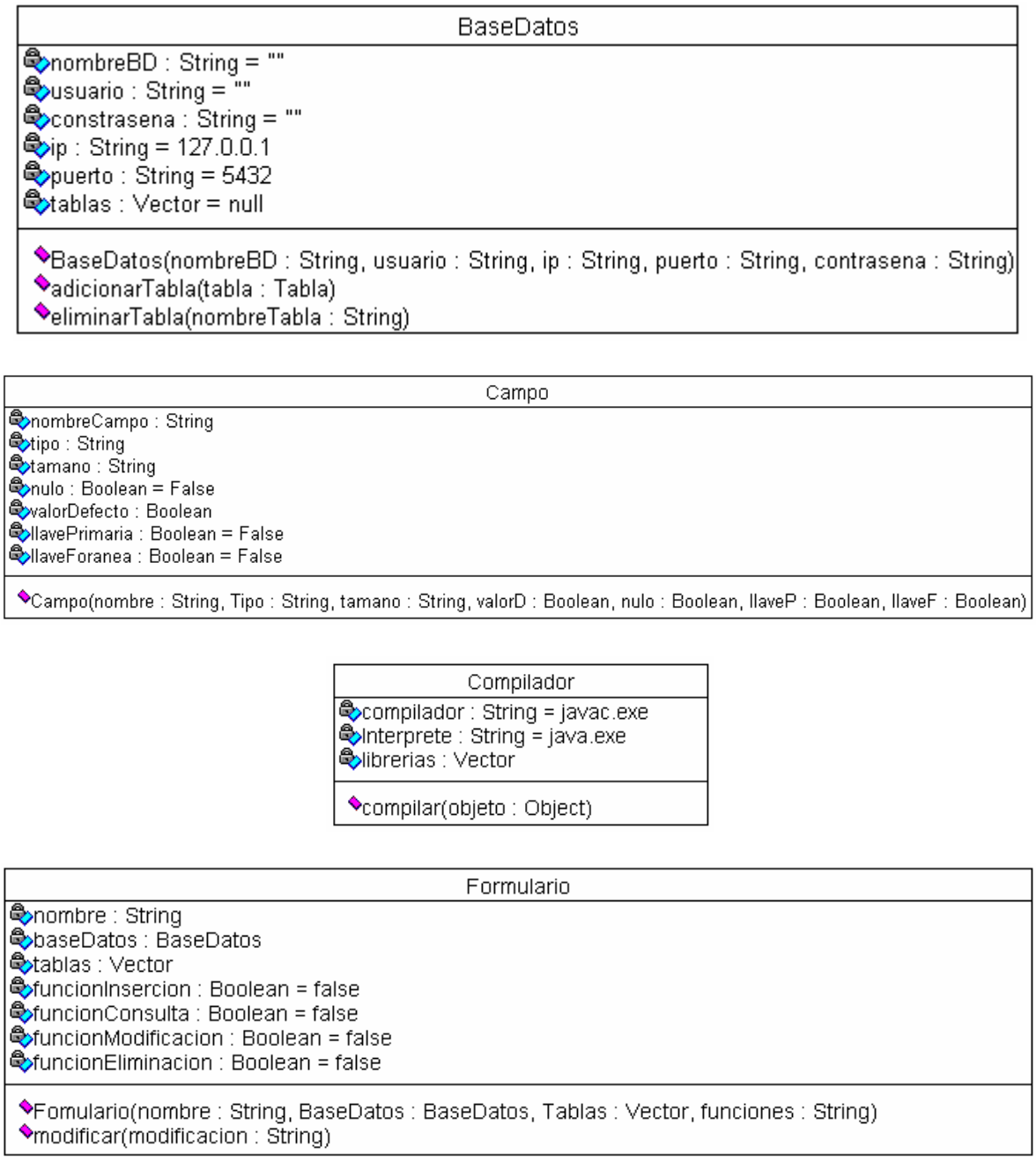
DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 5	Modelo Conceptual GRFPostgres
Figura 7	Diagramas de Colaboración

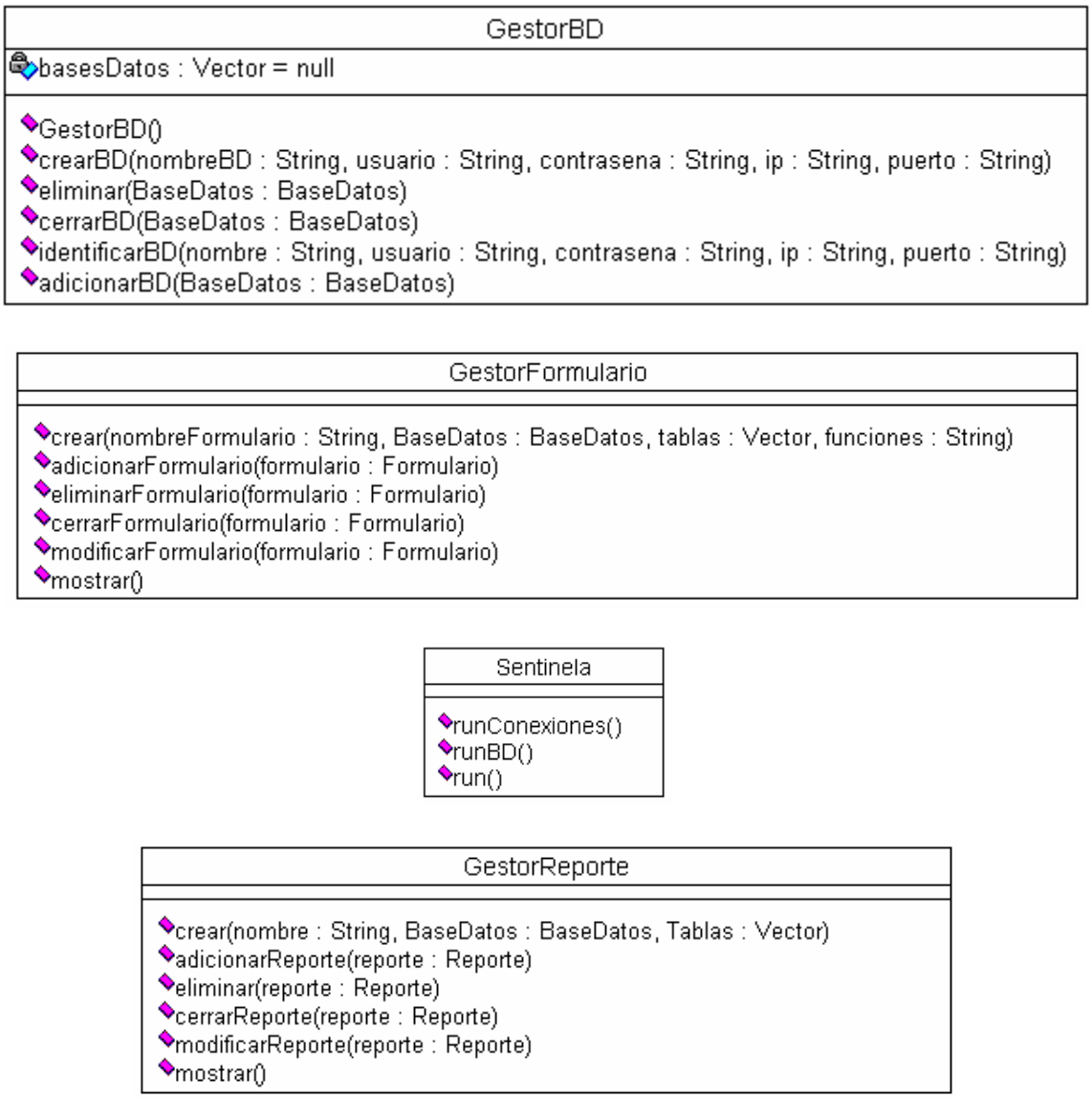
Figura 8. Diagrama de Clases



Detalle del Diagrama de Clases

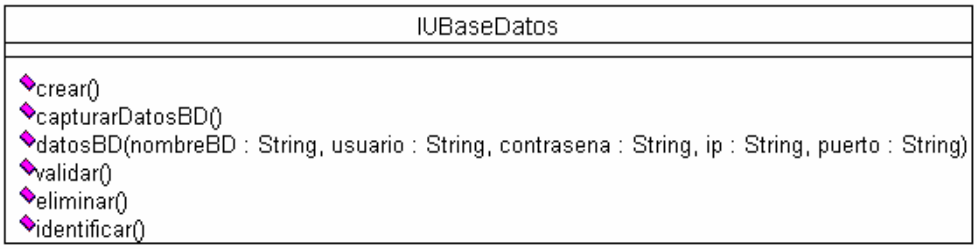
Figura 9. Detalle del Diagrama de Clases

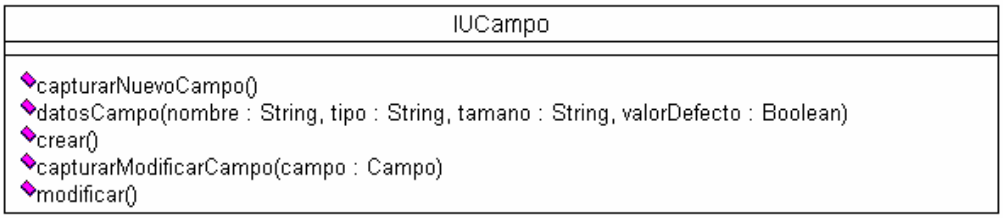


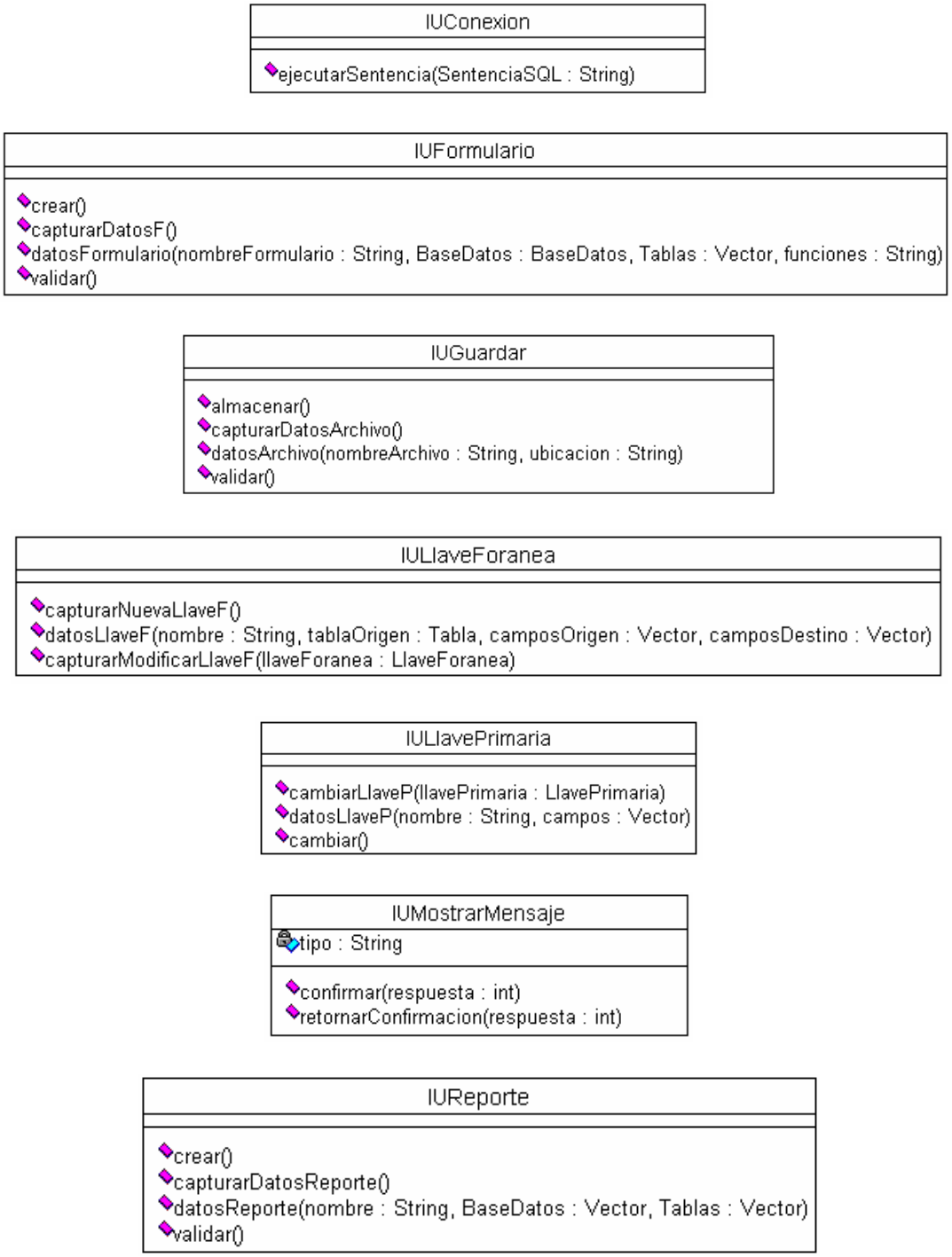

GestorBD
basesDatos : Vector = null
GestorBD()
crearBD(nombreBD : String, usuario : String, contrasena : String, ip : String, puerto : String)
eliminar(BaseDatos : BaseDatos)
cerrarBD(BaseDatos : BaseDatos)
identificarBD(nombre : String, usuario : String, contrasena : String, ip : String, puerto : String)
adicionarBD(BaseDatos : BaseDatos)
GestorFormulario
crear(nombreFormulario : String, BaseDatos : BaseDatos, tablas : Vector, funciones : String)
adicionarFormulario(formulario : Formulario)
eliminarFormulario(formulario : Formulario)
cerrarFormulario(formulario : Formulario)
modificarFormulario(formulario : Formulario)
mostrar()
Sentinela
runConexiones()
runBD()
run()
GestorReporte
crear(nombre : String, BaseDatos : BaseDatos, Tablas : Vector)
adicionarReporte(reporte : Reporte)
eliminar(reporte : Reporte)
cerrarReporte(reporte : Reporte)
modificarReporte(reporte : Reporte)
mostrar()

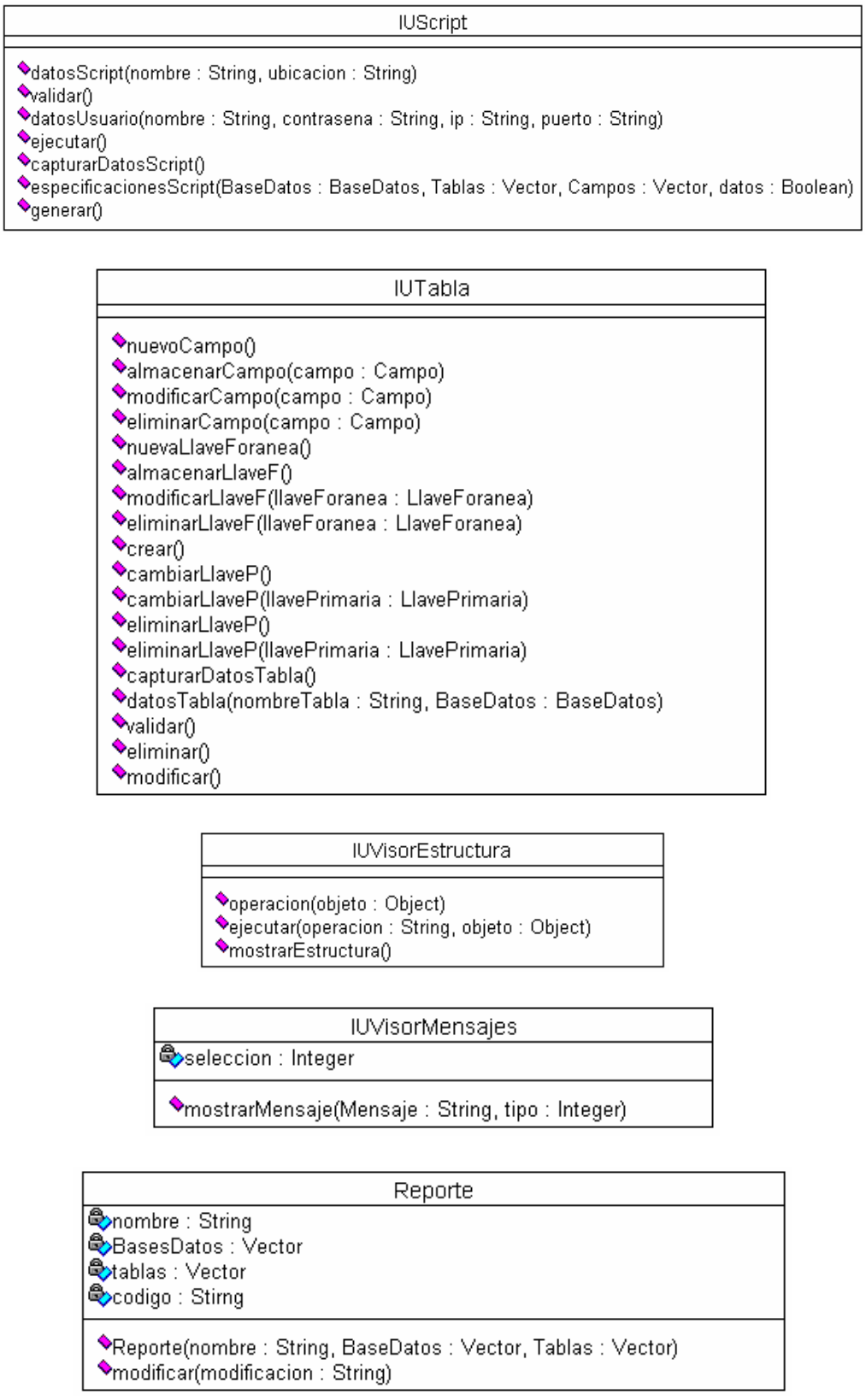

GRFPostgres
gestor1 : GestorBD
gestorFormulario : GestorFormulario
gestorReporte : GestorReporte
Formularios : Vector
Reportes : Vector
gestorIntegridad : GestorIntegridad
venBD : IUBD
venTabla : IUTabla
venFormulario : IUFormulario
venReporte : IUReporte
venGuardar : IUGuardar
venScript : IUScript
visorMensajes : IUVisorMensajes
compilador : Compilador
compilar()
CapturarObjeto()
ejecutar()
eliminarFormulario()
eliminarReporte()
capturarBD()
capturarFormularios()
capturarReportes()
main()


GestorTabla
crear(nombre : String, BaseDatos : BaseDatos, campos : Vector, llavePrimaria : LlavePrimaria, llavesForaneas : Vector)
adicionarTabla(Tabla)
eliminar(nombreTabla : String, BaseDatos : BaseDatos)
modificar(nombre : String, baseDatos : BaseDatos, campos : Vector, llavePrimaria : LlavePrimaria, llavesForaneas : Vector)


IUBaseDatos
crear()
capturarDatosBD()
datosBD(nombreBD : String, usuario : String, contrasena : String, ip : String, puerto : String)
validar()
eliminar()
identificar()


IUCampo
capturarNuevoCampo()
datosCampo(nombre : String, tipo : String, tamano : String, valorDefecto : Boolean)
crear()
capturarModificarCampo(campo : Campo)
modificar()


IUConexion
ejecutarSentencia(SentenciaSQL : String)
IUFormulario
crear()
capturarDatosF()
datosFormulario(nombreFormulario : String, BaseDatos : BaseDatos, Tablas : Vector, funciones : String)
validar()
IUGuardar
almacenar()
capturarDatosArchivo()
datosArchivo(nombreArchivo : String, ubicacion : String)
validar()
IULlaveForanea
capturarNuevaLlaveF()
datosLlaveF(nombre : String, tablaOrigen : Tabla, camposOrigen : Vector, camposDestino : Vector)
capturarModificarLlaveF(llaveForanea : LlaveForanea)
IULlavePrimaria
cambiarLlaveP(llavePrimaria : LlavePrimaria)
datosLlaveP(nombre : String, campos : Vector)
cambiar()
IUMostrarMensaje
tipo : String
confirmar(respuesta : int)
retornarConfirmacion(respuesta : int)
IUReporte
crear()
capturarDatosReporte()
datosReporte(nombre : String, BaseDatos : Vector, Tablas : Vector)
validar()


IUScript
datosScript(nombre : String, ubicacion : String)
validar()
datosUsuario(nombre : String, contrasena : String, ip : String, puerto : String)
ejecutar()
capturarDatosScript()
especificacionesScript(BaseDatos : BaseDatos, Tablas : Vector, Campos : Vector, datos : Boolean)
generar()
IUTabla
nuevoCampo()
almacenarCampo(campo : Campo)
modificarCampo(campo : Campo)
eliminarCampo(campo : Campo)
nuevaLlaveForanea()
almacenarLlaveF()
modificarLlaveF(llaveForanea : LlaveForanea)
eliminarLlaveF(llaveForanea : LlaveForanea)
crear()
cambiarLlaveP()
cambiarLlaveP(llavePrimaria : LlavePrimaria)
eliminarLlaveP()
eliminarLlaveP(llavePrimaria : LlavePrimaria)
capturarDatosTabla()
datosTabla(nombreTabla : String, BaseDatos : BaseDatos)
validar()
eliminar()
modificar()
IUVisorEstructura
operacion(objeto : Object)
ejecutar(operacion : String, objeto : Object)
mostrarEstructura()
IUVisorMensajes
seleccion : Integer
mostrarMensaje(Mensaje : String, tipo : Integer)
Reporte
nombre : String
BasesDatos : Vector
tablas : Vector
codigo : Stirng
Reporte(nombre : String, BaseDatos : Vector, Tablas : Vector)
modificar(modificacion : String)

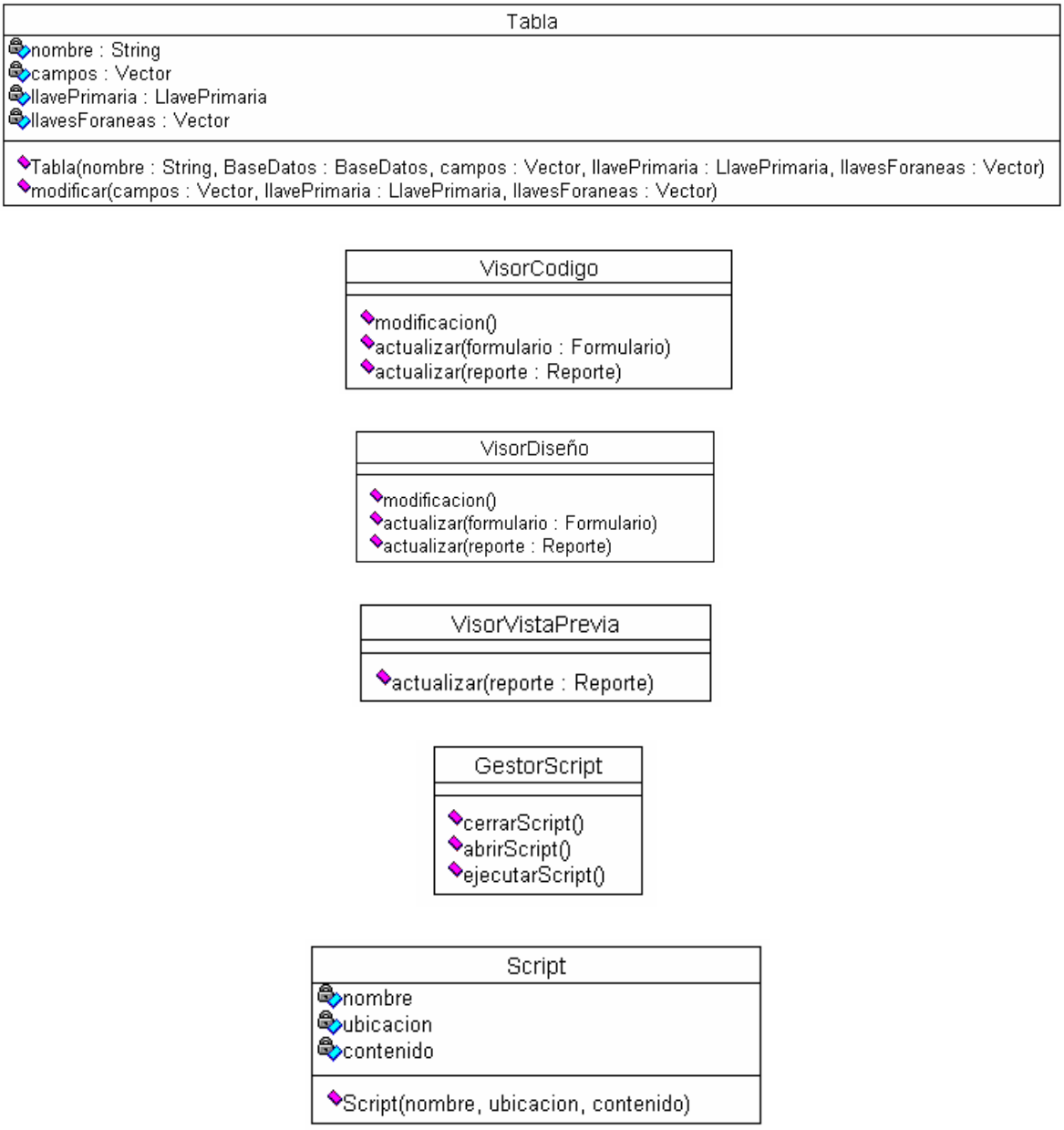
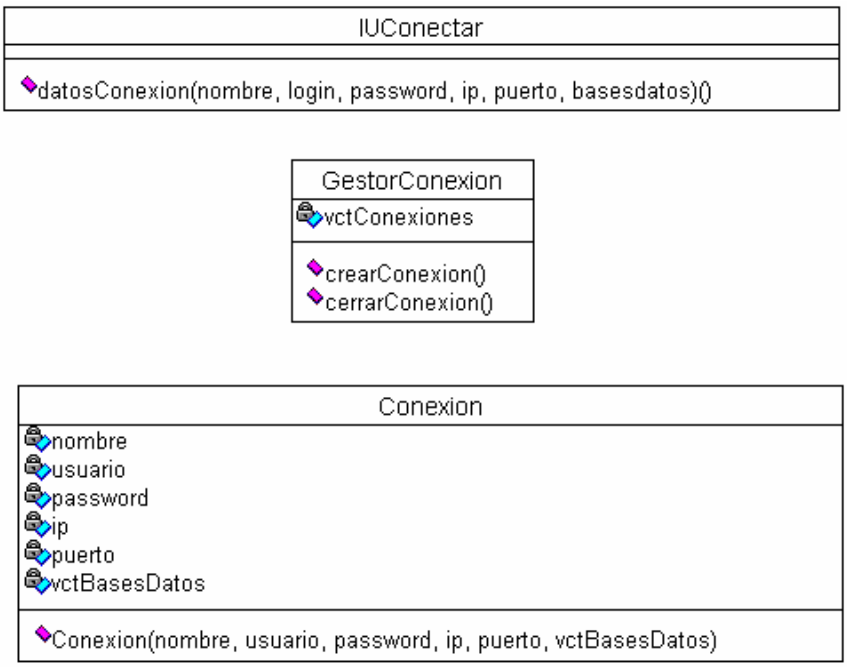


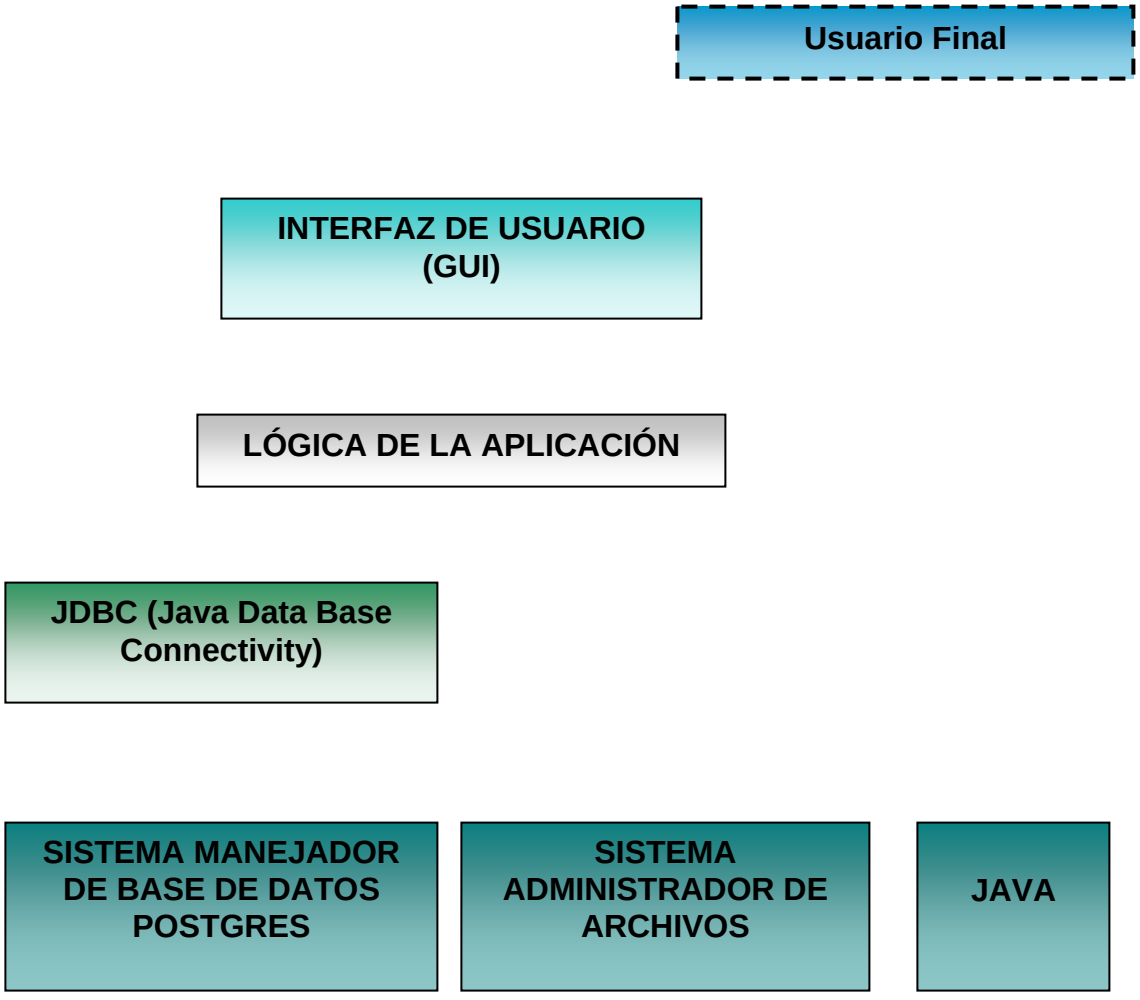
Tabla
nombre : String
campos : Vector
llavePrimaria : LlavePrimaria
llavesForaneas : Vector
Tabla(nombre : String, BaseDatos : BaseDatos, campos : Vector, llavePrimaria : LlavePrimaria, llavesForaneas : Vector)
modificar(campos : Vector, llavePrimaria : LlavePrimaria, llavesForaneas : Vector)
VisorCodigo
modificacion()
actualizar(formulario : Formulario)
actualizar(reporte : Reporte)
VisorDiseño
modificacion()
actualizar(formulario : Formulario)
actualizar(reporte : Reporte)
VisorVistaPrevia
actualizar(reporte : Reporte)
GestorScript
cerrarScript()
abrirScript()
ejecutarScript()
Script
nombre
ubicacion
contenido
Script(nombre, ubicacion, contenido)


IUConectar
datosConexion(nombre, login, password, ip, puerto, basesdatos)()
GestorConexion
vctConexiones
crearConexion()
cerrarConexion()
Conexion
nombre
usuario
password
ip
puerto
vctBasesDatos
Conexion(nombre, usuario, password, ip, puerto, vctBasesDatos)

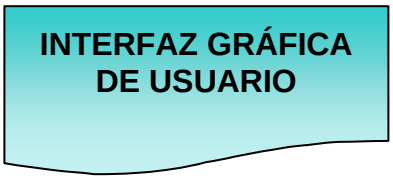
8.3.3 Diagrama de Paquetes de Arquitectura

Figura 10. Arquitectura del Sistema

DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 5	Modelo Conceptual GRFPostgres
Figura 7	Diagramas de Colaboración
Figura 8	Diagrama de Clases

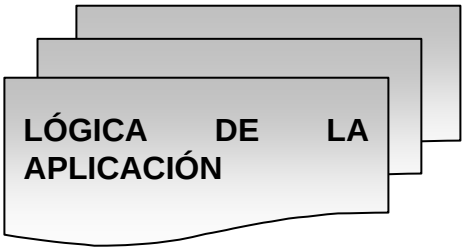


Esta arquitectura consta de cuatro capas, las cuales son:

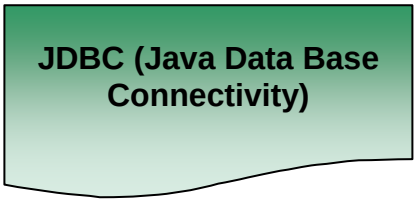


La herramienta proporciona una interfaz gráfica al usuario de tipo MDI, con ventanas internas hijas, área de trabajo, árbol de estructura, visor de mensajes, menús y barras de herramientas; la cual le permite al usuario una interfaz agradable y de fácil manejo

---

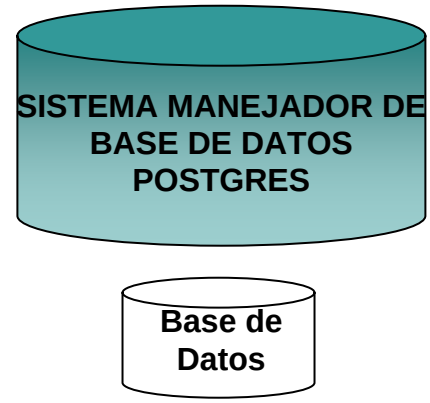


Las clases que componen la lógica de la aplicación permiten el control y gestión de las Bases de Datos y sus estructuras, así como de los formularios, los reportes y el funcionamiento de la aplicación en general.



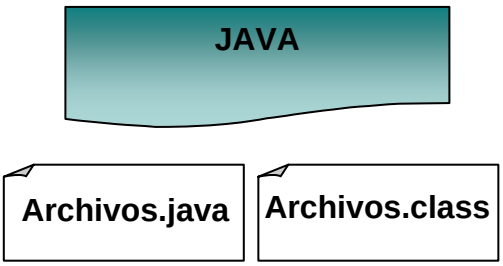
Brinda una interfaz con el servidor de Base de Datos, en este caso Postgres, para lo cual se utiliza el drive apropiado para conectarse con este sistema en particular.

En esta última capa se encuentran sistemas que interactúan en operaciones diferentes pero complementarias en los distintos módulos que posee la herramienta.



Postgres es el cargado del manejo directo sobre las bases de datos, la herramienta se encarga de interactuar con él y suministrarle al usuario una forma fácil y amigable de interactuar con él, las bases de datos y sus estructuras.

El sistema gestor de archivos interviene en la construcción de Scripts, formularios y reportes, este sistema varia de acuerdo al sistema operativo actual en el que esta instalada la herramienta, ya que la característica de aplicativo Java le permite al sistema ser instalado en diversos sistemas operativos.

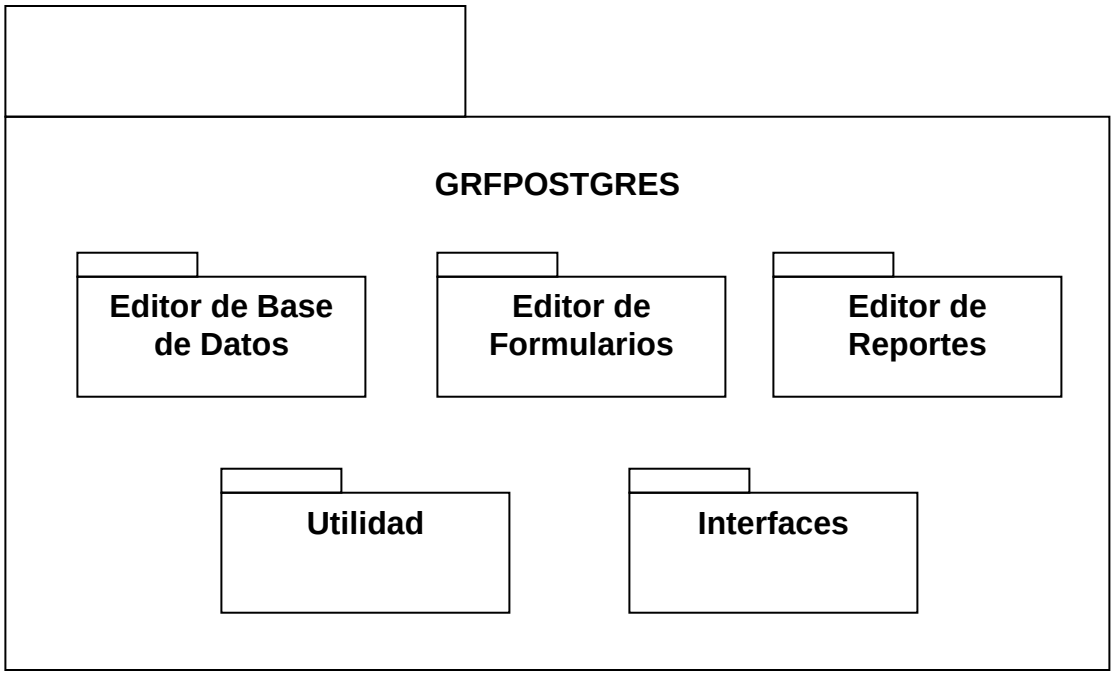


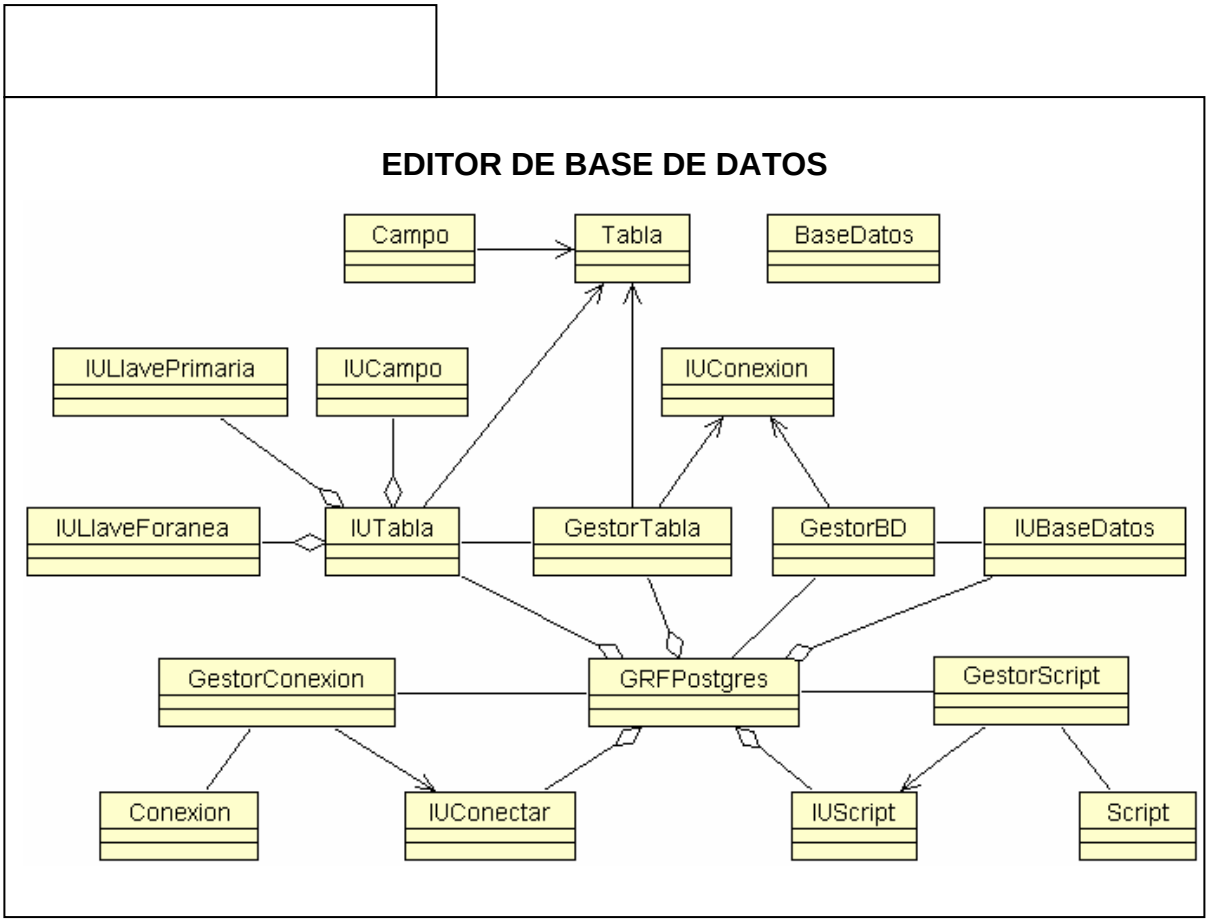
En la herramienta se permite la compilación y ejecución de los programas java que se estén construyendo en la misma (formularios y reportes), en este momento interviene el compilador y el intérprete de java.

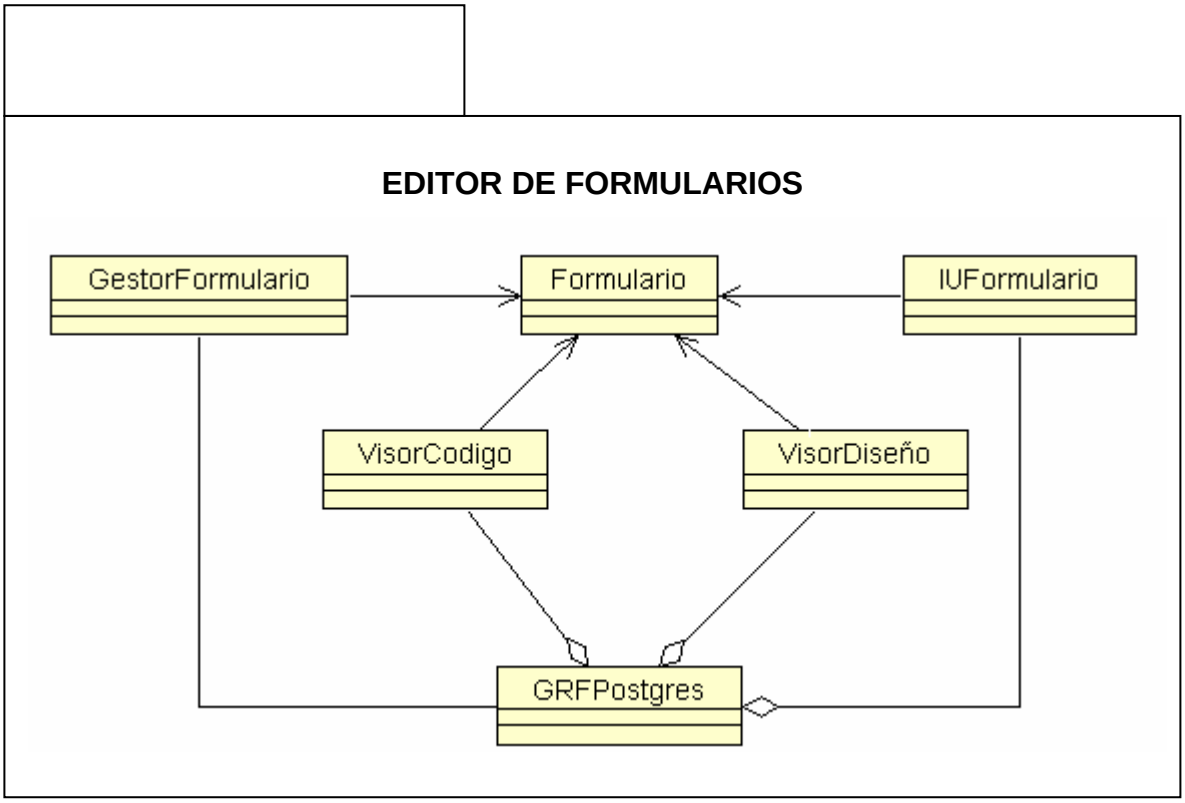
Figura 11. Paquetes

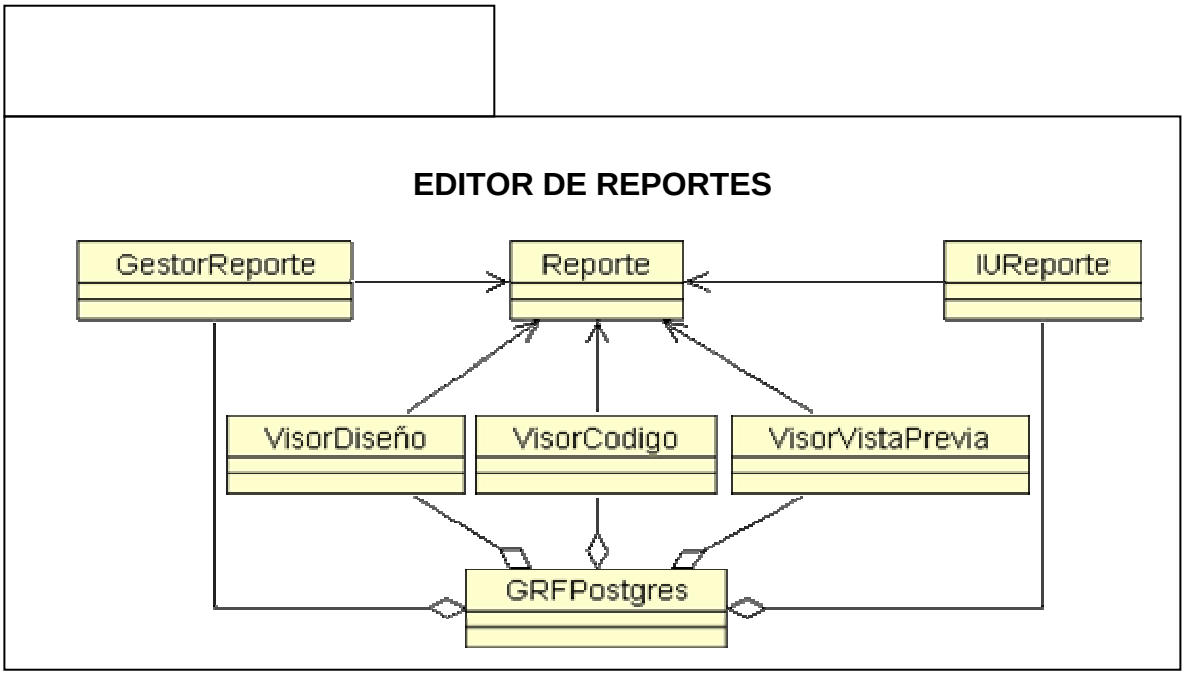
DOCUMENTOS DE REFERENCIA	
Documento número	**Título**
Figura 5	Modelo Conceptual GRFPostgres
Figura 7	Diagramas de Colaboración
Figura 8	Diagrama de Clases

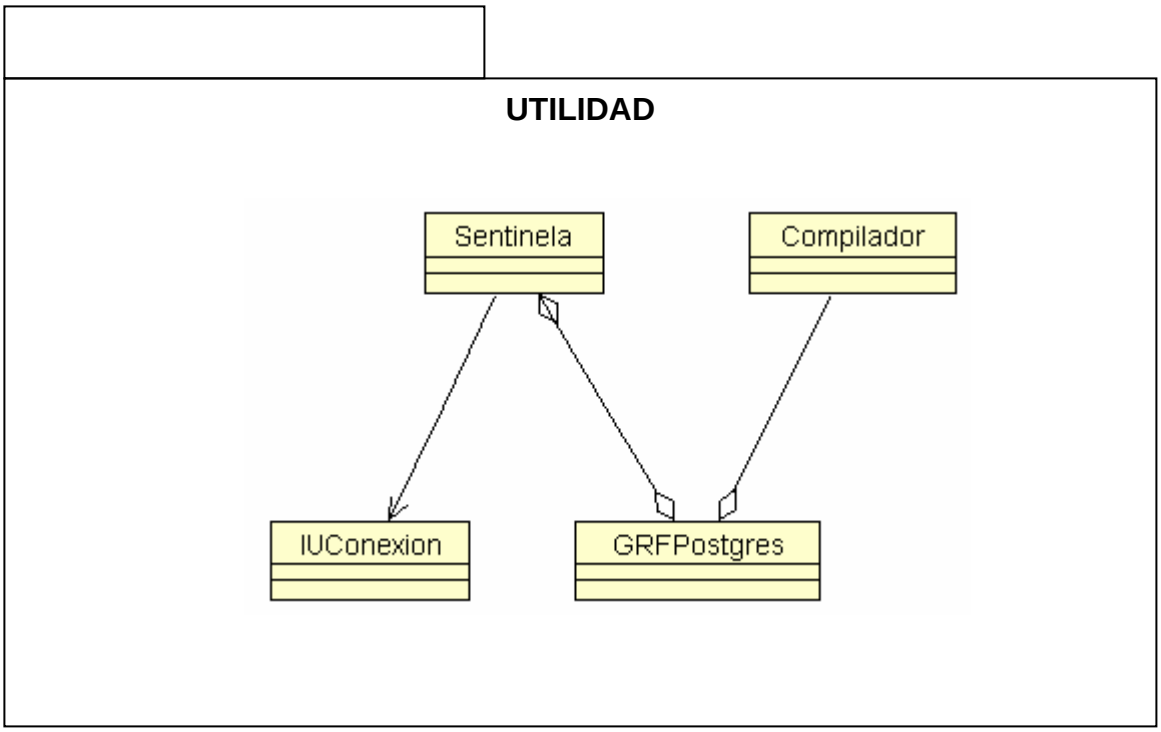
Paquete principal GRFPostgres

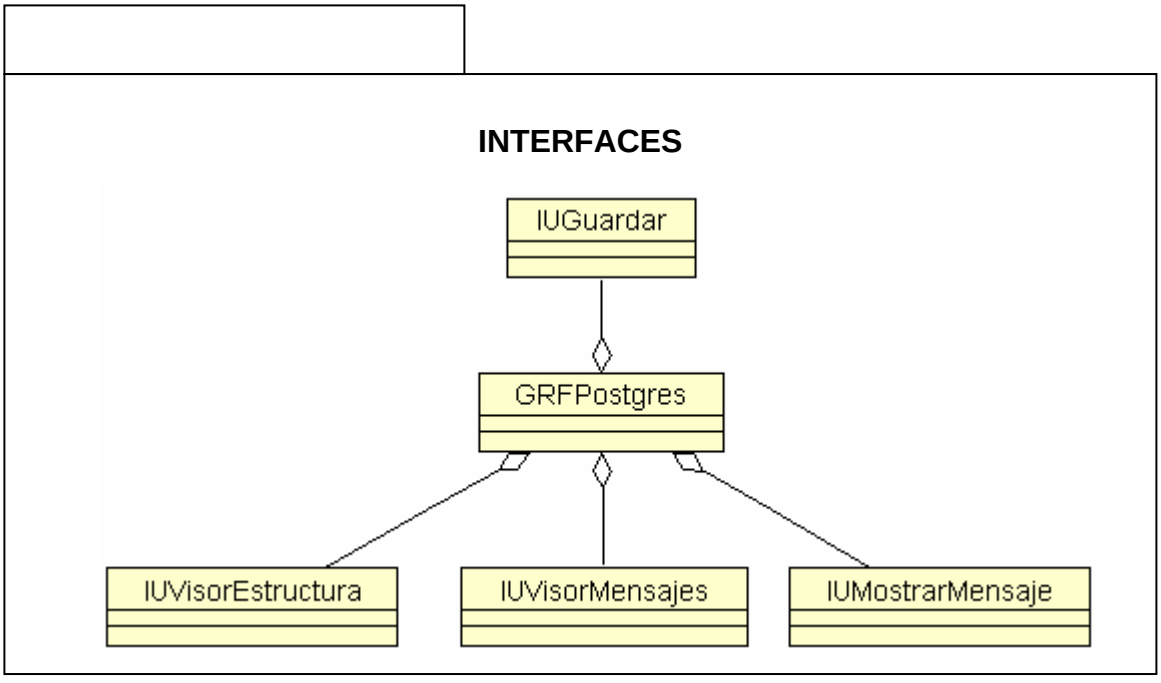



EDITOR DE BASE DE DATOS
Campo
Tabla
BaseDatos
IULlavePrimaria
IUCampo
IUConexion
IULlaveForanea
IUTabla
GestorTabla
GestorBD
IUBaseDatos
GestorConexion
GRFPostgres
GestorScript
Conexion
IUConectar
IUScript
Script


EDITOR DE FORMULARIOS
GestorFormulario
Formulario
IUFormulario
VisorCodigo
VisorDiseño
GRFPostgres


EDITOR DE REPORTES
GestorReporte
Reporte
IUReporte
VisorDiseño
VisorCodigo
VisorVistaPrevia
GRFPostgres


UTILIDAD
Sentinela
Compilador
IUConexion
GRFPostgres


INTERFACES
IUGuardar
GRFPostgres
IUVisorEstructura
IUVisorMensajes
IUMostrarMensaje

9. RESULTADOS

Como resultado de este proyecto de investigación y desarrollo se obtuvo una Herramienta Gráfica para el motor de Base de Datos Postgres, llamada GRFPostgres; desarrollada bajo la filosofía de UML, programada en el lenguaje JAVA, conformada por los siguientes módulos principales: Editor de Bases de Datos, Editor de Formularios; y dos módulos complementarios: Utilidad e Interfaces.

La Herramienta esta constituida como un paquete completo, bajo la licencia de software libre, y puede ser instalado en diversas plataformas gracias a estar programada en JAVA.

9.1 QUE HACE LA HERRAMIENTA

La versión de la herramienta que concluyo con este trabajo, en el módulo relacionado con la edición de estructuras SQL sobre el motor de base de datos PostgreSQL, permite realizar las siguientes operaciones:

Operaciones relacionadas con bases de datos

- 🕮 Abrir una base de datos. Seleccionada una conexión establecida en la herramienta, abrir una base de datos existente en el motor PostgreSQL escribiendo el nombre de la base de datos respectiva
- 🕮 Crear una base de datos. Seleccionada una conexión establecida previamente en la herramienta crear una nueva base de datos, suministrando entre otros datos: el nombre de la base de datos, una descripción, permitiendo escoger un espacio de tablas, un juego de caracteres de codificación, una plantilla de tablas y escribir el dueño de la base de datos.
- 🕮 Eliminar una base de datos. Seleccionada una base de datos previamente identificada o creada en la herramienta, eliminarla del motor PostgreSQL, pidiendo la confirmación de la operación por parte del usuario.

Operaciones relacionadas con tablas:

- 🕮 Crear una tabla. Seleccionada una base de datos, previamente identificada en la herramienta, crea una tabla en el motor postgreSQL y la registra en la herramienta, entre los datos que el usuario puede manipular al crear una tabla están:
  El nombre, el manejo de OIDs, el espacio de tablas, la herencia entre tablas
  Puede adicionar y eliminar campos con su interfaz apropiada en la cual se especifica: El tipo, el tipo especifico, dependiendo del tipo escogido, la precisión y la escala, el valor por defecto y si el campo permite valores nulos o no.

Puede adicionar y eliminar restricciones, de diferentes tipos como son:

1) Llaves primarias: con una interfaz específica que permite al usuario escoger el o los campos que formarán parte de la llave primaria.
2) Llaves foráneas, en su respectiva interfaz el usuario puede además de proporcionar el nombre, escoger la tabla referencia y la concordancia entre campos de ambas tablas, otras opciones a especificar en esta interfaz incluyen: el nivel de coincidencia, la acción al eliminar y/o al modificar un registro
3) Índices únicos y restricciones de chequeo, cada uno con la interfaz apropiada para definirlos.

- Editar una tabla. Seleccionada la tabla, la herramienta cuenta con la interfaz apropiada para modificar los campos y las restricciones de la tabla, algunas características de la tabla no se pueden modificar, pero se informan al usuario.
- Eliminar una tabla. Seleccionada la tabla, la herramienta presenta una interfaz que permite escoger el modo mediante el cual se eliminara una tabla, restringido o en cascada.

Como se puede apreciar lo que hace la herramienta en el módulo Editor de Base de datos, es básicamente capturar la información de una sentencia SQL especifica a través de una interfaz gráfica apropiada, dicha sentencia puede ser para crear, modificar o eliminar una estructura de base de datos, después la herramienta la transforma a una sentencia SQL textual valida, y utilizando el JDBC apropiado para el motor de base de datos PostgreSQL, ejecuta dicha sentencia SQL construida. Este mecanismo, implementado en la herramienta, es eficiente porque efectivamente realiza los cambios necesarios en el motor, pero adicionalmente se debe tener en cuenta que la herramienta posee el mecanismo apropiado para conservar la coherencia entre el motor y la información relacionada que la herramienta registra.

La versión de la herramienta en el módulo de Formularios permite realizar las siguientes operaciones:

- Crear un nuevo formulario. Seleccionado una conexión, una base de datos y una tabla, y proporcionando un nombre de formulario y un título, se genera un formulario con un solo control ya incluido, un control que conecta al JDBC de PostgreSQL, usando los datos de conexión seleccionados por el usuario.
- Generar un formulario. Seleccionando una conexión, una base de datos y una tabla, escogiendo los campos de la tabla cuyos registros se desean manipular por el formulario y proporcionando un nombre y titulo para el formulario. Se genera un formulario con los controles necesarios para manipular los registros de los campos de la tabla.
- Cerrar un formulario. Seleccionado un formulario creado o cargado en una base de datos en la herramienta al cerrarlo, el formulario se desvincula de la misma.
- Guardar y Abrir un formulario. Un formulario se guarda en lenguaje XML, referenciado a una base de datos, pero solo queda enlazado a la misma base

de datos original siempre y cuando la base de datos se cargue en la herramienta usando una conexión anteriormente realizada y almacenada en la herramienta. Para abrir un formulario el usuario debe haber seleccionado una base de datos, en el momento no se hace una verificación entre la base de datos seleccionada y la base de datos desde original con la cual se creo el formulario.

- 🕮 Guardar el código JAVA de forma independiente. Permite guardar el código Java generado por la herramienta en un archivo independiente para que el usuario pueda modificarlo en otra aplicación.
- 🕮 Compilar y ejecutar el código JAVA. Haciendo uso del compilador e interprete de JAVA, permitir que el código generado por la herramienta pueda ejecutarse sobre la misma y que el usuario pueda verificar los cambios realizados

El módulo de formularios fue diseñado para que un formulario ya sea creado o generado quede enlazado a una base de datos con el propósito de que los registros que se manipulen a través de este, estén relacionados por defecto a una tabla de una base de datos.

Para conseguir tal fin, se diseñaron componentes especiales que controlen la manipulación de registros en el formulario, entre ellos están:

- Un control que establezca la conexión con el motor PostgreSQL a través del JDBC apropiado con los datos de la conexión y ejecute una sentencia SQL de selección de todos los registros de una tabla específica o de acuerdo a la sentencia de selección suministrada por el usuario. Este objeto también contiene una relación de cada control del formulario relacionado con un campo de la sentencia SELECT que origina la colección de registros, mediante este objeto se manipulan el dato del campo en el registro actual, y también se supervisan el dato mostrado en dicho control de acuerdo con la posición del registro en un momento dado.
- Este control tiene además una serie de métodos que permiten insertar modificar y eliminar registros en la tabla seleccionada. Controla si la conexión con el motor esta abierta o cerrada.
- Las validaciones al insertar, modificar y eliminar un registro son capturadas directamente desde el motor PostgreSQL.
- Un control con un conjunto de opciones encargado de realizar las operaciones de inserción, modificación y eliminación de registros, este control debe estar enlazado al anterior y convoca a las operaciones apropiadas para realizar cada operación, los mensajes de error que se puedan generar en cada operación se obtienen del control enlazado y se muestran al usuario en un cuadro de dialogo.
- Un control destinado a recibir el dato que se quiere insertar, modificar o mostrar para un campo de un registro especifico. Este control debe referenciar al primer control y también debe contener una referencia de un campo que va a manipular.

Como aporte adicional al diseño del formulario, se incluyen otros controles que no están directamente relacionados con la manipulación de registros de una tabla, pero que se pueden programar para realizar otras funciones específicas, y otros simplemente sirven para el diseño.

El Editor de Formularios se pensó para que pueda incorporar nuevos controles al diseño de formularios, específicamente controles diseñados por el usuario para la manipulación de registros, de esta manera se libera la cantidad de controles que puede manipular la herramienta. Para realizar esto el usuario debe crear una clase especial que implemente interfaces de programación JAVA, ya diseñadas en la herramienta y que permiten su acoplamiento en la misma. La clase compilada de dicho componente debe ubicarse en un directorio específico desde el cual carga los controles la herramienta.

Otra característica adicional y que permite manipular las propiedades de un control es el visor de propiedades, que es un cuadro de dialogo mediante el cual el usuario pueden cambiar las características de diseño de un control especifico. Este visor solo muestra los atributos de diseño de un control de acuerdo a lo detallado en la clase que carga la herramienta relacionada con el control.

En cuanto a la manipulación del código JAVA generado, se analizo la estructura de archivo de programación realizado en este lenguaje, la estructura un archivo JAVA es siempre la misma. Una clase publica principal cuyo nombre es el mismo del archivo JAVA, y opcionalmente una secuencia de clases secundarias no públicas definidas a continuación.
Cada clase también posee una estructura definida, unos atributos, un método constructor y una secuencia de métodos opcionales. Una clase puede opcionalmente también tener clases internas declaradas.
La declaración de una clase en este lenguaje tiene una semántica definida, así como la declaración de un atributo global a la clase, de un método constructor y de un método específico, todo esto es parte de las reglas de producción del compilador del lenguaje JAVA. Esta semántica se tuvo en cuenta para diseñar el visor de código, un cuadro de dialogo mediante el cual se puede modificar el código generado parcialmente.

Todo esto se logro usando la tecnología Beans, incorporada en la plataforma JAVA, este conjunto de procesos, proporciona los medios estándar para crear los componentes de tal forma que JAVA posteriormente, pueda retornar la descripción de cada propiedad del mismo, el método de ese componente que permite modificar dicha propiedad y leer su valor actual. Además retorna la descripción de cada uno de los métodos que un componente tiene registrados, la descripción de cada uno de los parámetros que debe recibir dicho método. También retorna la descripción del conjunto de eventos registrados para ese componente.
Es conveniente aclarar, que el código generado para un control por el editor de formularios cuando se crea, o se modifica una de las propiedades no se puede

alterar por el editor de código de esta manera se garantiza la coherencia entre lo que se muestra en el diseño con lo que se muestra en el código.

Ahora, también se permite guardar el formulario diseñado a través de la herramienta usando el lenguaje XML, JAVA ofrece la posibilidad de guardar y posteriormente leer un archivo con este formato, pero además permite que un objeto pueda almacenarse en este formato conservando los valores de las propiedades del mismo, y luego, al momento de leerlo se pueda reconstruir dicho objeto con las propiedades almacenada. De esta forma se guardan los formularios y posteriormente se pueden abrir, siempre y cuando se enlacen a una base de datos específica.

El módulo Editor de Reportes, se analizo y diseño, incluso se programaron algunas interfaces que se utilizarían en la herramienta pero debido a la dimensión del proyecto, y tras haber cumplido e incluso extendido en más de una ocasión el período propuesto para el desarrollo del mismo no se logro terminar la implementación del módulo. En versiones anteriores de la herramienta el módulo alcanzo a ser implementado.

Algunos módulos complementarios pero trascendentales involucran la edición de scripts, se entiende por un script un archivo con un conjunto de instrucciones SQL, con operaciones como la generación de scripts sobre estructuras de la base de datos registradas en la herramienta, la exportación de un script en archivos que se puedan manipular a otros motores y la importación de un script externo para ser ejecutado en la herramienta.
La ejecución de un script utiliza el mismo mecanismo implementado para la ejecución de sentencias SQL construidas a partir de interfaces graficas de usuario.

Otro módulo importante es la Gestión de Conexiones, una conexión es un elemento importante en la herramienta, en ella se obtienen los datos mediante los cuales se la herramienta se conecta con una o más bases de datos que se relacionen con los datos de una conexión.
La herramienta permite crear nuevas conexiones y cerrar conexiones existentes, además con cada operación realizada verifica la coherencia herramienta – motor PostgreSQL.

9.2 ESPECIFICACIONES DEL DESARROLLO

Al finalizar este proyecto, la herramienta esta desarrollada, de acuerdo a las siguientes especificaciones:

Utilizando el lenguaje de programación JAVA2, utilizando el JDK (Java Development Kit) version1.5, utilizando especialmente las librerías:
- java.sql;
- java.beans;

- javax.swing: Librería grafica para la creación de interfaces de usuario. Entre las ventajas que posee: un amplio conjunto de controles de diseño, la navegación con el teclado entre ellos es automática, cualquier aplicación Swing se puede utilizar sin ratón, sin tener que escribir ni una línea de código adicional. Las etiquetas de información, o "tool tips", se pueden crear con una sola línea de código. Además, en Swing la apariencia de la aplicación se adapta dinámicamente al sistema operativo y plataforma en que esté corriendo.
- java.io;
- java.lang.reflect;

Utilizando el JDBC (Java Data Base Connectivity) versión 3.0, compatible con el motor de base de datos PostgreSQL.

La lógica de programación se desarrollo teniendo en cuenta la compatibilidad con el motor de PostgreSQL, versión 8.0, que trabaja sobre el sistema operativo Windows en la versión XP.

Las pruebas finales se realizaron ejecutándose la herramienta en el sistema operativo Windows XP.

9.3 REQUERIMIENTOS DEL SISTEMA

La versión de la herramienta que se desarrollo con este proyecto, requiere para su instalación y funcionamiento:
En el software:
- Instalar el motor de base de datos PostgreSQL versión 8.0, sobre la cual fue probada la herramienta
- Realizar la configuración necesaria para levantar el demonio postmaster, con la opción adecuada para que postgreSQL levante el puerto o socket para hacer transmisiones vía TCP/IP.
- En el momento se entrega la herramienta con su código fuente y los archivos compilados correspondientes, por lo tanto la ejecución de la herramienta se debe hacer utilizando el archivo interprete de JAVA, java.exe para la versión SDK 1.5, en futuras versiones se creará el instalador de la herramienta para usar en su ejecución la máquina virtual del sistema donde se instale.

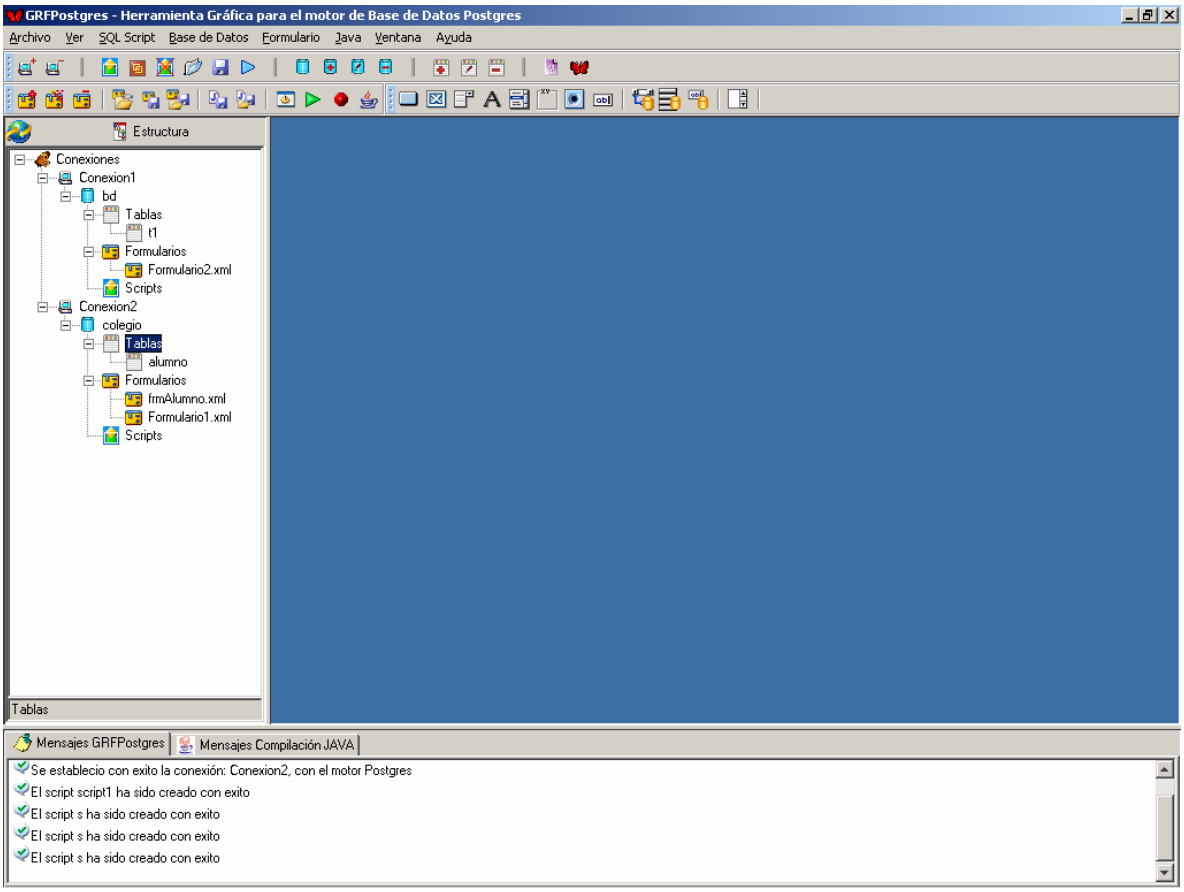
En el hardware:
- Un computador con 5 MB disponibles en disco duro para instalar la herramienta, en memoria RAM mínimo 128 MB, pantalla, teclado y Mouse.

9.4 MANUAL DEL USUARIO DE LA APLICACIÓN

La herramienta GRFPostgres, inicia con una interfaz de usuario principal de tipo MDI, que encapsula las demás interfaces de usuario y cuadros de diálogo, liderando y controlando las operaciones ha realizar.

Esta ventana principal se divide en cuatro secciones: Menús y Barra de Herramientas, Árbol de estructura, Área de trabajo y Visor de Mensajes.

Figura 12. GRFPostgres – Interfaz Principal



A continuación se explicarán con más detalle cada una de las secciones del aplicativo.

1. Menús y Barras de Herramientas. En la parte superior de la ventana principal se encuentran los menús y las barras de herramientas, cuya función es la de guiar al usuario hacia las funciones y operaciones que realiza GRFPostgres.

Aquí se encuentran ocho menús con sus respectivas operaciones: archivo, ver, base de datos, tabla, formulario, reporte, herramientas y ventana; y dos barras de herramientas: principal y de diseño.
En los menús se encuentran:

Figura 13. Menús y Barras de Herramientas

Menú Archivo. En este menú se encuentran: Conectar, Desconectar, Desconectar Todo, Conexiones Recientes y Salir.

Menú Ver. Opciones de visibilidad como vista de las diferentes barras de herramientas y secciones de la herramienta.

Menú SQL Script. Operaciones relacionadas con script como son: Abrir SQL, Guardar SQL, Nuevo SQL, Exportar SQL, Cerrar SQL, Ejecutar.

Menú Base de Datos. Menú con operaciones sobre las bases de datos como son: abrir BD, cerrar BD; y operaciones sobre bases de datos y tablas como son: crear, editar y eliminar.

Menú Formulario. Contiene las operaciones que pueden realizarse sobre los formularios como son: nuevo, generar y cerrar, además de abrir, guardar, guardar como y recientes formularios abiertos.

Menú Java. En este menú están las siguientes operaciones: Guardar Java, Guardar como Java, Compilar, Iniciar, Terminar y Configurar.

Menú Ventana. Encapsula las operaciones de visión de ventanas hijas MDI que se encuentren actualmente visibles en el área de trabajo de la ventana principal. Estas operaciones son: Cascada, vertical, horizontal, minimizar, cerrar, cerrar todas y selección de ventanas.

Menú Ayuda. Con las opciones de ayuda GRFPostgres y acerca de GRFPostgres.

En las barras de herramientas están:

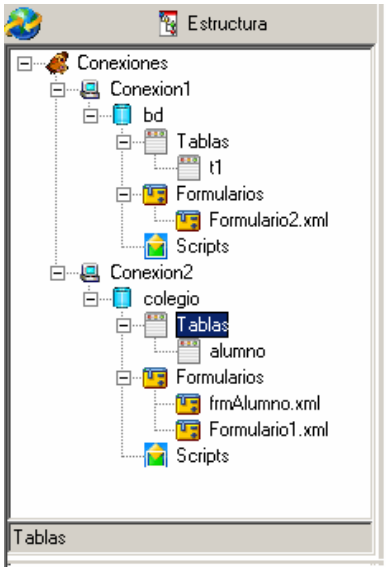
- Barra Base de Datos. Con subgrupos de conexiones, scripts, bases de datos, tablas y ayuda.
- Barra de Formularios. Las operaciones básicas que se pueden realizar sobre los formularios.
- Barra de Diseño de Formularios. Con elementos básicos Swing, para diseñar formularios, además de elementos propios de la herramienta relacionados con el manejo de bases de datos.

2. Árbol de estructura. En el árbol de estructura se pueden observar un nodo principal del cual se desprenden las conexiones, dentro de cada conexión se

observan las bases de datos que la componen, en cada nodo base de datos se tienen tres nodos: tablas, formularios y scripts. En cada uno de los cuales se listan las tablas, formularios y scripts relacionados con la base de datos correspondiente.

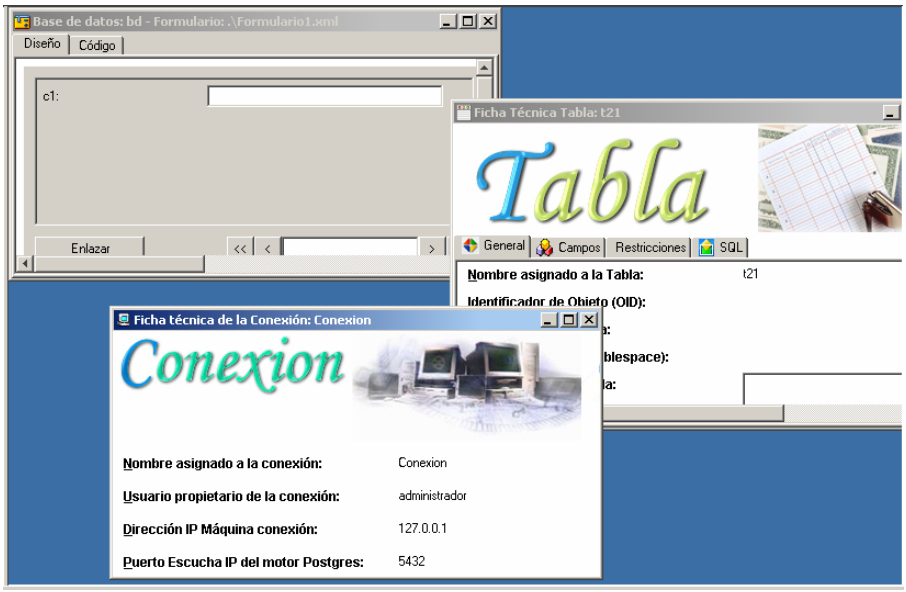
Cada uno de los nodos proporciona al dar clic derecho un menú emergente con opciones típicas de ese objeto. Ejemplo: en el nodo Base de Datos se proporcionan las opciones de editar, cerrar, eliminar y nueva tabla, además de una opción de expandir o contraer que permite ver u ocultar los nodos hijos respectivos.

Figura 14. Árbol de estructura



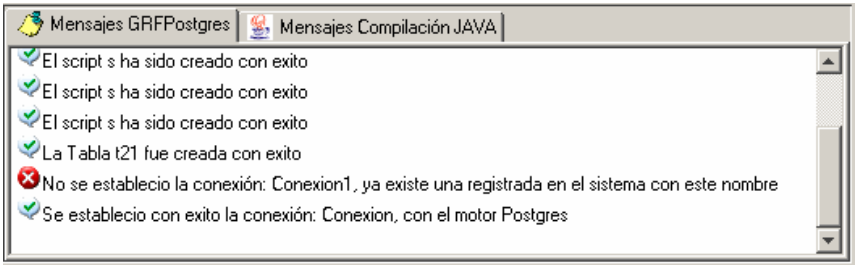
3. Área de Trabajo. En esta área de trabajo se encontrarán las ventanas hijas MDI, sobre esta parte de la ventana principal se aplican las opciones del menú ventana. Estas ventanas hijas son: la vista de formularios compuesta de vista código y diseño, fichas técnicas de los elementos: conexiones, base de datos, tabla script.

Figura 15. Área de Trabajo



4. Visor de Mensajes. En este elemento constituido por dos interfaces diferenciadas por pestañas la herramienta emite mensajes de registro de las operaciones que se realizan. La primera pestaña representa los mensajes propios de la herramienta, mientras en la segunda se tienen los mensajes emitidos por el compilador Java en procesos de compilación y ejecución.

Figura 16. Visor de Mensajes

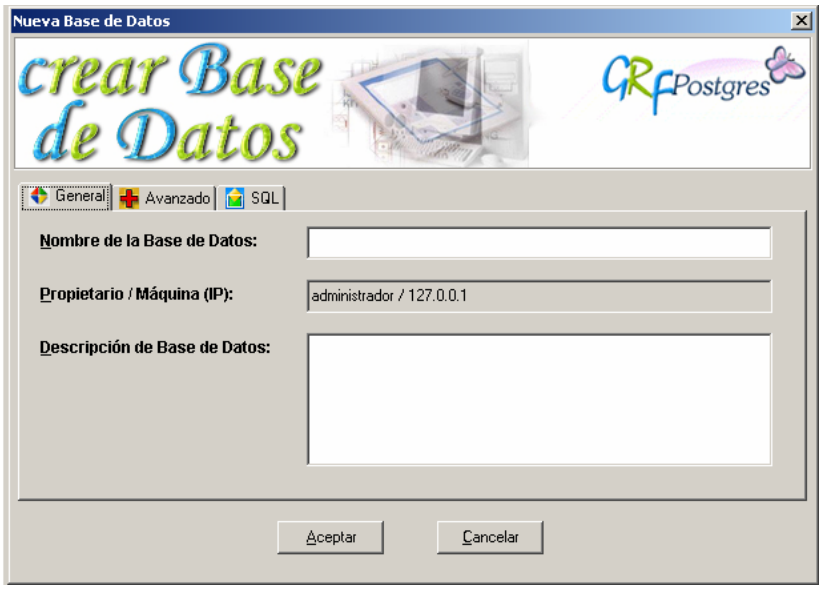


9.4.1 Editor de Base de Datos. El editor de base de datos esta conformado por las operaciones básicas que se puedan realizar sobre las bases de datos y sus estructuras: como son tablas, campos, llaves primarias y foráneas, índices únicos y restricciones de chequeo.

Las operaciones que se pueden realizar son: abrir, crear, editar, cerrar y eliminar sobre bases de datos; crear, modificar, consultar y eliminar sobre tablas, campos, llaves primarias y foráneas, índices únicos y restricciones de chequeo.

Para el manejo de Bases de Datos existen interfaces de tipo cuadro de diálogo, en la cuales se pueden realizar las operaciones respectivas anteriormente descritas.

Figura 17. Interfaz para crear Bases de Datos



En cuanto a las tablas se tiene interfaces para crear, modificar y eliminar.

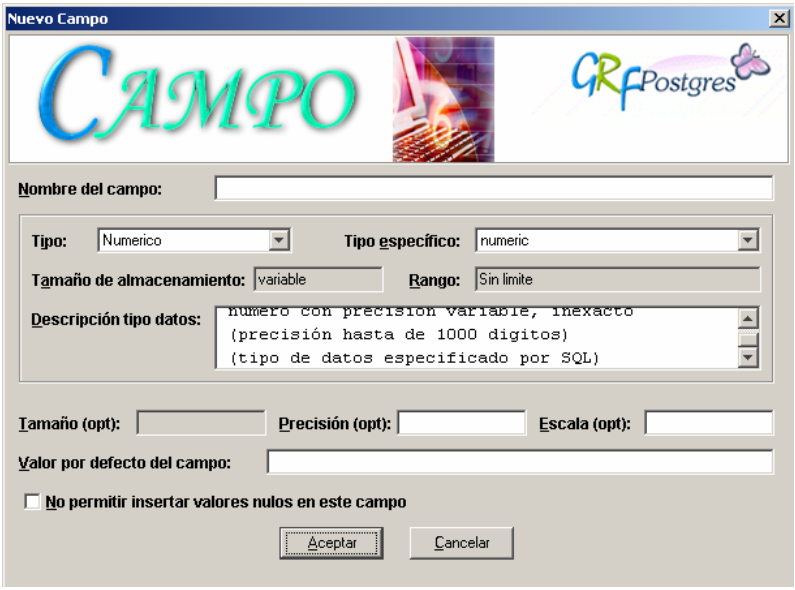
Esta interfaz se divide en cuatro secciones:

- Sección General
- Sección de Campos
- Sección de Restricciones
- Sección de SQL

Figura 18. Interfaz de creación de Tabla

Para la asignación y modificación de campos se tiene el siguiente cuadro de diálogo que se desprende de la interfaz de tabla, en el cual se pueden entrega datos del campo como son su nombre, tipo, tamaño si aplica, valor por defecto y otros datos de información como nulos, duplicados.

Figura 19. Interfaz de Campo

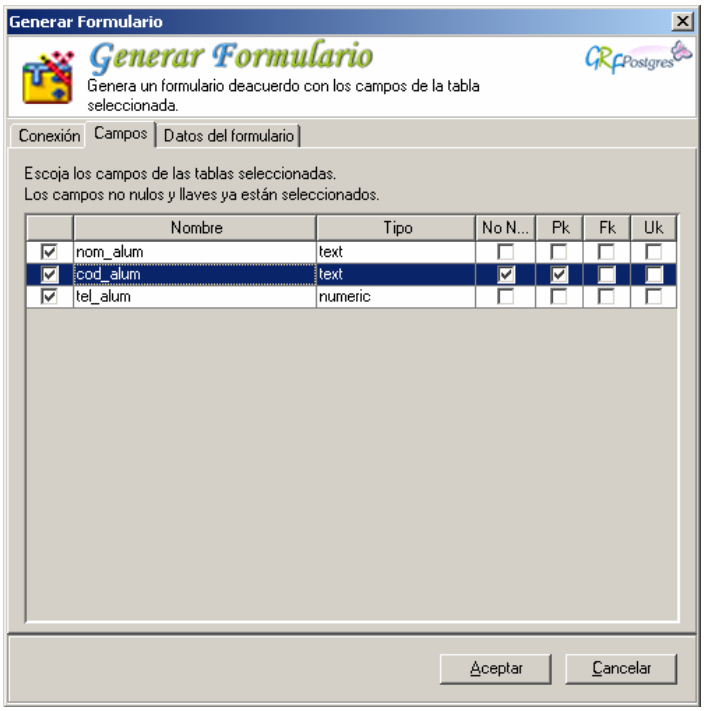


En cuanto a restricciones se tiene una interfaz en la cual el usuario define su nombre y los datos que la componen; esta interfaz es dependiente de la interfaz de tabla.

Figura 20. Interfaz de Restricción

9.4.2 Editor de Formularios. El editor de formularios en cuanto a interfaces se trata se compone de una estructura más compleja. En primer lugar se posee un asistente que guía al usuario para la generación automática de formularios relacionados con una o varias tablas de una base de datos.

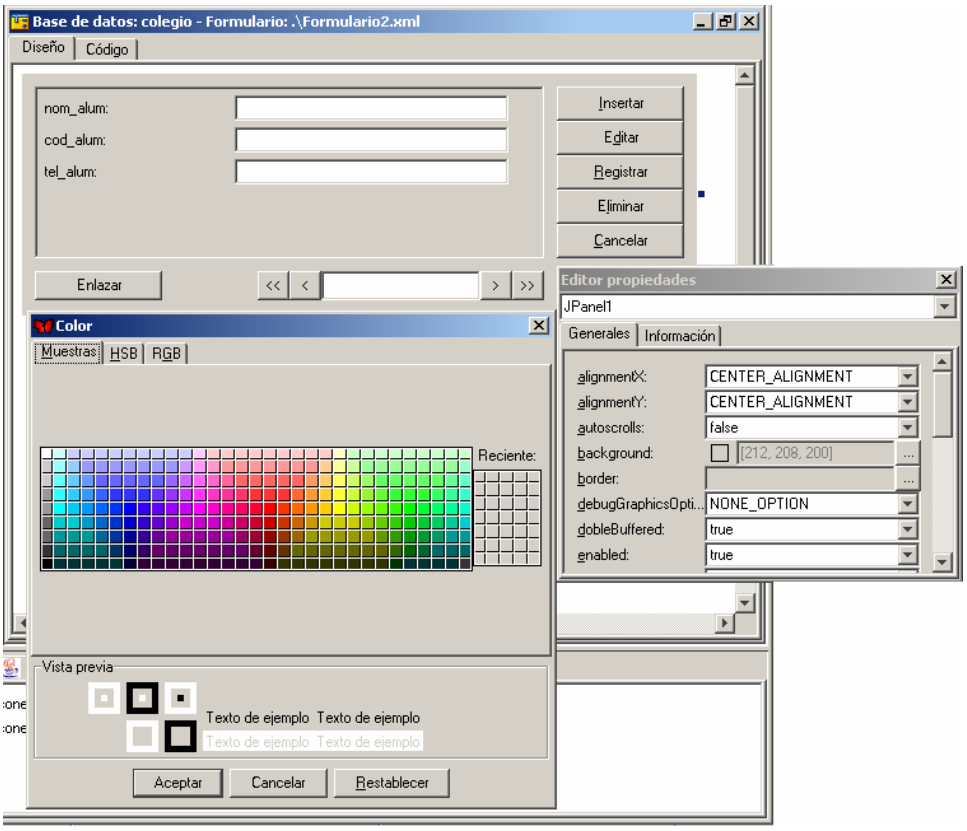
Figura 21. Asistente en la generación de formularios



Luego de este asistente se pasa a dos vistas del formularios: una vista de diseño y una vista código, en estas vistas que se encuentran como ventanas hijas en el área de trabajo, el usuario puede realizar modificación a su gusto, guardar el formulario como archivo independiente JAVA, compilar dicho formulario, e incluso verlo en ejecución desde la herramienta.

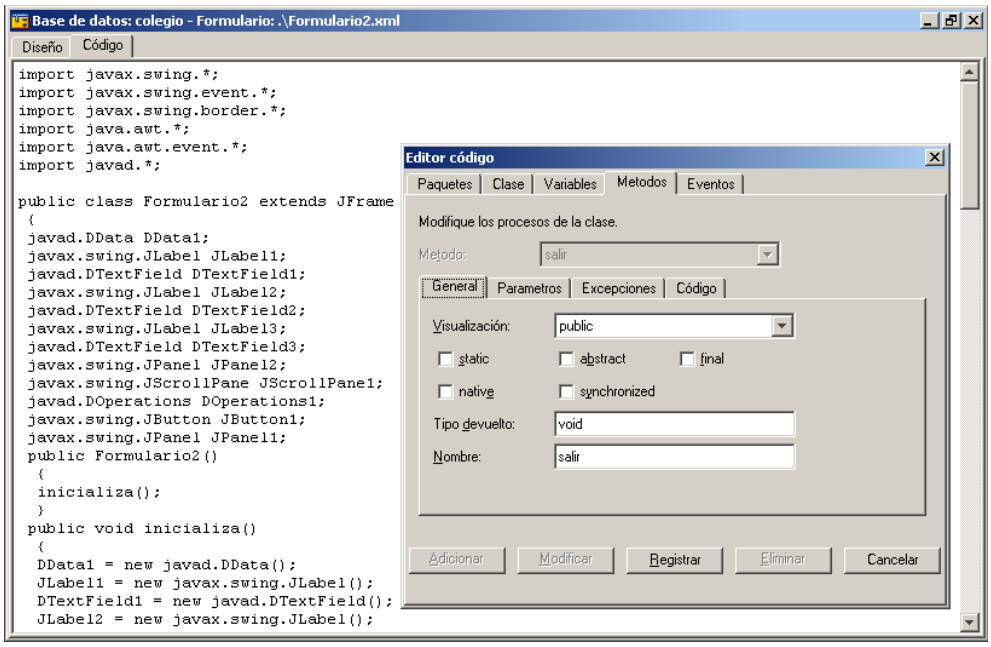
Vista de diseño. En esta vista se pueden realizar modificaciones de diseño sobre el formulario. Se presenta una imagen del formulario, sobre la cual el usuario puede mover elementos como cajas de texto, etiquetas, combos, etc. y modificar las propiedades de estos elementos a través de una ventana de diálogo dispuesta para este trabajo, como son colores de fondo y texto, nombres, ubicación, tamaño, etc.

Figura 22. Vista de diseño de un formulario



Vista código. En la vista código el usuario tiene una visión del código Java que se genera para el funcionamiento de este formulario, también puede adicionar código de eventos y métodos, a través de un cuadro de diálogo dispuesto para eso.

Figura 23. Vista código de un formulario

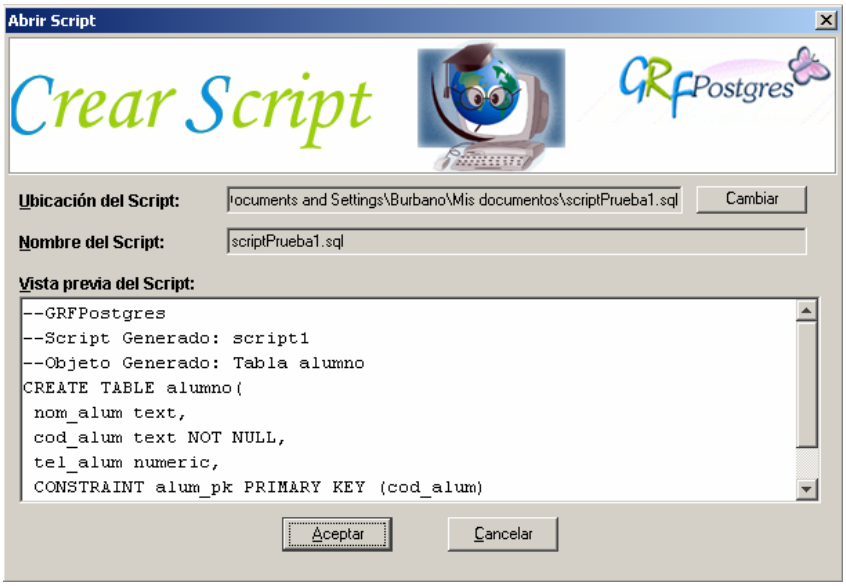


A parte de la generación automática de formularios por medio del asistente, el usuario puede generar un nuevo formulario en blanco para ser completamente diseñado a su gusto en las vistas de diseño y código. Para esto está la opción Nuevo en el menú formulario.

9.4.3 Conexiones y Scripts. Estas secciones de la herramienta permiten el manejo de conexiones y de script de estructuras de bases de datos y tablas. La sección conexiones permite operaciones como crear y cerrar conexiones. La sección de scripts tiene acciones como: nuevo Script, exportar Script, guardar Script, y ejecutar.

Figura 24. Interfaz de conexión

Figura 25. Interfaz de abrir Script

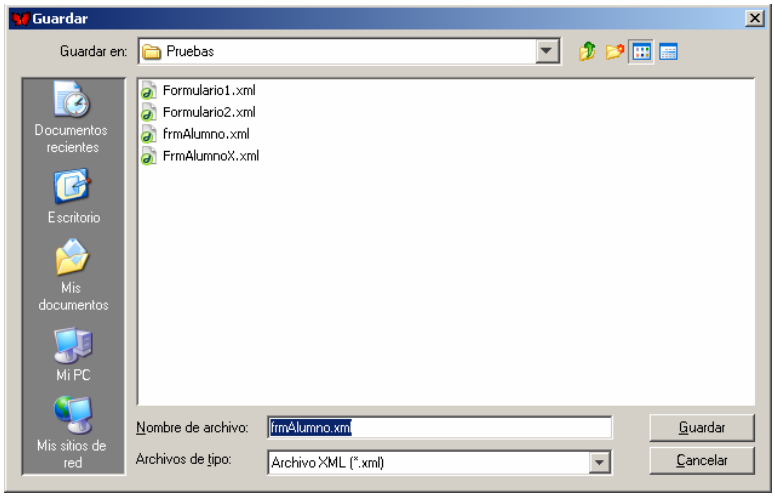


9.4.4 Utilidad e Interfaces. Estos módulos se componen de utilidades complementarias a los módulos principales descritos anteriormente, entre las que encontramos:

Guardar Elemento. Para guardar elementos como scripts, formularios y el código JAVA, se tiene esta operación a la cual puede accederse desde el menú archivo,

opciones guardar o guardar como, o desde la barra de herramientas principal. En ambos casos se lanza una ventana estándar de guardado en la cual se puede elegir la ubicación y nombre del archivo.

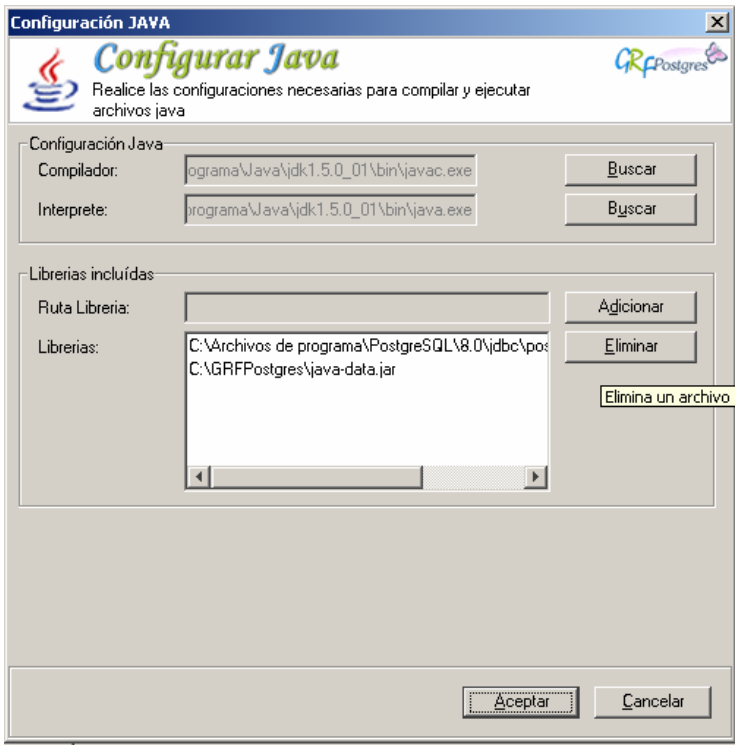
Figura 26. Ventana Guardar



Compilación y Ejecución. Estas operaciones se aplican sobre formularios. Se accede a estos procedimientos a través de la barra de herramientas principal. Y su función es vincular el compilador JAVA a los archivos JAVA para obtener una compilación o ejecución de los mismos, si ocurre un error este mensaje es emitido por medio del visor de mensajes en la pestaña de “Mensajes Compilación JAVA”.

Configurar. En la opción de configuración del menú herramientas, se puede entregar la ubicación del compilador e interprete JAVA, por medio de una ventana de diálogo sencilla.

Figura 27. Ventana de Configuración



10. TRABAJOS FUTUROS

Una visión global del desarrollo de la herramienta permite concluir que los módulos implementados hasta la versión actual de la herramienta pueden ser complementados, y desde luego se puede terminar de implementar el último módulo de la herramienta relacionado con la Edición de Reportes.

Algunas de las propuestas cuya implementación puede realizarse rápidamente puesto que se analizaron y diseñaron, pero no se terminaron de hacer son:

- Hay muchas otras sentencias SQL que se pueden implementar en la herramienta a través de interfaces graficas eficientes, pero siempre se ejecutaran en el motor usando el mismo mecanismo ya implementado con el JDBC.

- La perspectiva para el mecanismo que permite la incorporación de nuevos objetos para la manipulación y consulta de registros, es diseñar nuevos controles apropiados para cada tipo de datos que puede tener un campo, entre otros: VARCHAR, CHAR, BOOLEAN, FLOAT, DATE, etc., propios del motor PostgreSQL. En el momento solo se tiene diseñado un control que permita recibir el dato de un campo que se quiera insertar, modificar o mostrar que esta funcionado para los tipos VARCHAR, CHAR, BOOLEAN, NUMERIC, INTEGER FLOAT y DATE.

- La implementación de todo el módulo Editor de Reportes, en una versión inicial la cual ya se ha planeado. Entre las características proyectadas desarrollar en este módulo están:
  - El módulo Editor de Reportes se diseño para reutilizar clases y mecanismos implementados en el módulo Editor de Formularios. Esto se debe a que ambos módulos conservan características comunes como la manipulación de las propiedades de los controles seleccionados.
  - Un reporte también estará ligado a una base de datos específica, un reporte debe tener la posibilidad de crear un nuevo reporte en limpio con el objeto que enlaza a la base de datos ya incorporado, o de generar un reporte de una tabla con los campos seleccionados que formaran parte del mismo. Se debe solicitar el nombre del reporte y el titulo del mismo. También se debe especificar la distribución del reporte de acuerdo a esquemas preestablecidos como tabla, columnas, estándar. Pro último la opción de cerrar un reporte que desvincula el reporte elegido de la base de datos desde la cual se origino.

- La manipulación de controles relacionados con el editor de reportes es similar a la desarrollada en el Editor de Formularios, con la diferencia de que los controles deben ser generados para desempeñar funciones especificas de los formularios, como el objeto que permita totalizar y subtotalizar por atributos siempre y cuando sean de tipo numérico, el objeto que permite la incorporación de algunas formulas como la del pie de pagina, el numero de pagina y la actualización de la fecha actual del reporte. Los objetos que permitan dibujar líneas, dibujar cuadros e incorporar imágenes.
- El reporte también será guardado utilizando el lenguaje XML que facilita la reconstrucción de un formulario cuando se abre en la herramienta, además debe permitir guardar el código fuente JAVA de la aplicación del reporte, todo esto se puede realizar reutilizando varías clases ya implementadas del módulo editor de formularios. De la misma forma se puede implementar la compilación y ejecución de reportes.
- La vista de diseño de un reporte estará conformada por los siguientes objetos: el objeto hoja, que permite manipular la orientación del papel y el tamaño, el objeto sección, un reporte siempre tendrá tres secciones definidas, la sección encabezado, la sección detalle y la sección pie de página y el objeto línea sobre el cual se ubica los demás controles que conforman el reporte
- El editor de reportes incluirá una vista previa del reporte que contendrá un objeto diseñado para mostrar las páginas que forman parte del mismo.
- El resultado de un reporte se podrá almacenar en formato RTF, para lograr esto se tiene pensado hacer uso del proyecto iText, un proyecto software libre que incluye las librerías y clases necesarias para construir documentos en formatos HTML, PDF y RTF, mediante instrucciones.
- También se incluirá la posibilidad de imprimir el reporte, haciendo uso de la librería java.awt.print, que se encarga de controlar la impresora o impresoras del reporte.
- Otra de las características para manipular el reporte es la implementación de la función zoom o ampliación sobre las páginas del reporte.
- En el caso de la vista código, el código fuente generado para un reporte no se puede editar, la herramienta entregara un formulario con los objetos necesarios para mostrar el resultado de un reporte que se actualizará de acuerdo con los datos, la aplicación como tal también permitirá guardar el reporte en formato RTF, imprimir y la función zoom.

Otras propuestas factibles que pueden complementar los módulos hasta ahora desarrollados, pero que se deben analizar y diseñar para acoplarlas a la lógica de la aplicación hasta ahora realizada, son:

- Un módulo de seguridad exclusivo para el motor de base de datos PostgreSQL, que permita controlar el registro de nuevos usuarios, de nuevos

grupos, las vistas sobre estructuras de bases de datos dadas a cada usuario y los permisos sobre operación en esas mismas estructuras.

- Se sabe que el SQL es un lenguaje estándar, que todo manejador de base de datos que este en el mercado debe implementar ajeno a las demás características que la casa productora del manejador quiera implementar como valor agregado para la competencia, entonces cabe la posibilidad real que la herramienta se pueda funcionar con otros manejadores de base de datos, siempre y cuando exista el JDBC apropiado respectivo, sin que las interfaces hasta ahora desarrolladas en la herramienta y las futuras, y el mecanismo que transforma los datos que reciben dichas interfaces en sentencias SQL textuales válidas sufran alguna alteración en cuanto al análisis, diseño e implementación.

- La implementación de formularios más complejos con funciones más definidas como la relación maestro – detalle, donde un formulario esta contenido dentro de otro, y cada formulario manipula los registros de la tabla respectiva sobre la cual fue creado. Las tablas de cada uno de los formularios deben tener alguna relación de tipo foráneo entre sus campos.

Otras características que se pueden implementar, pero que implican módulos más complejos, pueden ser:

- El trabajo orientado a la WEB. Este módulo permitiría que los formularios y reportes sean construidos como páginas dinámicas desde el diseño. Para ello se debe hacer uso del código HTML y de un lenguaje que permita el acceso a base de datos incrustado en el primero como puede ser JSP.

- El trabajo multiusuario. Puesto que un proyecto puede ser desarrollado por un equipo de trabajo, se debe permitir que la herramienta trabaje en varios computadores que accedan a una misma base de datos que se encuentre en un servidor, La aplicación que se este desarrollando debe tener control de versiones y atender múltiples solicitudes de varios usuarios en una red local.

11. CONCLUSIONES

Como se explico en este documento, el objetivo del módulo Editor de Base de Datos era implementar un mecanismo eficiente y sencillo para que el usuario pudiera especificar los parámetros de algunas de las sentencias más comunes del lenguaje SQL en un ambiente gráfico. Este objetivo se alcanzó, al implementar el mecanismo ya descrito mediante el cual y usando el JDBC apropiado se puede convertir los datos obtenidos en las interfaces gráficas en sentencias SQL válidas para posteriormente ejecutarlos en el motor. Este mecanismo, el cual se demostró que era eficiente porque efectivamente realiza los cambios necesarios en el motor y considerando la posibilidad de que el lenguaje SQL tiene muchas más sentencias cada una semántica definida, permite afirmar que es probable implementar nuevas sentencias SQL mediante interfaces que cumplan con el objetivo de la sentencia.

Para alcanzar el objetivo que se perseguía con el módulo Editor de Formularios, el cual era la generación de código JAVA de aplicaciones relacionadas con los campos de una tabla de una base de datos, fue muy importante haber programado la herramienta en le lenguaje de programación JAVA, siendo este un lenguaje orientado a objetos, la versatilidad que ofrece JAVA para manipular todos los elementos lo que se necesitan y que vienen incorporados en el lenguaje, la amplía diversidad de librerías que contiene, cada una con funcionalidades específicas permitieron que las diferentes metas propuestas en este módulo se alcanzaran, como en el caso de la creación de componentes, usando la tecnología beans de JAVA, que fácilmente permitió manipular las propiedades de cada componente, que permitieron la generación eficiente del código respectivo y de la misma permitió que la herramienta no quedará ligada a unos pocos componentes que ya se ofrecen en la herramienta sino que se puedan incorporar componentes diseñados por el usuario, siempre y cuando se implementen respetando algunas interfaces de programación ya definidas.

Una de las ventajas de licenciar un proyecto como software libre es la posibilidad permitir que otras personas puedan intervenir en el desarrollo de un proyecto computacional de gran dimensión, y que beneficie la a comunidad de programadores que comparte esta manera de desarrollo de software. Para lograr este objetivo también propuesto en este proyecto y que pueda ser considerado por otras personas no basta con licenciarlo como software libre, también se necesita hacer público el trabajo desarrollado hasta el momento, lo cual implica hacer disponible toda la información necesaria, incluyendo el código fuente de la aplicación, a la comunidad de programadores en general. También es importante hacer un seguimiento y brindar toda la colaboración necesaria especialmente al grupo de personas que estén interesadas en continuar con el desarrollo del

proyecto, siempre dejando en claro cual es la idea principal del proyecto, sin olvidar que la herramienta debe cumplir con las características básica de funcionalidad, versatilidad y portabilidad, sobre todo si se trata de una aplicación destinada a ayudar al programador en el desarrollo de su código.

Lastimosamente, y por las razones expuestas en este documento, el módulo de Editor de Reportes no alcanzo su implementación en la versión de la herramienta entregada. Pero las expectativas para este módulo son grandes, en este documento se entrega el análisis y diseño de este módulo, y como en versiones anteriores que la herramienta tuvo a lo largo del desarrollo de este proyecto, incluso hay interfaces ya programadas que surgieron con ese análisis y diseño expuesto. Lo importante es continuar con el desarrollo del proyecto, encontrar el apoyo necesario, encontrar las personas interesadas en continuar con el mismo, de esta forma se abre el proyecto para que sea analizado por otras personas desde sus puntos de vista y que nuevas ideas puedan surgir en cuanto a análisis, diseño e implementación del mismo.

BIBLIOGRAFÍA

- BOOCH, G., et al. El Lenguaje Unificado de Modelado: El libro introductorio a UML escrito por sus creadores. 1 ed. Madrid: Addison Wesley, 1999. 432 p.

- EFEBER.NET. Clientes: psql en modo texto, PgAccess en modo gráfico (Online). Disponible en Internet. URL: http://www.efaber.net/formacion/fp/curso_acs/index.html#clientes (Citado 13 de Marzo del 2004)

- GOMEZ, Julio Cesar. Beneficios del software Libre. En: II CONGRESO INTERNACIONAL SOFTWARE LIBRE GNU/LINUX. (2º: 2003: Manizales). Ventana Informática – Edición Especial II Congreso Internacional Software Libre GNU/Linux: Departamento de Publicaciones Universidad de Manizales, 2003. 123 p.

- ITAPAZICO. Tutorial de Java. Disponible en Internet: URL: http://www.itapizaco.edu.mx/paginas/JavaTut/froufe/index.html.

- JACOBSON, Ivar, et al. El proceso unificado de desarrollo del software. Madrid: Pearson Educación S.A., 2000. 464 p.

- JOYANES, Luís, et al. Java 2: Manual de Programación. 1 ed. Madrid: McGraw-Hill, 2001. 542 p.

- LARMAN, Craig. UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. Madrid: Pearson Educación S.A., 2003. 624 p.

- MELO, Diego Samir. Webs dinámicas utilizando PHP. En: II CONGRESO INTERNACIONAL SOFTWARE LIBRE GNU/LINUX. (2º: 2003: Manizales). Ventana Informática – Edición Especial II Congreso Internacional Software Libre GNU/Linux: Departamento de Publicaciones Universidad de Manizales, 2003. 123 p.

- MOREA, Lucas. Tutorial de Java (Online). Disponible en Internet: URL: http://www.monografias.com.

- _____ , _____. Documento COMO para el RDBMS-SQL-Base de datos para Linux (Sistema de base de datos objeto-relacional Postgresql) Herramientas de Gestión de PostgreSQL (Online). Disponible en Internet: URL:

http://linux.dsi.internet2.edu/docs/LuCaS/Postgresql-es/web/navegable/Howto/PostgreSQL-COMO-10.html, (Citado 13 de marzo del 2004).

- POSTGRES. Página Oficial de Postgres. Disponible en Internet: URL: http://www.postgresql.org/. Disponible también: FTP: ftp://ftp.postgresql.org/.

- REDHAT Linux. Sitio Oficial de la Distribución RedHat. Disponible en Internet: URL: http://www.redhat.com/. Disponible también: FTP: ftp://ftp.redhat.com/.

- ROBLES, Tomas J; TURIENZO, Raúl. Introducción a PostgreSQL (online), Disponible en Internet: URL: http://programacion.com/bbdd/tutoriales/PostgreSQL/, programación en castellano (URL: http://programacion.com), 11 de marzo del 2003 (citado el 13 de marzo del 2004).

- SUN MICROSYSTEM. Tutorial oficial JDBC (Java). Disponible en Internet: URL: http://java.sun.com/docs/books/tutorial/jdbc/basics/index.html.