ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO DE INGENIERÍA DEL SOFTWARE

LANZADERA JAVA PARA LA PROGRAMACIÓN DE CONSULTAS SQL SOBRE LA CAPA DE PERSISTENCIA DE UN SOFTWARE DE APLICACIÓN.

JAVA SHUTTLE FOR THE DEVELOPMENT OF SQL QUERIES OVER THE PERSISTENCE LAYER OF AN APPLICATION SOFTWARE

Realizado por
JUAN MIGUEL DE LOS RIÓS CAPARRÓS

Tutorizado por
MANUEL NICOLÁS ENCISO GARCIA-OLIVEROS

Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA
MÁLAGA, Diciembre de 2015

Fecha defensa:
El Secretario del Tribunal

Resumen:

Este trabajo fin de grado tiene como finalidad el desarrollo de dos aplicaciones java, independientes en su uso, pero relacionadas entre sí en un mismo objetivo final. Una estará destinada a un usuario que, sin conocimiento alguno sobre modelos de datos relacionales ni consultas SQL, podrá ejecutar operaciones de exploración de los datos sobre la capa de persistencia de un software de aplicación, generando un modelo de extracción de consultas como subconjunto del QBE (Query by Example). Esta misma plataforma no estará hecha ad-hoc para un modelo concreto de base de datos relacional, sino que podrá ser configurado para su uso sobre cualquier capa persistente que subyace en un software de aplicación. La otra plataforma estará destinada al programador del software, que aprovechando los recursos de esta segunda plataforma, modelará el front-end de la primera para que ofrezca un manejo amigable y sin tecnicismos, guiado por la lógica de un usuario alejado de cualquier conocimiento sobre base de datos, revirtiendo en un mejor provecho de los recursos ofrecidos por la explotación de sus datos.

Palabras claves: Bases de datos, SQL, QBE, Exploración de datos.

Abstract:

This Degree Thesis (TFG) aims to develop two java applications. They are independent in use, but linked to the same goal. First focuses to a non-technical user, who without any knowledge of relational data models neither SQL queries , can execute operations of data explorer over the persistence layer from any application, and creating a model of queries extractor as a subset of QBE (Query by example). This platform is not developed "ad hoc" to a specific database. It can be configured by the second platform to use it over relational data model of any software, executes sql queries and collects datas. The second platform is intended for the software developers. They can use the resources of this second application to model the front-end of the first, so it will provide a friendly and non-technical management, searching the "way of thinking" of an user away from any knowledge of databases, and providing a best use of the resources offered by the exploration of the datas.

Keywords: databases, SQL, QBE, exploration of datas

Índice

1. Introducción

El desarrollo tecnológico ha propiciado que la implementación de los sistemas avance a un nivel superior, donde el desarrollador y el usuario final se relacionan de una forma u otra a través de la automatización de los procesos. Esto permite disminuir el tiempo de solicitud y recuperación de los datos. Al día de hoy se está logrando una mayor eficiencia en los resultados finales del software gracias al desarrollo de sistemas que pueden ser configurados partiendo desde las necesidades reales del usuario.

En el presente trabajo se ha desarrollado dos plataformas independientes en su uso, pero relacionadas entre sí en un mismo objetivo final. Una de las aplicaciones permitirá que usuarios ajenos a la gestión de las bases de datos, sin conocimientos sobre modelos relacionales ni consultas SQL, puedan ejecutar operaciones de exploración de los datos sobre la capa de persistencia de un software de aplicación. Esta plataforma no estará hecha ad-hoc para un modelo concreto de datos, sino que podrá ser configurado para su uso sobre cualquier capa persistente que subyace en un software de aplicación.

La otra plataforma estará destinada al programador del software, que aprovechando las herramientas que ofrece esta segunda aplicación, modelará el *front-end* de la primera para ofrecer un manejo amigable y sin tecnicismos, guiado por la lógica de un usuario alejado de cualquier conocimiento sobre base de datos, revirtiendo en un mejor provecho de los recursos ofrecidos por la explotación de sus datos. Esta aplicación deberá comportarse en un modelo de extracción de consultas como subconjunto del QBE (Query by example)

Para la puesta en marcha del trabajo se han realizado diversas tareas que ha servido de esbozo para la organización de las primeras ideas. Se hizo uso de varias herramientas informáticas así como de la creación de diversos recursos con la que se desplegó el código más adecuado a los objetivos específicos del desarrollo.

Para la implementación de este proyecto se utilizó la tecnología Java JDK 1.7.

El trabajo está estructurado en 11 capítulos incluyendo las conclusiones generales y bibliografía consultada.

En las primeras secciones se efectúa un examen teórico del tema de investigación, mostrando la motivación que guió el desarrollo del trabajo, así como la metodología y los objetivos trazados.

En la sección 5 se expone los requisitos funcionales y no funcionales que debe cumplir el sistema y guían el desarrollo hacia el resultado correcto. Estos serán analizados con mayor profundidad en la sección 7.

En la sección número 6 se describe el estado del arte, permitiendo conocer la base teórica que sustenta el trabajo.

La sección 8 describe el entorno tecnológico y la sección que relaciona los aspectos vinculados al desarrollo de la aplicación. Define el modelo de casos de uso, establece los diagramas de clases de diseño y de secuencia, así como las fases que guían el proyecto.

Por último, en la sección 10 se muestran algunas capturas de pantallas de la aplicación, dando paso en la sección 11 a las conclusiones generales del trabajo.

Utilizamos el formato APA como norma para las referencias bibliográfica utilizadas en esta memoria, donde citamos la referencia en el texto y posteriormente lo agregamos en la lista de referencias

2. Motivación

La motivación fundamental para el desarrollo de este trabajo es la creación de una herramienta que permita la ejecución de consultas a base de datos por parte de usuarios sin conocimiento alguno sobre modelos relacionales ni SQL, a los que hemos identificado como usuarios finales de un aplicativo software, pero ligando estrechamente esta motivación principal con la creación de una segunda herramienta que facilitará el trabajo de los desarrolladores de ese aplicativo software para afrontar el desafío que supone el desarrollo de una herramienta de consultas con estas características.

Actualmente, cuando los usuarios finales de las aplicaciones se enfrentan a la utilización de sus nuevas herramientas, echan en falta funcionalidades que no se habían previsto al inicio del desarrollo y que en la mayoría de las ocasiones tienen como objetivo la exploración de los datos almacenadas por el propio software. Esto ocasiona que los desarrolladores tengan que rehacer o agregar algunas de las funcionalidades implementadas para ajustarlas a esas nuevas solicitudes.

La herramienta desarrollada en este proyecto consigue ahorrar tiempo de desarrollo y aumenta la satisfacción e independencia en los usuarios finales de las aplicaciones, ya que ofrece la posibilidad de explorar la información almacenada, hacer consultas y generar informes sin tener que agregarle nuevos elementos ah-hoc.

El usuario final podrá realizar las consultas que necesite sobre su modelo de datos, sin tener conocimiento alguno en consultas SQL.

3. Metodología

Con la metodología utilizada para modelar el sistema se define y explica los pasos que se han seguido para desarrollar el proyecto, de manera que los desarrolladores, y todo aquel que pueda estar interesado en el producto final, tenga la misma visión.

Durante el desarrollo del presente trabajo se hizo una recopilación y revisión de los distintos materiales y documentación que están relacionados con ambas plataformas, y que son necesarios para el buen desempeño en la toma de decisiones.

Se realizó un estudio de los lenguajes usados en la parte que abarca las tecnologías *JAVA WEB START,* así como el estudio de los lenguajes propios de la parte cliente, junto con algunas herramientas de gestión de base de datos.

RUP sirvió de guía para el proceso de desarrollo. Aunque no se transitó por todo su ciclo de vida, sí se siguió sus fases de trabajo. Se utilizó el lenguaje de modelación UML para graficar los casos de uso y los diagramas.

El trabajo fue planificado según 4 fases:

A) Fase de documentación: En esta primera fase se recopiló y revisó el material a integrar y se consultó las distintas herramientas informáticas y recursos software que serían utilizados. En este punto se tomó la decisión de adoptar la tecnología Java Web Start y la API de Java Swing Framework para la implementación de aplicaciones escritorio, frente a la propuesta inicial de desarrollo que se proponía con el uso de STRUTS. Se entendió que el desarrollo de una plataforma "portable" que no requiriese de la instalación de ningún servicio, framework o contenedor en el servidor (véase Apache, Apache- Tomcat, etc…) facilitaría un uso que estaría más cerca al objetivo final del proyecto: su puesta en marcha de forma rápida y sin exceso de requerimientos técnicos.

B) Fase de análisis y diseño: Para abordar el desarrollo de la plataforma había que establecer la arquitectura definitiva para que pudiese ser implementada de forma adecuada.

- Diseño de la parte funcional, identificando los criterios funcionales y los aspectos relativos al uso de la herramienta

- Modelado de la aplicación utilizando el patrón Modelo-Vista-Controlador.
- Diseño del modelo Entidad - Relación
- Diseño de las interfaces gráficas de usuario.

C) Fase de implementación: En esta fase se realizó la codificación del software, teniendo en cuenta los resultados obtenidos en la fase anterior.

D) Fase de prueba: En esta fase se realizaron una serie de casos que pondrían a prueba el sistema. Con estas pruebas, llamadas casos de prueba, tratamos de probar el abanico de posibilidades más amplio que fuese viable.

- Prueba y depuración de la aplicación.
- Implementación de los cambios necesarios hasta lograr una versión funcional estable.

E) Fase de elaboración de la memoria: En esta última fase, hemos reorganizado toda la documentación generada para elaborar la memoria final del trabajo.

4. Objetivos

El objetivo general de este trabajo ha consistido en el desarrollo de una lanzadera java para la ejecución, sobre modelos estructurados XMLs, de cualquier consulta SQL que se desee lanzar sobre la capa de persistencia de un software de aplicación, ligándolo con una segunda plataforma que lanzará las consultas SQL obtenidos a partir del modelo XML que fue generado en la primera plataforma.

Cabe reseñar que por la naturaleza del modelo de programación utilizado en la plataforma, además de la obtención de un esquema XML que modele una base de datos concreta de forma abstracta, también se ha requerido que la misma herramienta genera de forma automático las clases que se corresponden con Modelo dentro del esquema MVC, donde cada clase mapea una tabla del modelo relacional. Estas clases quedan explicadas con más detalle en el apartado 9.

Existen además otros objetivos específicos como son:

- Generar consultas a medida, según necesidades específicas.
- Aceptar variaciones del modelo de datos sin afectar a los esquemas que ya están generados.

5. Requisitos

Para la implementación del sistema se tuvo en cuenta los requisitos que fueron capturados durante el desarrollo del proyecto y que tenían como principal objetivo obtener un producto de calidad.

Los requisitos fueron aportados por "clientes reales" que utilizan aplicaciones software "ad-hoc" y que han echado en falta una herramienta que les permitiese lanzar consultas sobre los datos almacenados en su software.

RNF-01	**Apariencia e interfaz externa**
Tipo de requisito	No Funcional
Dependencias	
Descripción	La interfaz debe ser sencilla y ajustarse a los estándares establecidos para un buen diseño.
Importancia	Alta
Prioridad	Baja

RNF-02	**Rendimiento**
Tipo de requisito	No Funcional
Dependencias	
Descripción	Debe garantizar una velocidad de respuesta genuina.
Importancia	Alta
Prioridad	Baja

RNF-03	**Soporte**
Tipo de requisito	No Funcional
Dependencias	
Descripción	Posibilidad de mantener o extender con la mayor simplicidad posible.
Importancia	Alta
Prioridad	Baja

RNF-04	Requerimientos de software
Tipo de requisito	No Funcional
Dependencias	
Descripción	Sistema Operativo Linux Ubuntu (cliente) y Linux Centos (servidor) J2SE Development Kit 8.0. Base de datos MySQL Herramientas de desarrollo Java NetBeans. Procesador de textos Open Office 4.4.1. Procesador de textos VIM en consola de comandos.
Importancia	Alta
Prioridad	Baja

RNF-05	Requerimientos de hardware
Tipo de requisito	No Funcional
Dependencias	
Descripción	Ordenador portátil TOSHIBA SATELLITE L655 Intel Core i5-3337U CPU 1,80 GHz 8 Gb RAM 500 GB SSD
Importancia	Alta
Prioridad	Baja

RF-01	Gestionar modelo de datos
Tipo de requisito	Funcional
Dependencias	
Descripción	Gestionar los modelos de la capa de persistencia de software de aplicaciones.

Importancia	alta
Prioridad	alta

RF-02	**Exportar configuración XML**
Tipo de requisito	Funcional
Dependencias	RF-01
Descripción	Generar XML a partir de la configuración del modelo.
Importancia	alta
Prioridad	alta

RF-03	**Exportar configuración TDs**
Tipo de requisito	Funcional
Dependencias	RF-01
Descripción	Generar clases TDs a partir de la configuración del modelo.
Importancia	alta
Prioridad	alta

RF-04	**Gestionar generador de consultas.**
Tipo de requisito	Funcional
Dependencias	RF-02
Descripción	Permite realizar consultas sobre el modelo de datos. Gestionar las consultas realizadas e imprimir resultados.
Importancia	Alta
Prioridad	Alta

En el capítulo 7 se analizarán con un mayor nivel de detalles los requisitos funcionales.

6. Fundamentos y estado del arte

El objetivo que se persigue en este epígrafe es la descripción del estado del arte relacionado con la tecnología utilizada en el desarrollo del sistema.

6.1. Introducción

Existe gran documentación sobre base de datos (BD), modelo relacional y esquemas. Son diversas las soluciones sobre entornos dinámicos para consultas de datos y herramientas de gestión de Bases de Datos.

Dubretic hace referencia a que "los sistemas de gestión de bases de datos relacionales (RDBMS)se han convertido en el tipo de base de datos estándar para una gran cantidad de industrias. Como su nombre lo indica, estos sistemas se basan en el modelo relacional que organiza los datos en grupos de tablas que se relacionan por el tipo de datos que contienen" (Dubretic, 2014).

Estos sistemas son explotados en gran medida por personas con conocimientos técnicos avanzados como desarrolladores o expertos en bases de datos. Ellos mismos pueden gestionarlas y a través de consultas pueden realizar extracción de datos para obtener la información que se requiere.

Los usuarios estándar que acceden a estas BD a través de aplicaciones, no pueden explotar todas sus bondades, por este motivo existe la tendencia de implementar *Generadores de consultas.* Estos programas permiten crear consultas a través de sintaxis sencillas, acceder a todos los datos de la BD y evaluarlos según las necesidades.

Para realizar un análisis del estado del arte es necesario aclarar algunos conceptos que son de vital importancia para el trabajo, como por ejemplo: bases de datos, capa de persistencia y modelo relacional. Al igual se mencionan los gestores de bases de datos más conocidos según el campo de acción que se centra el trabajo.

6.2. Análisis de conceptos.

Alcolea y Álvarez(2007) conceptualizan a la BD como un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y

un conjunto de programas que manipulen ese conjunto de datos(Alcolea y Álvarez, 2007).

Las bases de datos pueden clasificarse según el contexto manejado (código abierto o comercial), su utilidad, las necesidades que satisfagan o el modelo que sigan (modelo relacional o No SQL).

Existe un tipo de *software* exclusivamente dedicado a tratar con BD relacionales, conocido como Sistema de Gestión de Bases de Datos Relacionales. Entre los gestores actuales más populares se encuentran MySQL, PostgreSQLy Oracle, además de otros igual de pujantes como SQL Server o MongoDB (http://db-engines.com/en/ranking). Para este trabajo vamos a considerar solo a los tres primeros dentro de la comparativa que detallaremos en el punto siguiente.

Para referirse a la estructura de las BDs se utilizan los Esquemas. En una RDBMS, el Esquema define cada tabla, campos y sus relaciones (Edu4Java, s.f).Dichos esquemas se "diseñan a través del modelo entidad-relación de manera que sirvan posteriormente para su implementación " (Gutiérrez, 2013).Esto modelos son muy útiles para los usuarios casuales, debido a que pueden entender las estructuras de la BD.

En todo sistema es de vital importancia garantizar la persistencia de los datos. Por este motivo se han trazado diferentes mecanismos, entre los que se encuentra el de Persistencia."Estos mecanismos no son más que técnicas que permiten resolver la persistencia de datos a través del tiempo" (UDELAR, s.f).“Los mismos son ejecutados en la capa de persistencia; de esta manera se vinculan los datos guardados en una base de datos relacional, con los objetos de una aplicación“(Junta de Andalucía, s.f).

6.3. Sistemas Gestores de Base de Datos.

A) PostgreSQL

Es un sistema de base de datos relacional orientada a objetos que está publicado bajo una licencia BSD (Berkeley Software Distribution). Es un proyecto de código libre, y debido a estas características sus mejoras han sido un poco más rápidas en comparación con otros sistemas de base de datos. Originalmente este programa se

llamaba postgre. Posteriormente cambio su nombre postgres95 hasta llegar a ser llamado PostgreSQL como es conocido en la actualidad.

B) Mysql

Es un sistema de base de datos relacional, multihebra y multiusuario. Está publicado bajo una licencia GLP, esto quiere decir que es un software propietario y está sustentado por un empresa privada que posee el copyright de la mayor parte del código. Es desarrollado en su mayor parte por ANCI C, un estándar para el lenguaje de programación C. Fue comercializado por primera vez en 1981 por IBM.

C) Oracle

Es un sistema de base de datos relacional que fue desarrollado por *Oracle Corporation*. Se considera como uno de los sistemas de base de datos más completos. Hace algunos años su dominio en el mercado era casi total, pero en la actualidad ya no lo es tanto debido ala gran competitividad que existe. Surge a finales de los 70. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNL/Linux (Scribd, 2015)

6.3.1 Cuadro comparativo

RDBMS	Ventajas	Desventajas
MySql	Consume pocos recursos tanto de procesador como de memoria. Mejores utilidades de administración No hay límites en el tamaño de los registros. Multiplataforma	En sus versiones más reciente ha pasado a ser privativo. Algunas utilidades no documentadas.
PostgreSQL	Por su arquitectura de diseño, escala muy bien al aumentar el número de máquinas y cantidad de	Consume más recursos y carga más el sistema. Límite del tamaño de cada fila de las tablas a 8k

	memoria. Mejor soporte para procedimientos almacenados en el servidor. Multiplataforma.	Es de 2 a 3 veces más lento que MySql.
Oracle	Primero sistemas de bases de datos, por lo que es más completo. Permite el uso de particiones. Multiplataforma	Elevado costo en licencias. Fallos en las últimas versiones.

Según listados publicados en la Web donde se evidencia los RDBMS más utilizados, se pudo apreciar dentro de las categorías Libres que PostgreSQL y MySQL compiten entre los dos primeros lugares. Entre los de categoría Comercial, Oracle mantiene su liderato

Muchas son las tendencias que buscan realizar comparaciones entre estos gestores de bases de datos. Cada autor las realiza teniendo en cuenta los criterios que para su entender son los más favorables, por ejemplo: velocidad, costos, actualizaciones, consumo de recursos, entre otros. Es necesario destacar que la mejor manera de descubrir las principales ventajas y desventajas es con su utilización.

El RDBMS a utilizar en este proyecto será MySql debido a que se adapta mejor a las necesidades del entorno de implementación. Además este gestor de BD responde muy bien ante necesidades de velocidad y números de acceso concurrentes. Hemos entendido que su uso está más generalizado que otros RDBMS y que por tanto el proyecto se ubicaría en entorno de “más uso” que con otro RDBMS.

6.4. Opiniones sobre entornos dinámicos de consultas de datos.

Muchas opiniones coinciden que el desarrollo de sistemas de consultas a bases de datos "permiten crear herramientas útiles para hacer más flexible la comunicación

con las computadoras por usuarios no especializados, logrando de esta manera una interacción hombre – máquina en forma natural" (Soto, Cruz y Díaz, s.f).

Su utilidad es amplia, dependiendo de los intereses de cada sector. Desatacándose su uso en Universidades, centros científicos, empresas y otras entidades donde el análisis de los datos es de vital importancia para el desarrollo del trabajo. Los métodos que se utiliza para lograr la equivalencia entre el lenguaje formal con el utilizado en las consultas SQL, permite que el sistema interprete cada sentencia de acuerdo a la nomenclatura que se utilice. Las mismas tienen su correspondencia con las operaciones que utiliza SQL para realizar consultas.

En este punto, la búsqueda mediante ejemplo (QBE) nos permite orientar el resultado de nuestro proyecto a un modelo similar, en cierta medida subconjunto de éste, en la que el usuario final aporta cierta información sobre unas tablas de ejemplo para ejecutar las consultas SQL (Introducción al QBE, en CSUP)

7. Análisis de requisitos

En este capítulo se realiza un análisis más exhaustivo de los requisitos funcionales identificados en el capítulo 5.

7.1. RF-01

Gestionar modelo de consulta de los datos

Este requisito permitió identificar el caso de uso del sistema a implementar: Gestionar el modelo de consulta a los datos. La filosofía seguida fue la de obtener una herramienta que permitiera, a partir de esquemas, establecer relaciones entre los datos que se encuentran en la capa de persistencia para generar, a posteriori, las consultas SQL en una segunda aplicación que supiese interpretar el modelo XML proporcionado por esta primera aplicación, además de que aceptase en su compilación las clases definidas dentro del modelo de su esquema MVC que mapean las tablas de la base de datos, y que habrán sido generados junto al XML en eventos separados.

Una de las precondiciones que debe existir para lograr este requerimiento es que exista conexión con un RDBMS en el cual se puedan consultar los esquemas necesarios para el trabajo.

7.2. RF-02

Exportar configuración XML

Requisito asociado al caso de uso Generar XML. Muestra cómo está estructurada la configuración según la gestión del modelo de datos. Una precondición necesaria es la existencia de relaciones entre las tablas y los campos, de esta forma se pueden generar un archivo XML con la estructura necesaria para ser utilizados en otras aplicaciones.

7.3. RF-02

Exportar configuración TDs

Requisito asociado al caso de uso Generar TDs. Igual que en el requisito anterior, muestra cómo está estructurada la configuración según la gestión del modelo de datos. Una precondición necesaria es la existencia de relaciones entre las tablas y

los campos, de esta forma se pueden generar las clases TD que se corresponde con el modelo en el esquema MVC, con la estructura necesaria para ser utilizados por las librerías importadas en el código.

7.4. RF-04

Gestionar generador de consultas

Este requisito permitió diseñar los casos de uso del sistema: Gestionar consultas y Consultar reportes. Los mismos son la base de la implementación de la lanzadera de consultas. Este módulo permite a los usuarios no técnicos, crear reportes y recoger información a través de consultas sobre el modelo de datos. Los resultados de estas consultas las pueden gestionar según sus propias necesidades, para luego imprimirlos o exportarlos a modelos Excel de fácil interpretación por las herramientas ofimáticas que ofrece el mercado

8. Plataforma tecnológica utilizada en el desarrollo.

Las tendencias y tecnologías que actualmente se utilizan para la implementación de aplicaciones son numerosas. Los desarrolladores deben tener dominio de éstas y saber cuáles son las que deben emplear según las necesidades de la entidad. Las escogidas en este proyecto fueron:

A) Java Web Start

Permite arrancar aplicaciones Java que están en un servidor Web de aplicaciones comprobando previamente si el cliente tiene la versión actualizada de dicha aplicación. En caso contrario descargará la última versión y se ejecutará en local. Mediante esta tecnología se asegura que una aplicación es distribuida siempre en su última versión.

B) Java JDK 1.7

Java es una tecnología que se usa para el desarrollo de aplicaciones. Muchos desarrolladores coinciden que el manejo de punteros de memoria, la gran API, su sintaxis, su sistema de herencias y la sintaxis del compilador le hacen unos de los lenguajes más fáciles para aprender y para usar en muchos campos.

Específicamente, JDK incluye herramientas útiles para desarrollar y probar programas escritos en el lenguaje de programación Java y se ejecuta en esta misma plataforma .La versión de JDK 7, utilizada para el desarrollo de estos proyectos, incluye varias novedades, como por ejemplo.

- Project coin
- Strings en switchs
- Gestión automática de recursos en sentencias try-catch
- Multicatch
- Rethrow de excepciones más precisos
- Operador diamante <>
- Literales binarios
- Guiones bajos en literales numéricos
- Fork/Join y utilidades de concurrencia
- Mejoras en trabajo con archivos y carpetas

Otro elemento a destacar es la completa librería de clases que contiene el JDK.

Como es natural, también tiene sus desventajas, como por ejemplo: Lenta evolución, no es muy eficiente para la creación de multimedia. Además, según criterios de los especialistas, su principal objetivo no es el rendimiento. Esto último, aunque aporta una ventaja también es un inconveniente. Estos criterios son útiles para conocer la plataforma, pero no son un obstáculo para trabajar con Java JDK 1.7

Todo está en función de la utilización que se le quiera dar y de los intereses del desarrollo.

C) JExcelApi

Es un API de Java de código abierto que permite a los desarrolladores leer, escribir y modificar dinámicamente las hojas de Excel. Cuenta con varias librerías que se usan en dependencia de las necesidades de los desarrolladores. Esta herramienta permite generar un archivo Excel con los valores de los campos de una BD montada en un servidor.

D) Netbeans 8.0.2

Es una herramienta de desarrollo que soporta lenguajes como JAVA, PHP, C/C++ entre otros.

Entre sus principales características se encuentran: asistentes para la creación y configuración de distintos proyectos, buen editor de código, multilenguaje, control de versiones, comprobaciones sintácticas y semánticas, plantillas de códigos, coding tips, herramientas de refactorización, entre otras.

En cuanto al acceso a base de datos, desde el propio NetBeans se puede conectar a distintos RDBMS, como pueden ser Oracle, MySql, etc. Además se puede ver las tablas, realizar consultas y modificaciones, todo ello integrado en el propio IDE. Se integra con diversos servidores de aplicaciones de tal manera que se puede gestionar desde el propio IDE: inicio, parada, arranque en modo debug, despliegues. Entre otros podemos usar Apache Tomcat, GlassFish, JBoss, WebLogic, Sailfin, Sun Java System Application Server. (GENBETA, 2014)

D) Apache Subversion

Es una herramienta de control de versiones open source basada en un repositorio cuyo funcionamiento se asemeja enormemente al de un sistema de ficheros. Es software libre bajo una licencia de tipo Apache/BSD. (SUBVERSION, 2015)

Se ha utilizado para controlar las versiones del código en local y contra un servidor subversión instalado en una máquina CentOS.

9. Desarrollo

En esta sección se abordarán los temas relacionados al desarrollo de las aplicaciones.

9.1. Estructura de los proyectos

El proyecto sigue un Modelo Vista Controlador (MVC), basado en la reutilización de código y separación de conceptos, con el objetivo de facilitar su posterior mantenimiento. El mismo está estructurado por paquetes para administrar la complejidad. Los que comienzan con *op* son controladores, los *dlg* pertenecen a las vistas y los *form* a los formularios.

Debemos destacar que en *tabledefs* se encuentra el mapeo de los campos en las tablas de la base de datos MYSQL del modelo MVC. Estas clases heredan de la clase TableDef del cominergis.

La nomenclatura que se usó para nombrarlas es TD"nombre tabla". Por lo general existirá una clase por cada tabla de la BD, aunque puede suceder que exista más de una para hacer determinados JOINS complejos. Estas clases se generan de forma automática a partir del esquema, junto al XML (aunque en eventos separados) y son distintas para cada modelo de datos sobre el que se ejecutará la lanzadera de datos.

Libraries, contiene las librerías externas necesarias para la aplicación.

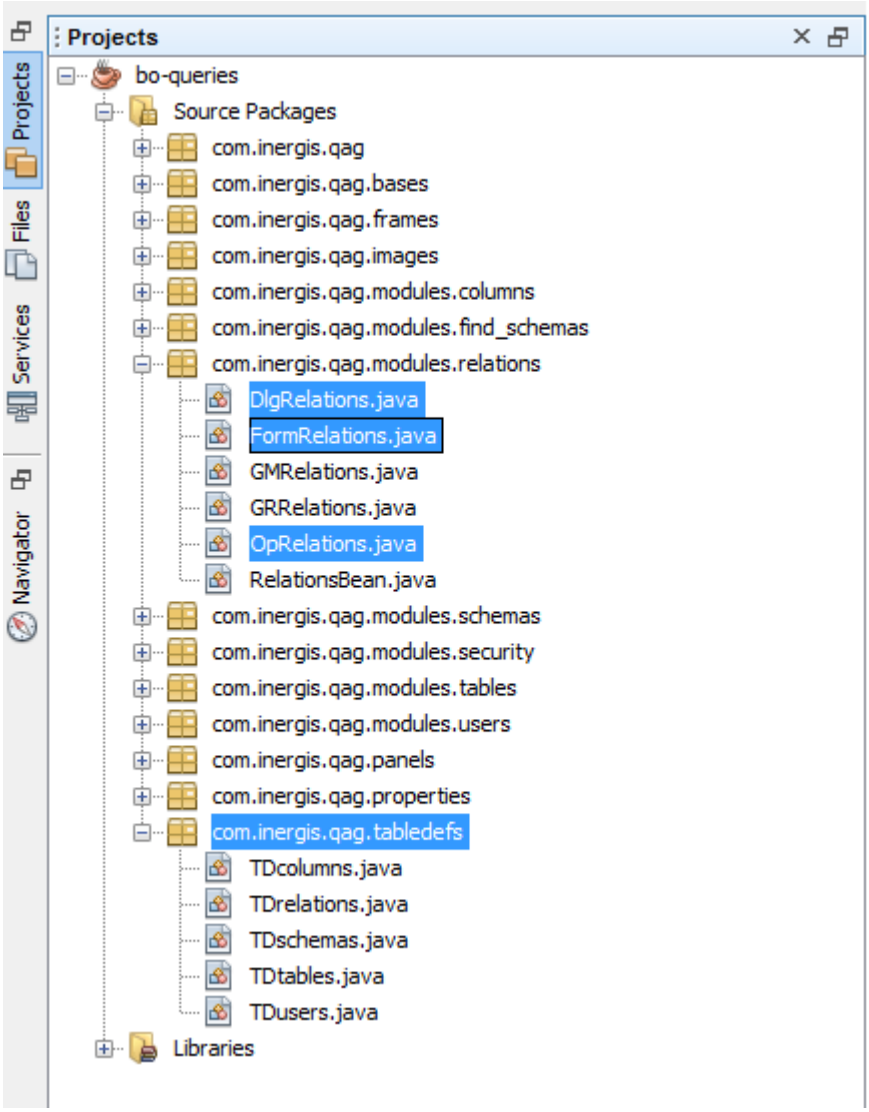


Fig. 1 Estructura del Generador de Asistente para Consultas.

La figura 1 representa la estructura del árbol de paquetes del BackOffice para la aplicación que genera de asistenta para Consultas (en nuestro proyecto, la 1era plataforma). En la misma se encuentra:

- **Bases:** Contiene clases que van a ser utilizadas en múltiples lugares de la aplicación
- **Frames:** Contiene las clases que son instancias de JFrame en la aplicación, estás son la ventana de autenticación y la ventana principal.
- **Images:** Contiene las imágenes de la aplicación.
- **Modules:** Contiene todos los paquetes funcionales de la aplicación. En general habrá paquetes por cada tabla, como columns, schemas y relations.

- **Properties:** Contiene los ficheros de configuración de la aplicación, habrá dos, una para el entorno de desarrollo y otro para el entornos de producción. En nuestro caso solo usaremos el de producción.
- **Panels:** Contiene los paneles de la ventana principal.
- **Tabledefs:** Contiene las clases que mapean las tablas de la base de datos.

El paquete com.inergis.qag contiene además, las clases:

- **AppSession:** Contiene las variables relacionadas con la sesión del usuario que inicia la aplicación.
- **Ctes:** Contiene las variables que están activas durante toda la aplicación que no van a variar sus valores.
- **PreferencesAplication:** Contiene las variables que están activas durante toda la aplicación que pueden variar sus valores.
- **Runme:** Clase principal de la aplicación.
-

9.2. Definición del operador y los beans.

Los operadores y beans se definen una vez finalizado el mapeo de las tablas de la BD. El primero se utiliza para realizar las consultas y los beans para recoger los datos.

En com.inergis.qag.modules.relations se puede apreciar un ejemplo de la estructura interna de los paquetes. En el mismo se encuentran: el controlador OpRelations.java, que permite todas las operaciones *CRUD*. El DlgRelations.java muestra una ventana donde se encuentra un listado con las entidades asociadas, una caja de búsqueda para filtrar los datos, y en la parte derecha se encuentran los botones asociados a las entidades (editar, añadir, borrar). El FormRelations.javapermite, según la operación, añadir o editar formularios con los datos de las entidades.

EL paquete *com.inergis,qag.tabledefs*, se encuentra el mapeo de las tablas de la BD.

En el caso específico de la lanzadera de consultas se siguió la misma política de trabajo. Se estructuró la implementación en paquetes y se trabajó siguiendo el modelo MVC.

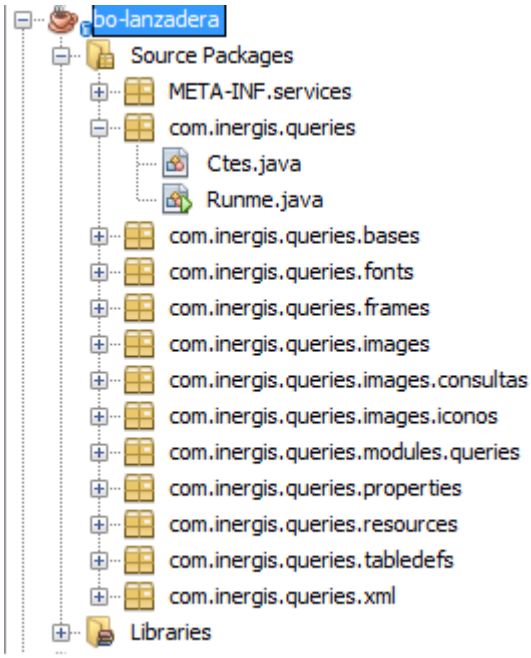


Fig.2 Paquete que integra las funcionalidades de la lanzadera de consultas.

La figura 2 representa la estructura del árbol de paquetes del BackOffice en la lanzadera. En el mismo se encuentran los paquetes: bases, frames, images, modules, properties, tabledefs estos realizan las mismas funcionalidades que en el generador de consultas explicado con anterioridad.

- **Fonts:** Contiene la fuente que se utilizará en los ficheros que se van a generar.
- **XML:** Contiene el fichero XML que se utilizará en las consultas.
- **Queries:** Contiene todas las funcionalidades de la lanzadera.

9.3. Librerías cominergis

Toda la gestión de los datos a nivel visual se realiza siguiendo un mismo esquema:

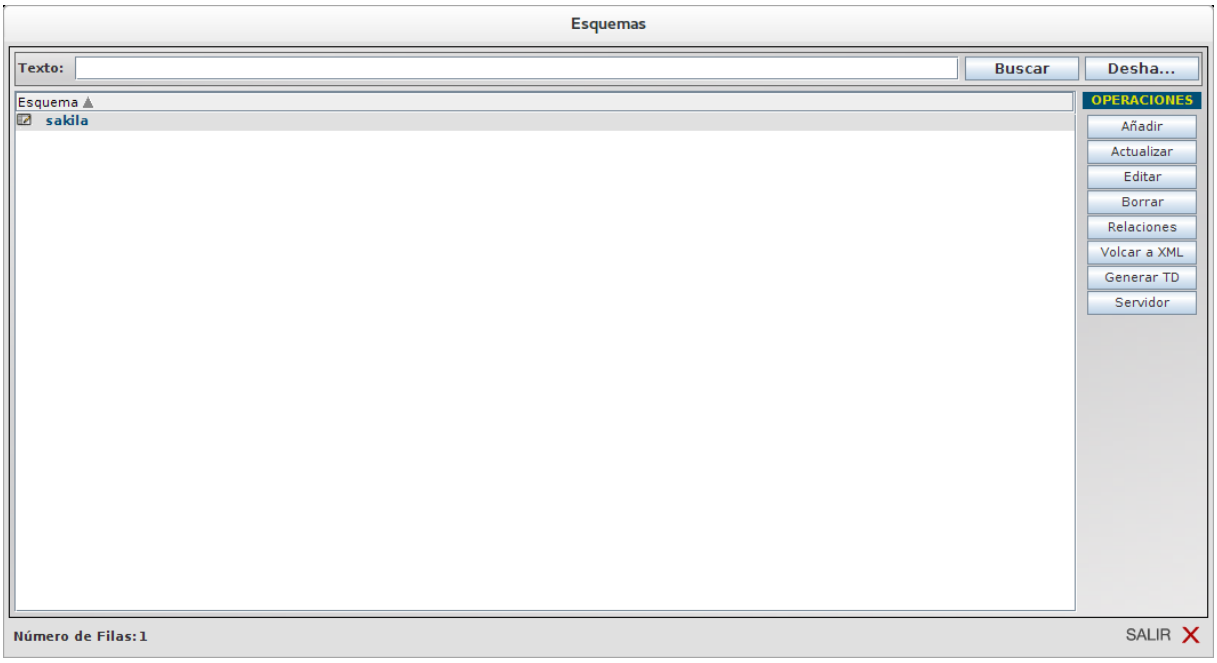


Fig.3 Patrón de esquema a seguir.

El diseño está distribuido de la siguiente forma:

Un buscador en la parte superior de la ventana que permite realizar filtrados de los datos. A la derecha se encuentran todas las operaciones que se pueden efectuar. La información se muestra dentro de un grid. Este componente debe ser del tipo JGrid de la librería cominergis y permite mostrar los datos de la base de datos utilizando los beans y tabledef que se definieron con anterioridad.

Para cada grid usado, se debe definir el modelo y el render. Las clases que definen el modelo van a tener la nomenclatura GM<Nombre>, las que definen el render va a tener GR<Nombre>. En el modelo se configuran los datos que se desean mostrar del bean y en el render nos dice como se visualiza los datos.

La librería cominergis permite la generación de trazas o logs. Las propiedades que se pueden configurar de este fichero son:

- **Active:** Permite activar o no el fichero de logs.
- **Filepath:** Ruta donde se van a almacenar.
- **Filename:** Nombre del fichero.
- **Filesize:** Tamaño que puede tener el fichero de logs.

La librería cominergis contiene varios componentes Swing, para usos específicos que facilitan el trabajo.

9.4. Análisis práctico para el mapeo de la tabla TDcolumns.

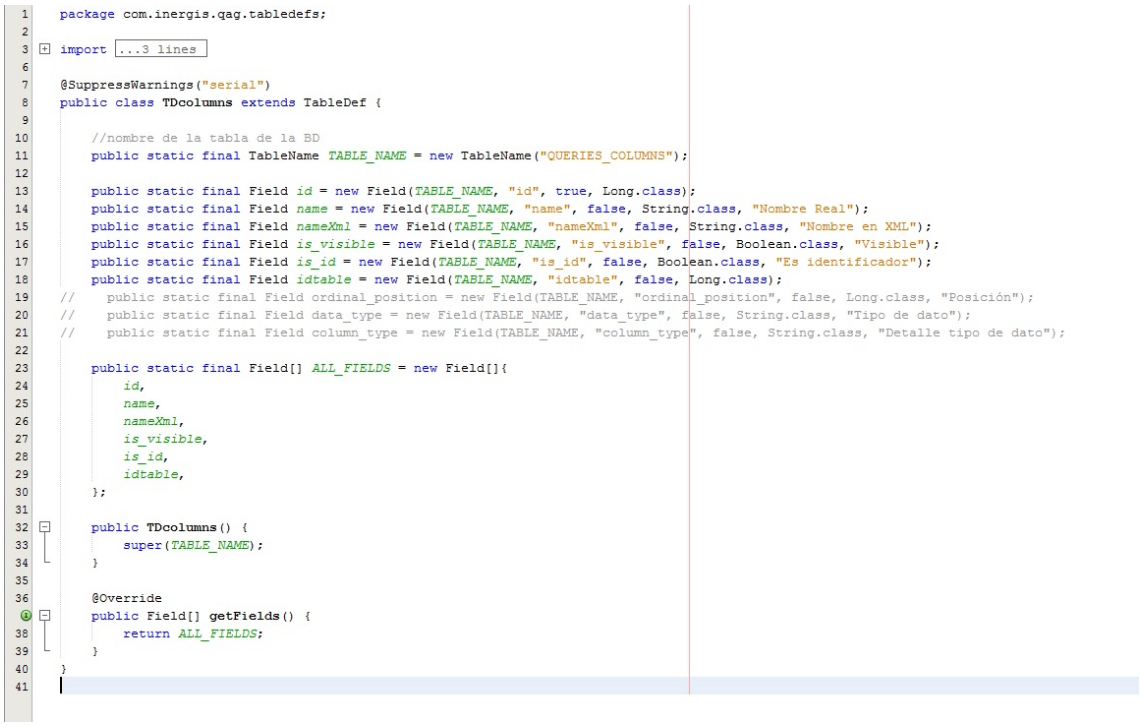
En el proyecto Generador de asistente se encuentra el paquete tabledefs. Dentro de este se definieron varios ficheros donde cada uno mapea a una tabla de la base de datos en Mysql. Para el caso específico de TDcolumns, se definieron varias variables estáticas de clases. Todos los TD heredan de una misma clase padre denominada TableDef.

Existe una del tipo tablename, donde se define el nombre real de la tabla en la Base de datos, a continuación viene varios campos de tipo field. Cada uno representa a un campo de la BD. El constructor de estos campos tipo field, reciben varios parámetros:

- Primero: Nombre de la tabla en la BD, que es el mismo valor que se creó anteriormente.
- Segundo: String que representa el nombre real del campo en la BD Mysql.
- Tercer: Booleano que representa si este campo va a ser llave de la tabla.
- Cuarto: Tipo de campo, puede ser (Long, String, Inter, boolean, etc.)
- Quinto: Etiqueta.

Posteriormente a la definición de los fields, se define una variable del tipo arreglo de campos, que va a contener todos los campos que se definieron con anterioridad, para este mapeo.

Seguidamente se define un constructor TDcolumns que invoca el constructor padre y se le pasa como parámetro el table_name que se definió. Se debe definir el método getFields, que en la clase padre está declarado como abstracto. Ver figura 4.



```
package com.inergis.qag.tabledefs;

import ...3 lines

@SuppressWarnings("serial")
public class TDcolumns extends TableDef {

    //nombre de la tabla de la BD
    public static final TableName TABLE_NAME = new TableName("QUERIES_COLUMNS");

    public static final Field id = new Field(TABLE_NAME, "id", true, Long.class);
    public static final Field name = new Field(TABLE_NAME, "name", false, String.class, "Nombre Real");
    public static final Field nameXml = new Field(TABLE_NAME, "nameXml", false, String.class, "Nombre en XML");
    public static final Field is_visible = new Field(TABLE_NAME, "is_visible", false, Boolean.class, "Visible");
    public static final Field is_id = new Field(TABLE_NAME, "is_id", false, Boolean.class, "Es identificador");
    public static final Field idtable = new Field(TABLE_NAME, "idtable", false, Long.class);
//    public static final Field ordinal_position = new Field(TABLE_NAME, "ordinal_position", false, Long.class, "Posición");
//    public static final Field data_type = new Field(TABLE_NAME, "data_type", false, String.class, "Tipo de dato");
//    public static final Field column_type = new Field(TABLE_NAME, "column_type", false, String.class, "Detalle tipo de dato");

    public static final Field[] ALL_FIELDS = new Field[]{
        id,
        name,
        nameXml,
        is_visible,
        is_id,
        idtable,
    };

    public TDcolumns() {
        super(TABLE_NAME);
    }

    @Override
    public Field[] getFields() {
        return ALL_FIELDS;
    }
}
```

Fig.4 Definición de la clase TDcolumns.

9.5. Análisis práctico para la creación de unbean.

Dentro del paquete columns, se define una clase ColumnsBean con el formato nombre table+Bean. Estas clases siempre heredan de la clase BaseBean.

Para el vean se definen los mismos campos que se definieron en la tabla, menos el Id, pues la clase BaseBean ya lo tiene definido. Se implementa además un constructor ColumnsBean, al que se le pasa como parámetro un objeto *Register*. Se implementa además uno sin parámetros, para inicializar los campos a conveniencia.

El método abstracto *Initialize, está* definido en la clase padre, con esto se le indica al bean en que campo debe almacenar cada campo-valor del *Register.* La clase *Register,* es la que cominergis utiliza para los datos recogidos de la BD.

Posteriormente se implementa el método abstracto *toRegister,* definido en la clase padre, con este método se hace la conversión Register -->ColumnsBean. Se le indica al Register donde debe almacenar cada campo del bean.

```java
package com.inergis.qag.modules.columns;

import ...3 lines

@SuppressWarnings("serial")
public class ColumnsBean extends BaseBean {

    protected String name;
    protected String nameXml;
    protected Boolean is_visible;
    protected Boolean is_id;
    protected Long idtable;

    public ColumnsBean(Register reg) {
        this();
        this.initialize(reg);
    }

    public ColumnsBean() {
        super();
        this.idtable = 0L;
        this.name = "";
        this.nameXml = "";
        this.is_visible = false;
        this.is_id = false;
    }

    @Override
    public void initialize(Register reg) {
        super.id = (Long) reg.getFieldValue(TDcolumns.id);

        this.setName((String) reg.getFieldValue(TDcolumns.name));
        this.setNameXml((String) reg.getFieldValue(TDcolumns.nameXml));
        this.setIs_visible((Boolean) reg.getFieldValue(TDcolumns.is_visible));
        this.setIs_id((Boolean) reg.getFieldValue(TDcolumns.is_id));
        this.setIdtable((Long) reg.getFieldValue(TDcolumns.idtable));

    }

    @Override
    public Register toRegister() {
        Register reg = new Register(TDcolumns.ALL_FIELDS.length);
        reg.add(TDcolumns.id, super.id);

        reg.add(TDcolumns.name, getName());
        reg.add(TDcolumns.nameXml, getNameXml());
        reg.add(TDcolumns.idtable, getIdtable());
        reg.add(TDcolumns.is_visible, getIs_visible());
        reg.add(TDcolumns.is_id, getIs_id());
```

Fig.5 Definición de los beans.

A continuación se implementan los método getter y setter de cada atributo.

```
    return reg;
}

/**...3 lines */
public String getName() {...3 lines }

/**...3 lines */
public void setName(String name) {
    this.name = name;
}

/**...3 lines */
public String getNameXml() {
    return nameXml;
}

/**...3 lines */
public void setNameXml(String nameXml) {
    this.nameXml = nameXml;
}

/**...3 lines */
public Boolean getIs_visible() {
    return is_visible;
}

/**...3 lines */
public void setIs_visible(Boolean is_visible) {
    this.is_visible = is_visible;
}

/**...3 lines */
public Boolean getIs_id() {
    return is_id;
}

/**...3 lines */
public void setIs_id(Boolean is_id) {
    this.is_id = is_id;
}

/**...3 lines */
public Long getIdtable() {
    return idtable;
}
```

```
        return reg;
    }

    /**...3 lines */
    public String getName() {...3 lines }

    /**...3 lines */
    public void setName(String name) {
        this.name = name;
    }

    /**...3 lines */
    public String getNameXml() {
        return nameXml;
    }

    /**...3 lines */
    public void setNameXml(String nameXml) {
        this.nameXml = nameXml;
    }

    /**...3 lines */
    public Boolean getIs_visible() {
        return is_visible;
    }

    /**...3 lines */
    public void setIs_visible(Boolean is_visible) {
        this.is_visible = is_visible;
    }

    /**...3 lines */
    public Boolean getIs_id() {
        return is_id;
    }

    /**...3 lines */
    public void setIs_id(Boolean is_id) {
        this.is_id = is_id;
    }

    /**...3 lines */
    public Long getIdtable() {
        return idtable;
    }
```

Fig.6 Definición de funciones set y get.

9.6. Análisis práctico para la creación del operador OpColumns.

Dentro del paquete columns se define una clase OpColumns. Este es un operador para las consultas a la base de datos de la tabla columns. Estas clases operadoras siguen el formato de Op+nombreTable y heredan de la clase tableQueryOp del cominergis. Se le debe pasar a esta clase como parámetros el TD y el Bean correspondiente a la tabla columns.

Se define igualmente un constructor sin parámetros. Se llama al padre indicando el TD, el Bean y el nombre del pool de conexiones.

Se define siempre un constructor que tiene como parámetro un objeto TableQueryOp. Este constructor se utilizará cuando se tenga que llamar al operador desde una transacción iniciada por otro operador. Se llama al padre indicando el TD, el Bean y el operador que ha abierto la transacción.

Posteriormente se definen todos los métodos que se necesitan para lanzar consultas sobre la tabla columns según los requerimientos necesarios.

```java
package com.inergis.qag.modules.columns;

import ...14 lines

@SuppressWarnings("serial")
public class OpColumns extends TableQueryOp<TDcolumns, ColumnsBean> {

    public OpColumns() throws Exception {
        super(TDcolumns.class, ColumnsBean.class, Ctes.DS_NAME);
    }

    public OpColumns(TableQueryOp qop) throws Exception {
        super(TDcolumns.class, ColumnsBean.class, qop);
    }

    public ArrayList<Register> _getListColumns(String schema, String table) throws Exception {
        //Lista de selección
        SelectableItem[] selectItemArray = new SelectableItem[]{
            TDcolumns.name
        };

        return super.execSqlQuery(selectItemArray, "show columns from " + schema + "." + table);
    }

    public ArrayList<ColumnsBean> getListForTable(Long idtable) throws Exception {
        WhereAND wSearch = new WhereAND();
        wSearch.add(TDcolumns.idtable, ComparisonOp.EQ, idtable);

        OrderBy ord = new OrderBy();
        ord.add(TDcolumns.name, OrderOp.ASC);

        return super.select(wSearch, ord);
    }

    public ColumnsBean[] getArrayForCombo(Long idtable, boolean idonly) throws Exception {
        WhereAND wSearch = new WhereAND();
//          wSearch.add(TDcolumns.is_visible, ComparisonOp.IS, SQLCtes.TRUE);
        wSearch.add(TDcolumns.idtable, ComparisonOp.EQ, idtable);
        wSearch.add(TDcolumns.nameXml, ComparisonOp.NE, "");
        if(idonly){
            wSearch.add(TDcolumns.is_id, ComparisonOp.IS, SQLCtes.TRUE);
        }

        OrderBy or = new OrderBy();
        or.add(TDcolumns.nameXml, OrderOp.ASC);

        ArrayList<ColumnsBean> list = super.select(wSearch, or);
        return list.toArray(new ColumnsBean[list.size()]);
    }
```

Fig.7 Definición OpColumns.

```
public ColumnsBean[] getColumnsByTable(String searchTable) throws Exception {

    SelectableItem[] selectItemArray = new SelectableItem[]{
        TDcolumns.id,
        TDcolumns.name,
        TDcolumns.nameXml,
        TDcolumns.is_visible,
        TDcolumns.is_id
    };

    JoinTable[] joinArray = new JoinTable[1];

    WhereAND wAND0 = new WhereAND(TDcolumns.idtable, ComparisonOp.EQ, TDtables.id);

    joinArray[0] = new JoinTable(TDcolumns.TABLE_NAME,
            JoinOp.INNER_JOIN,
            TDtables.TABLE_NAME,
            wAND0);

    WhereAND wSearch = new WhereAND();
    wSearch.add(TDtables.name, ComparisonOp.EQ, searchTable);

    OrderBy or = new OrderBy();
    or.add(TDcolumns.nameXml, OrderOp.ASC);

    ArrayList<ColumnsBean> list = super.select(null, joinArray, wSearch, or);
    return list.toArray(new ColumnsBean[list.size()]);

}

public ArrayList<Register> _getList(Long idtable, String search) throws Exception {
    WhereAND wSearch = new WhereAND();
    wSearch.add(TDcolumns.idtable, ComparisonOp.EQ, idtable);
    wSearch.add(TDcolumns.name, ComparisonOp.LK, "%" + search + "%");

    OrderBy or = new OrderBy();
    or.add(TDcolumns.name, OrderOp.ASC);

    return super._select(null, wSearch, or);
}

public Long insertColumn(ColumnsBean bean) throws Exception {
    try {
        super.startTransaction();

        Long id = super.insert(bean);
        super.commitTransaction();
        return id;
    } catch (Exception ex) {
```

Fig.8 Funciones

```
    public Long insertColumn(ColumnsBean bean) throws Exception {
        try {
            super.startTransaction();

            Long id = super.insert(bean);
            super.commitTransaction();
            return id;
        } catch (Exception ex) {
            super.rollbackTransaction();
            throw (ex);
        }
    }

    public void updateColumn(ColumnsBean bean) throws Exception {

        try {
            super.startTransaction();

            super.update(bean);

            super.commitTransaction();

        } catch (Exception ex) {
            //Retroceder cambios
            super.rollbackTransaction();
            //Propagar la exception hacia arriba
            throw (ex);
        }
    }
}
```

Fig.9 Definición de funciones.

9.7. Modelos

9.7.1. Casos de Uso

En el presente capítulo se describen los diagramas de los principales casos de uso del sistema. Los mismos muestran la relación entre los actores y los casos de uso descritos.

Se identificaron dos actores principales para cada plataforma:

A) Desarrollador: Interactúa con la aplicación y su objetivo es generar un fichero XML con la estructura de la BD al el fin de incorporarlo en la aplicación del cliente.

B) Usuario: Actor genérico que interactúa con el sistema para obtener un resultado concreto. En este caso específico, interactuar con la plataforma a través de una interfaz amigable para realizar consultas simples a la base de datos y obtener los resultados esperados.

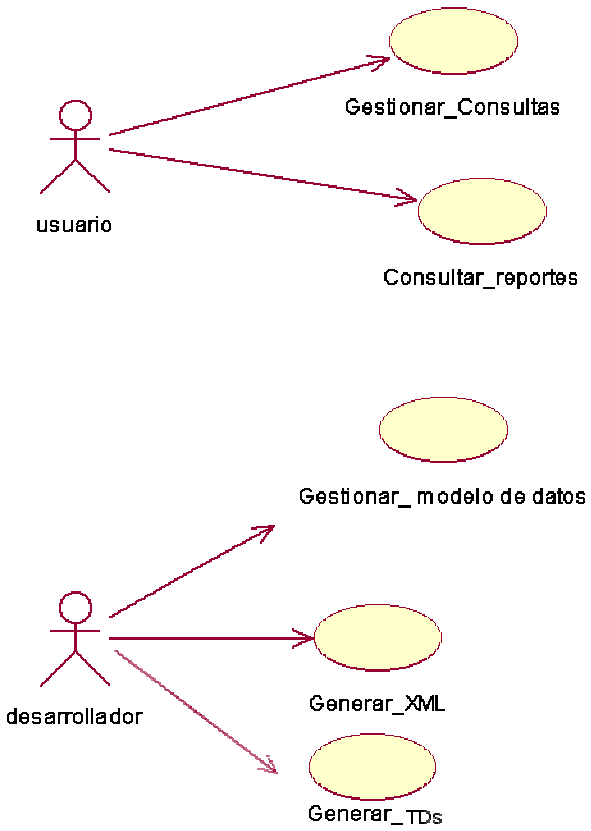


Fig. 10 Diagrama de casos de uso.

Caso de uso	Generar XML
Actor	Desarrollador
Propósito	Obtienes esquemas a partir de mapeos de bases de datos. Muestra cómo está estructurada esa configuración según la gestión del modelo de datos. Muestra las tablas, campos, y la relacione entre ellos.

Caso de uso	Generar TDs
Actor	Desarrollador
Propósito	Obtienes los TDs que mapearán las tablas de la base de datos.

Caso de uso	Gestionar modelo de datos
Actor	Desarrollador
Propósito	Posibilidad de almacenar las consultas realizadas, para poder realizarlas posteriormente sin tener que construirla nuevamente.

Caso de uso	Gestionar Consultas
Actor	Usuario
Propósito	Permite crear nuevas consultas y modificar o eliminar las realizadas.

Caso de uso	Consultar reportes

Actor	Usuario
Propósito	Ver los resultados de las consultas realizadas. Permite filtrar los datos e imprimirlos o exportarlos a Excel.

9.7.2. Diagramas de Clases

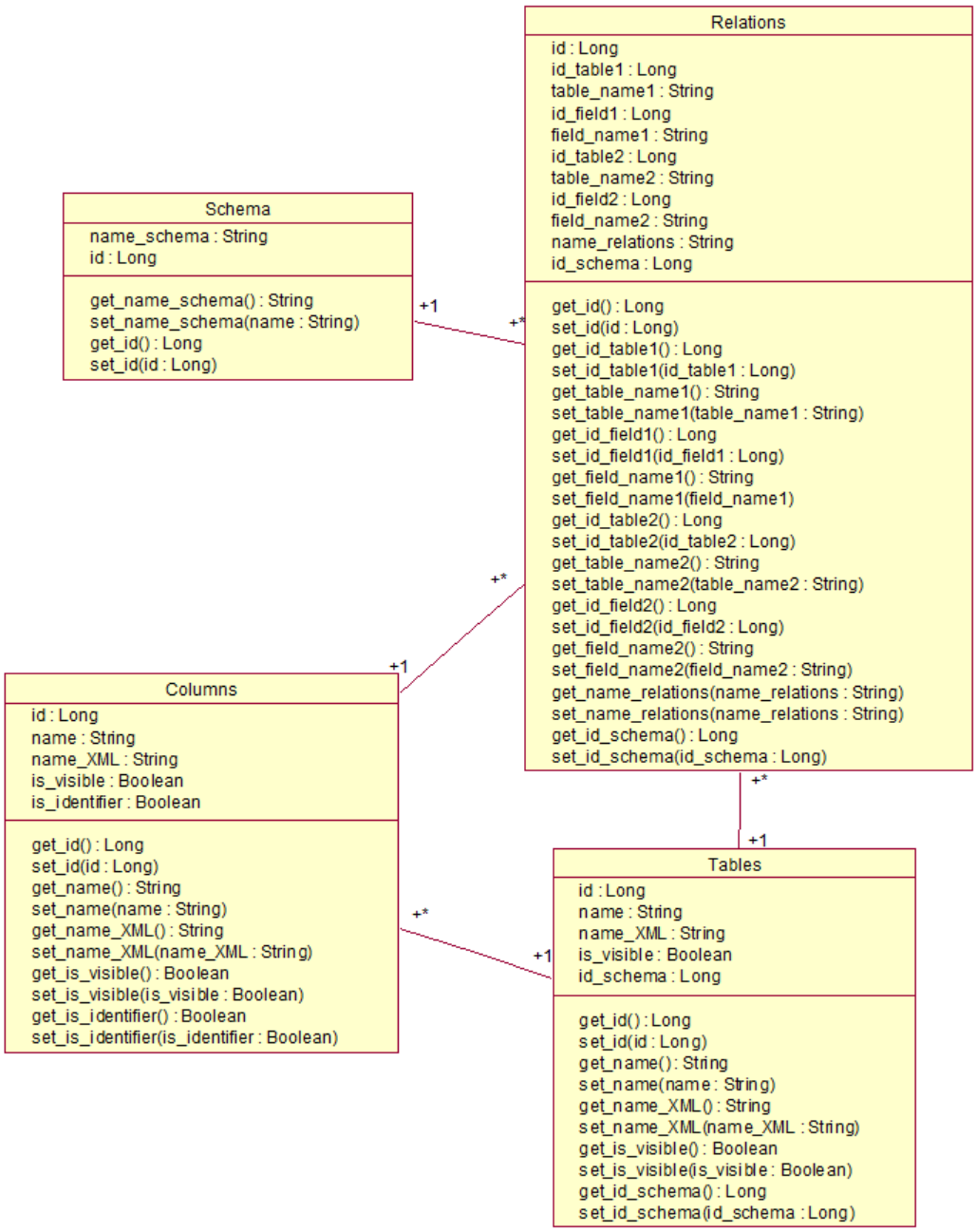


Fig.11 Diagrama de clase de implementación. Generador de Consultas

En la siguiente figura se muestra el diagrama de clases de implementación para la persistencia de datos. La misma se ejemplificará utilizando las tablas que se muestran en la figura: TDcolumns, ColumnsBean, OpColums. Para cada una de las demás entidades se realiza de forma similar.

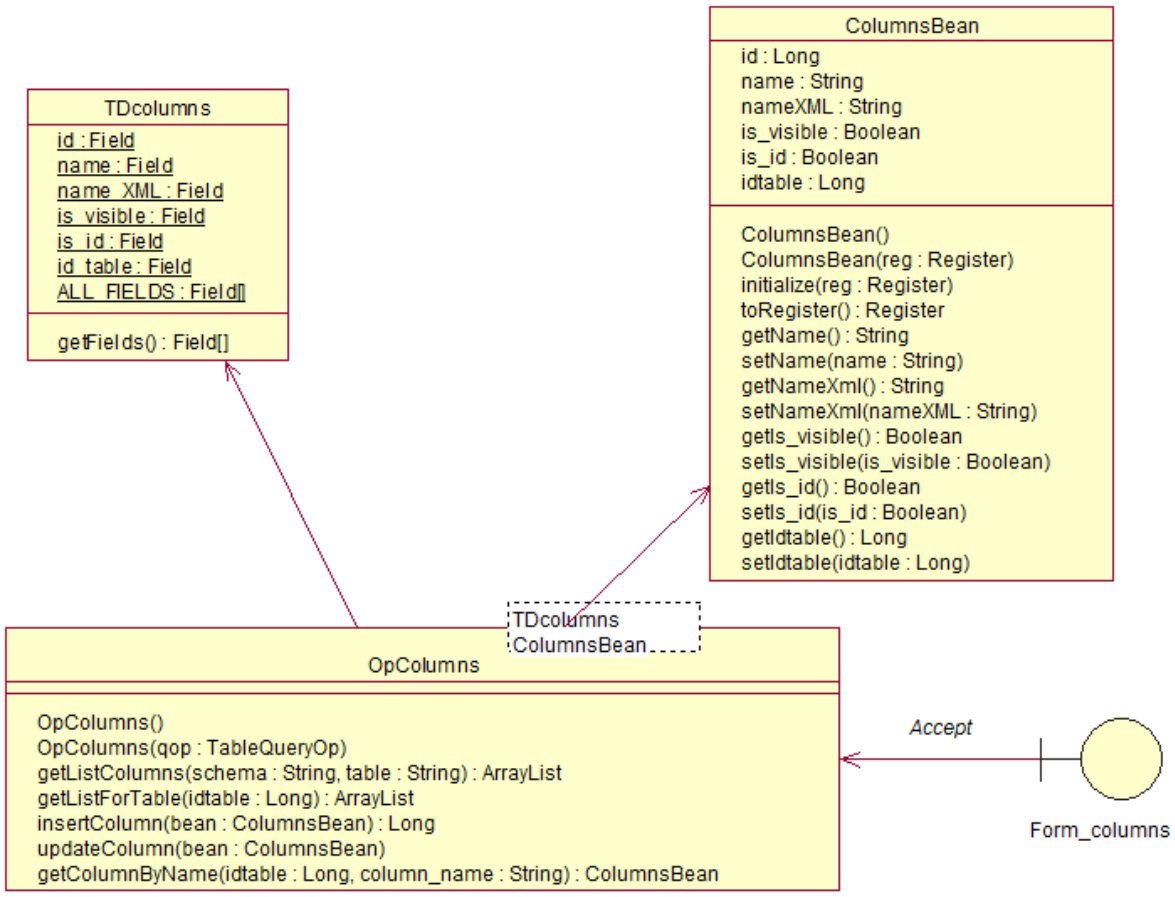


Fig.12 Diagrama de clase de implementación para la persistencia de datos.

9.7.3. Modelo lógico de datos

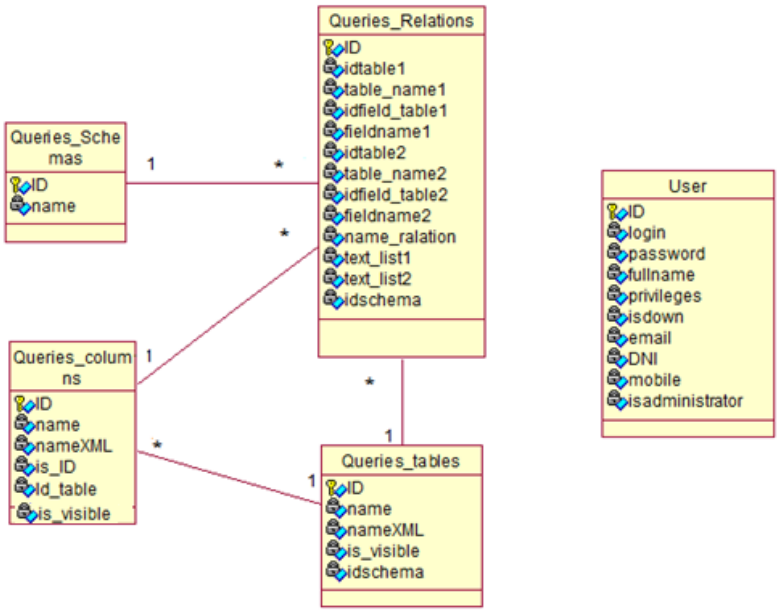


Fig.13 Modelo lógico de los datos Proyecto Generador de esquemas.

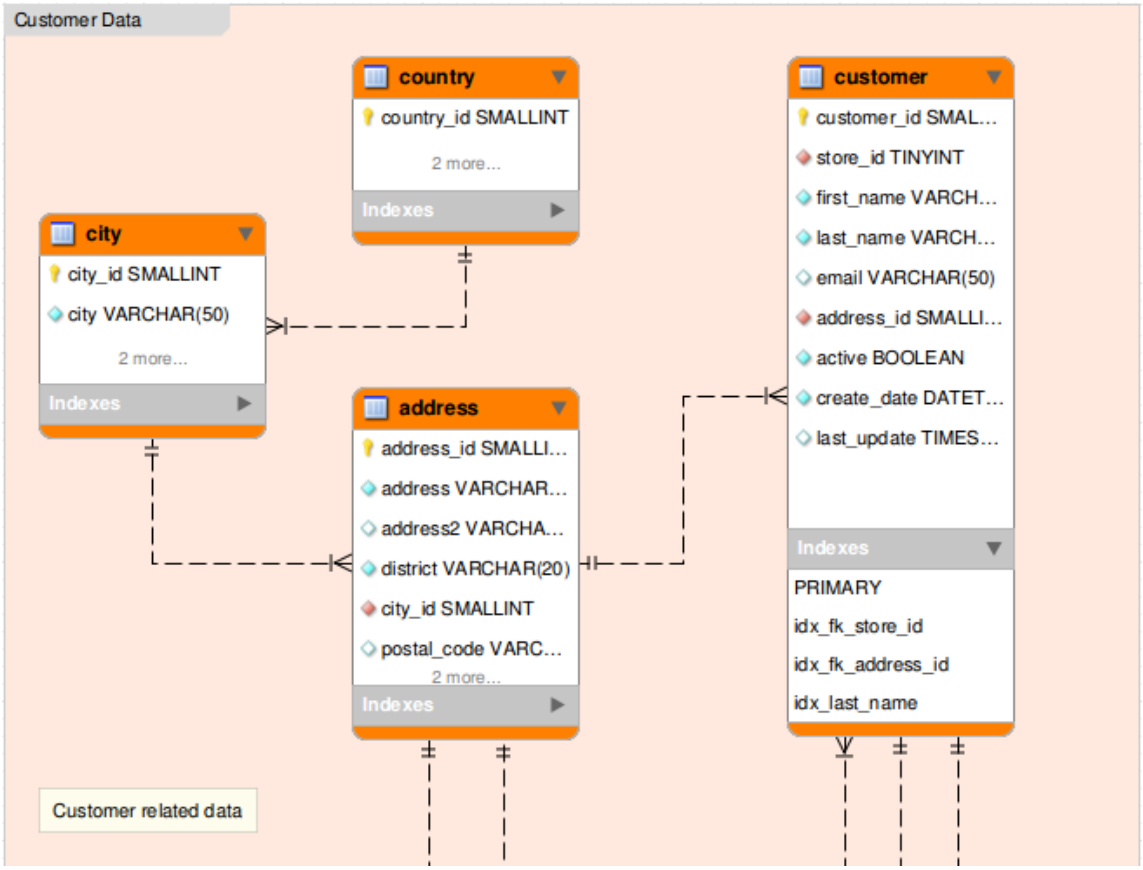


Fig.14 Modelo lógico de los datos Proyecto lanzadera.

Se debe aclarar que el modelo lógico representado en la figura 12 no está completo. Solo se expusieron las tablas relacionadas con el ejemplo de funcionamiento de la lanzadera expuesto en el capítulo 10.1.

NOTA: Tal como se ha especificado en los objetivos del proyecto, la capa persistente sobre la que operará la lanzadera podrá ser cualquier base de datos, nunca será una en concreto, en contra de lo que pudiera entenderse en la figura 12. En esta representación sólo se ha querido plasmar una de las bases de datos que se usará de ejemplo en el proyecto. No se debe entender que ese es el modelo relacional que subyace en su capa de datos.

9.7.4. Diagrama de Secuencia

En esta sesión se muestra un ejemplo de diagrama de secuencia perteneciente al generador de consultas. En el mismo se evidencia el comportamiento básico y exitoso de las acciones a realizar en el caso de uso escogido: Editar Usuario. Se puede ver, además de las relaciones que existen entre las clases representadas, que cada una de ellas cumple la función para las cuales fueron diseñadas, logrando conseguir una arquitectura de base sólida para lograr el éxito de los procedimientos.

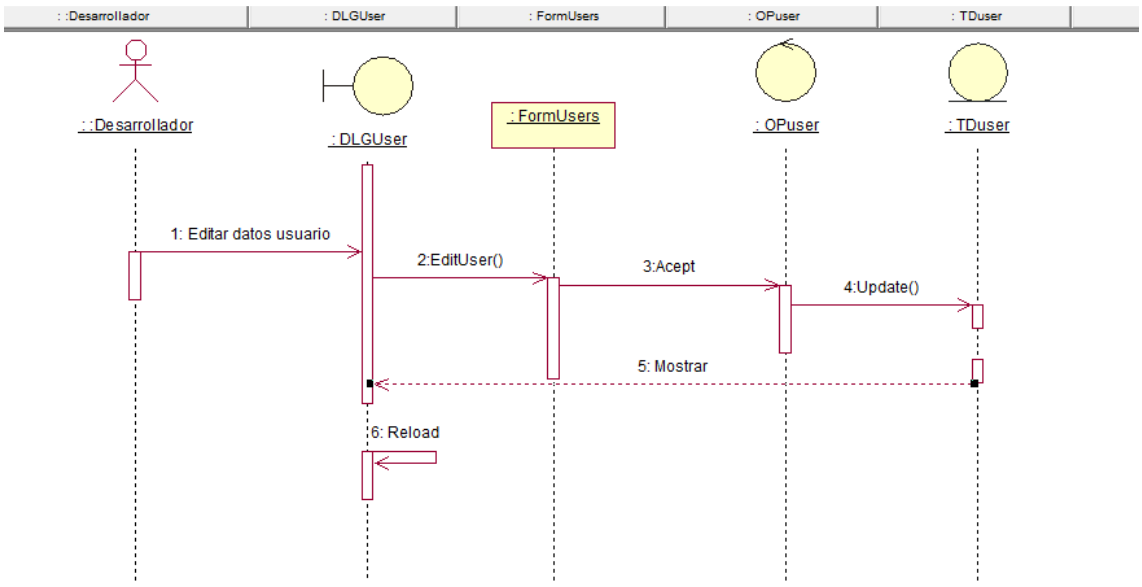


Fig.15 Diagrama de secuencia editar usuario en el Generador de consultas

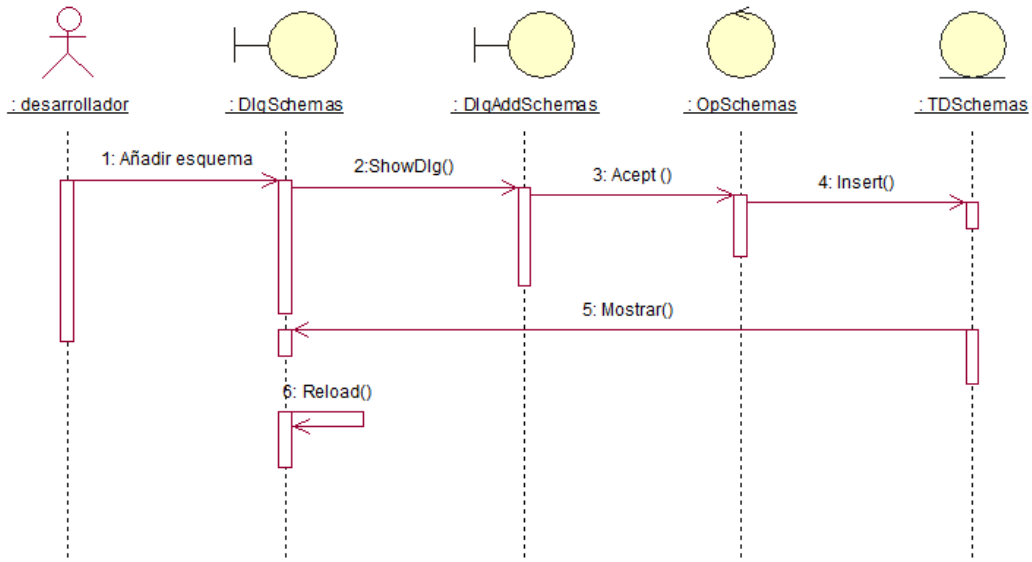


Fig.16 Diagrama de secuencia Añadir Esquema

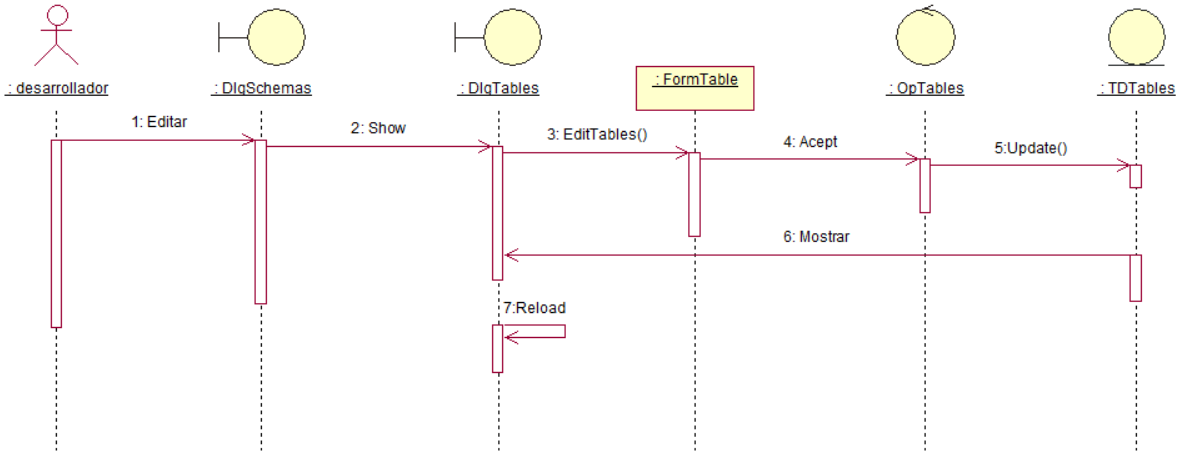


Fig.17 Diagrama de secuencia Editar tablas

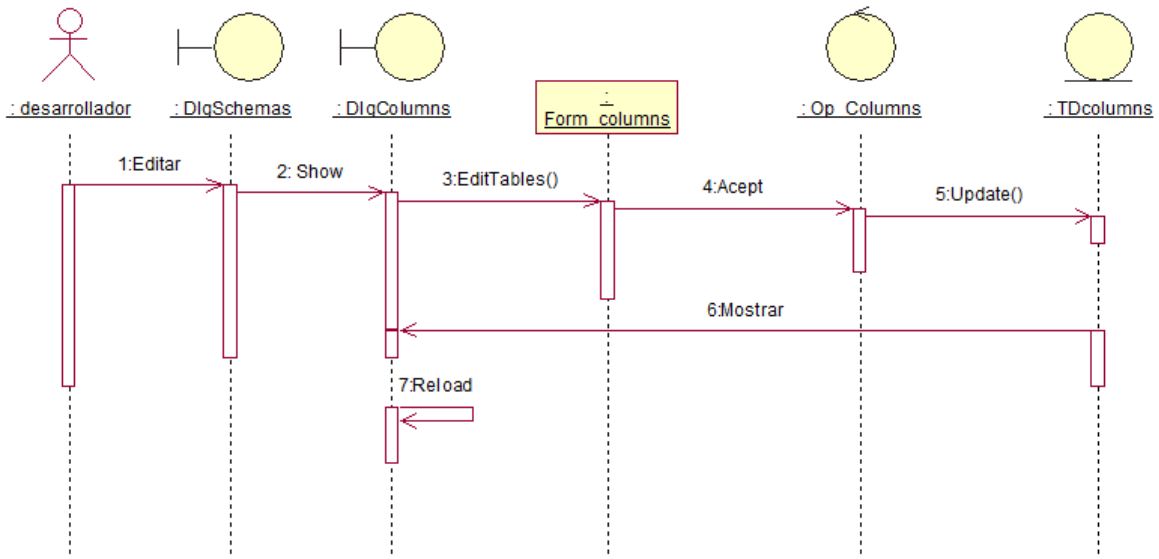


Fig.18 Diagrama de secuencia Editar columnas

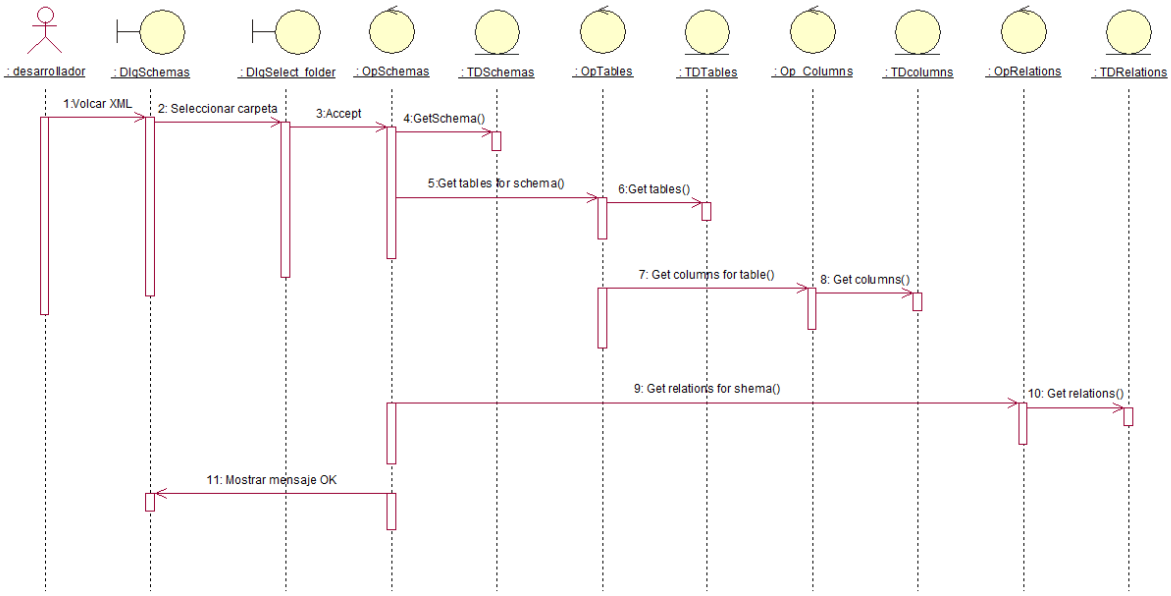


Fig.19 Diagrama de secuencia Generar XML

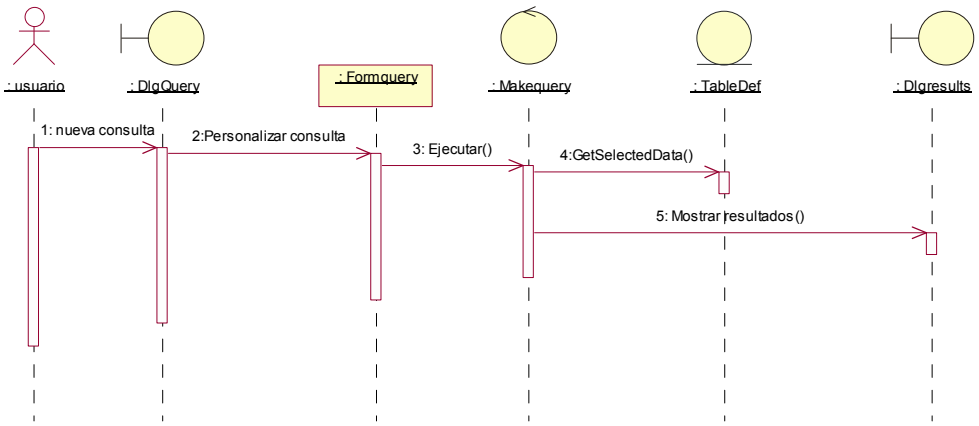


Fig.20 Diagrama de secuencia para crear nueva consulta en la aplicación Lanzadera.

9.8. Pruebas

Siguiendo el ciclo de vida del desarrollo propuesto por RUP, se realizaron pruebas de los sistemas implementados. Estas pruebas se centraron más bien en la arquitectura, cuando el grueso de los sistemas se estaba construyendo. Se planificaron las misma para que el desarrollo no se detuviera. Se logró avanzar en la implementación y a la vez detectar errores en los primeros prototipos del producto.

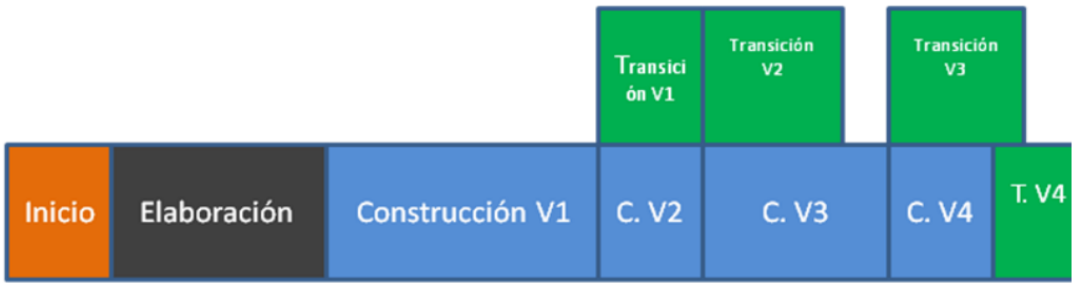


Fig.21 Planificación de prueba según versiones.

Se comprobó si la interfaz de usuario era utilizable y consistente. Se escogieron varios procedimientos como por ejemplo, Añadir condición y Crear relación, pertenecientes a la lanzadera y al generador de consultas respectivamente, con el fin de realizarles pruebas de funcionamiento con datos reales. Esto permitió comprobar el buen estado de todas las operaciones y corregir situaciones anómalas, como por ejemplo:

Generador de consultas:

- En el generador de consulta, se fijaba las mismas relaciones a todas las tablas.
- Introducir campos vacíos.
- En la lanzadera se debe especificar una condición antes de ejecutarla.

10. Interfaz de usuario

10.1. Lanzadera

A continuación se muestran los principales elementos de la interfaz de usuario para la lanzadera de consultas junto con una breve explicación.

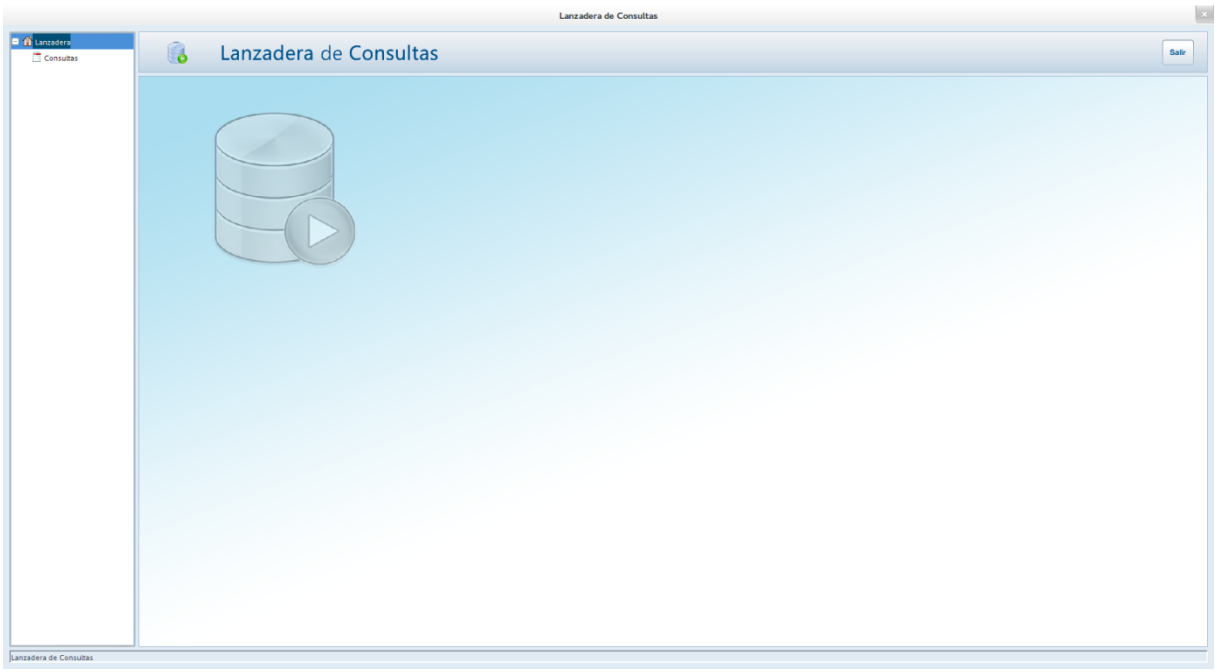


Fig.22 Lanzadera de consultas.

La ventana principal (Fig.22) nos presenta las diferentes funcionalidades que la aplicación nos permite.

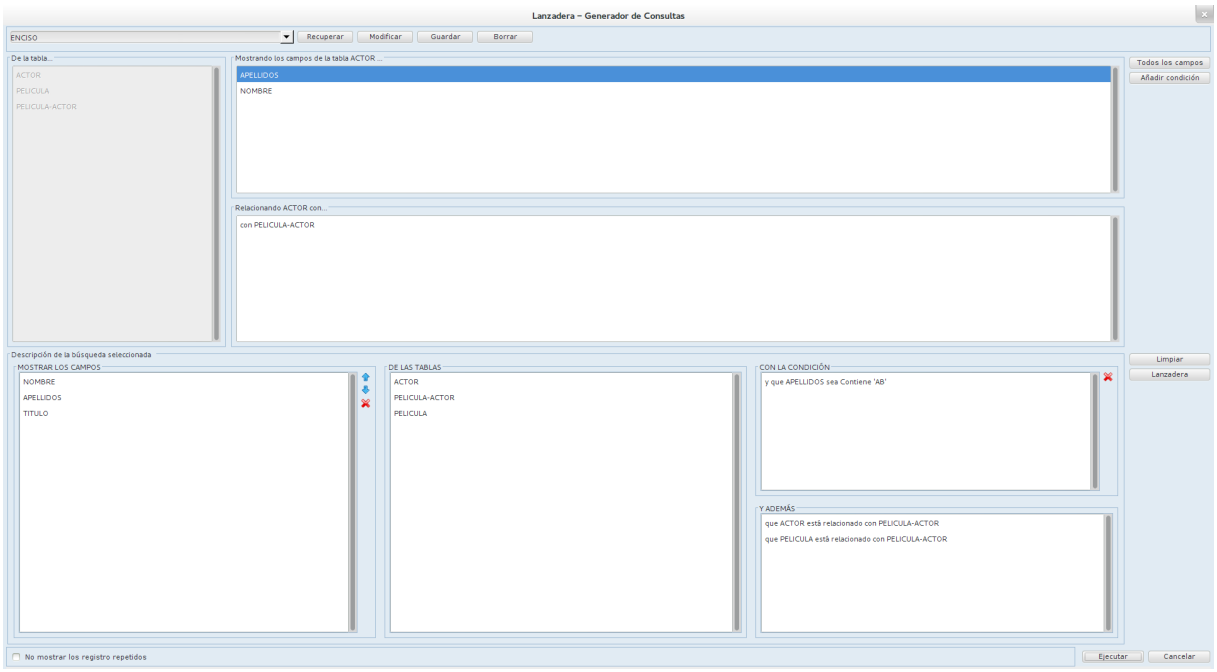


Fig. 23 Pantalla principal

La primera funcionalidad muestra un combo de selección donde se puede escoger alguna consulta que se ha generado y que fue guardad en una sesión anterior. A la misma se le puede realizar las siguientes acciones:

- **Recuperar:** Recupera los resultados de la consulta seleccionada.
- **Modificar:** Modifica la consulta seleccionada.
- **Guardar:** Guarda la consulta.
- **Borrar:** Elimina la consulta seleccionada.

En el panel izquierdo se lista todas las tablas que conforman la base de datos. En la sección derecha podemos observar dos listados:

- **Mostrando el campo**: Muestra los campos de la tabla que son visibles en el XML.
- **Relacionando tabla con**: Se muestran las relaciones que existen con la tabla seleccionada.

En la parte inferior se muestra una descripción de la búsqueda seleccionada, es decir, se listan los campos que se seleccionaron y las tablas a las que pertenecen.

Para añadir la condición deseada se debe dar clic en el botón Añadir condición.

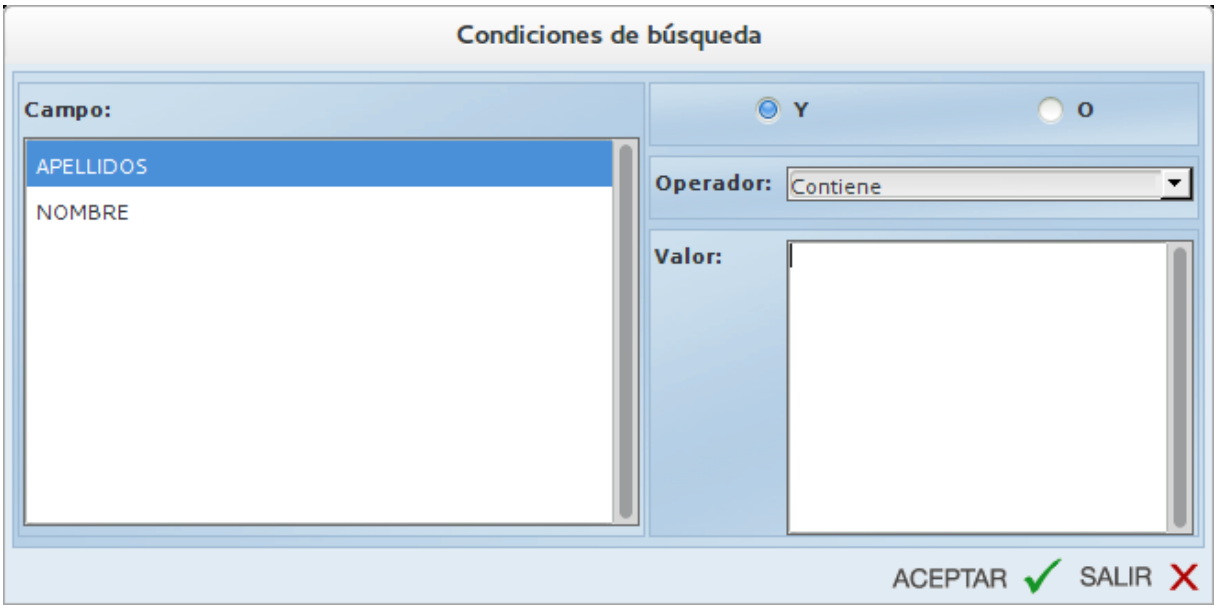


Fig.24 Condiciones de búsqueda.

La ventana Condiciones de búsqueda posibilita crear las condiciones para realizar el filtrado de los datos. En la parte izquierda se muestran los campos y en la parte derecha te permite escogerlos operadores booleanos, relacionales y de posición que se va utilizar (Y=AND, O=OR, NO contiene=NOT, Contiene=LIKE, Mayor que, Menor que,…) y asignarle el valor que se utilizará para guiar la búsqueda.

Una vez establecida la condición se debe ejecutará la Lanzadera, con los campos, el operador y el valor definidos.

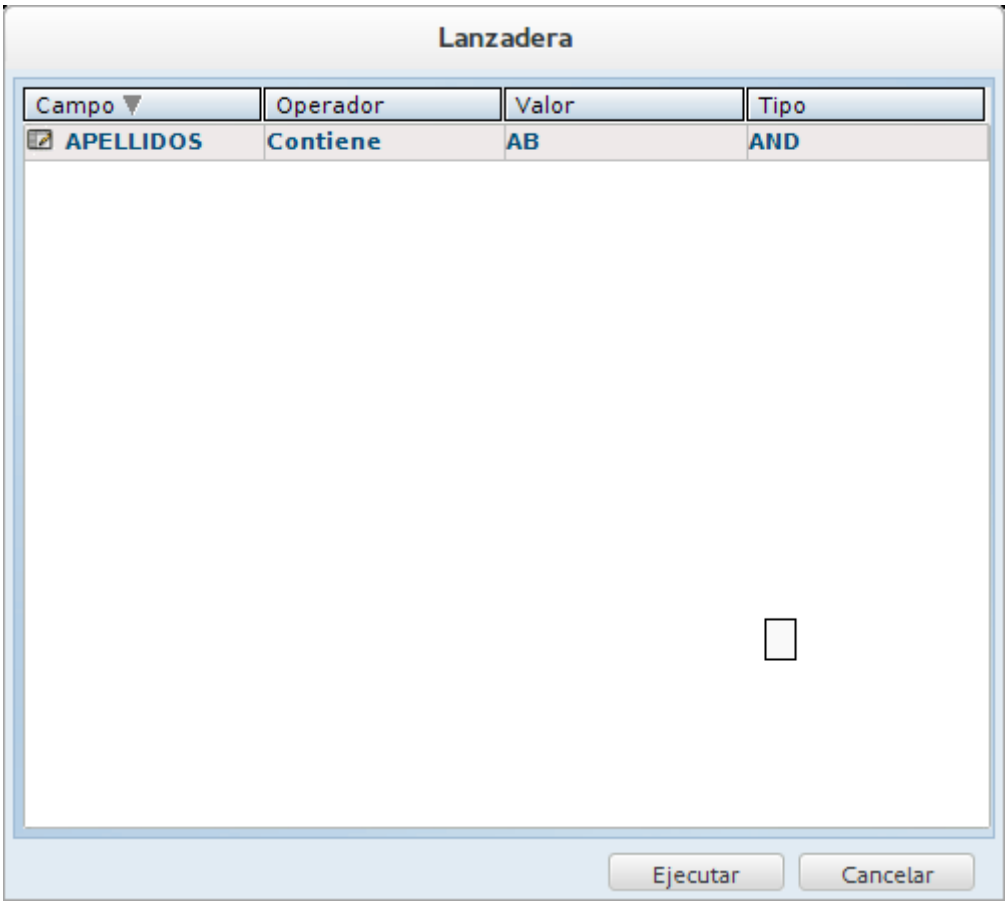


Fig.25 Lanzadera.

El resultado de la ejecución de la consulta será el listado con los valores que cumplen con la consulta especificada. Estos valores se pueden imprimir o exportar a un archivo Excel o a un PDF (impresión en formato fijo con paginación). Además se permite eliminar una fila en caso de que no se desee mostrarla en el reporte.

También se puede aplicar sobre cada columna las funciones de count, max, min y sum

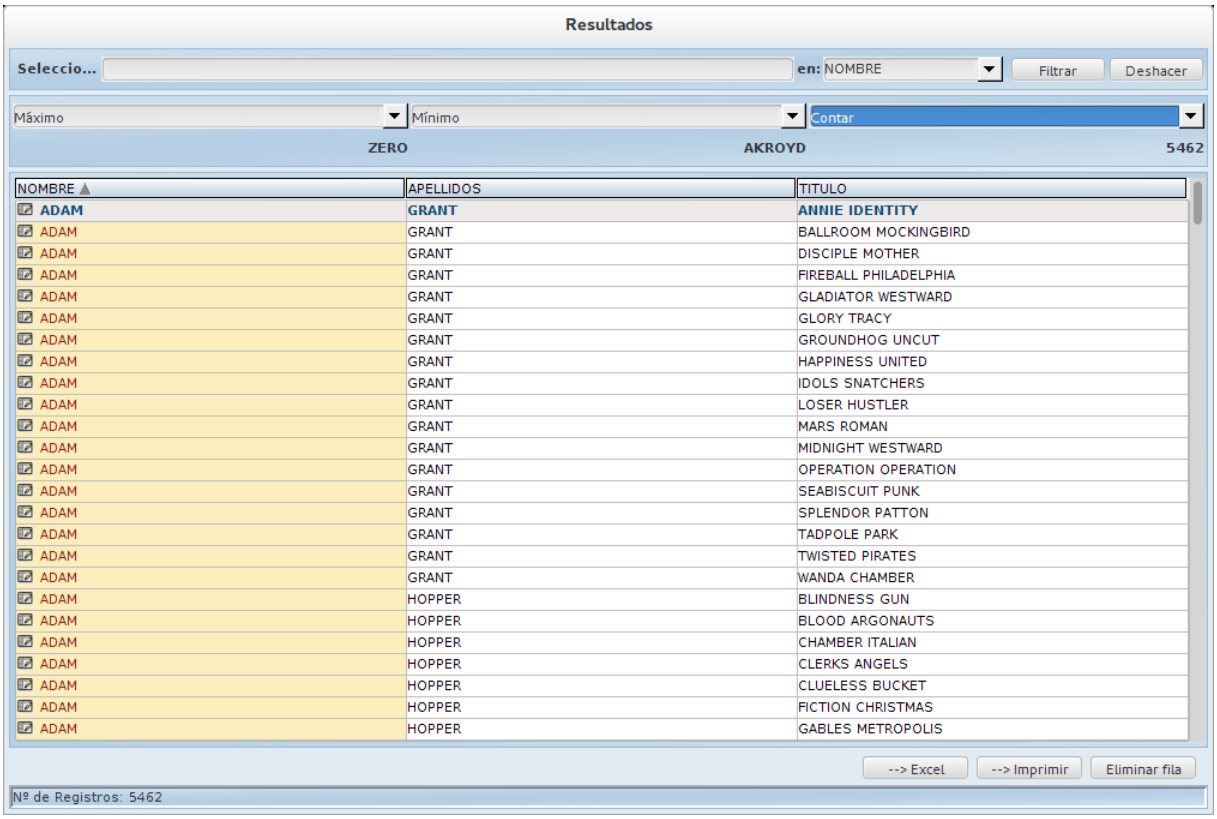


Fig.26 Resultado de la ejecución de la consulta.

10.2. Aplicación Generador de asistente para consultas.

La página principal de la aplicación muestra sus principales operaciones: Usuarios y Esquemas.

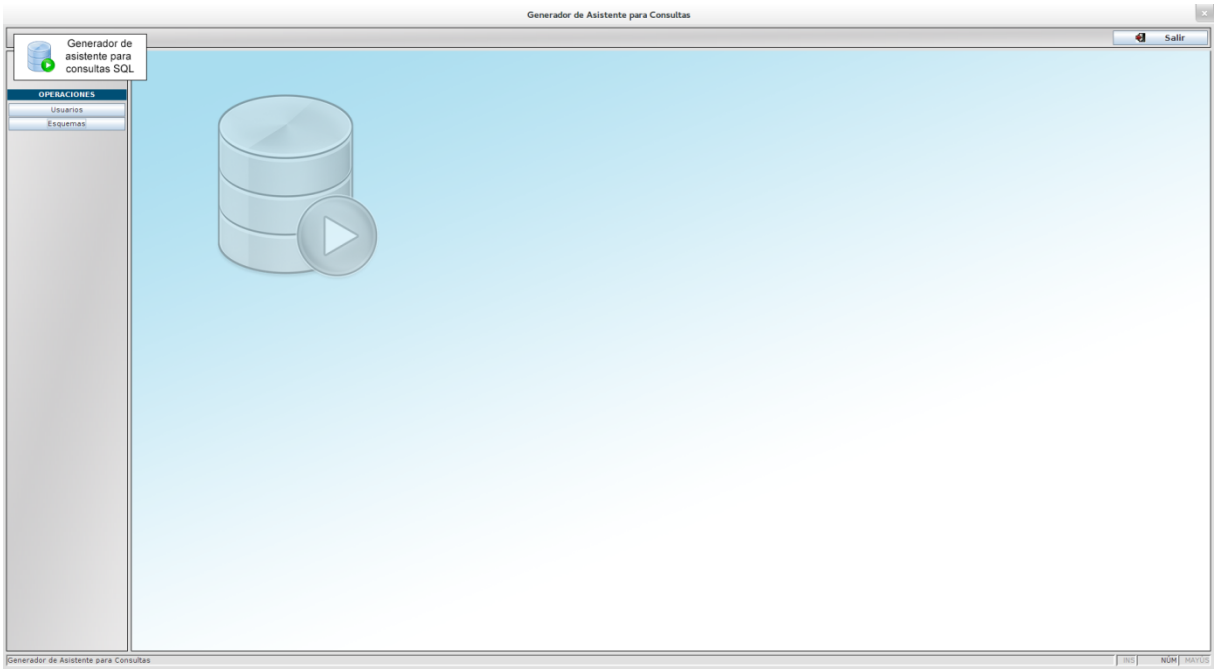


Fig.27 Página principal del Generador de Asistente para Consultas.

El módulo de usuarios permite gestionar a todos los usuarios de la aplicación. Sus principales funciones son:

- **Añadir**: Permite añadir nuevos usuarios.
- **Editar**: Edita los datos de los usuarios creados.
- **Borrar**: Elimina a los usuarios.

Nota: Se debe destacar que a los usuarios que se creen en esta aplicación se le deberá darles los permisos de Select (grant select privileges) a las BD en MySQL, para realizar cualquier operación sobre los datos almacenados. Esto será transparente para el usuario final.

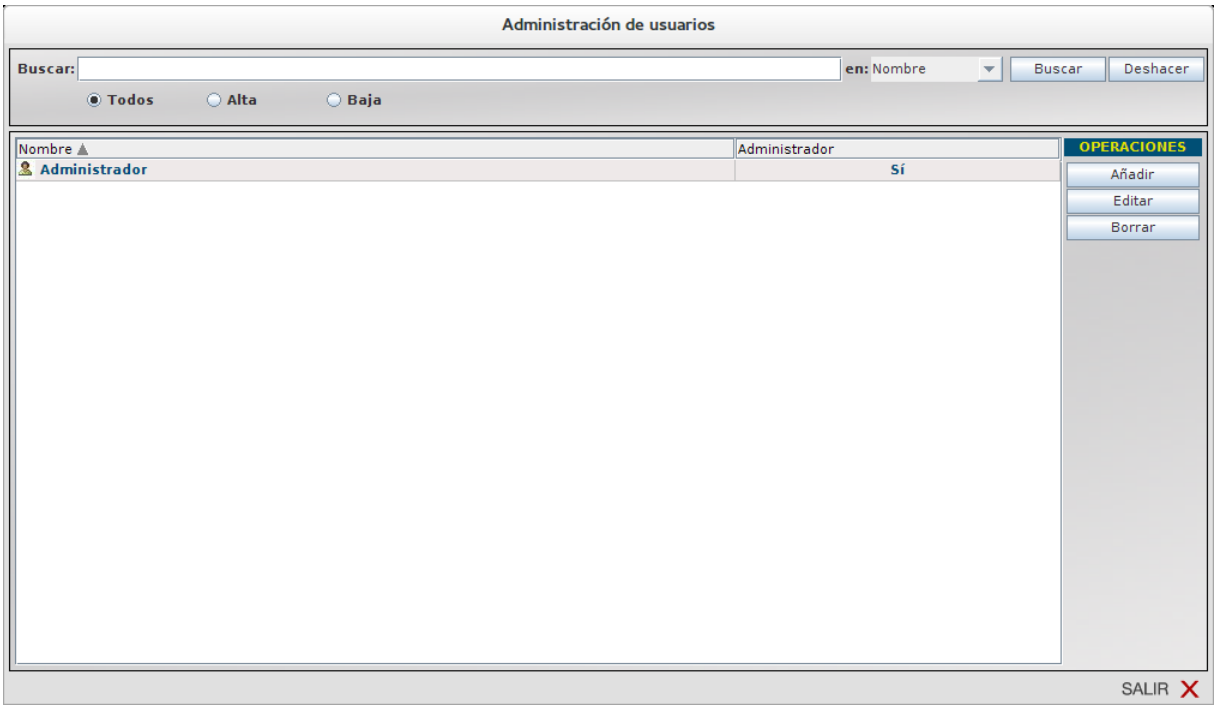


Fig.28 Administrar usuarios.

Se puede realizar búsquedas de usuarios según los criterios de selección deseados: Usuarios dados de baja o los que están activos, por nombres o si son administrador o no. Los resultados de la búsqueda se visualizarán en la tabla del centro de la ventana.

Para añadir a un usuario se tiene en cuenta los siguientes parámetros: Nombre, usuario, contraseña. Es necesario además especificar el nivel de acceso que va a tener, Administrador (puede añadir nuevos usuarios) o Usuario común (generador de

los XML y los TDs). En caso de baja por algún motivo, se debe macar la opción *dado de baja*.

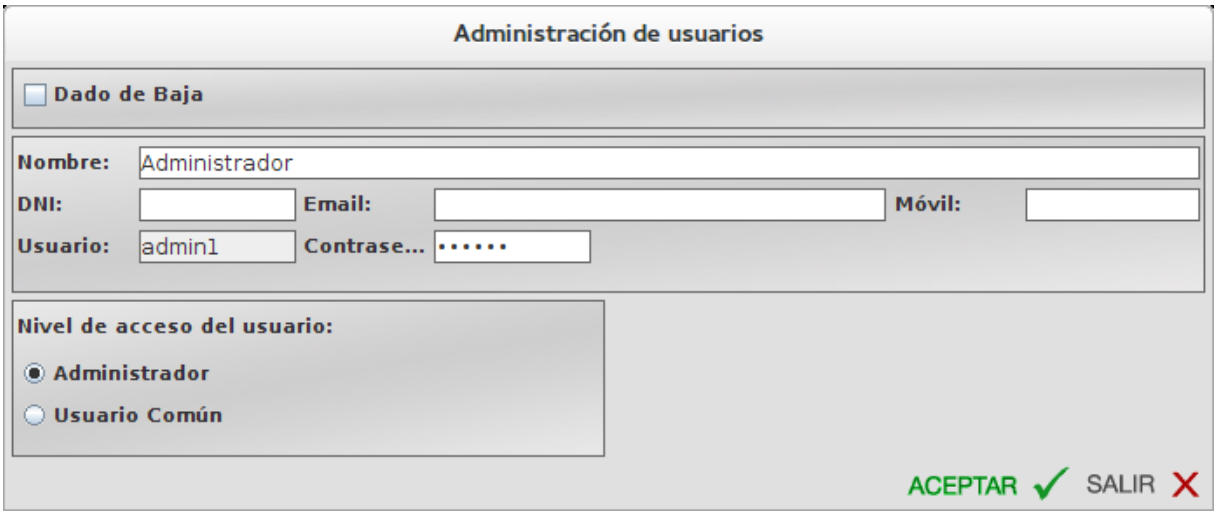


Fig.29 administrar usuarios

El módulo Esquema, permite gestionar los esquemas existentes.

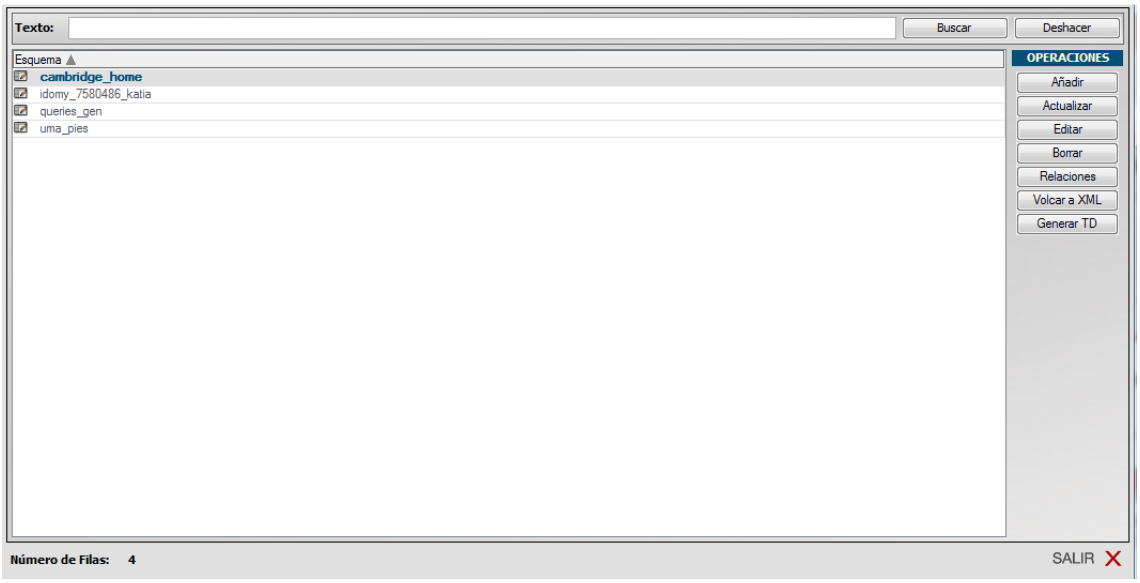


Fig.30 Administrar esquemas.

Las operaciones asociadas son:

- **Añadir**: Permite añadir nuevos esquemas que se encuentran en el gestor de base de datos.

- **Actualizar**: Permite actualizar los datos de los esquemas.
- **Editar**: Permite modificar los datos de los esquemas creados.
- **Borrar**: Elimina el esquema seleccionado.
- **Relaciones**: Define las relaciones de las tablas.
- **Volcar a XML:** Genera el XML del esquema, que posteriormente será utilizados en la lanzadera de consultas de las otras aplicaciones.
- **Generar TD:** Genera los ficheros TD que mapean el esquema seleccionado.

Si se selecciona un esquema, dando doble clic sobre la fila o editándolo, se abrirá una ventana para gestionar las tablas de ese esquema.

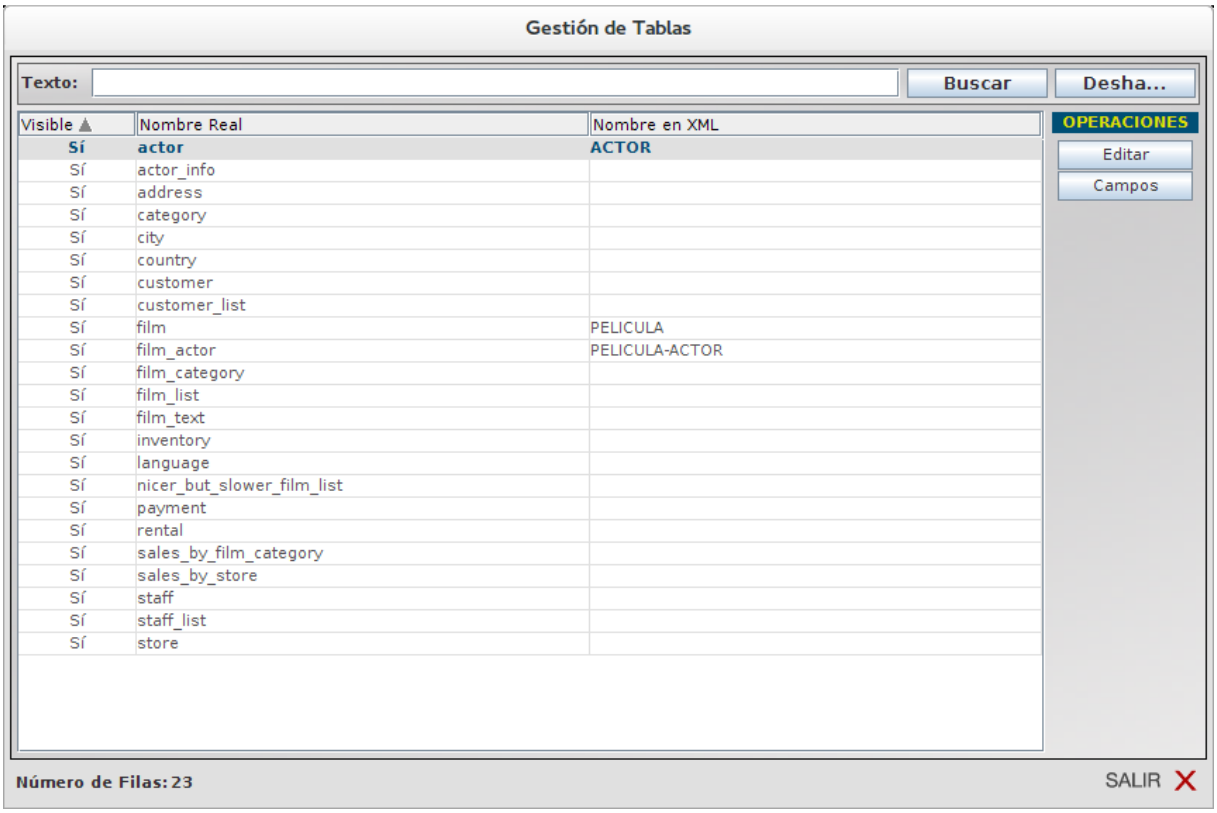


Fig.31 Gestión de tablas.

Por defecto se muestran todas las tablas pertenecientes al esquema escogido con sus nombres originales. Si se desea buscar específicamente una, se puede realizar el filtrado en el espacio habilitado para esta operación. Los datos a mostrar son: visible, nombre real y nombre en XML. En la derecha están habilitadas las operaciones a realizar:

- **Editar**: Permite modificar los datos de la ficha de la tabla.
- **Campos**: Muestra los campos asociados a la tabla seleccionada.

La ventana Gestión de campos, lista todos los campos que pertenecen a la tabla seleccionada.

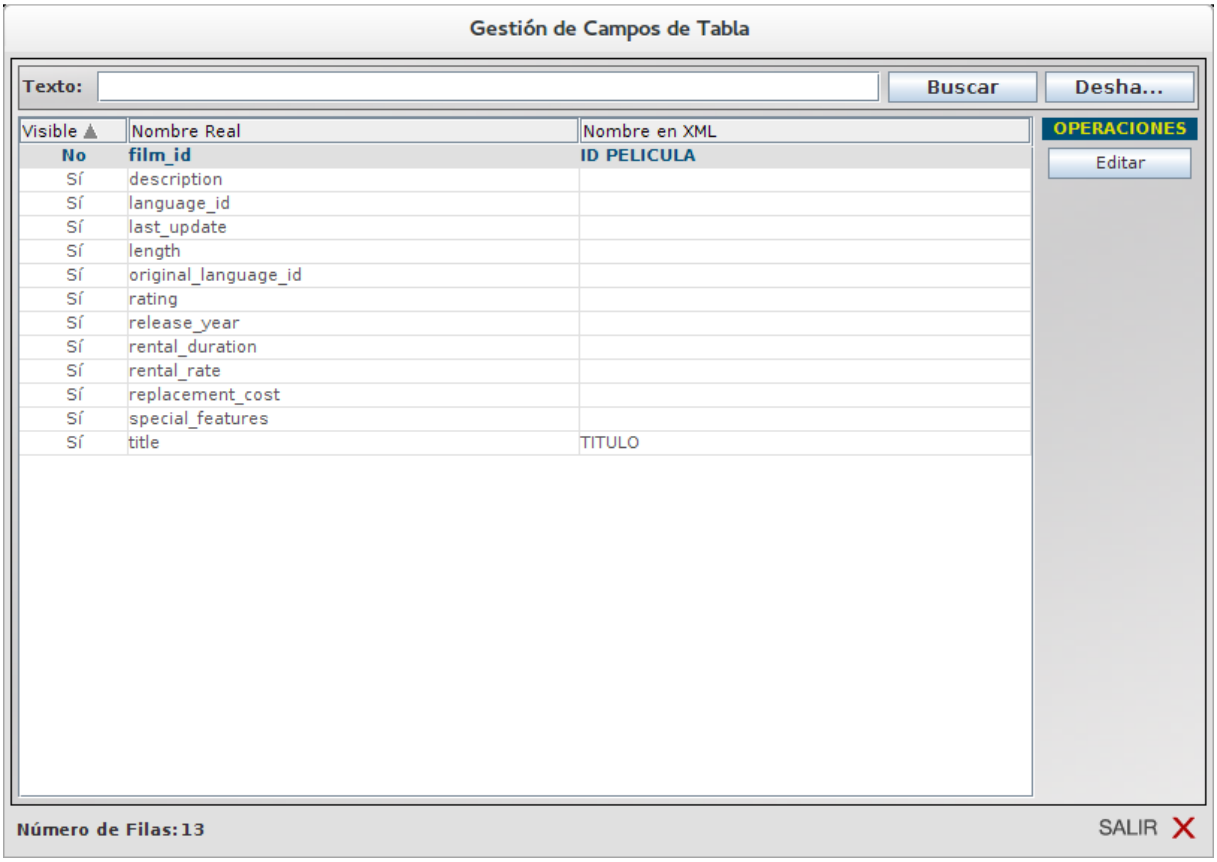


Fig.32 Gestión da campos de tabla

En la opción editar permite modificar los datos de la ficha del campo seleccionado.

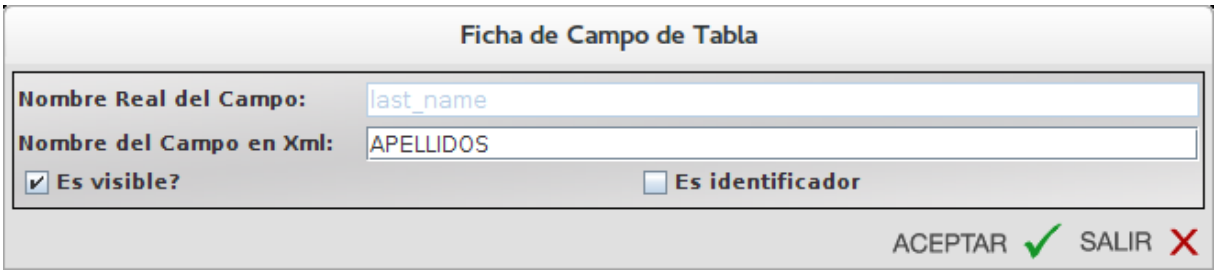


Fig. 33 Campo tabla

El checkbox es ID, me indicará que ese campo es identificador y me servirá para establecer las relaciones que quiero establecer de forma manual por ausencia de los FOREIGN KEYS establecidas en la creación original de las tablas

Se debe aclarar que los datos reales de los esquemas, campos y las tablas no se pueden modificar, están bloqueados.

La etiqueta relaciones, de la ventana Esquemas, permite indicar las relaciones que hay entre las tablas. Se debe destacar que sólo se pueden relacionar aquellas tablas que sean visibles y que sus campos sean identificadores.

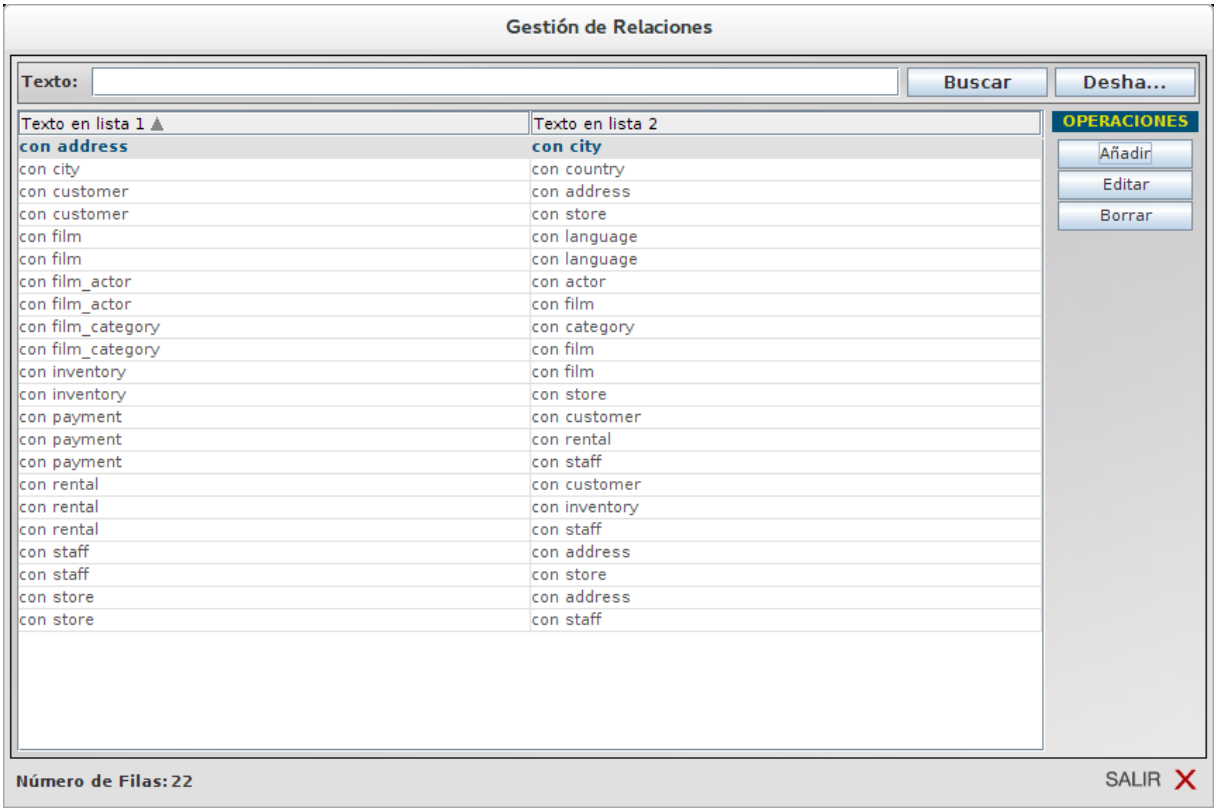


Fig.34 Gestión de relaciones.

Las operaciones a realizar son:

- **Añadir**: Permite crear nuevas relaciones.
- **Editar**: Permite modificar las relaciones creadas.
- **Borrar**: Elimina la relación seleccionada.

Si no existen relaciones (porque no se han establecido FOREIGN KEYS), la ventana aparecerá vacía. a medida que se creen, o se establezca desde secuencias SQL, se visualizará el listado de las mismas con los textos de la lista1 y de la lista 2.

Para crear nuevas relaciones se deben llenar los campos de la ficha relación

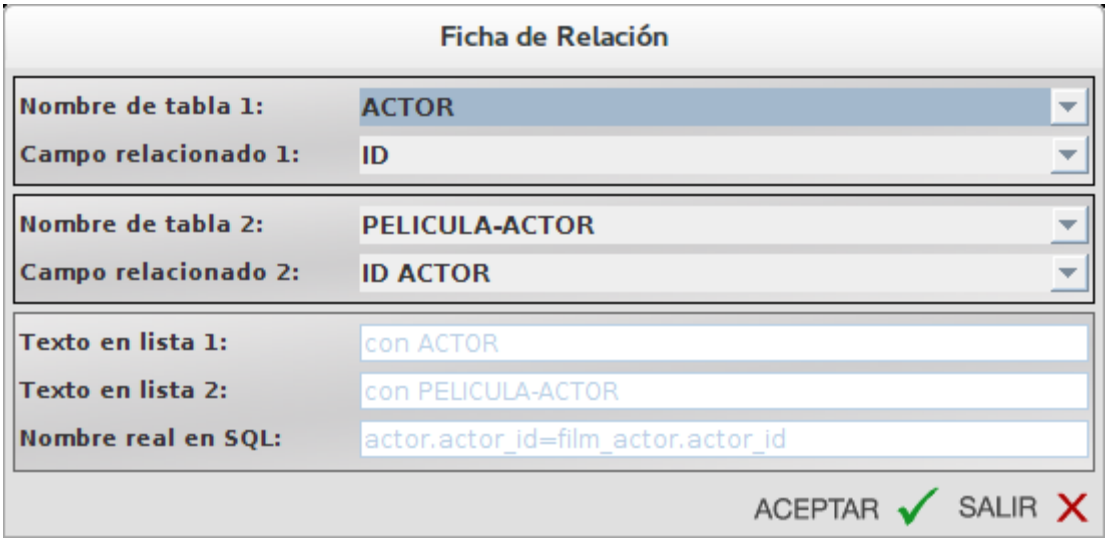


Fig.35 Añadir relación.

En la etiqueta con nombre de la tabla 1, se selecciona la tabla a relacionar. Una vez elegida, en el campo relacionado1 se listan todos los campos identificadores de esa tabla. Lo mismo sucede con la tabla y campo 2.

Los campos Texto en lista 1 y 2, así como nombre real SQL, se crearán automáticamente y no son modificables.

Una vez realizadas todas las relaciones se pueden exportar a un archivo XML, dando clic en la opción Volcar a XML de la ventana Esquemas. Esta acción me abrirá una ventana para seleccionar la carpeta donde se desea almacenar el fichero generado. Se debe destacar que se guarda por defecto la última ruta seleccionada.

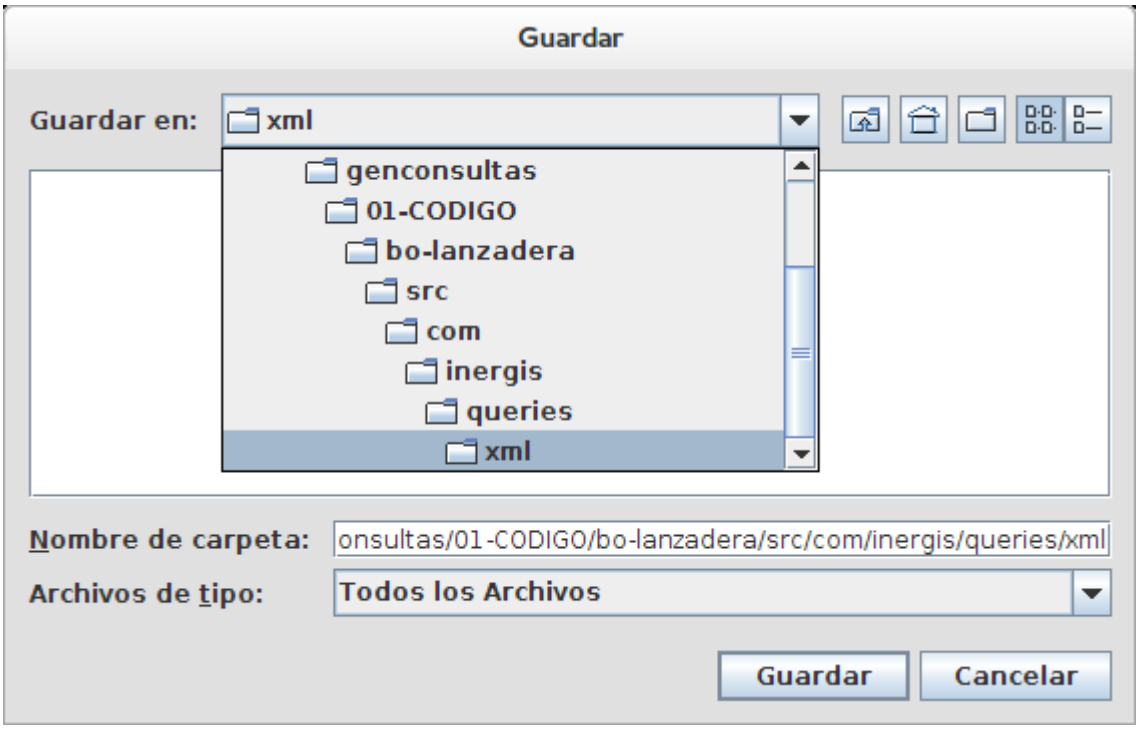


Fig.36 Ventana para especificar la ruta donde se almacenará el fichero XML.

Siguiendo el mismo ejemplo de la relación Pacientes y Provincias la estructura generada dentro del XML sería la siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
- <SCHEMA>
  - <TABLES>
    - <TABLE>
        <NAME>pacientes</NAME>
        <TRUE_NAME>pacientes</TRUE_NAME>
       - <FIELDS>
          - <FIELD>
              <NAME>ci</NAME>
              <TRUE_NAME>ci</TRUE_NAME>
            </FIELD>
          - <FIELD>
              <NAME>ci</NAME>
              <TRUE_NAME>ci</TRUE_NAME>
            </FIELD>
          - <FIELD>
              <NAME>edad</NAME>
              <TRUE_NAME>edad</TRUE_NAME>
            </FIELD>
          - <FIELD>
              <NAME>prov</NAME>
              <TRUE_NAME>id_provincia</TRUE_NAME>
            </FIELD>
          - <FIELD>
              <NAME>nombre</NAME>
              <TRUE_NAME>nombre_apellidos</TRUE_NAME>
            </FIELD>
          - <FIELD>
              <NAME>nombre</NAME>
              <TRUE_NAME>nom_apell</TRUE_NAME>
            </FIELD>
        </FIELDS>
      </TABLE>
    - <TABLE>
        <NAME>provincia</NAME>
        <TRUE_NAME>provincia</TRUE_NAME>
       - <FIELDS>
          - <FIELD>
              <NAME>idprov</NAME>
              <TRUE_NAME>id_provincia</TRUE_NAME>
            </FIELD>
          - <FIELD>
              <NAME>provincia</NAME>
              <TRUE_NAME>nombre</TRUE_NAME>
            </FIELD>
        </FIELDS>
      </TABLE>
  </TABLES>
  - <RELATIONSHIP>
    - <RELATION>
        <NAME_FROM>pacientes</NAME_FROM>
        <NAME_WITH>provincia</NAME_WITH>
        <NAME_LIST_FROM>con paciente</NAME_LIST_FROM>
        <NAME_LIST_WITH>con provincia</NAME_LIST_WITH>
        <TRUE_NAME>pacientes.id_provincia=provincia.id_provincia</TRUE_NAME>
      </RELATION>
  </RELATIONSHIP>
</SCHEMA>
```

Fig.37 Estructura del XML generado de la relación paciente-provincia.

Al accionar el botón Generar TD, me abre igualmente una ventana que permite especificar la ruta donde se van a almacenar los ficheros TD que mapean el esquema seleccionado y que corresponden al modelo de la aplicación (MVC)

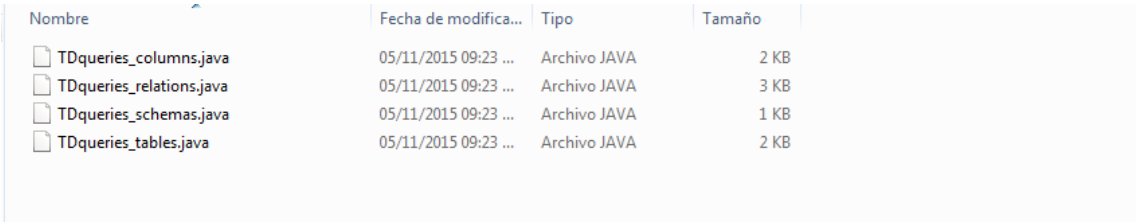


Fig.38 Ficheros TD.java generados.

11. Conclusiones.

Con la realización de este proyecto y la implementación de las aplicaciones se cumplió el objetivo central del trabajo. Se demostró además que la relación entre el generador de consultas y la lanzadera Java es factible tanto para los desarrolladores como para los usuarios finales de los aplicativos. Se mostraron las bases que sustentan el proyecto, lográndose una correcta concepción e implementación de los sistemas que responden a los procesos objetos de informatización. Se diseñó una BD en MySql para el generador de consultas donde se almacenará la información persistente de los esquemas generados.

La forma dinámica del archivo XML volcado permitirá su integración en los diferentes sistemas.

La lanzadera posibilita obtener mayor personalización en las consultas a realizar, logrando mayor exactitud en los resultados a obtener.

Es recomendable, debido al carácter iterativo e incremental del proyecto, continuar la revisión de la calidad, con vistas a mantener el buen funcionamiento.

Bibliografía

[1] Alcolea Velázquez, J. Álvarez Arrieta y F. Moreno Iglesias, L. (2007). *Generador del Modelo Relacional y Esquemas de Bases de Datos a partir del modelo Entidad/Relación*. (Proyecto de Sistemas Informáticos).Universidad Complutense de Madrid, facultad de Informática. Madrid, España.

[2] Dubretic, M. (2014). Oracle vs MySQL vs SQL Server: una comparación entre los Sistemas Gestores de Bases de Datos Relacionales más Populares.[Mensaje en un blog]. Recuperado de: https://blog.udemy.com/es/oracle-vs-mysql-vs-sql-server-una-comparacion-entre-los-sistemas-gestores-de-bases-de-datos-relacionales-mas-populares/

[3] Edu4Java. (s.f). *Tipos de bases de datos, Clientes SQL y Esquemas en Bases de Datos.* Edu4Java. Recuperado en: http://www.edu4java.com/es/sql/sql2.html

[4] Gutiérrez, P. (2013). *Fundamento de las bases de datos: Modelo entidad-relación.* GENBETA: dev. Recuperado de: http://www.genbetadev.com/bases-de-datos/fundamento-de-las-bases-de-datos-modelo-entidad-relacion.

[5] DB-ENGINES. The DB-Engines Ranking ranks database management systems according to their popularity. Recuperado de http://db-engines.com/en/ranking

[6] Junta de Andalucía (s.f). Funcionalidades de la capa de persistencia[Sitio Web Accedido el: 13 de Agosto 2015]. Recuperado enhttp://www.juntadeandalucia.es/servicios/madeja/contenido/libro-pautas/13

[7] Scribd. (2015). Read Unlimited Books.Recuperado de: http://es.scribd.com/doc/36570454/Diferencia-Entre-PostgreSQL-MySQL-Oracle#scribd

[8] Soto Corzo, JM. Díaz Portillo, D y Cruz Zamora, JA. (s.f). Sistema de Consultas en Lenguaje Natural para Bases de Datos. Instituto Tecnológico de Apizaco. México. Recuperado en: newton.azc.uam.mx/mcc/01_esp/08_sitios/micai_06/WORKSHOPS/LOLACOM06/LOLACOM06_01.PDF

[9] Introducción al QBE. Computer Sciences User Pages. Recuperado en: http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/qbe.pdf

[10] UDELAR. (s.f). *Persistencia en Sistemas O: O. Taller de programación.* Instituto de Computación. Facultad de Ingeniería. Universidad de la República. Uruguay. Recuperado de: http://www.fing.edu.uy/tecnoinf/mvd/cursos/progapli/material/teo/PA-03-Persistencia.pdf

[11] GENBETA: dev Desarrollo y software. (2014). NetBeans.Madrid. España. Recuperado de: http://www.genbetadev.com/herramientas/netbeans-1

[12] SUBVERSION (2015). *Apache TM subversión ¨Enterprise-class centralized versión control for the masses¨.* Recuperado de: https://subversion.apache.org/